University of Wisconsin-Madison
Computer Sciences Department

Database Qualifying Exam
Fall 2017

INSTRUCTIONS

1. Answer each question in a separate book.
2. 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.
3. **Do not write your name on any answer book.**
4. 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.
5. 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: QUERY CONTAINMENT

1. Consider the following two UCQs (Union of Conjunctive Queries):

```
P(x,y) :- R(x,z), R(z,w), R(w,y).

Q(x,y) :- R(x,y).
Q(x,y) :- R(x,z), R(z,y).
```

Is P contained in Q, and is Q contained in P? Explain the reason for your answers in detail.

2. For queries q and q' in CQ, recall that it is NP-complete to decide whether q is contained in q'. One proof of this statement uses the idea of a canonical database, where we construct a database D from the query q such that if q' is true on D, then q is contained in q'. This suggests that answering a query on a database is NP-complete as well. On the other hand, every day relational databases across the globe efficiently answer conjunctive queries (and more!). Explain this seeming contradiction.

3: STRING MATCHING

This question is about string matching, a ubiquitous operation in many data integration applications.

1. Edit distance is a popular string distance score. The edit distance $d(x, y)$ computes the minimal cost of transforming a string x into string y, by deleting a character, inserting a character, or substituting a character with another. The cost of each operation is 1. For example, $d(madison, maddisom) = 2$. Briefly describe an efficient solution to compute the edit distance between two given strings x and y. You do not have to fully describe the algorithm; sketching out the key ideas is sufficient. What is the time and space complexity of your algorithm?

2. Two popular string similarity measures are the Overlap measure and the Jaccard measure. To compute these measures for two strings x and y, first we tokenize the strings, say by generating all 2-grams (sequences of 2 consecutive characters) of the strings. Thus, string "madison" is tokenized into the set of 2-grams {"ma", "ad", "di", "is", "so", "on"}.

   Let Bx and By be the set of 2-grams generated for strings x and y, respectively. We then define:

   - the Overlap measure $O(x, y) = |Bx \cap By|$
   - the Jaccard measure $J(x, y) = \frac{|Bx \cap By|}{|Bx \cup By|}$

   where $|S|$ is the number of elements in set S.

   Prove that for t in the range $[0, 1]$:

$$J(x, y) \geq t \Leftrightarrow O(x, y) \geq \frac{t}{1+t}(|Bx| + |By|)$$

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.