DB Qual 2008

This exam consists of three parts. Please answer each part in a separate blue booklet. You have a total of 60 minutes for the entire exam. You can allocate time among the parts as you like, but be sure not to get stuck in one part without spending some time in the other parts.

The exam is open book and notes. No laptops are allowed. Simple calculators are allowed.

If you feel that a question is ambiguous or unclear, state any reasonable assumptions you need to make in order to answer the question.

DB Qual 2008; Part A

Problem A1 (10 points)

Let $R(A, B, C, D)$ be a relation.

(a) If R satisfies $AB \rightarrow C$ and $B \twoheadrightarrow D$, it does not necessarily follow that R satisfies $B \twoheadrightarrow C$. Give an example (an instance of relation R) that demonstrates this statement. That is, your instance must satisfy the FD $AB \rightarrow C$ and the MVD $B \twoheadrightarrow D$, but not satisfy the MVD $B \twoheadrightarrow C$.

(b) If R satisfies $A \rightarrow B$ and $C \twoheadrightarrow A$, then R must satisfy $C \twoheadrightarrow D$. Prove this statement.

Problem A2 (10 points)

Let $S(A, B)$ be a relation. Say A-values a_1 and a_2 are *neighbors* if

1. a_1 is not equal to a_2.
2. There is some B-value b such that both (a_1, b) and (a_2, b) are tuples in S.

Write a Datalog program to define the predicate OneNeighbor(x) that is true if and only if x has exactly one neighbor in S.

DB Qual 2008; Part B

Problem B1 (3,4,5 points for (a),(b),(c) respectively)

Consider a relation $R(A)$.

(a) Using conventional SQL, write a query to compute the largest value in R, without using the "Max" operator. You may assume there are no duplicates in R. Your answer will be graded on simplicity as well as correctness.

(b) Using conventional SQL, write a query to compute the median value in R. You may assume there are no duplicates in R, and R contains an odd number r of tuples. Note there is no "Median" operator in conventional SQL. Your answer will be graded on simplicity as well as correctness.

(c) Now assume R may have duplicate values. Write a query to compute the "mode" of R, i.e., the value (returned just once) that appears most often. If multiple values satisfy the criteria (for example, if two values appear four times each, and no values appear more than four times), then all of them should be returned. Note there is no "Mode" operator in conventional SQL. Your answer will be graded on simplicity as well as correctness.

Problem B2 (4 points)

Consider a conventional query optimizer that estimates cardinality and cost to decide among different access methods, join methods, and query plans. In this setting, nested-loops joins are considered more "dangerous" than simple hash joins in the presence of poor cardinality and cost estimates. Why?

Problem B3 (4 points)

Consider the execution of a simple nested-block join over two very large relations. Suggest a buffer page replacement strategy that is likely to provide good I/O performance for this operation. Briefly explain why.

Problem C1 (10 points)

Consider an extensible hash table. The hash function used generates $b = 4$ bits. Initially the hash table is empty, and we insert 360 records into it. The following table shows how many records hash into each hash key. For example, there are 30 records that hash to 0011.

Hash	Num. Records
0000	10
0001	20
0010	30
0011	30
0100	20
0101	90
0110	40
0111	40
1000	10
1001	10
1010	10
1011	10
1100	10
1101	10
1110	10
1111	10

Each hash table bucket can hold 100 records. Within each disk bucket, the records are sorted by their search key. There are no duplicate search keys in the input records.

(a) Does the order in which we insert the 360 records impact the *final* state of the extensible hash table? Briefly explain your answer.

(b) After all records are inserted, how large (number of entries) is the directory? Assume that the records are inserted by hash key order, e.g., first all 10 records with hash 0000 are inserted, then the 20 records with hash 0001 are inserted, and so on.

(c) For the directory of part (b), how many directory entries point to the bucket that contains the 0000 records?

(d) For the directory of part (b), how many directory entries point to the bucket that contains the 1000 records?

Problem C2 (10 points)

Consider a multi-granularity locking system. The database consists of 4 records (it's a small database!) organized in the following hierarchy:

- The root is an object X with two children
  - Object Y_1 has two children
    - Object $Y_{1,1}$ and
    - Object $Y_{1,2}$.
  - Object Y_2 has two children
    - Object $Y_{2,1}$ and
    - Object $Y_{2,2}$.

The system produces legal schedules, and all transactions are well-formed and two-phase. Assume that when a transaction T needs to write all children of node Z, it gets a write lock on Z, as opposed to getting an IX lock on Z and then getting a write lock on all the children. If T is not going to write *all* the children, then it gets an IX lock on Z.

(a) Consider the following schedule of two transactions T_1 and T_2:

$$w_1(Y_{1,1}) \;\; w_1(Y_{1,2}) \;\; w_2(Y_{1,2}) \;\; w_2(Y_{2,2}) \;\; w_1(Y_{2,1})$$

Show all the lock and unlock requests that are issued by the transactions. Use the notation $L(o, m, t)$ for a lock on object o in mode m (X, IX, ...) at "time" t. If $t = 1$, the lock occurs before the first action, i.e., before $w_1(Y_{1,1})$. If $t = 2$, the lock occurs between the first and the second action, i.e., between $w_1(Y_{1,1})$ and $w_1(Y_{1,2})$, and so on. Use $U(o, m, t)$ for unlocks. If a lock (or unlock) can occur at several times, then pick one of the times arbitrarily.

(b) Consider the modified schedule (only the fourth action has changed):

$$w_1(Y_{1,1}) \;\; w_1(Y_{1,2}) \;\; w_2(Y_{1,2}) \;\; w_2(Y_{2,1}) \;\; w_1(Y_{2,1})$$

Show that this schedule *cannot* be produced by our system. You can use the results and the notation of part (a).