DB Qual 2006

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 2006; Part A

Problem A1 (5 points)

The relation $R(A, B, C, D, E)$ has functional dependencies $A \rightarrow B$, $B \rightarrow C$, $C \rightarrow AD$, $D \rightarrow E$, and $E \rightarrow A$. If we project R onto the set of attributes ACE, what dependencies hold in ACE? Briefly explain your reasoning.

Problem A2 (5 points)

Suppose we have a relation $Male(name)$ that holds the set of names of males. There is also a relation $Marriages(name1, name2)$ that relates pairs of married people. It is not determined which of a married couple is listed first, and you should not assume that $Marriages(a, b)$ implies $Marriages(b, a)$. A *Bachelor* is a male who is not married. Write a Datalog program that defines the set of Bachelors. Briefly explain your reasoning.

Problem A3 (10 points)

In this problem, assume we have a data cube R with n dimension attributes and one dependent attribute. There are n different values in each dimension. Thus, the fact table has n^n tuples. You may assume n is large and give simple forms of answer that ignore lower-order terms.

a) (3 pts.) Recall that a *roll-up* is a projection of the data cube onto some proper subset of the dimensions. That is, a roll-up is a query of the form $\gamma_{a_1,a_2,\ldots,a_k,SUM(d)\rightarrow d}(R)$, where γ is the grouping operator of the extended relational algebra, the a_i's are a list of between 0 and $n-1$ of R's dimension attributes, and d is the dependent attribute of R. As a function of n, how many different roll-ups does R have? Briefly explain your reasoning.

b) (5 pts.) Recall that a *slice* is a selection for particular values in one or more attributes. That is, a slice is a query of the form $\sigma_{a_1=c_1\ AND\ a_2=c_2\ AND\ \ldots}(R)$, where the a_i's are one or more of the dimension attributes of R, and c_i is one of the constants appropriate for attribute a_i. As a function of n, how many different slices does R have? Briefly explain your reasoning.

c) (2 pts.) As n gets large, what is the limit of the ratio of the number of slices of R to the number of tuples in R? Explain briefly.

DB Qual 2006; Part B

Problem B1 (8 points)

During a foosball tournament in the Gates 4A lounge, pairs of students played matches against each other and the scores were recorded in a relational database table:

```
Match(A1, A2, PtsA, B1, B2, PtsB)
```

A tuple $(a_1, a_2, p_a, b_1, b_2, p_b)$ in `Match` records the fact that students (a_1, a_2) played against students (b_1, b_2), and the score was p_a to p_b. Assume no structure in the tournament: any two players could play against any other two players any number of times. Also assume there were no ties: In each match either `PtsA` > `PtsB` or `PtsB` > `PtsA`.

Write a single SQL query that returns the two players who, playing together, won more games than any other pair. If more than one pair has the highest number of wins, your query should return all such pairs.

Your answer will be graded on clarity as well as on correctness.

Problem B2 (12 points, 3 per part)

The four heuristics (a)–(d) below all have been used as ways to reduce the search space of query plans during query optimization. For each of the four heuristics:

i. Briefly state why applying the heuristic is generally expected to yield "good" plans. (An explanation consisting of a few well-chosen words is sufficient.)

ii. Argue that the heuristic can sometimes fail: Describe or show a query plan (logical or physical) and database characteristics (e.g., statistics, auxiliary structures, or anything else typically used in plan costing) such that applying the heuristic is very likely to result in a worse plan than not applying it. Give the simplest a plan and characteristics you can come up with that contradicts the heuristic. (Again, a short but clear answer can earn full credit.)

The heuristics are:

1. Push selection predicates down the plan.
2. If a selection predicate is expensive to evaluate, pull it up the plan.
3. Perform joins before cross-products.
4. If both operands of a join are already sorted, use a sort-merge join.

Problem C1 (10 points)

Please answer the following questions. **The answer for each part below should be at most 10 words.** (If you think you need more words, select the most important part of your answer and write that, keeping to the strict 10 word limit. This limit does not apply to other problems.)

(a) When are locks released when strict two-phase locking is used?

(b) Consider an extensible hashing index with 16 directory entries ($i = 4$ hash bits are being used). What is the smallest number of data blocks (buckets) that can be in use at this time? (If fewer blocks were in use, the directory size could be reduced.)

(c) Give one scenario where spanned records are a must.

(d) Consider an undo-redo logging scheme, and a particular transaction T. After a crash, we examine the log. The existence (or not existence) of what log record(s) tell us that T must be aborted?

(e) Consider a B-Tree of order n. Under what circumstances will the tree contain a single node?

(f) Give two classes of schedules that are subsets of recoverable schedules.

(g) With write-ahead logging, what needs to be flushed to disk before object Y is modified?

(h) What are record tombstones used for?

(i) What is a dense index?

(j) What type of data structure is used to detect deadlocks?

See problem C2 on next page.

Problem C2 (10 points)

Consider a multiple-granularity locking scheme. A particular object Y is currently locked in shared mode (S lock) by transaction T_1. A second transaction requests an intention exclusive (IX) lock on Y. (T_2 plans to write object Y_3 which is a child of Y in the hierarchy.) The locking protocol tells us that these two locks are *not* compatible, so T_2's request is not granted.

(a) Suppose that the protocol was implemented by a Berkeley graduate student and was *not* implemented correctly, so that the IX lock is granted. Except for this one bug, the locking protocol works as it is supposed to. Show a non-serializable schedule that can result from this broken protocol.

You can add transactions (beyond T_1 and T_2) and actions on other objects, but keep the schedule as simple as possible. Write your schedule with one action per line, and include the lock actions. For example, you could have the actions

- T_1 locks Y_3 X mode
- T_1 reads object X_3

(b) Prove that the schedule you give in Part (a) is not conflict-serializable.