CS411 Database Systems

Spring 2010, Prof. Chang

Department of Computer Science
University of Illinois at Urbana-Champaign

Final Examination

May 14, 2010
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				Total
Points	22	16	11	10	12	15	14				100
Score											
Grader											

Problem 1 (*22 points*) Misc. Concepts

For each of the following statements, indicate whether it is *TRUE* or *FALSE* by circling your choice, and provide an explanation to justify. You will get *2 points* for each correct answer with correct explanations, and ***no penalty*** **(of negative points) for wrong answers**.

(1) An *E-R* diagram will translate uniquely to a relational schema.

(2) In an *E-R* diagram, an entity will translate to a table in the relation model, while a relationship will translate to a join between tables.

(3) We can consider relational algebra as a query language.

(4) The *basic* operators in relational algebra are $\pi, \sigma, \rho, \bowtie, \cup, -$.

(5) In SQL, we can only use aggregate functions where there is a Group-By clause.

(6) In rule-based optimization, a commonly used heuristic rule is to push projection down, to reduce the number of columns early on.

(7) In optimizing join queries, we choose to assume only left-deep join trees, because such trees are most efficient.

(8) When we program a database transaction, we can use the "Abort" command to rollback a transaction that cannot be successfully completed.

(9) Grouping (*i.e.*, group-by) must be processed in a two-pass algorithm.

(10) For logging, we prefer the UNDO+REDO scheme, so that we can be more flexible in when to write out dirty data to disk.

(11) One particular motivation for uncertain database research is that data may be inherently uncertain, *e.g.*, weather forecast gives probabilistic predictions.

Problem 2 (*16 points*) Short Answer Questions

For each of the following questions, write your answer in the given space. You will get *2 points* for each correct answer.

(1) Consider relations $R(a,\ b)$ and $S(a,\ c)$, for the following query. Would a hash index on $S.a$ help in query processing? Briefly explain.

SELECT a FROM R, S WHERE b<10 and $R.a = S.a$

(2) For the above SQL query, suggest an efficient query plan, and explain why it is a good plan.

(3) Consider the following two tables $R(A,\ B,\ C,\ D)$ and $S(D,\ E)$. Identify one functional dependency (that is not violated in this example) that involves three attributes.

Table R:

A	B	C	D
1	3	2	2
1	3	2	4
3	1	3	6
3	1	1	6

Table S

D	E
1	3
2	2
4	1
6	2

(Problem 2, cont.)

(4) For the above tables, what is the result of this query: $\sigma_{A<2}(R \bowtie S)$? Show the result table.

(5) Name one difference of query semantics in an uncertain database as compared to deterministic SQL queries.

(6) Give one advantage of UNDO-only logging over REDO-only logging.

(7) Given a memory space of 1000 blocks, for hashing a relation of 10^6 blocks, what is the minimal size (as the number of blocks) of a bucket?

(8) In tuning system performance, we should use Hints sparingly. Why?

Problem 3 (*11 points*) Query Languages

The following questions refer to the database schema below:

Professor(pid, *pname, dept*), *Student*(sid, *sname, age*), *Course*(cid, *pid, cname, time, location*), *Grade*(cid, sid, *score*).

(a) Write a query, in *relational algebra*, to return the names of professors who work in "CS" department and teach at least two courses. (*4 points*)

(b) Write an SQL query, to return the name, and the average score of each student, ordered by the average score (descending) *(Note: If one student is registered for two courses and the scores are* 75 *and* 85*, respectively, his average score is* 80*)*. (*4 points*)

(c) The professor of CS411 would prefer to enhance the score of young students (younger than 20). Write a statement that scales these students' score by 1.07 (*3 points*)

Problem 4 (*10 points*) Indexing

Consider the following B+-Tree of order 4 (*i.e.*, $n = 4$, each index node can hold at most n keys and $n + 1$ pointers) shown in Figure 1.

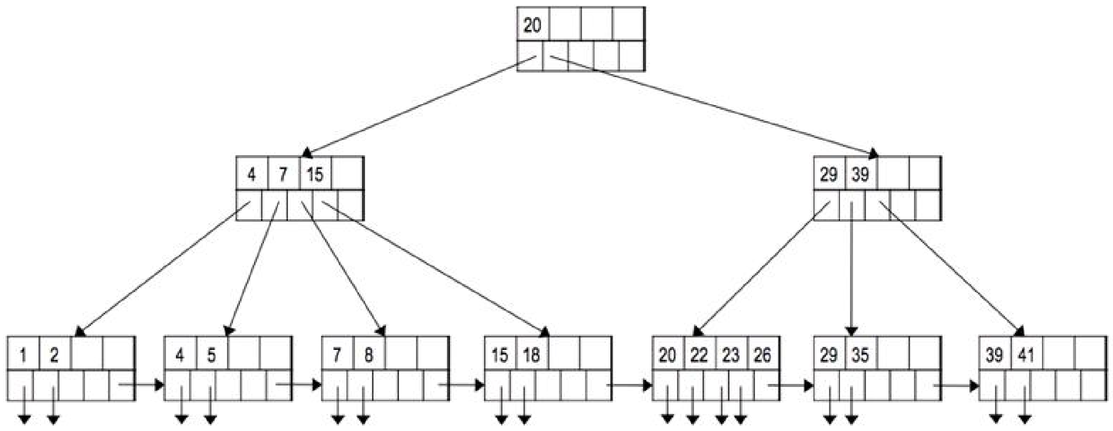

Figure 1: B+ tree.

(a) Show the steps in looking up all records in the range 16 to 30. (*2 points*)

(b) Show the resulting tree after deleting key 39 from the *original* tree. (*4 points*)

(c) Show the resulting tree after inserting key 24 into the *original* tree. (*4 points*)

Problem 5 (*12 points*) Query Processing

Consider two relations $R(x, y)$ and $S(x, z)$ with the following statistics:
$T(R) = 5{,}000$, $B(R) = 500$ (each block contains 10 tuples),
$V(R, x) = 100$ (number of distinct values of attribute x in R),
$T(S) = 8{,}000$, $B(S) = 1{,}000$ (each block contains 8 tuples),
$V(S, x) = 200$ (number of distinct values of attribute x in S),
$V(S, z) = 50$ (number of distinct values of attribute z in S) and $z < 100$.
Assume the memory buffer has 101 blocks (M = 101)

(a) Estimate the number of tuples in $R \bowtie_{z<50} S$ (*3 points*)

(b) Briefly describe the algorithm and compute the cost of join using a block nested-loop join. (*5 points*)

(c) Now suppose we want to use sort-based join. Assuming that the number of tuples with the same x value is not large, we can use the more efficient sort-based join approach below:

(1) Create sorted sublists of size M, using x as the sort key, for both R and S.

(2) Bring the first block of each sublist into the memory buffer.

(3) Repeatedly find the least x-value, and output the join of all tuples from R with all tuples from S that share this common x-value. If the buffer for one of the sublists is exhausted, then replenish it from disk.

Please estimate the cost of applying this sort-join on R and S. Also, please state the requirement on the number of memory blocks M (note: considering the maximum number of blocks K for holding tuples of R and S with the same x value.) (*4 points*)

Problem 6 (*15 points*) Query Optimization

(a) Judge the following two rules to be true/false, if true, give a short proof. If false, give a counterexample. (*6 points*)

(a.1) Projection can be pushed below set union. (*3 points*)

(a.2) Duplicate elimination can be pushed below projection. (*3 points*)

(b) Consider the physical query plans for the following expression:

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

We have the following assumptions:

(1) $B(R) = 5,000$, $B(S) = B(U) = 12,000$

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

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

(4) The memory buffer has 101 blocks ($M = 101$)

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 the table on the next page. 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. (*9 points*)

Problem 7 (*14 points*) Failure Recovery

Consider the following log sequence.

Log ID	Log
1	⟨START $T1$⟩
2	⟨$T1$, A, 10⟩
3	⟨START $T2$⟩
4	⟨$T2$, B, 15⟩
5	⟨$T2$, C, 20⟩
6	⟨START $T3$⟩
7	⟨$T1$, B, 25⟩
8	⟨COMMIT $T1$⟩
9	⟨$T3$, C, 30⟩
10	⟨COMMIT $T2$⟩
11	⟨START $T4$⟩
12	⟨$T3$, D, 35⟩
13	⟨$T4$, A, 40⟩
14	⟨COMMIT $T3$⟩
15	⟨$T4$, C, 45⟩
16	⟨COMMIT $T4$⟩
17	⟨START $T5$⟩

Note: For the questions (a)-(d), assume the given log sequence is a *UNDO* log.

(a) Please briefly explain the meaning of the record: $\langle T1,\ A,\ 10\rangle$ *(logID 2)* (*1 points*)

(b) When is the latest time for transaction $T1$, $T2$ that "dirty data" can be flushed onto disk (i.e. the time Output(X) for data X can be performed)? (*2 points*)

For $T1$: Output(s) ________ before logID _____
For $T2$: Output(s) ________ before logID _____

(c) Suppose we want to start checkpointing right after logID 6. In the space below, *where* and *what* the start checkpointing record would look like. Then, indicate *where* and *what* the earliest end checkpoint record would look like. (*2 points*)

(d) Continue from (c). Suppose the system crashes right after logID 13. Show which transactions/actions (*e.g.*: $\langle T1,\ A,\ 10\rangle$) need to be undone *in the correct order.* (*3 points*)

Note: For the questions (e)-(g), assume the given log sequence is a *REDO* log.

(e) Please briefly explain the meaning of the record: $\langle T3, D, 35\rangle$ *(logID 12)* (*1 points*)

(f) When is the earliest time for transaction $T3$, $T4$ that "dirty data" can be flushed onto disk (i.e. the time Output(X) for data X can be performed)? (*2 points*)

For $T3$: Output(s) ________ before logID _____
For $T4$: Output(s) ________ before logID _____

(g) Suppose we set the checkpointing in the following way (the original IDs do not change):

Log ID	Log
1	⟨START $T1$⟩
2	⟨$T1$, A, 10⟩
3	⟨START $T2$⟩
4	⟨$T2$, B, 15⟩
5	⟨$T2$, C, 20⟩
6	⟨START $T3$⟩
7	⟨$T1$, B, 25⟩
8	⟨COMMIT $T1$⟩
9	⟨$T3$, C, 30⟩
⟶ ⟨START CKPT, $T2$, $T3$⟩	
10	⟨COMMIT $T2$⟩
11	⟨START $T4$⟩
12	⟨$T3$, D, 35⟩
13	⟨$T4$, A, 40⟩
⟶ ⟨END CKPT⟩	
14	⟨COMMIT $T3$⟩
15	⟨$T4$, C, 45⟩
16	⟨COMMIT $T4$⟩
17	⟨START $T5$⟩

Now the system crashes right after logID 14 (the end checkpoint has been written out to disk). Show which transactions/actions (*e.g.*: ⟨$T1$, A, 10⟩) need to be redone *in the correct order*. (*3 points*)