CS411 Database Systems

Fall 2004, Prof. Chang

Department of Computer Science
University of Illinois at Urbana-Champaign

Final Examination

December 17, 2004
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 **11** 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	12	18	8	9	10	10	19	14			100
Score											
Grader											

Problem 1 (*12 points*) Misc. Concepts

For each of the following statements, indicate whether it is *TRUE* or *FALSE* by circling your choice. You will get *1 point* for each correct answer, *-0.5 point* for each incorrect answer, and *0 point* for each answer left blank.

(1) *True* *False*

The LRU buffer replacement algorithm should not be used for certain query operations, such as nested-loop join.

(2) *True* *False*

For relation $R(A,\ B,\ C,\ D)$, if $AB \rightarrow C$ and $C \rightarrow D$, then $\{A,\ B\}$ is a key.

(3) *True* *False*

To process join operation $R \bowtie S$, we can choose any join methods: nested-loop, index, sort-merge, hash, or hybrid– The only difference is their costs.

(4) *True* *False*

Given an SQL query, there are often multiple ways of writing it in relational algebra.

(5) *True* *False*

For the "ACID" properties of transactions, the "I" stands for *idempotency*– that is, the multiple executions of the same transaction should result in the same correct effect.

(6) *True* *False*

If a schema is in 3NF, then it must also be in BCNF.

(7) *True* *False*

B-Tree was invented by Dr. Rudolf Bayer, an alumnus of the computer science department of UIUC.

(8) *True* *False*

If a pointer is very likely to be followed for many times, then it will pay off to perform *automatic swizzling*.

(9) *True* *False*

An RDBMS performs its own buffer management, for not only efficiency but also *correctness*– because it needs to impose certain ordering in buffer replacement for transaction processing.

(10) *True* *False*

For disk latency, *seek time* refers to the time for the head to find the desired sector– *i.e.*, for the disk to rotate so the first of the sectors containing the desired data reaches the head.

(11) *True* *False*

From $A \rightarrow B$, we can derive $AC \rightarrow BC$, which further leads to $A \rightarrow BC$.

(12) *True* *False*

For building a database application, we often write the application mainly in some "host language," which interacts with a database for manipulating data. The different ways of operations between the host language and the underlying database, as many have observed, are called *impedance mismatch*.

Problem 2 (*18 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: _____
Consider relation $R(A,\ B)$ and $S(B,\ C)$, where $T(R) = 200$ and $T(S) = 100$, and B is a key for R. What is the estimate for $T(R \bowtie S)$?

(2) Answer: _____
Consider relation $R(A,\ B,\ C,\ D)$ with $A \rightarrow B$ and $C \rightarrow D$. What is the BCNF decomposition?

(3) Answer: _____
Write the following query in relational algebra, for relations $R(a,\ b)$ and $S(c,\ d)$:
select a, d **from** R, S **where** $R.a > 10$ and $R.b = S.c$.

(4) Answer: _____
Write the following relation algebra expression in SQL, for relations $R(a,\ b,\ c)$ and $S(a,\ b,\ e)$: $\pi_{a,\,b}(\sigma_{a\,>\,5}R) - \pi_{a,\,b}S$

(5) Answer: _____
Consider a B+-tree with $n = 100$ over a relation with 1 million records. What is the number of nodes in the tree that we have to examine when searching for a record?

(6) Answer: _____
Pointers (or addresses) are often used in databases. Give one example of when pointers are used.

(7) Answer: _____
For relation $R(A,\ B,\ C)$, suppose $AB \rightarrow C$ and $C \rightarrow B$. List all normal forms (if any) that R is in. You only need to consider 3NF, 4NF, and BCNF.

(8) Answer: _____
For relation *Student*(*sid, name, dept*), suppose *dept* can be one of {cs, ee, ce, me, chemistry}. If $T(Student) = 1000$, then what is the size estimate of $\sigma_{dept\,=\,cs}Student$?

(9) Answer: _____
Rewrite $R \bowtie_\theta S$ using only the *basic* relational algebra operators.

Problem 3 (*8 points*) Schema Decomposition

We perform decomposition to normalize an original schema to be of certain normal forms. For such a decomposition to be "equivalent" to the original schema, it is desirable to be *lossless*.

To study this concept, let's consider an original schema $R(A, B, C)$. Suppose we decompose R into $R1(A, B)$ and $R2(A, C)$.

(a) Is this decomposition always lossless? Answer *yes* or *no* and briefly explain why. (*4 points*)

(b) Give an example instance of R (*i.e.*, an example table with several tuples) and demonstrate its decomposition, to support your answer in (a). (*4 points*)

Problem 4 (*9 points*) Query Languages

Consider relation *Scores*(*name, exam, score*), which records the score of a student in an examination (either "midterm" or "final"); for example:

name	*exam*	*score*
Betty	midterm	85
Alex	midterm	57
Alex	final	90
Betty	final	78
...	...	...

(a) Write a query, in *relational algebra*, to return the final-exam score of Alex. (*2 points*)

(b) Write a query, in *relational algebra*, to return the names of those students who score higher in the final exam than in the midterm exam. (*3 points*)

(b) Write a query, in *SQL*, to return the "count" distribution of scores for the *midterm* exam, in descending order of score. That is, we want to list each score along with the number of students with that score, in the midterm exam. For example, the output may look like the following: (*4 points*)

score	*count*
85	6
82	5
78	8
...	...

Problem 5 (*10 points*) Indexing: B+tree

Consider constructing a B+tree of order 3 (*i.e.*, $n = 3$).

(a) Show the resulting tree after inserting keys 10, 20, 30, 40, 50, 60, 70, 80, 90, 100, in this order. (*4 points*)

(b) Is it possible, with the same set of keys, to construct a "shorter" tree– *i.e.*, one that has a smaller height? If *no*, explain why not. If *yes*, show an order of inserting the keys and the resulting tree. (*6 points*)

Problem 6 (*10 points*) Query Processing

We wish to join relations $R(a,\ b)$, $S(b,\ c)$, and $T(c,\ d)$, *i.e.*, to produce $R \bowtie S \bowtie T$. As our assumptions:

- Each relation holds B_r, B_s, B_t blocks of data respectively.
- The memory buffer for query processing has M blocks.
- R is already sorted by $R.b$.

(a) If we want to minimize memory requirement for processing this query, what is the minimal value of M? Describe a processing strategy that results in this minimal requirement. (*4 points*)

Note: In counting the memory requirement, as usual, we do not include the buffer space for writing the *final* output, *i.e.*, tuples of $R \bowtie S \bowtie T$. All other required space should be included.

(b) To contrast, suppose we want to process with the following strategy. What is the memory requirement M? That is, you will derive an inequality condition in terms of M, B_r, B_s, and B_t, under which the following procedure can be carried out. (*6 points*)

1. Perform the *first* phase of two-phase multiway merge sort on S. That is, as many times as necessary, we load the buffer with as many blocks from S as possible, sort the tuples in memory, and write out the sorted sublist. At the end of this step, S is stored as several sorted sublist (or "runs") on the disk. (We do *not* perform the second phase now.)
2. Read T entirely into the buffer, using as many blocks as necessary.
3. Merge R and the sorted sublists of S to produce $R \bowtie S$, and compare each of the resulting tuple with T (already in memory) to produce $(R \bowtie S) \bowtie T$. (As usual, any output tuple of the overall result is stored in an output buffer, which is not part of M.)

Problem 7 (*19 points*) Query Optimization

Consider the following query that joins *Student*(*sid, sname, sdept*), *Enrollment*(*sid, cid*), *Courses*(*cid, ctitle, iid*), *Instructor*(*iid, iname, iaddr*).

select *sname*, *ctitle*, *iname*
from *Student S*, *Enrollment E*, *Course C*, *Instructor I*
where join-conditions AND selection-conditions

In the **where**-clause, we have the following conditions:

- The join-conditions specify how the relations are joined. In this query, they are fixed to natural joins, *i.e.*: $S.sid = E.sid$ AND $E.cid = C.cid$ AND $C.iid = I.iid$.
- The selection-conditions are of the form c_1 AND c_2 AND $\cdots$ AND c_n, where each c_i is a selection condition on some relation, *e.g.*, $S.sdept =$ "CS" or $I.iaddr$="1234SC".

In this question, we are to consider that, although the join conditions remain the same, different scenarios with different selection conditions may need different join ordering. For our purpose, we consider the following join orders for processing $S \bowtie E \bowtie C \bowtie I$:

- *J1*: $((S \bowtie E) \bowtie C) \bowtie I$
- *J2*: $((I \bowtie C) \bowtie E) \bowtie S$
- *J3*: $(S \bowtie E) \bowtie (C \bowtie I)$

(a) We have only given three example orders in the above. If we want to consider *all* possibilities, how many different orders are there for processing $S \bowtie E \bowtie C \bowtie I$? (*6 points*)

Note, to simplify, let's assume that join orders are symmetric– *i.e.*, $A \bowtie B$ is equivalent to $B \bowtie A$. For instance, we consider $J1$ and $I \bowtie ((S \bowtie E) \bowtie C)$ as the same order.

(b) In practice, an optimizer often does not consider all possibilities. Suppose we only consider *left-deep* join ordering– Then, in the entire space just described in (a), how many join orders are left-deep? (*4 points*)

(c) Give an example scenario, for which $J3$ will clearly be the best choice. First, describe your scenario by specifying what the selection-conditions are, and explain why. (*5 points*) Second, give a *complete* query plan (in terms of relational algebra expression, or a query tree) for your query. (*4 points*)

Problem 8 (*14 points*) Failure Recovery

Consider the following *UNDO* logging.

Action ID	Action
1	$\langle$START $T1\rangle$
2	$\langle T1, A, 10\rangle$
3	$\langle$START $T2\rangle$
4	$\langle T1, B, 10\rangle$
5	$\langle$COMMIT $T1\rangle$
6	$\langle T2, B, 10\rangle$
7	$\langle$COMMIT $T2\rangle$
8	$\langle$START $T3\rangle$
9	$\langle T3, A, 10\rangle$
10	$\langle$START $T4\rangle$
11	$\langle T3, B, 20\rangle$
12	$\langle$COMMIT $T3\rangle$
13	$\langle T4, C, 10\rangle$
14	$\langle$START $T5\rangle$
15	$\langle$COMMIT $T4\rangle$
16	$\langle T5, D, 10\rangle$
17	$\langle$COMMIT $T5\rangle$

(a) We want to see when "dirty data" can be flushed to disk– *i.e.*, what time to perform Output(X) for data X (*e.g.*, Output(A), Output(B), *etc.*). Suppose we want to perform such "output" *as late as possible*. Insert these outputs on the figure to suggest their timing. (*3 points*)

(b) Suppose we want to start checkpointing right after Action 4: First, show on the figure this start checkpointing log record. (*2 points*) Then, show on the figure the end checkpointing log record. (*3 points*)

(c) Continue from (b). Suppose the system crashes after Action 15. How far back in the log must we look to find all actions that need to be undone? (*3 points*)

(d) Now, suppose this system is actually a *redo* log. To contrast with (a), if you *still* want to perform "output" (*e.g.*, Output(A), Output(B), *etc.*) *as late as possible*. When should such output be done? (*3 points*) You can show on the figure for the timing, but you should separate and distinguish your answer from that of (a).