University of Wisconsin-Madison
Computer Sciences Department

Database Qualifying Exam
Spring 2013

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 **four** (4) 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: CONCURRENCY CONTROL

Consider a DBMS that implements a Gray-style hierarchical locking for concurrency control. Applications in this DBMS often run transaction in which attributes are updated by adding or subtracting constant values. In addition to the lock types introduced in the Gray et al.'79 paper, such systems often implement increment and decrement locks. Such locks protect the attribute that is locked and allow the locked object to be incremented/decremented. Answer the following questions for this scheme:

- Explain why such locks make sense, i.e. explain the advantages of using these locks.
- What do you think is the right compatibility of these (increment and decrement) locks with the existing lock types proposed in the Gray et al. paper?
- Can you think of the disadvantages of using these special lock types, as opposed to using just the lock types introduced in the Gray et al.'79 paper?

2: NORMAL FORMS AND INFERENCE RULES

Fix a set of attributes Z in a relation R.

A **functional dependency** is a statement $X \rightarrow Y$ where X, Y are subsets of Z. It is said to hold if whenever s and t are tuples in R such that s[X] = t[X] then s[Y] = t[Y].

A **multivalued dependency** is a statement of the form $X \rightarrow\rightarrow Y$, where X and Y are subsets of Z**.** It is said to hold whenever there are tuples *s* and *t* in R such that
if s[X] = t[Y], then there is a tuple *u* in R such that u[XY] = s[XY] and
u[Z – XY] = t[Z – XY] .

(1) Prove or provide an explicit counterexample (i.e., a relation) for the following where X, Y, and U are sets of attributes, i.e., subsets of Z.

(a) If $X \rightarrow Y$ then $X \rightarrow\rightarrow Y$
(b) If $X \rightarrow\rightarrow Y$ then $X \rightarrow Y$
(c) If $X \rightarrow Y$ and $Y \rightarrow U$ then $X \rightarrow U$
(d) If $X \rightarrow\rightarrow Y$ and $Y \rightarrow U$, then $X \rightarrow\rightarrow U$

(2) Describe the connection between dependencies, normal forms, and update consistency.

3: INFORMATION RETRIEVAL

a) Describe the PageRank algorithm and show the formula that is used to compute the PageRank scores.

b) Most commercial search engines such as Google and Bing use not just the PageRank scores, but also many other types of information to compute the final scores of Web pages. Describe examples of such types of information. Describe an algorithm that uses such types of information as well as the PageRank scores to compute a final score for each Web page. If the algorithm has a set of parameters, describe how those parameters are initialized.

c) Describe how the above algorithm can be run in parallel on a cluster of machines.

4: JOIN ALGORITHMS

In this problem you are to consider a special kind of join. Rather than the more common "equijoin", "band joins" look for tuples that have joining attribute values "near" each other. In SQL this might look like:

```
SELECT *
FROM R, S
Where R.A <= S.B + c AND R.A >= S.B – c
```

Here the "band" is of size "c".

a) Describe how you would evaluate such a band join using Nested Loops, Index Nested Loops, and Sort-Merge. For each algorithm, say whether the standard equijoin version needs to be modified to evaluate the band join, and comment on the relative performance you would expect between the equijoin and band join versions.
b) Describe how you would evaluate such a band join on a shared-nothing cluster. Again, comment on the relative performance of the band join and more standard parallel equijoin algorithms.
c) Can you give a real-world example of a join that would be a band join? That is, can you give a question (in English) that would be answered by a band join? (E.g., a real-world example of an equijoin could be "For each sailor, find all the reservations for that sailor.")