Clarification of Homework 4 part B.

I was away from my email over the weekend. I understand I was out-of-pocket at a bad time with respect to questions about Homework 4 part B.

Here are a number of points/hints that may help you with the assignment.

1a) This assignment is quite small compared to homework 3 part B. I think part of the confusion is that you are reading too much into it.

Recall at the end of the lecture on secondary indexes there was an example:

Select *
From …
Where city = ‘Austin’.

In that example we went through what would happen in the table contained 10 different cities or it contained 1,000 different cities. When it contained just 10 cities, we concluded that a secondary index was not likely to be beneficial, but when the table contained 1,000 different cities the secondary index would be beneficial.

The goal of this homework is for you to determine, if you were using the “city” example, the exact number of cities where the performance using a secondary index is equal to the performance if there is no secondary index.

You are asked to express your final solution as a proportion of the number of pages needed to determine the output of the query. That is, if you use a secondary index, only those data pages containing records will be read into memory. Compare that number of pages to the total number of data pages to hold the table.

Recall, if you don’t use a secondary index, then the query will sequentially scan the entire table. Measured as time-per-block, a sequential scan of an entire table is much faster than reading each block, one at a time, using a secondary index. Even if many of the blocks are not needed, it might be faster to read the blocks sequentially, rather than one at a time.

1b) Exercise (not to hand in). You may want to check that this tradeoff is real. To do so, load the table but only have 2 unique values in columnA. With only two different values, it is near certain that every data page will contain rows that appear in the output. Thus, a sequential scan will be much faster than using a secondary index.

If you try this, and you find roughly the same query execution time, it may be that the query optimizer makes exactly the calculation we made in class, such that, even if a secondary index exists, the optimizer will choose a sequential scan (i.e. the optimizer knows the sequential scan will be faster, to does not choose to use the secondary index).
If the RDBMS has a query optimizer that is smart enough to do this, rather than measuring execution times of the query, you may use “Explain” command. That is, you simply have to ask the query engine to execute the query, and the explain command will tell you if it would choose a sequential scan, or use the index. (See “explain” in your RDBMS’ user manual.)

2) Some of you have expressed concern over the time it takes your RDBMS to load 1,000,000 rows. The assignment asks you to determine “a proportion of pages”. All things being equal, the proportion of pages will be the same for 100,000 rows or 1,000,000 rows. Thus, you are free to use a smaller table.

3) In response to your questions I double checked that mySQL has an Explain command; it does (this is part of the SQL standard). I also took a look as to the precise command mySQL uses for updating its data catalog. (This is not part of the SQL standard.) Where many databases have an explicit command, (e.g. DB2 its runstats), per the following, mySQL with InnoDB has no explicit command. What it does do has implications on making your measurements cold start vs. warm start. So I replicate the key passage here:

> “Unlike MyISAM, InnoDB does not store an index cardinality value in its tables. Instead, InnoDB computes a cardinality for a table the first time it accesses it after startup. With a large number of tables, this might take significant time. It is the initial table open operation that is important, so to “warm up” a table for later use, access it immediately after startup by issuing a statement such as SELECT 1 FROM tbl_name LIMIT 1.