University of Wisconsin-Madison
Computer Sciences Department

Database Qualifying Exam
Fall 2014

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. Indexing

Traditional database indices like B+trees and Hash-based indices are optimized for the case when the search finds match(es) in the index, i.e. when evaluating the effectiveness of these indices, we focus on the case when the search key *hits* in the index.

Consider a scenario in which you have many search queries that will not have a match in the database (index). Design an optimized index for the case when the search term *misses* in the index search. Think of both a tree-based and a hash-based index.

If your index is different than a traditional index (that is optimized for the case of the hits), show examples of when your index would perform better than a traditional index.

Also, discuss the efficiency of your index for both equality and range predicates.

2. Data Integration

(a) Briefly discuss the similarities and differences between a virtual data integration system and a RDBMS.

(b) Name at least four key challenges in building a data integration system.

(c) Define and compare GAV and LAV. Give a concrete example of a set of source schema and a mediated schema, and then show how to relate the schemas using each of the above approaches (i.e., GAV and LAV).

(d) Formally define the notion of maximally contained rewritings.

3. Data Matching

Consider two tables A and B that contain employee descriptions. Suppose both tables have the following attributes: ename, affiliation, city, phone (where ename is the name of the employee). Suppose further that each table contains 500,000 tuples of such employee descriptions.

(a) Name at least four main approaches to perform entity matching between the two tables (i.e., find all tuple pairs [x,y] such that x is from A and y is from B and x and y refer to the same real-world employee).

(b) Consider a rule-based approach to matching tables A and B. Matching 500,000 tuples in A with 500,000 tuples in B is extremely time consuming. Discuss how you can scale up the rule-based approach to matching A and B. Discuss scaling it up on a single machine, as well as on a cluster of machines.

(c) In practice, the two tables may contain a lot of synonyms. For example, a tuple in A may have city = "Los Angeles", and another tuple in B have may city = "LA". Here "Los Angeles" and "LA" are synonyms. As another example, the values "UW", "UW-Madison", and "UWisc-Madison" in affiliation are all synonyms for "the University of Wisconsin, Madison".

Given two tables A and B, suppose a developer wants to find all synonyms in a particular attribute, say affiliation, and standardize them (before performing entity matching). Briefly discuss a solution to this synonym finding problem.

4. Joins

Consider the equijoin:

```
SELECT *
FROM R, S
WHERE R.A = S.B
```

Suppose that we have clustered B+tree indices on R.A and S.B. Consider the join algorithms merge-join, hybrid hash join, and index nested loops.

a) Give Shapiro-style cost formulas for the performance of these three algorithms. However, in your formula, distinguish between random and sequential I/O operations. State any assumptions you make in deriving these formulas.
b) Somehow characterize the relative performance of these algorithms as a function of properties of the input relations. For example, will one algorithm always dominate another? If so, when? If not, when will one tend to dominate the other? State any assumptions you are making.
c) In the classical setting we have assumed that random I/O is much slower than sequential I/O. However, new kinds of memory are either here or on the horizon that greatly reduce the difference between random and sequential I/O. If the performance of random I/O changes to be closer to that of sequential I/O, how does your analysis in part B change?

5. Recovery

This question deals with the two-phase commit protocol.

a) Give the basic, presumed-abort, and presumed-commit versions of 2-phase commit.
b) Consider a system in which data is replicated among multiple servers, perhaps for performance or availability. Suppose we adopt the policy that every data item has a “master” server, with the implication that every update to a data item X has to be made on the copy of X at the master of X first before being propagated to other servers. Do we need 2-phase commit in this scenario? Why or why not?