UNIVERSITY OF WISCONSIN - MADISON
COMPUTER SCIENCES DEPARTMENT

DATABASE QUALIFYING EXAM
FALL 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. Views

One of the important features of modern relational DBMSs is the fact that they provide the ability to define user/group-appropriate views of the data that they contain. In particular, both the view definition syntax and the role(s) of views in a database are defined in the SQL standard.

(a) List several reasons that views are considered to be so important.

(b) Consider the following (terribly surprising, no doubt) SQL database definitions:

```
CREATE TABLE Emp(emp_id int, ename char(40), address char(100),
   salary float, birth_date date, dept_id int,
   CONSTRAINT ec1 PRIMARY KEY (emp_id),
   CONSTRAINT ec2 FOREIGN KEY (dept_id) REFERENCES (Dept))
CREATE TABLE Dept(dept_id int, dname char(60), mgr_id int,
    budget float, strategic_plan varchar(10000),
    CONSTRAINT dc1 PRIMARY KEY (dept_id),
    CONSTRAINT dc2 FOREIGN KEY (mgr_id) REFERENCES (Emp))
```

(Note that ec1, ec2, etc are just the names of constraints; these names are required by SQL92.) Illustrate each of the reasons that you gave in (a) by defining, in SQL, one or more example views over this database that clearly illustrate your points. Be sure to explain how the examples illustrate your points. (Don't worry about getting the view definition syntax exactly right, by the way!)

(c) Since we're entering the mid-1990's, all relational DBMS vendors are hot at work converting their relational DBMSs into "object-relational" DBMSs. In light of this fact, you are to do two things here. (i) Explain what the Emp/Dept database schema is likely to look like when expressed in the DDL of such a system (feel free to make the syntax up). (ii) Consider how the SQL view model can be extended to work in this brave new world: Does everything extend gracefully, or are there some potential difficulties, and in the latter case, what difficulties do you envision? (HINT: Think about the problem of providing an Emp view and a Dept view to some group of users who need to use a subset of the data in both relations; assume that the group should not be allowed to access either of the base relations.)

2. Recovery

In the basic approach to concurrency control using 2PL and recovery using WAL, transactions obtain R/W locks before reading/writing objects, and a transaction's log records are forced (in LSN order) to disk before the corresponding updates to data pages. In particular, a transaction is considered to ''commit'' at the instant that its COMMIT log record is forced to disk.

(a) When a transaction is ready to commit, what can you say about the number of log records that must be forced to disk in order to commit it?

(b) Immediately after a transaction commits, what can you say about the data pages written by the transaction?

(c) When a transaction is ready to commit, in the basic approach its locks cannot be released until its COMMIT log record is forced to disk. Faced with the task of implementing a high-performance DBMS, you consider appending a COMMIT record to the log (possibly in main memory, with no attempt to force to disk) and releasing the locks. *With this one exception, 2PL and WAL are followed — log records are still written in LSN order, and precede the writing to disk of corresponding data pages.* Consider a specific transaction T, and discuss this strategy in the case that:

(i) There is no subsequent crash before the COMMIT log record for T eventually is forced to disk.

(ii) There is a crash before the COMMIT log record for T is forced to disk.

3. Normal Forms

(a) Define lossless-join and dependency-preservation for decompositions. Explain why these properties are important.

(b) It is often said that BCNF ‘‘eliminates all redundancy’’, in the following sense: Consider a tuple t in an instance of a relation R that satisfies all the dependencies associated with R. If we ‘‘hide’’ a value in some field of R, it is not possible to determine what this value is based on the values of other fields of tuple t and/or other tuples in R. Thus, no field value is made ‘‘redundant’’ by the rest of the relation instance and the known dependencies.

(i) Is this claim correct? If so, sketch a proof. If not, state any further assumptions that are necessary to make it correct, and sketch a proof of correctness with these additional assumptions.

(ii) Discuss your answer to (i) in the context of the following example: R = ABC, S = ABD, the only FDs are those implied by the fact that A is a key for both R and S. Is this design in BCNF? Can you think of any redundancies? Would your answer change if you know that R and S were obtained by decomposing the relation schema ABCD?

(c) Can you give an example of a redundancy that is eliminated by BCNF but not 3NF? Why do people even consider 3NF? In particular, why would one settle for a 3NF design when a BCNF design ‘‘eliminates all redundancy’’?

4. Concurrency Control in Spatial Access Methods

With the increasing interest in geographic information systems, R-Trees and their variants have become very popular spatial indexing methods.

(a) Describe the R-tree indexing structure and how lookup operations work.

Consider the operation of an R-tree in a multiuser environment and assume that you wish to support page-level locking using a non-2PL for more concurrency.

b) Design a non-2PL algorithm for R-trees. Explain how your algorithm increases concurrency for both readers, writers, and a mix of readers and writers.

c) How is your algorithm different than the corresponding B-tree algorithm.

5. System R Query Optimization

a) For a query on one table, how many access plans does the System R optimizer consider? (Hint: Your answer should involve the number of indices on the table.)

b) Disregarding "interesting orders", what is the maximum number of partial solutions stored when optimizing a query of n relations? Explain how you derive your answer, which should be a function of n.

c) Explain the difference, both in terms of language and processing, between a standard subquery and a "correlation" subquery.

d) State 3 assumptions made in the System R optimizer that may not be true.

6. SQL

Consider the following schema, designed to aid companies in junk mail marketing:

CUSTOMER(CustID, CustName, Salary);
PURCHASES(CustID, ItemID, Date, Price);
ITEMS(ItemID, ItemName, Quantity);

Translate the following queries into SQL.

(a) Give all customers who have made more than $100 worth of purchases a 10% raise.

(b) Print the name of all items that are only bought by people with salaries over $100,000.

(c) (Assume that different people may pay different amounts for the same item.) Print the ItemName and average price for all items such that the average price paid for the item was over $20.

(d) For each item, find the name of the customer who paid the most for that item.