CS 245
Final Exam – Winter 1996

This exam is open book and notes. You have 120 minutes to complete it.

Print your name:____

The Honor Code is an undertaking of the students, individually and collectively:

1. that they will not give or receive aid in examinations; that they will not give or receive unpermitted aid in class work, in the preparation of reports, or in any other work that is to be used by the instructor as the basis of grading;

2. that they will do their share and take an active part in seeing to it that others as well as themselves uphold the spirit and letter of the Honor Code.

The faculty on its part manifests its confidence in the honor of its students by refraining from proctoring examinations and from taking unusual and unreasonable precautions to prevent the forms of dishonesty mentioned above. The faculty will also avoid, as far as practicable, academic procedures that create temptations to violate the Honor Code.

While the faculty alone has the right and obligation to set academic requirements, the students and faculty will work together to establish optimal conditions for honorable academic work.

I acknowledge and accept the Honor Code.

Signed:____

Problem	Points	Maximum
1		10
2		10
3		10
4		10
5		10
6		10
7		10
Total		70

1. **(10 pts)** Indicate whether each of the following statements is TRUE or FALSE. If you chose FALSE, please explain why.

   (a) We want to perform a merge join of two unsorted relations R and S. The minimum memory requirement for the whole process is $\lceil\sqrt{\max(|R|, |S|)}\,\rceil$ blocks (ignoring constant factors that do not depend on the size of R or S).

   (We assume that $|R|$ means the number of blocks in relation R, and that $|S|$ means the number of blocks in S.)
   TRUE

   (b) We want to perform a hash join of two relations R and S. The minimum memory requirement for the whole process is $\lceil\sqrt{\max(|R|, |S|)}\,\rceil$ blocks (ignoring constant factors that do not depend on the size of R or S).

   FALSE

   (c) Checkpointing (as discussed in class) can significantly reduce recovery time when using *UNDO* logging.

   TRUE

   (d) If a schedule is not conflict serializable, then it cannot possibly maintain consistency constraints.

   FALSE

   (e) There are serializable schedules that are not conflict serializable.

   TRUE

   (f) Two schedules with identical acyclic precedence graphs must be conflict equivalent.

   TRUE

   (g) When an access authorization is given with *grant option*, the recipient of the grant can give it in turn to an unlimited number of other users.

   TRUE

   (h) A semi-join is only useful when each relation participating in the join is partitioned across more than one site.

   FALSE

   (i) In a two phase commit protocol, the transaction reaches its commit point between the first and the second phase.

   TRUE

(j) A distributed database system is not necessarily more reliable than a centralized one.

TRUE

2. **(10 pts)**

Consider the following execution schedule:

$$S = w_3(B)\, w_3(A)\, w_1(B)\, r_2(B)\, r_2(A)\, w_1(A).$$

You are trying to determine if this schedule is view serializable.

(a) Show the singleton arcs (those where there is no choice) of the labeled precedence graph for S. Do not show the poly-arcs. (Of course, you also need to show the nodes of the graph. Also, if two arcs are identical, please only show one.)

(see attached handwritten solutions)

(b) Show the poly-arcs for this labeled precedence graph. Do not include here the arcs you showed in your previous answer. (Again, show all nodes.)

(see attached handwritten solutions)

(c) Is schedule S view serializable? Briefly explain your answer.

(see attached handwritten solutions)

3. **(10 pts)** Consider a multi-granularity locking scheme where the database is divided into *files*, which are divided into *blocks*, which are divided into *records*. (That is, we have a three level hierarchy.)

   (a) Consider a transaction T_1 that wishes to read and modify a particular record r_1 in block b_{25} in file f_3. What locks does T_1 need to obtain before performing the desired action?

   IX on f_3. IX on b_{25}. X on r_1.

   (b) Consider a transaction T_2 that wishes to read all of file f_5. What locks does T_2 need to obtain before performing the desired actions?

   S on f_5.

   (c) Consider a transaction T_3 that wishes to read all of file f_5 and while doing that will modify records r_1 (in block b_{25}) and r_2 (in block b_{26}). What locks does T_3 need to obtain before performing the desired actions?

   SIX on f_5. IX on b_{25}. IX on b_{26}. X on r_1. X on r_2.

   (d) Consider the lock for a specific file f_3 and the following requests, made in this order: IS (by T_4), IS (by T_5), IX (by T_6), IS (by T_7), S (by T_8), SIX (by T_9), IX (by T_{10}). After these lock requests are issued (and none of the granted locks are released), what transactions will be waiting? Assume that your lock manager wishes to avoid starvation. IS to T_4 granted. IS to T_5 granted. IX to T_6 granted. IS to T_7 granted. S to T_8

   not granted. SIX to T_9 not granted. IX to T_{10} not granted (to avoid starving T_8). T_8, T_9, and T_{10} are waiting.

   (e) For exactly the same set of requests as in part (e), but assuming that you allow starvation (in order to grant as many locks as quickly as possible), what transactions will be waiting after these requests?

   IS to T_4 granted. IS to T_5 granted. IX to T_6 granted. IS to T_7 granted. S to T_8 not granted. SIX to T_9 not granted. IX to T_{10} granted (let T_8 starve). T_8 and T_9 are waiting.

4. **(10 pts)** For each schedule below:

- indicate if the schedule is recoverable or not; and
- indicate if the schedule is conflict serializable or not.

(a)

$$S_1 = w_1(A)\, w_2(B)\, w_3(C)\, c_1\, r_3(A)\, c_2\, r_3(B)\, c_3$$

Recoverable, Conflict-Serializable.

(b)

$$S_1 = w_1(A)\, w_2(B)\, w_3(C)\, r_3(A)\, c_2\, r_3(B)\, c_3\, c_1$$

Not Recoverable. Conflict-Serializable.

(c)

$$S = r_1(A)\, w_2(A)\, w_2(B)\, c_2\, r_1(B)\, c_1$$

Recoverable. Not Conflict-Serializable.

(d)

$$S = w_1(A)\, w_1(B)\, r_2(A)\, w_2(B)\, a_2\, w_1(C)\, c_1\, r_3(B)\, w_3(C)\, c_3$$

Recoverable. Conflict-Serializable.

5. **(10 pts)** Your database system is executing a hash join with the following parameters:

- k: the number of buckets into which you will partition the relations.
- m: the total amount of memory, in blocks. Remember that one of these blocks is used for reading the relations.
- s: the number of blocks for each relation. (Assume that both relations are of equal size.)

(a) How large does m need to be so that the first phase of the algorithm can run?

$m \geq k + 1$

(b) How large does m need to be so that the second phase of the algorithm can run?

$m \geq \lceil s/k \rceil + 1$

(c) Hybrid hash join is an optimization of hash join that keeps some of the buckets in memory in the bucketizing phase. Derive an equation that gives the maximum number of buckets that can be kept in memory when joining two relations. Your answer should be in terms of parameters given above.

Let b be the number of buckets that we can keep in memory.

$m = b\frac{s}{k} + (k - b)1 + 1$
$b = \frac{m}{\frac{s}{k} - 1} - k + 1$

6. **(10 pts)** When the recovery manager runs after a crash, it can execute the following types of actions:

- *write(Z, v):* This action stores the value v into database object Z.
- *abort(T_j):* This action writes an abort T_j record into the log. (The record is not written to the log until all T_j writes generated during recovery have been flushed to disk.)

(a) The Megatron 1000 system uses *UNDO* logging only. Given the contents of the log below, list the actions performed by the recovery manager (using the two action types given above). In the log shown here, the format of each record is `<Tid, ObjectId, Old value>`. The oldest record is shown in the left most position of line 1. The second log record written is the second one of line 1. The records on line 2 follow all of those on line 1. The last record written before the crash is the last one of line 2.

```
<T1, start>   <T1, X, 10>   <T2, start>
<T2, Z, 12>   <T2, Y, 20>   <T2, commit>
```

$write(X, 10)$
$abort(T_1)$

(b) The Megatron 1001 system uses *REDO* logging only. The contents of the log at recovery time are given below. Lists the actions performed by the recovery manager. The format of the each log record is `<Tid, ObjectId, New value>`.

```
<T1, start>   <T1, X, 10>   <T2, start>
<T2, Z, 12>   <T2, Y, 20>   <T2, commit>
```

$write(Z, 12)$
$write(Y, 20)$
$abort(T_1)$

(c) The Megatron 9500 uses *UNDO, REDO* logging with checkpointing (as described in class). Given the contents of the log below, lists the actions performed by the recovery manager. The format of each log record is `<Tid, ObjectId, New value, Old value>`.

```
<T1, start>   <T1, X, 10, 11>   <T1, commit>
<checkpoint>  <T2, start>       <T2, Y, 20, 21>
<T3, start>   <T3, Z, 30, 31>   <T4, start>
<T3, commit>  <T4, W, 40, 41>   <T2, Z, 80, 30>
```

(Must do undo before redo.)
$write(Z, 30)$ (Undo)
$write(W, 41)$ (Undo)
$write(Y, 21)$ (Undo)
$write(Z, 30)$ (Redo)
$abort(T_2)$
$abort(T_4)$

7. **(10 pts)** Your goal is to estimate the cost of performing the natural join between relations $R(A, B, C)$ and $S(C, D)$, in terms of number of IOs (i.e., number of blocks that will be read into memory). We are told that

- $n_R = 300$
- $n_S = 1,200$
- $Dom(C, R) = Dom(C, S) = 10^4$
- Values in R and S are uniformly distributed over the domain of the corresponding attribute.
- There are no duplicate C values in R.
- There are no duplicate C values in S.

To execute the join you will execute the following plan: (1) First, scan all tuples of R. For each tuple, probe an in-memory index of S over attribute C. (2) If the C value is found in the S index, then fetch the S tuple, join it to the R tuple, and output the joined tuple. While executing this plan, no blocks are cached, so every access incurs an IO.

(a) Assume that the R records are stored contiguously in a file, each block holding 3 R records. The S tuples are stored in a separate file, contiguously but in random order. Each block of this file holds 6 S records. How many IO do you estimate it will take to perform the join? (Ignore IOs needed to output result.) Briefly explain your answer.

$300(1 + \frac{1200}{10000}(1)) = 336$ IOs

(b) Now assume that the records are placed differently on disk. The R and S records are clustered, so that each block contains 2 R records and 2 S records. If an R record has a C value that appears in S, then that S record is placed in the same block. If an R record has a C value that does not appear in S, then there is some other S record in the block. (This S record has a C value that is not referenced by any R record.) How many IOs are needed for the join in this scenario? (Ignore IOs needed to output result.) Briefly explain your answer.

300 / 2 = 150 IOs