Final exam for CS-E4610

Thu, Apr 6, 2017

This is an **open book** exam. It is allowed to use any textbook, printed material, or personal notes brought in the room. Use of electronic devices is **not** allowed.

There are **four** problems. To receive full points you need to answer **all** problems.

The teaching staff will come in class around **10am** to answer clarifying questions about the problems.

Problem 1 (20 points)

1. Mark all that apply. In a database system, you build an index to ...

   (a) ... allow many users to access the same table at the same time.

   (b) ... keep the data in memory for easy access.

   (c) ... speed-up certain types of queries.

   (d) ... make sure that the data stays consistent.

   Explain briefly your answer.

2. In relational database systems, an *equality join* between two tables returns the cartesian product of the corresponding relations.

   (a) True

   (b) False

   Explain briefly your answer.

3. Consider a relation *R(a, b, c, d)* stored over 1024 disk pages. The relation is to be sorted on attribute a with the *external sorting algorithm* with B = 2000 buffer (memory) pages available. How many times will the algorithm load each page from disk before it outputs the result?

   - Exactly once.
   - Exactly twice.
   - More than twice.
   - Once if attribute a is a key; up to 1024^2 otherwise.

   Explain briefly your answer.

4. Consider a relational table *R(a, b, c, d)* and a B+ tree index of type 2 built on attribute a of that table. Mark all that apply based on this information.

   (a) Data entries in each page of the index file are sorted by the value of a.

   (b) Data entries in each page of the relation file are sorted by the value of a.

   (c) As a new record is inserted into the relation, it is possible that new *index entries* are inserted into the non-leaf nodes of the tree, while no new *data entry* is inserted into a leaf node.

   (d) The index can be used to answer equality queries on attribute a, but not range queries on attribute a.

   Explain briefly your answer.

5. Consider a relational table *R(a, b, c, d)* and a hash index of type 2 built on attribute a. Attribute a is a key. The index is built using the extendible hashing scheme (i.e., it uses a directory). Each bucket of the index can hold up to 4 data entries.

   Originally both the table and the index are empty. For every record added to the table, the index is also updated. No deletions from the table occur.

   As records are added to the table, is it possible at any point that some buckets have *local depth* 3 while others have *local depth* 1? If so, describe such a case. If not, explain why.

6. Imagine that a law firm hired you to design and build a IR system for legal cases. The IR system should work as follows: A lawyer provides a few keywords and the system should return the past legal cases that contain those keywords. It is important to be able to retrieve *all* relevant past legal cases, otherwise the lawyers may miss important facts, and will not be able to serve the best interests of their clients. Assume that you have built a prototype of the IR system, and you want to evaluate it.

   Explain: is it more important to have *high precision* or *high recall* and why?

Problem 2 (25 points)

Consider two relations $R(a, b, c)$ and $S(a, d)$. Attribute a is a key for table S. Relation R is stored over $M = 1000$ pages on disk; relation S is stored over $N = 1500$ pages on disk. There are $B = 1002$ pages available in memory.

We wish to compute the join $R \bowtie_{R.a=S.a} S$, and the available join algorithms are (i) block-nested-loops join, (ii) sort-merge join, (iii) hash join.

We measure the **cost** of the algorithms in terms of the *average* number of times that a page of either relation is loaded from disk to memory.

1. Calculate the cost of each of the aforementioned algorithms, as defined above.
   Note Provide exact calculations, not $O(\cdot)$ notation.

2. Consider another instance of the problem where the available memory B remains the same, but the number of pages occupied by the relations is k times as large. In other words, relation R occupies $kM = 1000k$ pages on disk; and relation S occupies $kN = 1500k$ pages on disk.

   - What is the cost of the aforementioned algorithms as a function of k?
   - Which algorithm performs best for very large k?

   Note State clearly any additional assumptions you make in your calculations.

Problem 3 (25 points)

The database of an online store contains information about the purchases of its customers. For each purchased product in each session (i.e., visit to the online store), the database stores the following information:

- the ID of the customer who made the purchase;
- the ID of the product and the number of items that were purchased in the same session – e.g., the customer might have purchased five (5) items of the product 'Best Shampoo Ever' in one session;
- whether the product was purchased as a gift for another person;
- the delivery address of the customer (street, number, apartment, city, country, zipcode);
- the date of the purchase (day, month, year).

To store and manage the data, we have the following options: (i) store them in a single relational table in PostgreSQL; (ii) store them in a single collection in MongoDB; (iii) store them in a single text file over a Spark cluster.

For each of the aforementioned cases:

1. Provide examples of purchased products to show how the information is stored. (Show specific examples of relational-table rows, MongoDB documents, and text-file lines).
2. Provide a query to calculate *the total number of items purchased for each product ID*.
3. Provide a query to calculate *the total number of non-gift items purchased for all combinations of product ID, zipcode, and month* that appear in the data.