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) *False*

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

⇒ *Explain*: It can be translated into different relational schemas, depending on different merging strategies, as well as different subclassing methods.

(2) *False*

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.

⇒ *Explain*: A relationship in ER diagram normally would also translate into a table in the relation model. For many-to-one relationships, they can be absorbed into the table for entity.

(3) *True*

We can consider relational algebra as a query language.

⇒ *Explain*: It is a query language, although it is not declarative (SQL is declarative).

(4) *False*

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

⇒ *Explain*: $\bowtie$is not a basic operator. Instead, Cartesian product $\times$ is a basic operator.

(5) *False*

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

⇒ *Explain*: We can use aggregate directly in Select clause without a Group-By clause.

(6) *True*

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

⇒ *Explain*: It is true. Another heuristic is to push selection down to reduce the number of tuples.

(7) *False*

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

⇒ *Explain*: Not necessarily. Bush trees can sometimes be more efficient. We use left-deep only for the easiness of plan selection.

(8) *True*

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

⇒ *Explain*: True. "Abort" would rollback the operations to make it as if the transaction did not take place.

(9) *False*

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

⇒ *Explain*: False. It could be processed in one-pass algorithm as well, if there is enough resource.

(10) *True*

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

⇒ *Explain*: True. UNDO plus REDO gives us more flexibility.

(11) *True*

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

⇒ *Explain*: True. This reflects the real world, where many things are uncertain.

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) Answer: Yes. We can use the hash index to speed up the join between R and S. For each value of a in R, we can directly use the hash index to see if there is a match in S.

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) Answer: We can push the selection down to evaluate b<10 condition first. This will help to reduce many tuples, and thus the join can be done with fewer tuples.

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

(3) Answer: Many FDs are possible, for instance: $AD \rightarrow B$, $AC \rightarrow B$, ...

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) Answer: There are two tuples $< 1, 3, 2, 2, 2 >$ and $< 1, 3, 2, 4, 1 >$ for schema (A,B,C,D,E).

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

(5) Answer: There are many differences. For instance, uncertain database adopts the possible worlds semantics, where a probabilistic database is viewed as a set of possible instances (worlds) associated with their probabilities.

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

(6) Answer: In UNDO-only logging, data is flushed immediately after a transaction finished, thus it requires less buffer to keep the data. For more details on comparing the two logging schemes, refer to page 869 of the textbook.

Give one advantage of UNDO-only logging over REDO-only logging.

(7) Answer: $10^6/1000 = 1000$

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) Answer: Since the database may change and evolve, and in such situations Hints may do harm to the system performance.

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*)

Solution:

$\pi_{Professor.pname}((Course_1 \bowtie_{C1} Course_2) \bowtie_{C2} Professor)$
where, $Course_1$ and $Course_2$ are both type of $Course$
$C1 = (Course_1.cid <> Course_2.cid$ AND $Course_1.pid = Course_2.pid)$
$C2 = (Course_1.pid = Professor.pid$ AND $Professor.dept =$"CS")

(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*)

Solution:

select sname, AVG(score)
from Student, Result
where Student.sid = Result.sid
group by sname
order by AVG(score) DESC

(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*)

Solution:

update Student, Result
set score = score $*1.07$
where Student.sid = Result.sid AND Student.age $<$ 20 AND Result.cid = 'CS411'

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.

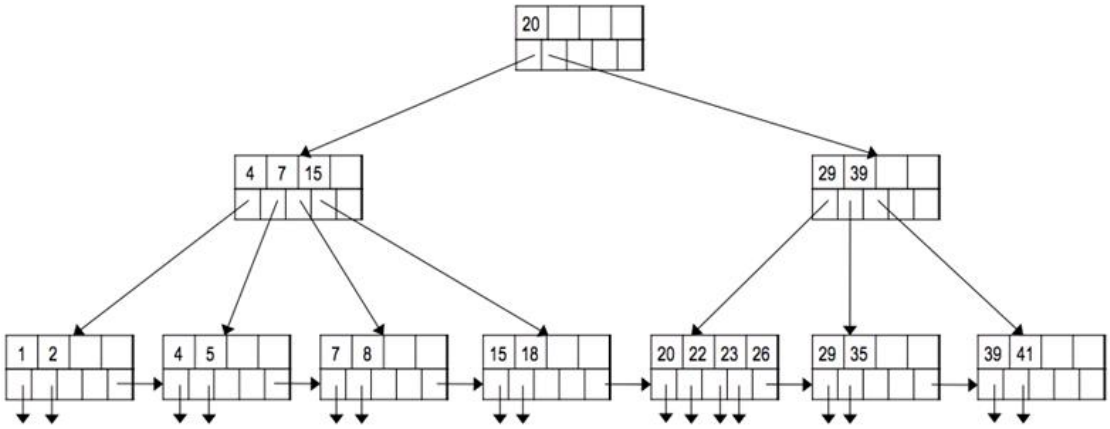

Figure 1: B+ tree.

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

Solution: $16 < 20$ go to 1st pointer, $16 > 15$ go to 4th pointer, get record with key 18, follow chain to 6th leaf getting records 20, 22, 23, 26, 29, and finally stopping at 25 as $30 < 35$.

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

Solution:

41 moves to join leaf [29,35]. Rightmost node in level 2 now only points to 2 leaves, problematic. Move [15,18] leaf over to right. Leftmost node in level 2 is now with 2 keys [4,7] and 3 pointers. Rightmost node in level 2 is now with 2 keys [20,29] and 3 pointers. Finally, fix the root node to be not 20 but 15.

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

Solution:

The 5th leaf must be split, there are 2 options.
Option 1. Split into [20, 22, 23] and [24, 26]. Rightmost node in level 2 adds a key and a pointer, and now has 3 keys, 24, 29, 39 and 4 pointers.
Option 2. Split into [20, 22] and [23, 24, 26]. Rightmost node in level 2 adds a key and a pointer, and now has 3 keys, 23, 29, 39 and 4 pointers.

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*)

Solution:

To Join:
T(R)T(S)/max(V (R, x), V (S, x)) = 5, 000 * 8, 000/200 = 200, 000

To then filter on $z < 50$ condition, use the heuristic of dividing by 3.
Final answer: $200,000/3 = 66,667$ (rounded)

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

Solution:

(1). Use 100 blocks of memory to buffer R in 100-block chunks
(2). For each chuck (50 chucks totally), read these blocks into main-memory buffers and build the search structure.
(3). Go through each block of S, reading each one into the last block of memory.
(4). Compare and output the joined tuples.
Cost: B(R) + B(R)B(S)/(M-1) = 5, 000 + 5, 000 * 8, 000/100 = 45, 000

(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*)

Solution:

Cost: 3B(R) + 3B(S) = 4,500.
We have 5 subsorted lists for relations R and 10 lists for S, 15 subsorted lists in total. Therefore, M $\geq$ K + 15 must hold.

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*)

Solution:
False.
Assume two relations: R(A,B)=(1,2) and S(A,B)=(1,4), we have:
$\pi_A(R \cup_S S)$ = (1,1), and $\pi_A R \cup_S \pi_A S$ = (1).

Solution is for bags. If sets were assumed, problem was graded accordingly.

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

Solution:
False.
Assume relation T(A,B)=((1,2),(1,4)), we have:
$\delta(\pi_A T)$=(1), and $\pi_A(\delta(T))$=(1,1)

Solution is for bags. If sets were assumed, problem was graded accordingly.

(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*)

Solution:

Correction to solution: ranges should be $k < 50; 50 <= k < 5,000; k > 5,000$. Also, third row third column entry should read 50-bucket, two-pass, not 70-bucket, two-pass

Range of k	Pipeline/Materialize	Algorithms for final join	Total disk I/O's
$k \leq 69$	Pipeline	one-pass	63, 000
$70 \leq k \leq 7000$	Pipeline	70-bucket, two-pass	87, 000 + 2k
$k > 7000$	Materialize	100-bucket, two-pass	87, 000 + 4k

Problem 7 (*14 points*) Failure Recovery

Consider the following log sequence.

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

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*)

Solution:

Transaction T1 changed elt A. A's old value was 10.

(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 _____

Solution:
For $T1$: Output A, B before Log ID 8
For $T2$: Output A, B before Log ID 10

(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*)
Solution:
Between Log 6 and 7, $< STARTCKPT(T1, T2, T3) >$
Between Log 14 and 15, $< ENDCKPT >$

(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*)
Solution:
The undo transactions in sequence: $< T4, A, 40 >, < T3, D, 35 >, < T3, C, 30 >$

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*)

Solution:

Transaction T3 changed elt D. D's new value is 35.

(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 _____

Solution:
For $T3$: Output C, D before Log ID 15
For $T4$: Output A, C before Log ID 17

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

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

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

Solution:
The redo transactions in sequence: $< T2, B, 15 >, < T2, C, 20 >, < T3, C, 30 >, < T3, D, 35 >$