Homework 3

Reading: see the calendar page linked to the class web site

Assigned: 2/11/09

Due: 11:59PM, 2/17/08, by email to the T.A., write CS386D HW3 in the subject line, Or hardcopy in class.

Special Supplementary “Hand-in”: For class on Thursday, 2/17, bring to class, on hardcopy, your answers to part B, questions 0 and 1. These will be used as the basis of class discussions.

Homework problems:

Part A:
Text Problems:
- 4.7.3, 4.7.4, 4.7.5
- 7.2.3 a,b
- 7.5.2a

Think about text problems:
- 4.7.8
- 7.1.5b
- 7.2.2b
- 7.3.1a
- 7.5.2c

Cont’d
Part B: Implementation Homework

Objective: Gain a firsthand feel for the performance, and physical optimization of a database. 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 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. (i.e. see slide 45 lecture 3)

Method: Build and query a database table using several different physical configurations and measure the ensuing performance. The assignment is modeled after homework 2 exercise 8.4.1.

Database: You may use a RDBMS of your choice on a machine of your choice. (We anticipate that most students will probably use mySQL.)¹ That said, Not about configurations:

1) When using MySQL: there are two versions of engines between the query layer and the storage. Use the one with the newer engine, called “innodb”. The old one doesn’t support important things, like cascaded deletes.
2) You need to turn auto commits to off. If not, the database commits after every insert. From java execute “con.setAutoCommit(false);” before your inserts. When finished resume commits by executing “con.commit()”

Schema:

```sql
CREATE TABLE benchmark (  theKey int PRIMARY KEY,  columnA int,  columnB int,  filler CHAR(247) ); // an extra column just to fill make sure your database takes some disk space
```

Different than text problem 8.4.1, one column always serves as the primary key. Two additional columns, columnA and columnB, may be augmented with secondary indexes.

---

¹ It would be of great interest for the class to compare results across vendors; so much so, I’ll add the following inducement. If you choose a different RDBMS, let me know which one, before the weekend, and if in the process your effort blows-up, comes talk to me about it, and you’ll be forgiven that part of the assignment. In other words, if you use something other than mySQL, you will bear no risk to your grade.
Database:

You must write a program that will generate data to populate the benchmark table. The benchmark table should contain 1,000,000 rows. The rows values are further defined as follows:

- theKey int PRIMARY KEY, a unique integer value
- columnA int, the integers [1-50,000] chosen at random
- columnB int, the integers [1-50,000] chosen at random
- filler CHAR(247), random, or don’t care. The intention of this column is simply to force the database to store a typical number of rows on pages and to try to better simulate real I/O time. Fixed length character strings, by definition, reserve space. That does not mean that if the character string is NULL or shorter than 247 characters that the database won’t still try to be clever and exploit space-saving techniques associated with variable length strings (VARCHAR).

Create two variations of a database generator to populate the benchmark table. The objective is to load the database in sorted order and in an unsorted order.

**Variation I**: generate the rows sequentially such that the primary key value is simply a counter, i.e., the rows will be generated in a sorted order.

**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, not the execution speed of your assignment. That means you must do everything you can to keep the differences in testing framework to a minimum. With respect to primary key generation this means you should not use a built-in DBMS method to generate primary keys for variation 1, and a program of your own making for variation II.

Juan’s solution to this was to create, in memory, a list of integers from 1 to 1 million. For variation 1, he simply sequentially accessed the list. For variation II Juan randomized the list and then sequentially accessed resulting list.

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, as follows.
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 the 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 (think about: why?).

Measurements:

[not necessarily 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.

Measure the query execution time for each the 3 queries on each of the 4 physical organizations having loaded the database using each of the 2 variations of the database generator. (24 measurements of query time.) Note, 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. You may get done with your work in a few hours, but the computer may take another day before you have results to turn in.

2) You may want to do the exercise on a much smaller database e.g. 10,000 rows, and after you are satisfied you have done the assignment correctly make your measurements for 1M rows.

Turn in:

0) Detail your software configuration, (for starters, DBMS, OS and version numbers) and hardware configurations, (for starters, processor, RAM, Disk etc.)
1) Performance results
   a) Fill in the following table of absolute measurements

<table>
<thead>
<tr>
<th></th>
<th>Load Time</th>
<th>Query 1</th>
<th>Query 2</th>
<th>Query 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>Data Generator</td>
<td>I</td>
<td>II</td>
<td>I</td>
<td>II</td>
</tr>
<tr>
<td>Physical Organization</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>2</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>3</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>4</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

   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?

   iii) Report your results by completing the following table.

<table>
<thead>
<tr>
<th></th>
<th>Load Time</th>
<th>Query 1</th>
<th>Query 2</th>
<th>Query 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>Data Generator</td>
<td>I</td>
<td>II</td>
<td>I</td>
<td>II</td>
</tr>
<tr>
<td>Physical Organization</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>2</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>3</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>4</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

2) Discuss your findings. (4-10 sentences)