CS411 Database Systems

Fall 2008

Department of Computer Science
University of Illinois at Urbana-Champaign

Final Examination

December 16, 2008
Time Limit: 180 minutes

- Print your name and NetID below. In addition, print your NetID in the upper right corner of every page.

  Name: _____ **NetID:** _____

- Including this cover page, this exam booklet contains **16** pages. Check if you have missing pages.
- The exam is closed book and closed notes. You are allowed to use scratch papers. No calculators or other electronic devices are permitted. Any form of cheating on the examination will result in a zero grade.
- Please write your solutions in the spaces provided on the exam. You may use the blank areas and backs of the exam pages for scratch work.
- Please make your answers clear and succinct; you will lose credit for verbose, convoluted, or confusing answers. *Simplicity does count!*
- Each problem has different weight, as listed below– So, plan your time accordingly. *You should look through the entire exam before getting started, to plan your strategy.*

Problem	1	2	3	4	5	6	7	8			Total
Points	15	12	10	12	16	12	12	11			100
Score											
Grader											

Turn over the page when instructed to do so. Good Luck!

Problem 1 (*15 points*) Basics

For each of the following statements, indicate whether it is TRUE or FALSE by circling your choice. If you change your mind, cross out both responses and write "True" or "False." You will get 1 point for each correct answer, 0 point for each incorrect answer.

(1) *True False*
When a fixed-length record is updated, it is sometimes necessary to create an overflow block.

(2) *True False*
We can only use a sparse index if the data file is sorted by the search key.

(3) *True False*
All the used pointers at the leaf nodes of a B-tree point to data records.

(4) *True False*
Every block in a B-tree is between half used and completely full.

(5) *True False*
In extensible hash tables, overflow blocks are permitted.

(6) *True False*
A linear hash table increases the size of its buckets based on the average number of records per bucket.

(7) *True False*
The cost of scanning a unclustered relation R is $B(R)$ where $B(R)$ is the number of blocks holding R.

(8) *True False*
When we join two relations with the block-based nested loop join algorithm, the smaller relation should be the outer relation of the algorithm.

(9) *True False*
Two-pass hash-based algorithms for binary operations (e.g., join) have a size requirement only on the smaller of two input relations.

(10) *True False*
In algebraic laws, the join operation is both commutative and associative.

(11) *True False*
Undo logging requires us to keep all modified blocks in buffers until the transaction commits and the log records have been flushed.

(12) *True False*
Every serializable schedule is conflict-serializable.

(13) *True False*
Two-phase locking scheme rolls back transactions that would engage in unserializable behavior.

(14) *True False*
Recoverable schedules sometimes require cascading rollback.

(15) *True False*
Both the wait-die and wound-wait schemes guarantee that every transaction eventually completes (i.e., no starvation).

Problem 2 (*12 points*) Storage

(i) A record header is a fixed-length region where information about the record itself is kept. Some information we may want to keep is the length of the record. Why would we want to keep the length information? (4 points)

(ii) Suppose each database block is 2^{20} bytes long. Each block contains a block header that is 40 bytes long. The blocks store Person records, which are fixed-length. The Person schema is shown below.

```
CREATE TABLE Person(
  name CHAR(30) PRIMARY KEY,
  gender CHAR(1),
  address VARCHAR(255));
```

In addition, each record contains a record header that has 12 bytes. Assume all fields must start at a byte that is a multiple of four.

(a) What is the length of a Person record? You only need to write down the equation. Do not calculate. (4 points)

(b) What is the maximum number of records we can fit in a block? You only need to write down the equation. Do not calculate. (4 points)

Problem 3 (*10 points*) Indexing

Consider indexing the following key values using an extensible hash table. Their corresponding hash value h(n) is already computed (see Table 1). Assume each bucket can hold at most 2 data items.

n	h(n)
A	1001
B	1111
C	1101
D	1011
E	1000
F	0101
G	1010

Table 1: Hash Function

(i) Suppose the keys are to be inserted in ascending alphabetical order, ie. A, B, C, ..., G. The hash table is initially empty. Draw the hash table **after all the keys are inserted**. Show the keys themselves in the buckets, not the hash value. Be sure to indicate i, the number of bits in the hash value that are used. For each bucket, also indicate the number of hash function bits that are used for that bucket. (5 points)

(ii) What are the advantages and disadvantages of dynamic hash tables compared to B-tree, in terms of querying? (5 points)

Problem 4 (*12 points*) Query execution

(i) Describe a one-pass algorithm for duplicate elimination (δ) of relation R and give the approximate size of main memory buffer M required for the algorithm. (4 points)

(ii) We analyzed the I/O requirements of simple sort-based join algorithm in the class. However, the algorithm needs additional disk I/O's if there are so many tuples with the same value in the join attribute that those tuples cannot fit in main memory. We consider relations $R(X, Y)$ and $S(Y, Z)$ where $B(R) = 1000$, $B(S) = 500$, $M = 101$. Calculate the total number of disk I/O's needed if there are only two Y-values, each appearing in half the tuples of R and half the tuples of S. (Hint: use the nested-loop join algorithm to deal with the issue of too many tuples with the same Y value.) (4 points)

(iii) When we join two relations with a two-pass, hash-based algorithm, the number of buckets k for each relation might be much smaller than the number of main memory buffer M. Then, there is more memory available on the first pass of the algorithm than we need to hold one block per bucket. To utilize this free main memory buffer, the hybrid-hash-join algorithm keeps one bucket in main memory to save disk I/O's of the original algorithm. Describe how to implement a two-pass hash-based algorithm for duplicate elimination (δ) using this hybrid-hash-join idea. Also, give the number of disk I/O's assuming that the number of buckets k is approximately equal to $B(R)/M$ where R is an input relation. Strictly speaking, the size of a bucket must be slightly smaller than M so that we have one block for each of the other $k-1$ buckets in memory buffer, but you do not have to worry about this issue. (4 points)

Problem 5 (*16 points*) Query optimization

(i) Give examples to show that:

(1) Projection cannot be pushed below set union. (2 points)

(2) Projection cannot be pushed below bag difference. (2 points)

(ii) Let us consider physical query plans for the expression

$$(R(w,x) \bowtie S(x,y)) \bowtie U(y,z)$$

We make the following assumptions:

(a) R occupies 2,000 blocks; S and U occupy 10,000 blocks.

(b) The intermediate result $R \bowtie S$ occupies k blocks for some k.

(c) Both joins will be implemented as hash-joins, either one-pass or two-pass, depending on k.

(d) There are 101 buffers available.

Consider three cases regarding the range of k and compute the cost of the best physical query plan for each case and fill in the blanks in Table 2. When you choose the two-pass algorithm for the final join, make sure to specify the number of buckets in the fields of the third column as well. (6 points)

Range of k	Pipeline or Materialize	Algorithm for final join	Total Disk I/O's

Table 2: Costs of physical plans as a function of k.

(iii) Consider a relation $R(a, b, c, d)$ that has a clustering index on a and nonclustering indexes on each of the other attributes. The relevant parameters are:
$B(R) = 1000$, $T(R) = 5000$, $V(R, a) = 20$, $V(R, b) = 1000$, $V(R, c) = 5000$, $V(R, d) = 500$.
Give the best query plan (index-based or table-based followed by a filter scan) and the disk I/O cost for each of the following selections:

(1) $\sigma_{(a=1)AND(b=2)AND(c=3)}R$. (3 points)

(2) $\sigma_{(a=1)AND(b=2)AND(c<3)}R$. (3 points)

Problem 6 (*12 points*) Recovery

(i) Assuming the log in Table 3 is an undo log:

(a) If we begin a nonquiescent checkpoint immediately after event number 2, when the <End CKPT> record is written? (2 points)

(b) Describe the action of the recovery manager if there is a crash, the last log record to appear on disk is number 12, and the recovery manager did the checkpointing mentioned in the previous question. (2 points)

(ii) Assuming the log in Table 3 is a redo log, and there are nonquiescent <Start CKPT> and <End CKPT> records between number 10 and 11. Describe the action of the recovery manager if there is a crash and the last log record to appear on disk is number 12. (4 points)

Table 3: Log Sequence

Number	Event
1	<Start T>
2	<T,A,10>
3	<Start U>
4	<U,B,20>
5	<T,C,30>
6	<Start V>
7	<U,D,40>
8	<V,F,70>
9	<Commit U>
10	<T,E,50>
11	<Commit T>
12	<V,B,80>
13	<Commit V>

(iii) Consider the undor/redo log in Table 4. Describe the action of the recovery manager if there is a crash and the last log record to appear on the disk is number 7. (4 points)

Table 4: Undo/Redo Log Sequence

Number	Event
1	<Start T>
2	<T,A,10,11>
3	<Start U>
4	<U,B,20,21>
5	<T,C,30,31>
6	<U,D,40,41>
7	<Commit U>
8	<T,E,50,51>
9	<Commit T>

Problem 7 (*12 points*) Concurrency control

Answer the following questions:

(i) Briefly explain why it is not necessarily desirable to execute multiple transactions as a serial schedule in a database system. (3 points)

(ii) Consider the schedule $S : w_3(A); r_1(A); w_1(B); r_2(B); w_2(C); r_3(C)$. Draw the precedence graph for the schedule S, and determine whether it is conflict-serializable or not. (3 points)

(iii) When we use timestamps as a concurrency method, the scheduler of a database system assigns each transaction T a unique timestamp $TS(T)$. The scheduler also associates each database element X with the read time of X, $RT(X)$, and the write time of X, $WT(X)$. The scheduler aborts a transaction T reading a database element X if $TS(T) < WT(X)$. Explain why. (3 points)

(iv) In the following sequences of events, $R_i(X)$ means that 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. Tell what happens when the following sequence of events is processed by a validation-based scheduler. (3 points)

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

Problem 8 (*11 points*) Deadlocks and distributed commit

Answer the following questions:

(i) For the sequence of actions below, tell which transaction wounds (i.e., rolls back) which transaction under the wound-wait deadlock avoidance system. We assume that locks are requested immediately before each read and write action and that unlocks occur immediately after the final action that a transaction executes. If a transaction has already obtained a lock on a database element, it does not need to request a lock on that element to perform further actions. We also assume the order of the timestamps of the transactions is the same as the order of subscripts for the transactions, that is, T_1, T_2, T_3, and T_4. (4 points)

$$r_1(A); r_2(B); w_1(C); w_2(D); r_3(C); w_1(B); w_4(D); w_2(A);$$

(ii) We describe sequences of messages that can take place during a two-phase commit with the notation as follows. Let (i, j, M) means that site i sends the message M to site j, where the value of M can be P (prepare), R (ready), D (don't commit), C commit, or A (abort). We consider a simple situation in which site 0 is the coordinator for a distributed transaction T, and sites 1 and 2 are the components of T. For example, the following is one possible sequence of messages that could take place during a successful commit of the transaction:

$$(0, 1, P), (0, 2, P), (2, 0, R), (1, 0, R), (0, 2, C), (0, 1, C)$$

Give an example of a sequence of messages that could occur if site 1 wants to commit and site 2 wants to abort. (3 points)

(iii) Suppose that site 1, which is a component of a distributed transaction T, fails during the two-phase commit process, and the last log record for T is $< Don't\ commit\ T >$. Describe briefly how site 1 figures out the global decision made by the coordinator. (4 points)