SPRING 1993
COMPUTER SCIENCES DEPARTMENT
UNIVERSITY OF WISCONSIN—MADISON
PH. D. QUALIFYING EXAMINATION

Database
Depth Examination

Monday, February 8, 1993
3:00 – 7:00 PM
113 Psychology

GENERAL 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. *Do not write your name on any answer book.*
3. Return all answer books in the folder provided. Additional answer books are available if needed.

SPECIFIC INSTRUCTIONS:

Answer any *four* of the *five* questions 1-5 *and* any *three* of the *four* questions 6-9. 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!

POLICY ON MISPRINTS AND AMBIGUITIES:

The Exam Committee tires 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.

UNIVERSITY OF WISCONSIN-MADISON
COMPUTER SCIENCES DEPARTMENT

DATABASE QUALIFYING EXAMS
SPRING 1993

BREADTH EXAM

Answer any *four* (4) of the following *five* (5) questions. You may *not* select questions from the Depth Exam. 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. Query Languages

Consider the following schema used by "The Generic Company" to keep track of various information about projects, departments, and employees:

PROJECT (*prno*, prname, fundingagency, cost, dno)
DEPT (*dno*, dname, mgr-id, budget)
EMPLOYEE (*eid*, ename, phone)
WORKSON(*eid, prno*, hoursperweek)

In the above, primary keys are in italics. Write SQL statements to do the following tasks. (In addition to the standard parts of the language, assume an SQL version that supports the keywords **union**, **minus**, **intersect**, **contains**, **not contains**, **exists**, and **not exists**.)

(a) Find the names of all projects managed by Gunga Din or funded by DGFA at a cost of over five million dollars. (DGFA is De Generic Funding Agency.)

(b) Find the names of all projects in which at least 10 employees work for more than 20 hrs a week.

(c) Find the names of all managers whose departments only have DGFA funded projects. (i.e., if the department has a project, the project is funded by DGFA.)

(d) Find the names of all employees who work on *every* DGFA funded project.

(e) Fire (i.e., remove) all employees who no longer work on any project.

2. Query Processing

The Sybase database system uses only the nested-loops join algorithm. In the event that an index does not exist on the join attribute of the inner relation, one is constructed "on the fly". They claim that this strategy offers equivalent performance to the sort-merge join algorithm.

(a) Are they correct? Explain by describing the I/O behavior of the nested-loops and sort-merge join algorithms. Assume that the index has to be created dynamically. Assume, further, that neither relation fits in memory.

(b) Under what conditions will the two algorithms have essentially the same I/O behavior — and hence equivalent performance?

3. Integrity Constraints

Consider the important concept of referential integrity in relational databases.

(a) Define the terms ‘key’ and ‘foreign key’. Give two examples, one that involves two relations and one that involves only a single relation.

(b) What events require referential integrity maintenance actions to be taken? For each one, what set of useful alternative actions should be provided for database designers to choose among?

(c) Should referential integrity constraints be enforced on a per-event basis, or should enforcement be deferred until end-of-transaction? Briefly discuss the issues and make a recommendation.

4. Database Design

Dean Martin of The Polytechnic Institute for Thespians (affectionately known as The PITS) has decided to put some of his data in a relational database. He has to keep track of the hours (H) and rooms (R) where courses (C) are taught, the students (S) who are enrolled in each course, and student grades (G) in each enrolled course. The following constraints are given:

- only one course can meet in a room at one time,
- each student has just one grade in each course,
- a student can only be in one room at one time.

Further, the dean is frequently interested in queries of the following form:

- What was John Smith’s grade in Comparative Histrionics?
- What are the courses that Jane Smythe is enrolled in?

(a) List the functional dependencies.

(b) Suggest a ‘good’ BCNF schema.

(c) If you use the BCNF schema suggested in part (b), can you efficiently check that the FDs given to you are maintained when the database is updated? Explain briefly.

(d) What are the conditions under which a 3NF schema is preferable to a BCNF schema?

(e) In this example, is there a 3NF (but not BCNF) schema that is better than the BCNF schema that you suggested in part (b)?

5. Update Problems

Consider the following database schema, where primary keys in italics:

EMP (*name*, sal, dname)

DEPT(*dname*, floor, mgr)

In what follows we use a hypothetical, SQL-like, syntax for updates. Consider the following update:

```
update EMP
set sal = (sal * 1.2)                                   (U1)
where sal < 100K and sal > 50K
```

(a) Assume that a secondary index exists on the sal field. What problem may occur if this update is not handled properly? (Keep in mind that updates also change the index.)

(b) Describe an algorithm that can be used to avoid the problem. Comment on the positive and negative aspects of your algorithm.

Consider now the following two updates:

```
update EMP
set sal =
        select f.sal                                    (U2)
        from EMP f, DEPT d
        where dname = d.dname and d.mgr = f.name

update EMP
set sal =
        select f.sal                                    (U3)
        from EMP f, DEPT d
        where name = d.mgr and d.dname = f.dname
```

(c) One of the above two updates presents a problem when executed. Identify the problem and the update. Explain the reason why the other update does not share the problem.

(d) Based on your explanation above, suggest how the problem may be detected without executing the problematic query.

DEPTH EXAM

Begin by taking the breadth exam, following the instructions given there. Then answer *three* (3) of the following *four* (4) questions.

6. Query Optimization

Access plans for join queries are very often represented as binary trees whose leaves are base relations and whose internal nodes represent joins of the relations corresponding to their children. The left (resp. right) child of a join node represents the outer (resp. inner) relation of the join. A *left-deep* tree is one where the right children of all join nodes are leaves (base relations). A *bushy* tree is one where there is at least one join node with both of its children being intermediate nodes (temporary relations).

Consider the query optimization algorithm proposed by Selinger et al., which identifies the optimal access plan for a query among those represented by left-deep trees.

(a) Explain the basic principles of how the algorithm works.

(b) What were the motivations for restricting the algorithm to operate on left-deep trees alone?

(c) How would you modify the algorithm to handle both deep and bushy trees?

(d) Consider join queries with three relations and give one example of such a query where the optimal plan is left-deep and another example where the optimal plan is bushy.

(e) Repeat (d) but for join queries with four relations.

7. Query Processing

Consider disjunctive SQL queries, i.e., those of the form

select attribute-list
from relation-list
where predicate1 **or** predicate2 **or** ...

Many current relational database systems handle such queries poorly.

(a) Why are disjunctive queries more difficult to handle efficiently than conjunctive queries? Explain briefly.

(b) What primitive algebraic operator(s) should be available for use in query plans for processing such queries? Explain, showing how your answer would help in efficiently processing an example query.

(c) Consider the special case of a single-relation disjunctive query where each predicate involves an attribute that is supported by a secondary index. Propose an efficient way to process this kind of disjunctive query. Under what conditions will your approach beat a full relation scan, and vice versa?

(d) Repeat (c) for the special case where all of the predicates are on the SAME secondary-indexed attribute.

8. Commit Protocols

Consider Skeen's three-phase commit protocol, which is non-blocking in the face of site crashes as long as at least one site manages to survive and carry out the transaction.

(a) Why can't Skeen's protocol be used as a non-blocking protocol in the face of network partitions? (A network partition is when several groups of sites can communicate only amongst themselves; i.e., a node in one group cannot communicate with a node in another group.) Explain by giving an example of what could go wrong if the associated termination protocol was simply run independently in each partition after such a network failure.

(b) The notion of a quorum (as used in the quorum-based concurrency control methods covered in the Bernstein and Goodman article) suggests an approach by which Skeen's protocol can be made to work (albeit with blocking) when network partitions occur. Propose a design for a quorum-based three-phase protocol. Explain how your proposal is able to correctly handle network partitions as well as site crashes.

(c) Is your proposed protocol still a non-blocking protocol with respect to combinations of site crashes, according to Skeen's definition? Why or why not? Should you care?

9. Logic and Databases

Consider the following recursive program for the transitive closure:

```
t(X,Y) :- e(X,Y).
t(X,Y) :- e(X,W), t(W,Y).
```

Assume that e is a base relation, while t is defined only by these two rules.

(a) Give a four-tuple instance of e such that seminaive evaluation of t and naive evaluation of t make the same number of tuple inferences.

(b) Give a four-tuple instance of e such that the seminaive evaluation of t makes far fewer tuple inferences than does naive.

Now consider the query t(1,Y)? over this same transitive closure program.

(c) Show the magic sets rewriting of the program for this query.

(d) Give a four-tuple instance of e such that the seminaive evaluation of the rewritten program materializes the same set of tuples for t as does seminaive applied to the original program.

(e) Give a four-tuple instance of e such that the seminaive evaluation of the rewritten program materializes fewer tuples of t than does seminaive applied to the original program.