CS245A Final Examination

Thursday, March 19, 1998, 3:30PM—6:30PM

Directions

The exam is *open book/notes*; any written materials may be used.

For each of the 32 questions, circle the letter (a), (b), (c), (d) or (e) of your chosen answer. Score = 5 times number right minus number wrong, so random guessing nets you nothing on the average.

If you wish to explain or demonstrate your solution to a problem for partial credit, you may use page bottoms or the backs of the pages (but warn us on the front). Please use this option sparingly, e.g., if you think the question is flawed or open to multiple interpretations, because we shall only be awarding partial credit in rare situations.

Do not forget to **sign the pledge** below.

I acknowledge and accept the honor code.

____

Print your name here: ____

Question 1: A certain disk unit stores 10 gigabytes. It has 10 surfaces and 10,000 tracks per surface. The average circumference of a track is 5 inches, and 20% of each track is used for gaps and parity checks. The average density of a track, in *bits* per inch is closest to:

(a) 100,000 (b) 200,000 (c) 250,000 (d) 500,000 (e) 800,000

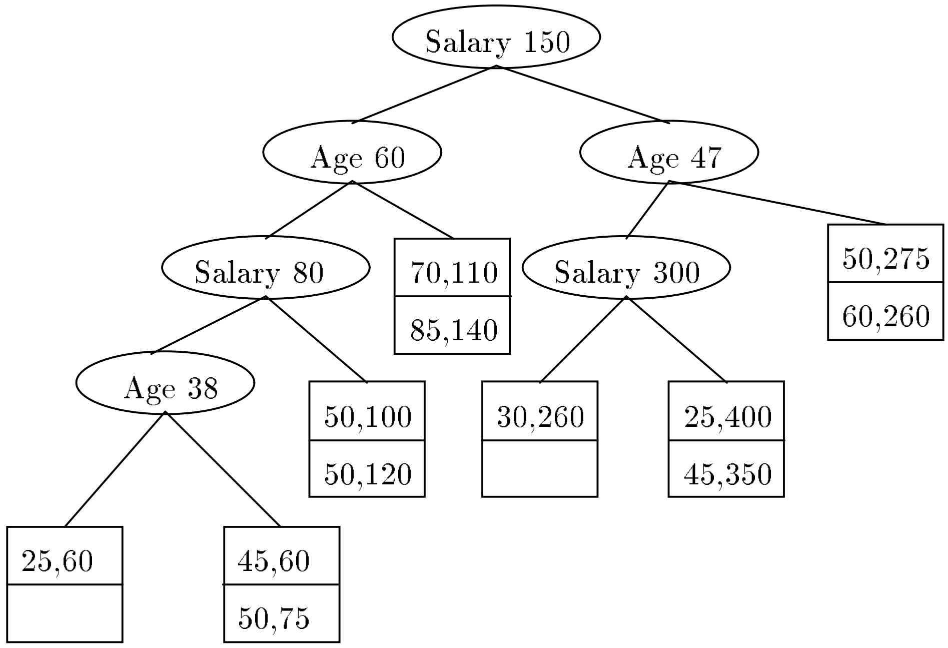
Question 2: Suppose 10 tuples of R can fit in one block, R has 100,000 tuples, and there are among these tuples 20,000 distinct tuples. We wish to compute $\delta(R)$ in one pass. That is, we read blocks of R into a single buffer, one at a time, use B buffers (block-sized chunks of main memory) to hold intermediate results, and write the answer somewhere (it is not important where the output goes). The smallest possible value of B is:

(a) 1000 (b) 2000 (c) 5000 (d) 10,000 (e) 20,000

Question 3: We have a file of 1,000,000 records, which we hash into a hash table of 1000 buckets. 100 records will fit in one block. We use zero or more blocks for each bucket, keeping them packed as full as possible. However, we do not allow two or more buckets to share a block. What is the *maximum* number of blocks we need to hold all the buckets of the hash table?

(a) 10,000 (b) 10,990 (c) 10,999 (d) 11,000 (e) 11,001

Question 4: In the following kd-tree:



the set of points that could be inserted into the leaf with $(30, 260)$ is best described by:

(a) $salary < 300$ and $age < 47$
(b) $150 \leq salary < 300$ and $47 \leq age < 60$
(c) $150 \leq salary < 300$ and $age < 47$
(d) $150 \leq salary$ and $age < 60$
(e) $salary < 300$ and $47 \leq age$

Question 5: The following schedule of transactions T_1, T_2, and T_3

$$r_1(A);\ r_2(A);\ r_3(B);\ w_1(A);\ r_2(C);\ r_2(B);\ w_2(B);\ w_1(C);$$

is conflict-equivalent to which of the following serial schedules?
(a) T_3, T_1, T_2 (b) T_2, T_3, T_1 (c) T_3, T_2, T_1 (d) T_2, T_1, T_3 (e) It's a trick question; there is no equivalent serial order.

Question 6: The following schedule of transactions T_1, T_2, and T_3

$$r_1(A);\ r_2(B);\ r_3(C);\ inc_2(C);\ r_3(B);\ inc_1(C);\ inc_2(A);$$

where *inc* stands for the increment action, is conflict-equivalent to which of the following serial schedules?
(a) T_3, T_1, T_2 (b) T_2, T_3, T_1 (c) T_3, T_2, T_1 (d) T_2, T_1, T_3 (e) It's a trick question; there is no equivalent serial order.

Question 7: Four hosts, C (the coordinator), X, Y, and Z are jointly running pieces of a transaction T, and attempt to engage in a two-phase commit of T. Unfortunately, there is a crash, and both C and X become inaccessible from Y and Z. When Y and Z examine their logs to see what to do about T, which of the following situations could arise?

I. Y's last log entry about T is `Abort` T, and Z's last log entry about T is `Ready` T.

II. Y's last log entry about T is `Commit` T, and Z's last log entry about T is `Abort` T.

III. Y's last log entry about T is `Don't commit` T, and Z's last log entry about T is `Abort` T.

(a) I only (b) II only (c) III only (d) I and II only (e) I and III only

The following three questions are based on four relations AB, BC, CD, and DE. In each relation, the schema is the two attributes whose names form the name of the relation (e.g., relation AB has attributes A and B). We have the following statistics:

$N(AB) = 100$; $N(BC) = 200$; $N(CD) = 300$; $N(DE) = 400$;
$V(A, AB) = 20$; $V(B, AB) = 50$; $V(B, BC) = 50$; $V(C, BC) = 40$;
$V(C, CD) = 60$; $V(D, CD) = 100$; $V(D, DE) = 50$; $V(E, DE) = 100$;

Question 8: Given these numbers and the usual rule for estimating join sizes (as in the notes or text), the best estimate of the size of $AB \bowtie BC \bowtie CD \bowtie DE$ is:
(a) 8000 (b) 10,000 (c) 12,000 (d) 15,000 (e) 20,000

Question 9: Let us disallow any expression tree in which a natural join is actually a product (i.e., there is no attribute in common between the left and right arguments). Then the number of distinct (not identical), left-deep expression trees for the join of all four relations is:
(a) 0 (b) 1 (c) 2 (d) 6 (e) 8

Question 10: For which of the following expressions is the sum of the estimated sizes of the two intermediate relations the least?
(a) $AB \bowtie \big(BC \bowtie (CD \bowtie DE)\big)$
(b) $(AB \bowtie DE) \bowtie (BC \bowtie CD)$
(c) $DE \bowtie \big(CD \bowtie (AB \bowtie BC)\big)$
(d) $(AB \bowtie BC) \bowtie (CD \bowtie DE)$
(e) $\big((CD \bowtie BC) \bowtie AB\big) \bowtie DE$

Question 11: Suppose blocks are sized so that they can either hold five records of relation R or be used as a B+ tree node with 10 keys and 11 pointers. If R has 1000 records, what is the smallest number of blocks that could be used to store R and a sparse, B+ tree index on the key of R?

(a) 200 (b) 210 (c) 221 (d) 223 (e) 245

The following two questions are based on a grid file with two dimensions, x and y. In each dimension, the space is divided into 1000 stripes by 999 grid lines, not necessarily spaced evenly. An index in either dimension may be thought of as a sorted list of the values of the grid lines. On each block holding this list, we put the number (from 0 to 999) of the stripe just below the lowest grid line on that block.

Also needed in this structure is an bucket array A, which in C would be declared `A[1000][1000]`, whose elements are pointers to the block(s) holding the members of one bucket of the grid file. To lookup the point (x, y), we search both dimensions' indexes to find the stripes in each dimension where this point would lie, say stripe i_x in the x-dimension and i_y in the y-dimension. We then consult the array element $A[i_x][i_y]$ for a pointer to the bucket where (x, y) may be found.

Assume integers take 4 bytes, pointers take 8 bytes, and each block holds 1024 bytes. Also assume that the array A is found in a known sequence of blocks, so that given i_x and i_y we may go directly to the block where $A[i_x][i_y]$ is found, without the need for another index.

Question 12: What is the minimum number of blocks needed to hold this structure (indexes and bucket array), not including any space needed to hold the buckets themselves?

(a) 6710 (b) 7821 (c) 8932 (d) 10043 (e) 11154

Question 13: If we add whatever additional levels of index to the indexes for the dimensions that we wish, then the minimum number of disk I/O's needed to lookup a point (assuming nothing is initially in memory and excluding any accesses to read the bucket itself) is:

(a) 3 (b) 5 (c) 7 (d) 9 (e) 11

Question 14: Suppose A and B are database elements and their initial values are both 0. After transaction T executes their values are both changed to 1. The contents of the log are given below:

$$< \texttt{START}\ T > < T, A, 0 > < T, ?, ? > < \texttt{COMMIT}\ T >$$

What kind of logging was used, and what are the values for the two question marks?

(a) Undo and B,0 (b) Undo and B,1 (c) Redo and B,1 (d) Redo and B,0 (e) Redo and A,1

Question 15: Suppose we are using timestamp-based concurrency control, and there are two transactions T_1 and T_2 with timestamps 100 and 200, respectively. They try to execute the following sequence of actions in the (wall-clock-time) order shown:

$$r_1(A);\ r_2(B);\ w_1(B);\ w_2(A);$$

Which of the following best characterizes what happens?

(a) All four actions succeed.
(b) $w_1(B)$ succeeds, but $w_2(A)$ fails and causes T_2 to abort.
(c) $w_1(B)$ fails and T_1 aborts; then $w_2(A)$ succeeds.
(d) $w_1(B)$ and $w_2(A)$ both fail and cause their transactions to abort.
(e) $w_1(B)$ fails and causes both T_1 and T_2 to abort.

Question 16: Consider the following schedule of actions, with one action replaced by ???:

$r_1(A)$; $r_2(B)$; ???; $w_1(C)$; $w_2(A)$;

Which of the following substitutions for ??? will make the schedule *not* be conflict-serializable?
(a) $r_2(A)$ (b) $r_1(B)$ (c) $w_2(B)$ (d) $r_1(C)$ (e) $r_2(C)$

The next two questions refer to the following sequence of lock requests, where $L_i(X)$ means that transaction T_i requests a lock on database element X.

$L_1(A)$; $L_2(B)$; $L_3(C)$; $L_1(B)$; $L_2(C)$; $L_3(A)$;

The timestamps of T_1, T_2, and T_3 are 100, 200, and 300, respectively. We make the following assumptions about timing:

1. If a transaction dies or is wounded, it instantaneously gives up its locks. It does not, however, restart, until a fixed amount of time t elapses. Moreover, t is sufficiently large that any current transaction will either commit or abort within time t. That is, both of the transaction's lock requests are moved to the end of the lock-request sequence.
2. If a transaction ever gets all the locks it wants, then it instantaneously completes work, commits, and releases its locks.
3. When a lock is released, it is instantaneously given to any transaction waiting for it (in a first-come-first-served manner, although that is irrelevant for this problem).

Question 17: If a wait-die strategy is used to handle lock requests and avoid deadlock, the order in which the transactions commit is:
(a) T_1, T_2, T_3 (b) T_1, T_3, T_2 (c) T_2, T_1, T_3 (d) T_2, T_3, T_1 (e) T_3, T_1, T_2

Question 18: If a wound-wait strategy is used to handle lock requests and avoid deadlock, the order in which the transactions commit is:
(a) T_1, T_2, T_3 (b) T_1, T_3, T_2 (c) T_2, T_1, T_3 (d) T_2, T_3, T_1 (e) T_3, T_1, T_2

The next two questions refer to the following schedule:

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

That is, S is a serial schedule of three transactions, T_1, T_2, and T_3, each of which reads database element A and then writes database element B. Note that S is therefore "a serializable schedule equivalent to S."

Question 19: How many schedules of the six actions in S are conflict-equivalent to S?
(a) 6 (b) 12 (c) 15 (d) 24 (e) 30

Question 20: How many schedules of the six actions in S are view-equivalent to S?
(a) 6 (b) 12 (c) 15 (d) 24 (e) 30

Question 21: Consider the sequence of actions:

$$r_1(A);\ r_2(B);\ w_1(C);\ r_3(B);\ r_3(C);\ w_2(B);\ w_3(A);$$

Assume that the read- and write-timestamps of A, B, and C are each 0 prior to execution of these actions. For which initial timestamps of the transactions T_1, T_2, and T_3 will no transaction abort because of the timestamp rule?
(a) $TS(T_1) = 100;\ TS(T_2) = 200;\ TS(T_3) = 300$
(b) $TS(T_1) = 200;\ TS(T_2) = 100;\ TS(T_3) = 300$
(c) $TS(T_1) = 300;\ TS(T_2) = 200;\ TS(T_3) = 100$
(d) $TS(T_1) = 300;\ TS(T_2) = 100;\ TS(T_3) = 200$
(e) $TS(T_1) = 100;\ TS(T_2) = 300;\ TS(T_3) = 200$

Question 22: In the following concurrency-by-validation problem we use $R_i(X)$ to mean "transaction T_i starts, and its read set is the list of database elements X." Also, V_i means "T_i attempts to validate," and $W_i(X)$ means that "T_i finishes, and its write set was X." The sequence of events is:

$$R_1(A, B);\ R_2(B, C);\ V_1;\ R_3(C, D);\ V_3;\ W_1(A);\ V_2;\ W_2(A);\ W_3(B);$$

Which of the following tells what happens?

(a) T_1 validates, but neither T_2 nor T_3 validates.
(b) T_1 and T_2 validate, but T_3 does not.
(c) T_1 and T_3 validate, but T_2 does not.
(d) T_2 and T_3 validate, but T_1 does not.
(e) All three transactions validate.

Question 23: Suppose we have 8 data disks, numbered 1 through 8, and three redundant disks, 9, 10, and 11. Disk 9 is a parity check disk on disks 1, 3, 5, and 7, while disk 10 is a parity check disk on disks 2, 4, 6, and 8. Assume that all pairs of simultaneous disk failures are equally likely. Then in order to maximize the probability that we can survive two simultaneous disk failures, we should make disk 11 a parity check on:

(a) Disks 1 through 8. (b) Disks 9 and 10. (c) Disks 3, 4, and 5. (d) Disks 1, 2, 3, and 4. (e) Disks 1, 2, 5, 7, and 9.

Question 24: We wish to join the relations $R(a, b)$, $S(b, c)$, and $T(c, d)$ under the following assumptions:

1. There are B buffers available to hold blocks of data from these three relations.
2. The relations occupy n_R, n_S, and n_T blocks, respectively.
3. R is stored sorted by b; the other relations are unsorted.
4. The particular strategy we shall use to perform the join is:
   - *i)* Perform the first phase of two-phase multiway merge sort on S. That is, as many times as necessary, load all buffers from S, sort the tuples on b, and write out the sorted sublist.
   - *ii)* Load T entirely into main memory, using as many buffers as needed.
   - *iii)* Merge (and join when appropriate) R and the sorted sublists of S, and compare each of the resulting tuples with the tuples of T. Any join tuple in the result is stored in an output buffer, not counted among the B buffers available for this process.

Which of the following inequalities is the closest approximation to the condition under which this sequence of steps can be carried out as described?

(a) $B^2 \geq n_S + Bn_T$
(b) $B^2 \geq n_T + Bn_S$
(c) $B^2 \geq n_T + n_S$
(d) $B^2 \geq n_R + Bn_T + B^2 n_S$
(e) $B^2 \geq n_R + n_T + Bn_S$

The following two questions are based on the following undo/redo log (ST stands for "start"):

$< \texttt{ST}\ T_1> \ <\texttt{ST}\ T_2> \ <\texttt{ST}\ T_3> \ <T_1, A, 0, 1> \ <T_2, A, 1, 2> \ <T_3, A, 2, 3> \ <\texttt{COMMIT}\ T_2>$

Question 25: Of the following statements:

I. At least one transaction performed a dirty read.

II. The value of A on disk at the end of this log *could be* 0.

III. The value of A on disk at the end of this log *could be* 2.

which are true?
(a) I only (b) I and II only (c) I and III only (d) II and III only (e) I, II, and III

Question 26: If we follow the undo/redo rule for recovery (and in particular, ignore any need for cascaded rollbacks), then the value of A on disk after recovery is:
(a) 0 (b) 1 (c) 2 (d) 3 (e) It cannot be determined from the information given.

Question 27: Suppose we have a distributed locking system, where one obtains a global shared lock on a database element A by obtaining local shared locks on s copies of A and obtains a global exclusive lock on A by obtaining local exclusive locks on x copies of A. Assume that local locks work as expected: two transactions may not hold local exclusive locks or a local shared and exclusive lock on the same element at the same copy. Also assume that s and x are chosen so that global locks work properly: $2x > n$ and $x + s > n$, where n is the number of copies. Finally, we make the reasonable assumption that $s \leq x$, i.e., it is no harder to get a shared lock than an exclusive lock globally.

Let d be the number of the n copies of A that are down; i.e., it is impossible to get a local lock on A at those copies. Which of the following conditions guarantees that it is still possible to get global shared locks and global exclusive locks on A?
(a) $d \leq n - x$ (b) $d \leq s$ (c) $d \leq x - s$ (d) $d \leq n - s$ (e) $d \leq x$

Question 28: Suppose we have B-tree nodes as in the notes, with room for three keys and four pointers. As in the notes examples, when we split a leaf, we divide the pointers 2 and 2, and when we split an interior node, we divide the node so 3 pointers go with the first (left) node and 2 pointers go with the second. Suppose we start with a leaf containing pointers to records with keys 1, 2, and 3, and we continue to insert records with keys $4, 5, 6, \ldots$. At the insertion of which key will the tree first reach four levels (i.e., three levels of interior nodes plus a level of leaves)?

(a) 28 (b) 32 (c) 54 (d) 63 (e) 81

Question 29: Below is the conflict matrix for three hypothetical lock modes, X, Y, and Z. As always, a lock on a database element A can be granted in the mode represented by column j if and only if there is *no* other transaction that holds a lock on A in some mode i, where the entry in row i and column j is "no."

	X	Y	Z
X	yes	yes	no
Y	no	yes	yes
Z	yes	no	yes

Of the three statements:

I. It is possible for different transactions to hold locks on the same database element in lock modes X and Z at the same time.

II. It is possible for more than one transaction to hold a lock on some database element in mode Z, at the same time another transaction holds a lock on the same element in mode Y.

III. It is possible for different transactions to hold locks on the same element in all three lock modes at the same time.

Which are true?

(a) I only (b) II only (c) I and II only. (d) I and III only. (e) I, II, and III.

Question 30: Suppose we have a large number of processors, and relation R is distributed randomly among the disks associated with n of these processors. Suppose also that R has T tuples. We wish to hash all the tuples of R and ship tuple t to the processor numbered $h(t)$, where the result of hash function h is a number from 1 to n. We also want to store the received tuples in the main memory of each processor, and there is room in each main memory for M tuples. Assume that it costs one "communication unit" to ship one tuple to any processor other than the one on which it is initially found. Finally, assume that h is completely random; i.e., it is equally likely to assign a tuple to any processor, each processor will receive exactly the average number of tuples, and h is not related to the initial position of tuples.

Suppose we are allowed to choose n as we wish, to minimize the total number of communication units, while performing the task described above. The minimum expected number of communication units is:

(a) M (b) $T - M$ (c) T (d) $T + M$ (e) T^2/M

The following two questions concern the two transactions:

T_1: $sl(A)$; $ul(B)$; $xl(B)$;
T_2: $ul(A)$; $xl(A)$;

where sl, ul, and xl stand for shared, update, and exclusive locks, respectively.

Question 31: If T_1 and T_2 obey the 2-phase-locking rule, i.e., they cannot release any lock until they have acquired all their locks, then the number of legal sequences of the 5 lock steps above is:

(a) 0 (b) 1 (c) 2 (d) 3 (e) 4

Question 32: If T_1 and T_2 are *not* required to obey the 2-phase locking rule, but cannot acquire a lock on a DB element X after having unlocked any lock they held on X, then the number of legal sequences of the 5 lock steps above is:

(a) 1 (b) 3 (c) 5 (d) 7 (e) 9