University of Wisconsin-Madison
Computer Sciences Department

Database Qualifying Exam
Fall 1996

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 that you have made in formulating your answer. Good luck!

1) Parallel Database Systems.

Gamma and most derivative parallel database systems employ a push model for parallelism.

a) What is meant by a "push" model?

b) Describe the key data structures and algorithms that Gamma used to implement this push model.

c) In several papers Graefe proposes the use of an iterator-based pull model for query processing in relational database systems in which the consumer pulls tuples as needed from the producer. Explain why an iterator-based pull model does not work as effectively as a push model in a parallel environment without shared memory.

d) Would your answer change in a parallel system that had shared-memory?

e) In parallelizing an object-relational DBMS, how does the presence of complex and sometimes large attributes such as video, images, text, and arrays affect the use of a push model of parallelism?

2) Query optimization in object-relational database systems.

Object-relational database systems such as Illustra and UniSQL as well as object-oriented database systems such as O2 and Orion support the notion of set-valued attributes and nested queries. For example, the definition of a Person type might contain the attribute "children: set of person".

a) Describe two alternative ways of implementing set-valued attributes, one of which requires variable length records even if each element of a set is of fixed-length.

b) Discuss the impact of each strategy on the task of optimizing queries involving set-valued attributes.

c) What kinds of statistics should an optimizer maintain for set-valued attributes? What problems do you foresee in maintaining such statistics? What solutions would you propose?

d) What impact do set-valued attributes have on the effectiveness of parallel query processing in a system like Gamma?

3) **Buffer Management**

a) Everyone says MRU is great for sequential scans. Why is this true?

b) Discuss the benefits and drawbacks of a scheme that always replaces clean pages (if any are available) before dirty ones.

c) A "hate" hint allows a buffer manager request to specify whether the page is "hated." During replacement selection, all "hated" pages are considered before any page that is not hated. Are hate hints plus LRU plus MRU enough to implement DBMin? Why or why not?

d) In doing a hash join, many DBMS actually build the hash tables in virtual memory outside of the buffer pool. Give some reasons why they might do so.

4) Histograms

a) Let R be a relation and A one of its attributes. Consider the following distribution of values in A:

T	Value	10	20	30	50	60	80	90
	Frequency	10	10	20	100	10	40	10

Also consider the following five potential approximations of it:

T1	Value	10	20	30	50	60	80	90
	Frequency	35	35	35	35	20	20	20

T2	Value	10	20	30	50	60	80	90
	Frequency	12	12	12	70	12	70	12

T3	Value	10	20	30	50	60	80	90
	Frequency	18	10	18	100	18	18	18

T4	Value	10	20	30	50	60	80	90
	Frequency	100	40	20	10	10	10	10

T5	Value	10	20	30	50	60	80	90
	Frequency	40	25	15	40	40	25	15

For **each** one of the above, indicate whether or not it is:

1) a histogram (corresponding to the original distribution)
2) a serial histogram
3) an end-biased histogram

Make sure you indicate all choices that are applicable.

b) What is the size of the result of joining R with itself on A? Which of the above five potential approximations estimates that result size better?

c) What are the relative advantages and disadvantages of serial and end-biased histograms?

d) How would a general serial histogram on a single-attribute distribution be stored in the database? How about an end-biased histogram? In your answers, make sure you indicate all the details regarding the appropriate data structures, precise information maintained, etc.

e) How would your answer above change for a multi-attribute histogram?

5) Magic Sets.

Consider the following two ways of computing the names of employees who make more than 100K, and whose age is equal to their manager's age. First, a nested query:

```
Select E1.ename
From   Emp E1
Where  E1.sal > 100 And E1.age = (Select E2.age
                                  From   Emp E2, Dept D2
                                  Where  E1.dname = D2.dname
                                         And D2.mgr = E2.ename)
```

Second, a query that uses a view definition:

```
Select  E1.ename
From    Emp E1, MgrAge A
Where   E1.dname = A.dname And E1.sal > 100 And E1.age = A.age

Create  View MgrAge (dname, age)
As  Select  D.dname, E.age
    From    Emp E, Dept D
    Where   D.mgr = E.ename
```

a) Describe a situation in which the first query is likely to outperform the second query.

b) Describe a situation in which the second query is likely to outperform the first query.

c) Can you construct an equivalent query that is likely to beat both these queries when every employee who makes more than 100K is either 35 or 40 years old? Explain briefly.

6) Normalization

The Globetrotters Club is organized into chapters. The president of a chapter can never serve as the president of any other chapter, and each chapter gives its president some salary. Chapters keep moving to new locations, and a new president is elected when (and only when) a chapter moves. The above data is stored in a relation G(C,S,L,P), where the attributes are chapters (C), salaries (S), locations (L) and presidents (P). Queries of the following form are frequently asked and you must be able to answer them without computing a join: ‘‘Who was the president of chapter X when it was in location Y?’’

a) List the FDs that are given to hold over G.

b) What are the candidate keys for relation G?

c) What normal form is the schema G in?

d) Design a ‘‘good’’ database schema for the club. (Remember that your design must satisfy the query requirement stated above!)

e) What normal form is your ‘‘good’’ schema in? Give an example of a query that is likely to run slower on this schema than on the relation G.

f) Is there a lossless-join, dependency-preserving decomposition of G into BCNF?