Homework 5a & 6a

CS386D Database Systems
Created on 3/2/20
Version: 1.0

Assigned: 3/2/2020

HW 5a (milestone 1) due: 3/6/2020

HW 6a due: 3/13/2020

Objective: First.experience with, one, interacting with a large quantity of data, two, physical database optimization, i.e. introduce and observe the impact of adding secondary indexes. You will see that this is actually a lab version of homework problem 8.4.1. Also, as a lab, you will learn a collection of techniques commonly needed and/or used in practice to keep a RDBMS running well.

Introduction:
This is the first lab where you engage with an RDBMS containing millions of rows. There will be elements system configuration to make the software work. You must overcome the default transaction behavior of an RDBMS in order to load the data. (The default behavior is to start and commit a transaction for each row inserted into the database. i.e. if you do not figure out how to avoid the default behavior the database will stop, write the transcation logs and then resume, millions of times.)

Expect the database load to take several minutes to 20+ minutes. Hint: Do the lab with a much smaller test size, e.g. 10,000 rows. Only after you are confident that you have captured the functionality, scale the test set to the required size. (If you find that your machine is taking much more than 20 minutes to do part one of the assignment you may contact me with details of your machine configuaration and I'll reduce your test set size. D.M.). Be aware of this challenge as you schedule your work.

5a: (Milestone 1) You must load a SQL database, of your choice, (but not mySQL or SQLite), on a machine of your choice, (or spinup a cloud instance of your choice). You may also use a computer language of your choice. Though the write-up below speaks to Java, you may substitute any language you are comfortable with.

Some Specific Lessons: In order to help you make sense of the assignment, the following are [some of] the precise lessons of the assignment

1) Secondary indexes add overhead to inserting rows into a table.
2) Bulk loading (inserting many rows) is (usualy) much faster on sorted data and, can result in faster query performance.
3) Secondary indexes can be used to expedite queries beyond simply providing a fast access path to data.
4) Building and rebuilding indexes in an RDBMS is a fast operation.

Method: Build and query a database table using several different physical configurations and measure the ensuing performance.

Schema:

```
CREATE TABLE benchmark (
    theKey   NUMBER   PRIMARY KEY,
    columnA  NUMBER,
    columnB  NUMBER,
    filler   CHAR(247)
);
```

The row values are further defined as follows:

- theKey: a unique integer value
- columnA: an integer [1 - 50,000] chosen at random
- columnB: an integer [1 - 50,000] chosen at random
- filler: random text. The intention of this column is simply to force the database to store a typical number of rows on a page and to try to better capture real I/O time. Fixed length character strings, by definition, reserve space. But implementations do not necessarily consume that fixed length.

Data: The *benchmark* table should be populated with 5,000,000 rows. Use two variations of a database generator to populate the benchmark table:

- **Variation I**: generate and load the rows in sorted order on the primary key.
- **Variation II**: generate the rows such that the primary key value is chosen at random, without replacement, from the integers.

The goal of the assignment is to measure database performance in a series of controlled experiments, i.e. The goal is not to be clever and quickly determine the fastest way of accomplishing the workload. That means you must do everything you can to keep the differences among runs to a minimum. For example, it may be very tempting to use features like auto-generation of keys in sequence for variation I (sorted load). But, since you cannot use that feature for variation II, you shouldn't use it at all.
:

Build your data loader as client JDBC code[1]. Load the database with variation 1 and variation II, timing how long each takes. Verify the consistency of your results. If both varations are taking the same amount of time, try boosting the number of rows, say to 15,000,000. Report to class on Piazza if this makes a difference. Include the hardware configuration of your machine in your report.

Milestone 1: Achieve loading 5,000,000 rows in a reasonable amount of time. Run the test query specified below, (just once). Turn in with the rest of homework 5, a screen shot of your query and the output of the database.

About the remainder of the assignment:

You will be timing many runs. There are often issues with making timing calls in virtualized environments. So if you are using a virtual machine, be careful.

If you are not expeienced with JDBC, below are points I am careful to detail in the undergraduate class. You will also find a couple of files I use in that class in the Reading folder on Box.

Remember core lessons of JDBC coding

a) Never make a JDBC method call without writing a meaninful exception handler.
b) Open your connnection (and thus verify authorization) as early as possible in the execution of your program.
c) Write a "try" of the JDBC driver to force it's loading as one of the first things in your main.
   - Do this even if you can establish a connection at the beginning of your program. Often if there is a problem the exceptions thrown from the try and from the construction of a connection are differeent and provide additional information critical to debugging.
d) Do not ever trust anyone telling you that you have compatible drivers on the client and server sides. Demand to see the configuration with your own eyes.

The list above is not derived from academic sources. At some point soneone under my supervision failed to do one of the above and cost an entire team one or more days.

[1] If you are not a Java programmer you may use another language and database interface.

The Assignment Proper:

The following details a number of physical organizations of the database, workloads (query + database), and a set of measurement to be made.

Physical Organizations: Consider the following index configurations:

1. No secondary indexes
2. Secondary index on columnA
3. Secondary index on columnB
4. Secondary index on both columnA and columnB

Queries: There are three kinds of queries to be considered for this benchmarking problem:

Query 1

```
SELECT * FROM benchmark
WHERE benchmark.columnA = 25000
```

Query 2

```
SELECT * FROM benchmark
WHERE benchmark.columnB = 25000
```

Query 3

```
SELECT * FROM benchmark
WHERE benchmark.columnA = 25000
    AND benchmark.columnB = 25000
```

When taking measurements, do not simply run each of these queries once, nor is the value 25000 special. You should run each query a large number of times (at least 10) and average the results (think about: why?). Also, when averaging, each query should have a different constant value, (do not reuse 25000). At the same time, for repeatability, you should not generate different random constants for each run. That is, if you are going to run 10 possible "Query 1's" you should pick 10 different constants and reuse them each time you run the set of 10 (think about: why?).

Measurements: [not necessarily measured in this order]

Measure the loading time of the database using the two different data generators for each for each of the 4 physical organizations

A side issue: You will see that loading a database without secondary indexes is much faster than one with secondary indexes. A common practice is to "drop" secondary indexes before a bulk load, and then to rebuild them (reissue the create command). Make an additional measurement. Find out the time it takes to load your database without secondary indexes and then build them, and compare that to loading the database with the secondary indexes in place.

Measure the query execution time for each of the 3 queries on each of the 4 physical organizations having loaded the database using each of the 2 variations of the database generator (so, 24 measurements of query time). It is possible that some of these runs might take several hours. So,
1) Do not wait until the day before the homework is due to get started.
2) You may want to do the exercise on a much smaller database, e.g. 10,000 rows first. After you are satisfied that you have done the assignment correctly make your measurements for 5M rows.

Turn in:

1) Performance results

a) Fill in the following table of absolute measurements

	Load Time		Query 1		Query 2		Query 3	
Data Generator	I	II	I	II	I	II	I	II
Physical Organization								
1								
2								
3								
4								

b) Measurements with respect to speed-up

i) Let the load time of the database on physical organization 1 on sorted inserts form a baseline. What is the speed-up, (other-organization/baseline), for the remaining load times?

ii) Similarly, for each of the queries, let the physical organization 1 on sorted inserts form a baseline. What is the speed-up, for the remaining executions? (That is what is the exeuction time of the result divided by the execution of the baseline. Notice table already has "1" inserted to represent the baseline.

Report your results by completing the following table.

	Load Time		Query 1		Query 2		Query 3	
Data Generator	I	II	I	II	I	II	I	II
Physical Organization								
1	1		1		1		1	
2								
3								
4								

2) Discuss your findings. (4-10 sentences) (Hint: Use the SQL explain command, or its GUI equivalent)