CS434 – Data Base Theory and Design

Project #5

Author: Mark McKenney
Organization: CS434, Department of Computer Science, SIUE
Points: This assignment is worth 35 points.

Team Database Application (TDA): Part 5 - Querying and Schema Tuning

Description

1. **(9 points)** Write three queries on your TDA database, using the select-from-where construct of SQL. To receive full credit, all of your queries should exhibit some interesting feature of SQL: queries over more than one relation, or subqueries, for example. We suggest that you experiment with your SQL commands on a small database (e.g., your hand-created database), before running them on the large database that you loaded in TDA part 4. Initial debugging is much easier when you're operating on small amounts of data. Once you're confident that your queries are working, run them on your complete database. If you discover that most or all of your "interesting" queries return an empty answer on your large database, check whether you followed the instructions in Assignment #4 for generating data values that join properly. You will need to modify your data generator accordingly. **Turn in a copy of your SQL queries, along with a script illustrating their execution.** Your script should be sufficient to convince us that your commands run successfully. Please do not, however, turn in query results that are thousands (or hundreds of thousands) of lines long! (Your actual query results should be thousands or hundreds of thousands of lines long, just truncate the output for what you are turning in. Or, use a `count( )` function to indicate the number of tuples being returned)

2. **(6 points)** Write three data modification commands (one query each for update, insert, delete) on your TDA database. Most of these commands should be "interesting," in the sense that they involve some complex feature, such as inserting the result of a query, updating several tuples at once, or deleting a set of tuples that is more than one but less than all the tuples in a relation. As for the queries in (1), you might want to try out your commands on small data before trying it on your full database. **Hand in a copy of your modification commands and a script that shows your modification commands running in a convincing fashion.**

3. **(5 points)** Create a view on top of your database schema. **Show your CREATE VIEW statement and the response of the system.** Also, **show a query involving the view and the system response** (but truncate the response if there are more than a few tuples produced). **Finally, show a script of what happens when you try to update your view**, say by inserting a new tuple into it. Is your view updatable? **Please state why or why not?**

4. **(15 points)** In part (1) you probably discovered that some queries run very slowly over your large database. An important technique for improving the performance of queries is to create indexes. An index on an attribute `A` of relation `R` allows the database to find quickly all tuples in `R` with a given value for attribute `A`. This index is useful if a value of `A` is specified by your query (in the where-clause). It may also be useful if `A` is involved in a join that equates it to some other attribute. For example, in the query

   ```
   SELECT  Bars.address
   FROM    Drinkers, Bars
   WHERE   Drinkers.name = 'joe'
   AND Drinkers.frequents = Bars.name;
   ```

   we might use an index on Drinkers.name to help us find the tuple for drinker Joe quickly. We might also like an index on Bars.name, so we can take all the bars Joe frequents and quickly find the tuples for those bars to read their addresses.

   If the attribute list contains more than one attribute, then the index requires values for all the listed attributes to find a tuple. That situation might be helpful if the attributes together form a key, for example.

   To get rid of an index, you can say DROP INDEX followed by the name of the index. Notice that each index must have a name, even though we only refer to the name if we want to drop the index.

   Create at least one useful index for your TDA. Run your queries from part (2) on your large database with the indexes and without the indexes. For example, in order to see SQL Server statistics on Oracle during your session, you must activate SQLPLUS' internal SQL server tuning options using the SET command. Your settings will remain valid for the duration of a session. Here are a few useful options:

   - SET TIMING ON. Displays the elapsed execution time in milliseconds.
   - SET AUTOTRACE ON. Causes SQLPLUS to automatically display the query plan and execution statistics for statements as you execute them.

   If you cannot find an execution statistics program for your database, you can resort to "wall clock time."

   Naturally, these performance times may be affected by external factors such as system load, etc. Still, you should see a difference between the execution times with indexes and the times without. **Turn in a script**

showing your commands to create the indexes, and showing the relative times of query execution with and without indexes. Provide a brief explanation for your observations.

Deliverables:

Turn in a **SINGLE PDF** document including the statements to create the items listed above. Also, turn in evidence that proves that the statements and commands ran successfully. These can be typescripts, screenshots, or some other form of evidence.

Attach a copy of your ER diagram.

Be sure you have included:

1. 3 queries and their results.
2. 3 data modification commands and proof that they ran (for example, a return statement indicated that some number of rows were affected).
3. A CREATE VIEW statement and evidence that it worked.
4. A CREATE INDEX statement, evidence that worked, and the time it took a run a query before the index was created, and a faster time that it took the same query to run after the index was created.