CS-245 Database System Principles

Final Exam – Winter 2001

This exam is open book and notes. You have 120 minutes to complete it. There are 8 questions.

Serial number: __________
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 condense 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		15
5		15
6		10
7		15
8		15
Total:		100

Problem 1 (10 points)

State if the following statements are **TRUE** or **FALSE**. If a statement is false, please explain why.

a) Any schedule produced by a lock scheduler using shared and exclusive locks is conflict serializable.

b) Multi-key indexes are better than partitioned hash tables for answering range queries on multiple key attributes.

c) Undo logging is suitable for use with archiving, though it's less efficient than redo and undo/redo logging.

d) No deadlocks can occur under 2PL.

e) Consider a disk that has 16 surfaces, each with 1024 tracks. Each track is divided into 36 sectors, and a sector holds 2048 bytes. Blocks consist of 2 consecutive sectors. 20% of the circumference of each track is occupied by gaps between the sectors. The disk rotates at 3840rpm, i.e., one rotation every 15.6 milliseconds. Given these parameters, the transfer time for one block is approximately 0.78ms.

f) Relation R has 4000 records. Suppose a clustered B+ tree index is built on R's key attribute. A block can hold a B+ tree node (interior or leaf node) with 20 keys and 21 pointers. In this case, at least 220 blocks are needed to hold the entire B+ tree index.

g) Consider an extensible hash table where the hash function generates a 20-bit hash key, but only the high order 8 bits are used. In this case, the hash table can have at most 2^{20-8} data blocks (containing records).

h) The "containment of value sets property" states that whenever two attributes appear with the same name, the values these attributes can take come from the same set of values.

i) In a Zipfian distribution, the frequency of occurrence of the i^{th} most common value is in proportion to 1 over the square root of i.

j) Sequence pointers are used in leaf nodes of B+ trees to speed up range queries.

Problem 2 (10 points)

Examine the schedule given below. There are three transactions, T1, T2, and T3. Initially, the salary = 1 and the tax = 2. The assignments happen within the local memory space of the transactions and the effects of these assignments are not reflected in the database until the WRITE operation.

```
     T1                       T2                  T3
---------------------------------------------------------------------
0                                                   start
1                                                   READ tax
2                                                   tax := tax + 1
3      start
4      READ salary
5      salary := salary + 1
6                                                   WRITE tax
7                                                   commit
8                              start
9                              READ tax
10                             READ salary
11                             tax := tax + salary
12                             WRITE tax
13                             commit
14    ------------------- checkpoint start --------------------------
15    READ tax
16    tax := tax + salary
17    WRITE salary
18    ------------------- checkpoint end --------------------------
19    commit
```

a) Show the undo/redo log file entries that would be generated by this execution. Use the same notation used in class. For each log entry, indicate what line above generates it.

b) Assume that the undo/redo recovery algorithm with checkpoints is being used. The database crashes immediately after statement 7. (Assume that all the log records up to this point are on disk.)

b.1) Which transactions would have to be undone?

b.2) Which transactions would have to be redone?

c) Again assume that the undo/redo recovery algorithm with checkpoints is being used, but now the databases crashes just after statement 18. (Assume that all the log records up to this point are on disk.)

c.1) Which transactions would have to be undone?

c.2) Which transactions would have to be redone?

Problem 3 (10 points)

Examine the schedule given below. There are four transactions, T1, T2, T3, and T4.

```
         T1              T2           T3              T4
--------------------------------------------------------------------
1                                                    READ tax
2     READ salary
3                                                    WRITE tax
4                    READ tax
5                    WRITE tax
6     READ tax
7     WRITE salary
8                                 READ salary
9     WRITE tax
10                                WRITE salary
11                                                   READ salary
12                                                   WRITE salary
```

a) Draw the precedence graph for this schedule.

b) What is the equivalent serialization order for this schedule? If no order is possible, then state 'none'.

c) Assume that transaction T4 did not run at all. What is the precedence graph in this case?

d) What is the equivalent serialization order for this second schedule? If no order is possible, then state 'none'.

Problem 4 (15 points)

Consider a four-dimensional data cube S(W, X, Y, Z), where the following sum aggregates have been materialized:

Materialized:	S(W, X, Y, Z)
	S(*, X, Y, Z)
	S(W, X, *, *)
	S(W, *, Y, *)

For each of the following queries, give the materialized aggregates (or the cube itself) from which the query can most *efficiently* be computed. If the result can be computed from two aggregates S_1 and S_2, but the computation from S_1 is clearly less expensive than the computation from S_2, then only give S_1 in your answer. If with the information given here, you cannot infer that S_1 is better than S_2 or vice versa, then give both S_1 and S_2 in your answer.

Note that lower case variables represent particular values of the attribute (e.g., x is a value for X).

a) S(*, x, y, z)

b) S(w, *, *, *)

c) S(*, *, *, *)

d) S(w, *, y, z)

Problem 5 (15 points)

Consider a database that uses a validation mechanism for concurrency control. There are only 4 kinds of transactions in the system. The read, write actions of the 4 kinds of transactions are given below.

(i):	r(A) r(B) w(A) w(C) w(E)
(ii):	r(C) w(F) w(B)
(iii):	r(D) r(B) w(B)
(iv):	r(c) r(D)

Note that (i) - (iv) are transaction *types* and not transactions. In particular we can have two different transactions of the same type. Two transactions are said to execute *concurrently* if at some point of time, both the transactions have started, but neither has finished.

a) Determine for each pair of transaction types *(m, n)* (including the case *m* = *n*), if the transaction of type *n* can be aborted due to a transaction of type *m* when executing concurrently (in the absence of any other transaction). Please indicate your answer in the table below by placing a **YES** in column *n* and row *m* if a transaction of type *n* can be aborted due to a transaction of type *m*, and **NO** otherwise.

Transaction of this type (*n*)

Can be aborted by type below (*m*)	(i)	(ii)	(iii)	(iv)
(i)				
(ii)				
(iii)				
(iv)				

b) Determine for each pair of transactions *(m, n)* if the two transactions **can** be run concurrently without either of them aborting (in the absence of any other transaction). Use the notation followed in part (a) to indicate the answer in the table below. In particular, a **YES** in column *n* and row *m* will mean that a transactions of type *m* and *n* can run concurrently (i.e., a transaction of type *m* need not be aborted by a type *n* transaction, and that a transaction of type *n* need not be aborted by a type *m* transaction.)

	(i)	(ii)	(iii)	(iv)
(i)				
(ii)				
(iii)				
(iv)				

c) What is the condition under which two transactions can *never* execute concurrently in the validation scheme (without one of them being rolled back)? Express your answer in terms of read-sets (RS(T1), RS(T2)) and write-sets (WS(T1), WS(T2)) of the two transactions T1 and T2.

Problem 6 (10 points)

For each of the following statements, indicate if it is TRUE or FALSE. If the statement is false, give a simple counter-example. To construct counter-examples, please only use schedules that use the following two transactions (and only these transactions). It is OK to leave out some actions of these two transactions if you wish. (However, you should not reorder the actions of transactions.)

For this problem, use the definition of recoverable we used in class, not the one used in the textbook.

- T1: w1(A) w1(B) r1(C) c1
- T2: w2(A) r2(B) w2(C) c2

a) All serial schedules are conflict-serializable.

b) All avoids-cascading-rollback schedules are recoverable.

c) All strict schedules are serial.

d) All recoverable schedules are conflict-serializable.

e) All strict schedules are conflict-serializable.

Problem 7 (15 points)

Consider a legacy source that offers data equivalent to a relation R(X, Y, Z).

For parts (a) through (d) below, assume that the capabilities of this source are described by the following two templates:

- f u b
- o[x_1,x_2] f u

a) Can the following query be *directly* answered by this source?

- R(x_1, Y, z_1)

b) Can the following query be *directly* answered by this source?

- R(X, y_1, z_1)

c) Can the following query be *directly* answered by this source?

- R(X, y_1, Z)

d) Can the following query be *directly* answered by this source?

- R(x_3, y_1, Z)

For parts (e) and (f) below, assume that the capabilities of this source are described by the following single template:

- b u' c[z_1, z_2]

e) The following query *cannot* be directly supported by the source:

- R(x_1, Y, z_3)

However, can it be answered by submitting a different (and supported) query and then post-processing the results to get the desired answer? If yes, please give the query to submit to the source and explain how its answer is processed. If no, briefly explain why not.

f) The following query *cannot* be directly supported by the source:

- R(x_1, y_1, Z)

However, can it be answered by submitting a different (and supported) query and then post-processing the results to get the desired answer? If yes, please give the query to submit to the source and explain how its answer is processed. If no, briefly explain why not.

Problem 8 (15 points)

We would like to sort the tuples of a relation R on a given key. The following information is known about the relation.

- The relation R contains 100000 tuples, i.e., T(R) = 100000.
- The size of a block on disk is 4000 bytes.
- The size of each R tuple is 400 bytes.
- Relation R is clustered, that is each disk block holding R tuples is full of R tuples.
- The size of the sort key is 32 bytes.
- A record pointer is 8 bytes.

Answer the following questions based on the information above.

a) If we use a two pass sorting algorithm, what is the minimum amount of main memory (in number of blocks) required?

b) What is the cost of the two pass sorting algorithm in terms of number of disk I/Os? Include the cost of the writing the sorted file to the disk in the end in your calculations.

c) Consider the following variant of the sorting algorithm. Instead of sorting the entire tuple, we just sort the <key, recordPointer> for each tuple. As in the conventional two pass sorting algorithm, we sort chunks of <key, recordPointer> in main memory and write the chunks to disk. In the merge phase, <key, recordPointer> entries from different chunks are merged. The record pointers to used to recover the rest of the tuple (from the original copy of R) and write the sorted relation to the disk. What is the minimum amount of main memory required for this operation? What is the cost in terms of number of disk I/Os?

d) Keeping all other parameters constant, for what values of tuple size is the variant discussed in (c) better (in the number of IOs)?