`University of Wisconsin-Madison
Computer Sciences Department

Database Qualifying Exam
Fall 06

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 (NOTE: this is different from some previous years, when you were only asked to answer 4 of 5.) 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. Gray et al. locking paper.

Suppose we have a hierarchy with two paths from the root (DB) to the leaf (Record):

DB – File – Page – Record

And

DB – Index – Record

a. Suppose we want to get an S lock on a page p. What other locks must the transaction acquire?

b. Suppose now that we want to upgrade the lock on p to be an X lock. What, if any, other locks must be upgraded/acquired?

c. Gray et al. say that the locks should be acquired top-down and released bottom-up. What problems if any could occur if some transaction does the reverse? (Gets locks bottom-up and releases them top-down.)

d. Joe Qualtaker has read this paper and decides to implement a system with relaxed degrees of consistency. He decides to offer degree 1 consistency and degree 3 consistency. His idea is that at startup time, each transaction can choose a level and then run at that level. His intuition is that if a transaction wants serializiblity, it can ask for degree 3; if not, it can just ask for degree 1. Is he correct? If yes, say why; if no, say which of the two types of transactions (degree 1 or 3) will be disappointed and give an example of how they might be disappointed.

2. Data Mining

Your task in this question is to implement the A-Priori Algorithm to discover frequently co-occurring items in a database of retail transactions.

a. Give a short overview of the A-Priori Algorithm and describe the main data structures and ideas.
b. Assume that there are only very few items overall (e.g, less than 100 individual items). (Note that the number of transactions can still be very large.) What modifications would you make to the A-Priori Algorithm to make it more efficient for this special case?
c. Assume that we partition the transactional database D into k partitions D1, ..., Dk. Assume that the minimum support is fixed at s. Show that if an itemset is frequent in D, then it is frequent in some Di. Describe an algorithm that uses this observation and finds all frequent itemsets in two scans over D.
d. Assume you are only interested in frequent itemsets that contain a fixed item X. One possibility is to generate all frequent itemsets using the A-Priori Algorithm and then filter those itemsets that do not contain X. Describe a more efficient algorithm for this problem.
e. Assume you have found all frequent itemsets for a given database D. Now you are given a new database D', and you want to find all frequent itemsets in (D union D').

One possibility is to completely rerun the A-Priori Algorithm on (D union D'). Can you make use of the observation in part c. to design a more efficient algorithm?

3. R-Trees

a. What criteria are used by an R-tree to select a leaf node to insert a new rectangle into?

b. Explain why there is frequently more than one leaf node to choose from?

c. Describe the quadratic splitting algorithm.

d. Design a non-2PL locking protocol for R-trees.

e. Explain how your algorithm increases concurrency for both readers, writers, and a mix of readers and writers

f. How does your algorithm differ from the corresponding B-tree algorithm

4. XML and XQUERY (from 764, spring 03)

Consider a database consisting of a collection of XML book elements as shown below

```
<bib>
    <book>
        <title> book1 </title>
        <publisher>  Morgan Kauffmann
        </publisher>
        <year> 1998 </year>
        <author> author1 </author>
        <author>  author2 </author>
        <author>  author3 </author>
    </book>
    <book>
        <title> book2 </title>
        <publisher>  Morgan Kauffmann
        </publisher>
        <year> 1996 </year>
        <author> author1 </author>
        <author>  author4 </author>
    </book>
      •
      •
      •
</bib>
```

There are two competing approaches for storing and querying collections of XML documents. One is to construct a "native" database system designed specifically for XML documents. The second is to use a relational database system.

a. Design a relational schema for the XML document above. Make sure that your design can handle an arbitrary number of author elements for each book (your design should be extensible to handle an arbitrary number of each element type in general).

b. Given your mapping, translate the following XML query into SQL.

```
FOR $x IN document("bib.xml")/bib/book
WHERE $x/year > 1995
RETURN $x/author
```

c. Comment on the relative efficiency of storing XML documents as trees on disk (perhaps with each element as a separate record on a slotted page) versus the relational approach with respect to the efficiency of query execution.

Consider the following query in XQuery:

```
FOR $p IN distinct(document("bib.xml")//publisher)
LET $b := document("bib.xml")/book[publisher = $p]
   WHERE count($b) > 100
RETURN $p
```

d. Explain what it computes.

e. Modify the above query to only consider books in which some paragraph (accessed as .../book//paragraph) contains the word "sailing".

5. Database System Architectures – A perfect storm

The following facts are true:

(1) In 30 years disks have gotten 10,000X larger but only 30X faster.

(2) CPU performance (and hence database performance) is highly dependent on obtaining very high data and instruction cache hit rates

(3) Relational tables have gotten much wider (100s or even 1000s of attributes is not uncommon).

a. Discuss how these trends effect database system performance with respect to executing queries.

b. To what extent is horizontal partitioning and parallelism a solution to the problems these trends pose on database system performance?

Recently, a very old idea (circa 1970) known as vertical partitioning has started to receive a lot of attention. The idea is quite simple. For example, consider a table Foo of 5 attributes (A, B, C, D, and E). A fully vertically partitioned representation of Foo would be stored as five separate columns: one per attribute value with each column stored in a separate file.

c. Explain why vertical partitioning might be expected to improve performance.

d. Consider the query select A,B, C from Foo where C > 25. As the file containing C values is scanned, the query execution engine must assemble qualifying C values with the corresponding A and B values to produce an answer that is equivalent to the result that would be obtained with the normal relational representation (termed NSM). Design an efficient algorithm for performing this operation.

e. Consider the following sequence of queries:

select A from Foo where C > 25
select A,B from Foo where C > 25
select A,B,C from Foo where C > 25
select A,B,C,D from Foo where C > 25
select A,B,C,D,E from Foo where C > 25

Discuss how the selectivity factor of the predicate C>25 effects whether a vertically partitioned version of Foo is faster or slower than a conventional NSM layout of the table Foo as the number of attributes in the target list increases.

f. Compression is another idea that has been considered for a long time as a way of making the disk appear faster. The original version of INGRES released in the late 1970s included compressed version of its access methods. Compare and contrast the effectiveness of compression in vertically partitioned tables compared to tables with a conventional layout.