UNIVERSITY OF WISCONSIN - MADISON
COMPUTER SCIENCES DEPARTMENT

DATABASE QUALIFYING EXAM
FALL 1993

Answer any *six* (6) of the following *eight* (8) 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. B-tree Concurrency Control

(1) A number of special locking protocols have been proposed for B+ trees. What problem do they address (e.g., why consider special locking protocols for B+ tree)?

(2) Would the same problem be better or worse if the system used an optimistic concurrency control method rather than locking? Explain your answer.

(3) Describe one B+tree locking protocol.

(4) Consider an index that is implemented with linear hashing rather than by a B+ tree. Would you need a special purpose locking protocol for this data structure? If so, outline such a locking protocol. If not, explain why not.

2. Recovery

(1) Describe the basic shadow mechanism used in System R.

(2) What problems did the System R designers encounter with shadows as a recovery mechanism?

(3) O2, an object-oriented DBMS, uses a combination of shadows and logs. Shadows are used for transaction undo and logs are used for transaction redo. Does this combination avoid the problems the System R designers encountered? If so, why? If not, why not?

(4) Why might shadows be an attractive recovery mechanism for an object-oriented DBMS?

3. Parallel Database Systems

Consider a DBMS that needs to execute the joins (R1 $\bowtie$ R2), and (R3 $\bowtie$ R4). For the moment, assume that these are the only two queries that the system runs. Suppose that the DBMS is a parallel DBMS with 32 processors and 32 disks. There are a huge number of ways to set up the system to do these two joins. Three of these are the following:

(a) Fully decluster R1, R2, R3, and R4 over all 32 disks. Run the joins (R1 $\bowtie$ R2) and (R3 $\bowtie$ R4) concurrently using 32 processors for each join.

(b) Fully decluster R1, R2, R3, and R4 over all 32 disks. First run the join (R1 $\bowtie$ R2) over all 32 processors; next run the join (R3 $\bowtie$ R4) over all 32 processors.

(c) Decluster R1 and R2 over 16 disks, and R3 and R4 over the remaining 16 disks. Run the join (R1 $\bowtie$ R2) on the 16 processors attached to the disks on which R1 and R2 reside; at the same time, run the join (R3 $\bowtie$ R4) on the 16 processors on which R3 and R4 reside.

Assume that the declustering is done ‘‘ahead of time’’, that is, before the joins begin executing, so that the declustering itself is not part of the execution time. Your task in this problem is to discuss the performance tradeoffs among the three alternatives. Your answer should involve response time, relationship of relation size to memory size, disk usage, communication costs, and whatever else you think is pertinent to the problem. This problem is fairly open — state any assumptions you need to make in order to do your analysis.

4. Joins in Object-Oriented DBMSs

Consider an object-oriented data model where objects can point to other objects via OIDs. For example, an EMP object might contain the OID for the DEPT object that models the employee’s department and the EMP object that models the employee’s manager.

(1) Some object-oriented DBMS vendors claim that object-oriented data models eliminate the need for certain joins. Do you agree? Briefly support or refute the claim.

(2) Given the example above, how would vendors that make such claims be likely to process queries such as ‘‘find the names of the departments that employees over age 18 work in’’? What does this most closely correspond to in terms of relational database query processing strategies?

(3) Will the approach that you identified in (2) work well for large data sets? Why or why not, and what do you recommend doing if not?

5. SQL Queries

Consider the following schema used by ‘‘The Generic Company’’ to keep track of the closing prices of stocks of various company on various dates:

Stock (*company*, *date*, price)

In the above, the attributes in italics form the primary key. Consider a version of SQL that, in addition to the standard parts of the language, supports the keywords **union**, **minus**, **intersect**, **contains**, **not contains**, **exists**, and **not exists**. Also assume that ‘‘date.year’’, ‘‘date.month’’ and ‘‘date.day’’ will give you the year, month and day for a given date. Further, ‘‘date-1’’ and ‘‘date+1’’ can be used to refer to the previous and next days corresponding to ‘‘date’’, ‘‘date1-date2’’ to find the number of days between date1 and date2, and =, <, >, etc, to compare dates.

Write SQL statements to do the following tasks. If you find it convenient to define intermediate relations in formulating some of these queries, you are free to do so. (*Some of these queries may NOT be expressible in SQL. If so, state this, and briefly explain why.*)

(1) Find the difference between the highest closing price (of any company) in 1990 and the highest closing price of IBM.

(2) A ‘‘company peak’’ is a closing-price that is higher than or equal to the price of the same company on both the previous day and the next day. Find the first company peak for HP that falls on or after January 22, 1990.

(3) Find the longest interval between two consecutive peaks of HP (during the period for which data is available).

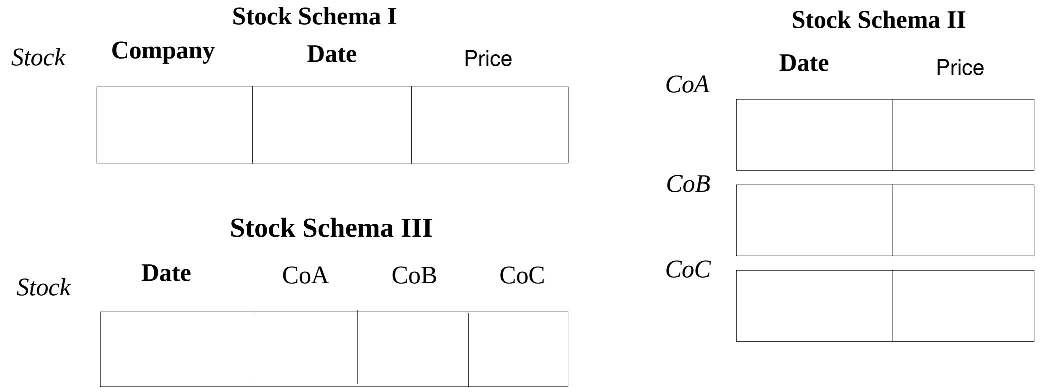
(4) The n-day moving average of a company on a given date is defined to be the average of the company’s closing-price over n days starting from the given date. (You can assume that the closing-price is defined for each company on every day in the period under consideration; i.e., there are no ‘‘holes’’ when the closing-price is not defined.) Find the 3-day moving average of DEC on 9/20/93.

(5) Find the smallest number N such that the N day moving average of GE on March 10, 1990 is greater than 100.

6. Schema Equivalence

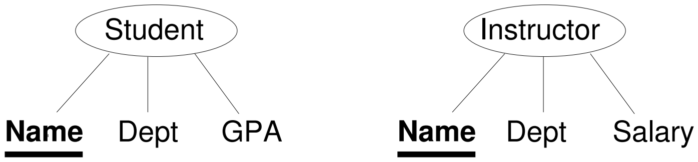
Two schemas expressed in a data model are considered equivalent if every instance of one (i.e., every possible database that can be expressed in it) can be translated into a unique instance of the other and vice versa. This type of equivalence is called *information capacity preserving*, because when a database instance is translated into another instance, no information is lost.

(1) Consider the following three relational schemas that capture information about closing prices of stocks on specific days:

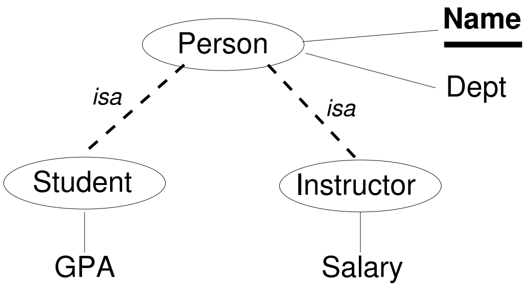


For each relation, the set of attributes indicated in bold forms the primary key in the relation, and key (functional) dependencies are the only kind of dependencies that hold in any schema. In addition, the domain of possible values for the ‘‘Company’’ attribute in Schema I is the set {CoA,CoB,CoC}. For each pair of the above three schemas, specify if they are equivalent or not based on the definition given earlier. In either case, justify your answer, e.g., if the schemas are not equivalent, be specific about why certain instances cannot be translated to a unique instance of the other schema, and specify what additional assumptions on the valid instances of the schemas would make them equivalent. If the schemas are equivalent, describe the translation of instances.

(2) Consider the following two object-oriented schemas that capture information about instructors and students:



Schema A



Schema B

In the first one, separate, unrelated classes are used for the two notions. In the second one, a class person is defined, which serves as a superclass to the two original classes. Again, specify whether or not the above two schemas are equivalent based on the definition given earlier. In either case, justify your answer as above.

7. Information Retrieval

Consider the idea of supporting a document retrieval system based on relational technology. Imagine that unique identifiers for various documents together with their titles are stored in a single relation called

DOC(*document-id*,title),

where document-id is the key. The system provides a stylized language to support the retrieval of documents based on keywords that may appear in the document's title. For example, in order to find the documents with the words ''War'' and ''Peace'' in their titles, one writes a request like

retrieve document-id
where keyword = "War" **and** keyword = "Peace"

This is translated internally into SQL and processed appropriately.

(1) Comment on the validity of the above design, identifying good and poor aspects of it and their implications.

(2) What other relations would you add to the above schema to improve it?

(3) Unlike in conventional systems, consider Grid files that use the same relation attribute on multiple dimensions. How could you use such an index structure on your schema of part (2) to improve on the performance of requests like the one given earlier?

8. Open Problems

Many of all the papers and book chapters that you've read while preparing for this exam leave open questions, address issues based on restrictive assumptions, or provide solutions that are inadequate or incomplete. Among all these, choose one such paper (or related set of papers) whose open or inadequately addressed problems you feel are most important, i.e., these are the problems whose solution you think would have the largest impact on database technology. Discuss these problem(s), explain why you think they are so important, elaborate on what the specific paper(s) leave unanswered, and optionally suggest approaches one might follow to solve them.