UNIVERSITY OF WISCONSIN - MADISON
COMPUTER SCIENCES DEPARTMENT

DATABASE QUALIFYING EXAM
SPRING 1994

Answer any *five* (5) of the following *six* (6) questions. Before beginning to answer a question, be sure to read it carefully. If you are confused about what the question means, state any assumptions you have made in formulating your answer. *Good luck*!

1. Normalization

Consider the problem of normalizing a relational database schema.

(a) What is the role played in this problem by (i) view definitions and (ii) integrity constraints?

(b) What is the key characteristic of unnormalized schemas, and what are their undesirable behaviors and features that result from it?

(c) What additional concerns arise when relations are decomposed for normalization?

(d) How does BCNF address the above concerns and to what extent does it solve the main problem mentioned in (b)?

(e) How useful and important is 3NF, which is a lower normal form than BCNF? What does it achieve and what does it not? (In your discussion, do bring up your answers to (b) and (c).)

(f) Similarly, what is the purpose of 4NF and 5NF (project-join normal form), which are higher normal forms than BCNF? (Again, in your discussion, bring up your answer to (b).)

(g) Consider the following conjecture:

> "Assume that a relational schema is in 3NF and that some candidate key is simple (i.e., it consists of a single attribute). Then the relation schema is in 5NF."

If it is correct, give a proof. If it is not correct, give two modifications of the above that result in correct theorems; no need to give proofs in this case.

2. Data Mining

You've been hired by Data Is Money, Inc. to help them analyze customer data for a major department store chain. Among other pieces of information, the chain has been amassing a customer purchase file containing information about all customer purchases over the past 10 years. Suppose you are given this information as the following three tables:

PURCHASE(date, buyer-id, item-id, item-price, quantity)
ITEMS(item-id, item-cost, item-type, description)
CUSTOMERS(buyer-id, street, city, state, zip, income)

They've been keeping this information around in the hopes that it will help them learn something about their customers and merchandise, but have not yet begun to "mine" this data to see what they might actually be able to learn. The information is all stored in a relational DBMS.

(a) As a first step, design an algorithm to identify "hot pairs" of items, which are pairs of items that tend to be bought together. The algorithm should be parameterized by two quantities:

PairProb - The probability that when either one of these items is bought by a given customer on a given date, he/she will also buy the other item on the same date. This is a measure of how "hot" a given "hot pair" really is, and pairs that are not at least this hot are deemed uninteresting.

MinSupport - The minimum number of purchases of either item that suffices to make a "hot pair" interesting to the data analyst requesting a list of hot pairs. E.g., two items can be "always bought together" - but not very interesting - if each item was bought just once, and it happened that one customer bought them as a pair at that time. The MinSupport parameter serves to filter out such low-frequency items.

Your algorithm should be able to work with very large PURCHASE relations. The algorithm should be centralized, i.e., designed to work on a database server or mainframe, for starters. What is the I/O complexity of your "hot pair" algorithm in terms of the size/characteristics of the input data?

(b) Would you recommend implementing your algorithm as a new primitive operation (e.g., utility) within the DBMS or as an application program that runs outside the DBMS, keeping in mind that the latter is probably simpler? Justify your recommendation.

3. Parallel Database Systems

As an alternative to the standard "shared nothing" approach to using large-scale, non-shared memory parallel computer system architectures, at least one vendor (Oracle) takes a "shared disk" approach. In the context of such a machine, this means that all processors are permitted to fetch and buffer disk pages from any of the processors that have disks attached, and as a result, that any query can run on any processor (or set of processors).

(a) Identify the potential performance advantages and disadvantages of the shared disk approach vs. the shared nothing approach. Do this comparison with respect to the following issues:

i) concurrency control using page-level locking

ii) parallelizing the Grace (or hybrid-hash) join algorithm

iii) recovery using write-ahead logging

(b) What problems might be simplified by the use a shared-disk approach.

4. Multidimensional Data Structures

(a) You have a relation *emp* with just two attributes *sal* and *age*, and queries of the form ‘‘Find all emps with sal in the range s1 to s2 and age in the range a1 to a2.’’ You are considering the use of an R-Tree index.

(i) What would the entries in the leaves contain?

(ii) What would the entries in the parents of the leaves contain?

(iii) What can you say about performance of the range queries in comparison with a Grid File?

(iv) Would you expect to see 2-disk access performance for exact-match queries? Explain.

(b) Your DBMS only supports Grid File indexes. You want to store a collection of 2-D spatial objects. It is suggested that you do this by entering each object in *every* grid block that it overlaps with.

(i) What are the potential problems with this approach?

(ii) Can you see a situation (data/queries) in which this could be superior to R-Trees?

5. Relational Query Processing

A weather monitoring system records information about various meteorological phenomena as they happen. There are two relations, VOLCANOS and QUAKES, each with a *time* field containing the time of the event, a *name* field, and other fields that are not relevant for our purposes. A scientist asks the question: "For which volcano eruptions was the strength of the most recent earthquake greater than 7.0 on the Richter scale?"

(a) Write an SQL query to compute the answer.

(b) Suppose that both relations are sorted in ascending order by time. What can you say about the evaluation plan produced by a Selinger-style optimizer?

(c) Can you suggest a better plan?

6. Buffer Management

Suppose that in some DBMS system there is a query, say Q1, that requires a full scan of a table, say T. Also, suppose that there is another query, say Q2, that uses an index to look up a few records of another table, say S (the exact set of records is different each time Q2 is called.) Assume that both T and S are bigger than the buffer pool. This question asks you to consider how DBMIN will perform under a variety of workloads built up of Q1 and Q2.

(a) Suppose that for some reason a bunch of users run Q1 one at a time. (One after another.) Will DBMIN provide high performance on this workload? Why or why not? (Your answer should include a description of how DBMIN would allocate buffer space under these conditions.)

(b) Now suppose that a large number of users run Q1 at the same time. Again, will DBMIN provide high performance on this workload?

(c) Finally, suppose that a bunch of users run Q2 one after another at the same time that another user repeatedly runs Q1. How will DBMIN performan on this workload?