University of Wisconsin-Madison
Computer Sciences Department

Database Qualifying Exam
Spring 2002

Answer all five (5) questions (NOTE: this is different from some previous years, when you were only asked to answer 4 of 5.) Before beginning to answer a question make sure that you 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. Concurrency Control

(a) In optimistic concurrency control as proposed by Kung and Robinson, one of the conditions that guarantees serializability can be stated as: if TN(Ti) < TN(Tj), then Write-Phase(Ti) ends before Write-Phase(Tj) begins and Write-Set(Ti) and Read-Set(Tj) do not overlap. Suppose that during the execution of a set of transactions T1, T2, …, Tn, each pair of transactions Ti and Tj in this set satisfy the condition. Explain why this schedule is serializable (it is not sufficient to just cite Kung and Robinson.)

(b) What is the relationship between the set of schedules allowed by Kung and Robinson and the set of schedules allowed at degree 3 consistency as defined by Gray?

(c) Is non-strict 2PL identical to any of the levels of consistency defined by Gray et al.? Explain your answer.

2. Expressive Power of SQL

Consider a subset of the SQL:1999 query language that includes recursive queries, but does not include grouping, aggregate operations, or cursors (i.e., no order-by). Nested queries and set-operations (e.g., union and difference) are allowed, and you can use views to store intermediate results. You can use the DISTINCT clause if you wish. You cannot write programs in a host language and embed SQL calls; you must stay strictly within the SQL query language with the above restrictions. Consider the relation Employees(eid, ename, mid, dept, sal) and answer the following questions; eid is the key and mid identifies the manager of the employee.

(a) Can you count the number of employee tuples?

(b) Can you find out if any departments employ more than one person?

(c) Can you identify departments that employ more than one person?

(d) Can you identify which department pays the highest salary?

(e) Can you identify employees who make less than someone they (directly or indirectly) manage?

3. Replication

All major data centers maintain two operational systems each with a complete copy of their database in the event that one of them falls into the Pacific Ocean or gets hit by a

tornado. Describe two alternatives for keeping the two copies of the database "synchronized" as the primary copy is updated by the normal workload. Focus on schemes that minimize the impact on the performance of the primary copy in exchange for having the backup copy slightly out-of-date, and discuss tradeoffs between your schemes.

As a hint, consider how database systems achieve robustness in the event of software and hardware failures and see how the different mechanisms can be extended to handle replicas that are geographically distributed.

Your schemes should avoid the use of distributed two-phase commit protocols. Why?

4. Bit-Mapped Indices

a) Describe how bit mapped indices work, contrasting with normal B-tree indices. Illustrate with an example that shows the same data stored in both types of indices.

b) Explain the advantage and disadvantages of bit-mapped indices. Under what circumstances will a bit-mapped index provide superior performance?

5. Parallel Database Systems

Over the 20 year period from 1980 to 2000, the capacity of commodity disk drives increased from 80MB to 80GB, while their bandwidth has grown from 30MB/s to 80MB/s, and their seek + rotational latency has gone from 10 msec to 9 msec. In 3 years, 1 TB disk drives are expected to be common. While databases have also grown in size over the past 20 years, it is commonly accepted that disks are getting bigger at a faster rate.

a) Eventually many databases will fit on a handful of disks. How would such a change affect the design of parallel database systems?

b) While it sounds great that most users will only need a few drives to hold all but their largest databases, what are the drawbacks?

c) In addition to becoming huge, disks have become incredibly cheap making it possible to replicate multiple copies of each table (or portions of each table). How might you exploit this situation? Be really creative here. For example, what would be wrong with replicating the entire database on each node in a shared-nothing parallel database system?