CS 631 / IT 603: ITDMBS: Midsemester Exam, 19 Sep 2005

Duration: 2 Hrs

1. Short answers

   (a) Suppose the disk transfer rate is 20 MB/sec, and a block is 4 KB, and seek time (incl. rotational latency) is 10 msec. What is the minimum number of contiguous blocks that you must read to ensure that not more than 50% of the time is wasted in seek time. ...3

   (b) Consider a selection with a disjunctive selection condition of the form ($P1$ **or** $P2$). Give a formula for estimating the selectivity of the selection condition, given the selectivities of $P1$ and $P2$ are $s1$ and $s2$ respectively; also mention under what assumption your formula holds. ...3

   (c) Write (in pseudocode) the functions open(), getnext() and close() to implement hash join. You may assume for simplicity that each tuple in the probe relation r matches at most one tuple in the build relation s. ...4

2. Query optimization

   (a) Explain how a materialization of $r ⟕ s$ can be incrementally maintained when tuples are inserted or deleted from r and s. ...5

   (b) Explain how to decorrelate the following query. ...3

   select *
   from r
   where $r.A = 5$ **and** $r.B$ **in** (**select** $s.B$ **from** s)

   (c) Suggest conditions under which decorrelation above may increase the cost of evaluation. ...2

3. Transactions:

   (a) What are the ACID properties? Describe them very briefly, in one sentence each. ...3

   (b) Give an example of a schedule that is not conflict serializable, but is still equivalent to a serial schedule in terms of the final state. ...3

   (c) Consider the following protocol: all data items are assigned a number; locks can be obtained only in increasing order of the number; locks may be released at any time, and need not be held in two-phase manner.

   Show a schedule generated by the above protocol, that is not serializable. ...4

4. Concurrency control:

   (a) Suppose that a transaction uses record-level locking, and does a relation scan, accessing millions of records. What problem would arise? ...2

   (b) Explain how multi-granularity locking can help avoid the above problem. ...2

   (c) Lock escalation refers to the process of replacing a number of fine-granularity locks by a coarse-granularity lock. Give pseudocode for a lock manager function *escalate*(T_i, N, mode), where T_i is a transaction and mode is one of S(hared) or X(clusive), which performs lock escalation; the function replaces locks on descendants of N by a lock on N. Only some of the descendants may be currently locked; assume also that N and its ancestors already have the required intention locks. Assume also that the lock manager knows the lock granularity tree. ...4

   (d) Does the release of fine-granularity locks in lock escalation above violate two-phase locking? Explain. ...2

Continued overleaf ...

5. B^+-trees: Suppose you insert a long sequence of entries, whose key values are sorted in increasing order, into a B^+-tree.

   (a) What would the expected occupancy of each leaf node? (Hint: recall the example you worked out in Quiz 2.) ...2

   (b) Describe, intuitively, a way of building a (new) B^+-tree bottom-up, for a given set of entries, first constructing the leaf level, then the next level, and so on, to ensure that occupancy is close to 100%. ...4

   (c) Explain how you can manage the above procedure, storing at most H nodes in memory at any time, where H is the height of the tree. ...2

   (d) Although filling nodes to 100% as above maximizes space utilization initially, it can lead to excessive space wastage if there are random inserts subsequently. Filling up nodes to a somewhat lower percentage utilization, say 80%, may be preferable. Explain why. ...2

Total Marks = 50