University of Wisconsin-Madison
Computer Sciences Department

Database Qualifying Exam
Spring 2011

GENERAL INSTRUCTIONS

Answer each question in a separate book.

Indicate on the cover of *each* book the area of the exam, your code number, and the question answered in that book. On *one* of your books list the numbers of *all* the questions answered. Return all answer books in the folder provided. Additional answer books are available if needed.

Do not write your name on any answer book.

SPECIFIC INSTRUCTIONS

Answer **all** five (5) questions. Before beginning to answer a question make sure that you read it carefully. If you are confused about what the question means, state any assumptions that you have made in formulating your answer. Good luck!

The grade you will receive for each question will depend on both the correctness of your answer and the quality of the writing of your answer.

Policy on misprints and ambiguities:

The Exam Committee tries to proofread the exam as carefully as possible. Nevertheless, the exam sometimes contains misprints and ambiguities. If you are convinced a problem has been stated incorrectly, mention this to the proctor. If necessary, the proctor can contact a representative of the area to resolve problems during the *first hour* of the exam. In any case, you should indicate your interpretation of the problem in your written answer. Your interpretation should be such that the problem is nontrivial.

1: Locking

A) Consider the locking protocol in Gray et al., which says:

1) Before requesting an S or IS lock on a node, the transaction must lock **ANY** path to the root in IS mode or greater.
2) Before requesting an IX, SIX or X lock on a node, the transaction must lock **ALL** paths to the root in IX mode or greater.

Would the protocol work if you flipped the ALL and ANY in the protocol? Explain.

B) What is the difference between a lock and a latch in a DBMS? Give clear examples of uses of locks and latches in a DBMS.

C) In lock-based concurrency control systems, why do short locks create problems for their recovery systems?

2: Join Algorithms

Consider two equijoin algorithms for joining tables R and S: the GRACE hash join, and sort-merge join. Furthermore, suppose that R and S can be partitioned in one pass, and that R and S can be sorted in two passes.

(A) Give cost formulas from the Shapiro paper for these two algorithms.

(B) Now focus on the I/O cost. We want you to be more precise than Shapiro. Some of the I/Os will be random. Some will be sequential. Assume that you have only one disk, and write and explain the formulas for the number of random and sequential I/Os done by the two algorithms. Note that some reads that would otherwise be sequential will be random because they are interspersed with other I/Os from other parts of the algorithm. You can assume that the answer tuples generated are not stored to disk (so writes of the answer will not affect the random/sequential nature of I/Os during the join.)

(C) Hash-based join algorithms are blocking algorithms. They must scan both R and S before they produce even the first output tuple. Propose a hash-based join algorithm that is able to produce some output tuples without necessarily scanning the entirety of R or S.

3: Buffer Pool Aware Query Optimization

In the past, researchers have proposed the idea of buffer-pool aware query optimization. In this approach, the query optimizer takes into account the current contents of the buffer pool when optimizing queries. That is, it checks what is in the buffer pool; it may choose different query plans depending upon what it does or does not find in the buffer pool.

A) Give a scenario where you expect this to be effective. That is, describe a situation where a buffer pool aware optimizer would produce a better plan than one that is not aware of the buffer pool contents. You should be specific, describing the query, query plans, and buffer pool contents involved.

B) Despite your scenario from part A, commercial query optimizers are not buffer pool aware. Speculate on some reasons for this - that is, give some arguments for why current query optimizers are not buffer pool aware.

4: Indexing

Consider the bitslice and column/projection indexing methods.

A) Give an example, one for each of the two indexing methods, for which that indexing method is clearly superior over the others for simple queries with only selection predicate clauses (that is, no aggregates or join predicates).

B) Consider the "AVERAGE" aggregate. Write the pseudocode (with comments) for computing this aggregates using each of the two indices above.

C) Outline an efficient algorithm for computing the MEDIAN aggregate using a bitslice index. (You are not required to write the pseudocode for this part, but make sure you are clear in conveying your solution. If you feel you can convey your solution more clearly with a pseudocode, then go for it.) Comment on the performance of your method compared to computing a MEDIAN using a column index by sorting the column index to compute the MEDIAN.

5: Information Retrieval

A) Briefly describe a solution to build an inverted index for a corpus of 100 million documents, assuming a cluster of 50 machines, using a parallel object-relational DBMS.

B) Suppose you constantly have new documents streaming into the above corpus (for example, you are building an inverted index for the Web, and you are continuously crawling the Web for new documents). Briefly describe how you can modify your solution to incrementally update the inverted index.