CS 441: Final Exam, Fall 1999

Write answers to Questions 1-10 on the question paper.

The remaining questions must be answered on the answer scripts provided.

1. For each of the following statements, (i) fill the blank with one of *never, sometimes,* or *always* and (ii) complete the statement by giving the reason(s) for your choice. ...16

(a) Query plan cost for computing $A \bowtie B \bowtie C$ is ———— less than the plan cost for $A \bowtie B \bowtie C \bowtie D$

(b) The size of the result of a join of n relations is ———— dependent on the order of the joins

(c) Given a θ–join, it is ———— better to do the natural join and then determine the tuples satisfying the θ

(d) Allocating more main memory buffers to store the results of an operation is ———— a good idea

(e) Sorting is ———— a blocking operation, i.e., the first output cannot be produced until all the inputs have been read

(f) Selection is ———— pipelinable

(g) It is ———— better to pipeline the execution of operations

(h) Given two plans for a query, the estimates for the result sizes are ———— going to be the same in the two plans

(i) When computing the union of 3 or more relations, it is ———— better to group the smallest relations first

(j) A sort–merge join is ———— a good choice if one or both arguments are already sorted

(k) Given updates u_1 and u_2 to two different pages by a transaction t, during the redo recovery phase, they can ———— be redone in either order

(l) The last record flushed to disk on behalf of a transaction is ———— the commit record

(m) Using two–version locking based concurrency control, a query can ———— finish without any rollbacks.

(n) In a database where no new tuples are added, repeatable reads ———— achieves serializability

(o) Deadlocks ———— occur with timestamp–based concurrency control

(p) The lectures for this course were ———— quite good

2. If we have a dense index on the primary key attribute of a relation, then it is possible to have pointers to tuples (or the records that represent those tuples) go to the index rather than to the record itself. What are the advantages of each approach? ...2

3. Give two reasons why *log shipping* might be used in multi–node systems. ...2

4. Give an example of a modification to a database (meta)data that need not be undone when the transaction that caused the modification aborts. Why is this (not) a violation of the atomicity property? ...2

5. The validation phase of optimistic concurrency control must be executed atomically. Why? ...2

6. A transaction creates temporary variables during its execution. Would you recommend logging the changes to these variables? Why (not)? ...2

7. Should the recovery manager be designed to deal with the following situation: A transaction t_1 that has committed and is redone read a value X written by some transaction t_2 that has not committed and is undone? ...2

8. A record has a fixed set of fixed–length fields. One such field often has a null value in many of the records, otherwise it is "long". Suggest a storage structure for this record. ...2

9. Give one advantage and one disadvantage of the following ...5

   (a) the elevator algorithm

   (b) having multiple disks

   (c) having mirrored disks

   (d) prefetching

   (e) No–steal/Force protocol

10. What ...5

   (a) is the cause for rotational latency in disks?

   (b) is an example of a database operation that needs two different hash functions for two different phases of the algorithm?

   (c) are the factors that contribute to the time taken by a transaction to modify - on disk - a field of type integer?

   (d) is the "point of no return" for a cohort in the two–phase commit protocol?

   (e) is the reason behind the `group by` operator being important for data warehousing?

11. Consider ...3

$\pi_L(R \bowtie S) = \pi_L(\pi_M(R) \bowtie \pi_N(S))$

where M is the list of all attributes of R that are either join attributes (in he schema of both R and S) or are input attributes of L, and N is the list of attributes of S that are either join attributes or input attributes of L.

(a) What is this equation saying? (b) Where is it useful?

12. A friend of yours is adamant that 2PL be used, but knows that it is prone to deadlocks. You throw up your hands and say, "O.K., but at least get all the locks together, that will avoid deadlocks", to which he says, "but, to do that I need to know beforehand all the locks I need, which I don't. But, I have a better way to avoid deadlocks." What might he have in mind? (Your friend does not like paying the overheads of transaction timestamping). ...3

13. The following is a sequence of log records written by two transactions t_1 and t_2: $< start\ t_1 >$; $< t_1, A, 10, 11 >$; $< start\ t_2 >$; $< t_2, B, 20, 21 >$; $< t_1, C, 30, 31 >$; $< t_2, D, 40, 41 >$; $< commit\ t_2 >$; $< t_1, E, 50, 51 >$.

Describe the action of the ARIES recovery manager, including changes to both disk and the log, if there is a crash and the last log record to appear on disk is:

(a) $< start\ t_2 >$ (b) $< commit\ t_2 >$ (c) $< t_1, E, 50, 51 >$ ...3

14. Consider the schedule ...5

$S: r_2(A); r_1(B); w_2(A); r_2(B); r_3(A); w_1(B); w_3(A); w_2(B);$

(a) Show that it is not conflict–serializable?

(b) With the minimum number of "moves" (where a move consists of changing the *position* of one of the operations) convert it into a conflict–serializable schedule.

(c) Is S serializable? Why (not)?

15. The *semijoin* of relations R and S is the bag of tuples t in R such that there is at least one tuple in S that agrees with t in all the attributes that R and S have in common. ...5

(a) Express the semantics of this operation using standard relational algebra operations. Argue the correctness of your expression.

(b) How would you estimate the size of the result of a semijoin?

(c) Where would this operation be useful?

16. Deadlocks due to lock escalation can be avoided by the use of *update* locks: an update lock holder (a) can read but not write, and (b) can upgrade to a write lock later. ...5

(i) Draw the compatibility table for this locking protocol.

(ii) What other condition(s) is/are needed to avoid deadlocks due to lock escalation?

17. Suppose a relation R has 20000 tuples that are packed 20 to a block. Let a be one of the attributes of R and there is an index on a. Consider the operation $\sigma_{a=0}(R)$. ...5

Give the number of disk I/Os required for the following scenarios:

(a) R is clustered but we do not use the index.

(b) R is not clustered and we do not use the index.

(c) V(R,a) = 100 and the index is clustering.

(d) V(R,a) – 100 and the index is nonclustering.

(e) Attribute a is a key.

18. Consider the "clock algorithm" used to approximate the LRU replacement algorithm: The buffers form a virtual circle with a "hand" pointing to one of the buffers. The hand rotates clockwise and examines the "flag" associated with each buffer. This flag is initially set to a value, usually 1. When a buffer is accessed, this flag is set again to this initial value. ...5

(a) Complete the rest of the details of the algorithm: In particular, how would the algorithm decide which buffer to replace?

(b) How can we implement pinning using this algorithm?

(d) How can we reduce the vulnerability of an "important" page?

19. Suppose certain data items in a database are replicated. ...6

(a) Argue why the *read-one, write-all* protocol works correctly?

(b) Consider the following "optimistic 2PL" version of this protocol: A transaction obtains a write lock on only one of the replicas R. Write locks on the rest of the replicas are taken when R is contacted by the coordinator of the transaction as part of the prepare-phase of 2PC. R responds with a "yes" only if it has done all the necessary work and all other replicas can obtain the necessary locks at that point.

(i) Argue why this a good protocol.
(ii) Argue why this is a bad protocol.

20. Suppose that blocks can hold either 10 records or 99 keys and 100 pointers. Also assume that the average B–tree node is 70% full, i.e., it will have 69 keys and 70 pointers. For each case below, determine (i) the total number of blocks needed for a 1 million record file, and (ii) the average number of disk I/O's to retrieve a record given its search key. Assume that there is nothing in memory initially and that the key is the primary key for the records. ...6

(a) The data file is a sequential file, sorted on the search key, with 10 records per block. The B–tree is a dense index.

(b) The same as (a), but the data file consists of records in no particular order, packed 10 to a block.

(c) The same as (a) but the B–tree index is a sparse index.

21. Consider $\delta(R \times S) = \delta(R) \times \delta(S)$ where δ is the duplicate elimination operator. ...7

(a) Argue the correctness of the expression.

(b) Is this correct if $\times$ is replaced by a $\bowtie$?

(c) Give an example to show that δ can be pushed below projection in a query plan.

(d) Give a one–pass algorithm for implementing δ.

(e) Describe a good way to parallelize δ.

22. Given the orders of magnitude difference in disk I/O costs relative to memory read costs, a DBMS attempts to maximize the usage of data and metadata that is computed or brought into memory/cache from disk. ...7

(i) What are the caching–related decisions that a DBMS must make, in general?

(ii) What are the differences in the factors that affect the caching–related decisions made by (a) an (update–intensive) OLTP system vs. (b) a (query–intensive) OLAP system.

Total Marks = 100