CS411 Database Systems

Fall 2005, Prof. Chang

Department of Computer Science
University of Illinois at Urbana-Champaign

Final Examination

December 14, 2005
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.*
- Problems that are related to homework (*e.g.*, in terms of concepts covered) are marked with [*HW*].

Problem	1	2	3	4	5	6	7	8			Total
Points	12	18	10	10	10	10	15	15			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*

Transaction management was one of the concepts Ted Codd created in his seminal work of defining the relational model.

(2) *True* *False*

A relation R in 3NF is also in 4NF.

(3) *True* *False*

The second and higher levels must be sparse in an index of multiple levels on sequential files.

(4) *True* *False*

Hash index is efficient in answering range queries, such as "finding products with price higher than 100".

(5) *True* *False*

The order of insertions into a B+ tree will affect the tree's final structure at the end.

(6) *True* *False*

The time needed to access a page on disk is comprised of two components: disk seek time and data transfer time.

(7) *True* *False*

If possible, we should always use a bushy join tree instead of a left-deep join tree, because the former is more efficient than the latter.

(8) *True* *False*

In SQL, without GROUP BY, we cannot use HAVING.

(9) *True* *False*

We can optimize query plans by pushing a selection down an expression tree, but not by moving a selection up the tree.

(10) *True* *False*

Query optimization is a major concept that enables declarative SQL queries, because it automatically generates query plans, without having users to write procedural queries.

(11) *True* *False*

For buffer management, a policy like LRU may be *inefficient* for transaction processing in RDBMS, but it will not affect *correctness*.

(12) *True* *False*

Pointer swizzling refers to the techniques that convert disk pointers to memory addresses.

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: _____
Write the following query in relational algebra, for relations $R(a,\ b)$ and $S(a,\ b)$:
(SELECT a FROM R WHERE b<10) EXCEPT (SELECT a FROM S WHERE b>5).

(2) Answer: _____
Write the following relation algebra expression in SQL, for relations $R(a,\ b)$ and $S(a,\ c)$: $\pi_{b,c}((\sigma_{b>10}R)\bowtie(\sigma_{c>5}S))$

(3) Answer: _____
Suppose one block can hold 10 pointers or 20 data tuples. Given a relation of 10000 tuples, how many blocks do we need for a 2-level index of this relation (given that the first level is sparse).

(4) Answer: _____
Consider relation R with five attributes ABCDE, and the following functional dependencies: $A \rightarrow B, BC \rightarrow D, D \rightarrow E$. Give the complete closure for {AC}.

(5) Answer: _____
A professor record consists of one fixed length field **SSN** and one variable length field **name**. The records are augmented by an additional repeating field that represents the courses a professor teaches. Each course is of fixed length. Show the layout of a professor record if the variable length field and repeating courses are kept within the record itself.

(6) Answer: _____
Transactions should be "ACID". What does "A" stand for?

(7) Answer: _____
In one sentence, explain the *principle of optimality*, which governs the correctness of dynamic programming.

(8) Answer: _____
Consider two relations R(a,b,c) and S(a,d,e). T(R)=200, T(S)=100. R.a is a foreign key referencing S.a and S.a is the primary key of S. What is the estimated size of R⋈S?

(9) Answer: _____
Among the four properties "ACID" achieved by transaction management, which properties are guaranteed by failure recovery?

Problem 3 (*10 points*) Schema Decomposition [*HW*]

Consider a relation R with five attributes ABCDE. The following dependencies are given: $A \rightarrow B$, $BC \rightarrow E$, $ED \rightarrow A$.

(a) List all keys for R. (3 points)

(b) Is R in 3NF? Briefly explain why. (3 points)

(c) Is R in BCNF? If yes, please explain why. Otherwise, decompose R into relations that are in BCNF. (4 points)

Problem 4 (*10 points*) Query Languages [*HW*]

The following questions refer to the database schema below: *Product*(*pid, price*), *Order*(*oid, cid, pid, quantity*), *Customer*(*cid, name, age*).

(a) Write a query, in *relational algebra*, to return the names of senior customers (older than 60). (2 points)

(b) Write a query, in *relational algebra*, to return the names of customers who didn't order any product. (3 points)

(c) Write a SQL query, to list each product along with its total ordered quantities, in descending order of the amount. (5 points)

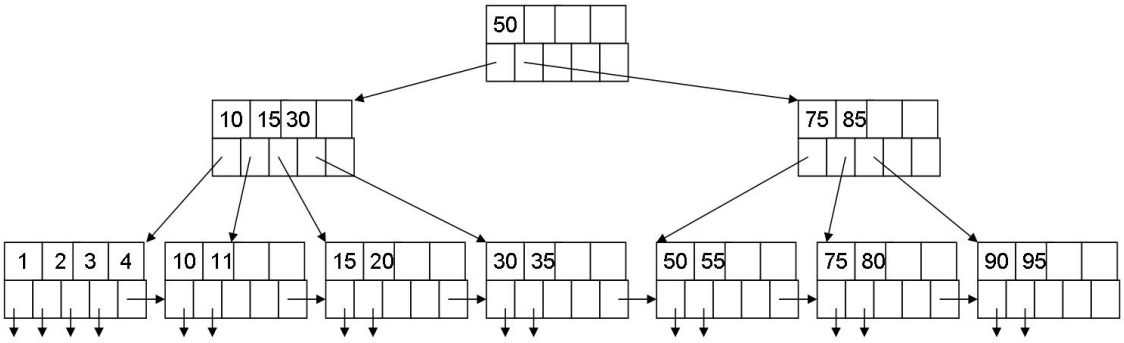

Figure 1: B+tree

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

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

(a) Based on the tree in Figure 1, show the resulting tree after inserting key 5.

(b) Based on the tree in Figure 1, show the resulting tree after deleting key 90.

(c) Based on the tree in Figure 1, show the steps in executing the following operation: Lookup all records in the range 40 to 100 (including 40 and 100).

Problem 6 (*10 points*) Query Processing [*HW*]

Consider joining two relations $R(x,y)$ and $S(x,z)$ on their common attribute x. The size of relation R is 150 blocks and the size of relation S is 100 blocks. Attribute x has 50 different values and is evenly distributed in both R and S. Suppose that both relations are not sorted by attribute x.

(a) Suppose the memory buffer has 15 blocks, compute the cost of join using a block-nested loop join. (3 points)

(b) Suppose the memory buffer has 15 blocks, compute the cost of join using a sort-merge join. (3 points)

(c) Can you estimate the size of the output relation by joining R and S on x. If yes, estimate it. If no, explain why. (4 points)

Problem 7 (*15 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".

(a) Use this example, explain why join ordering is important for minimizing the cost of a query plan. Give an example scenario to support your explanation. (*5 points*)

(b) Consider dynamic programming for generating the optimal join order. Without actually working through the whole process, calculate how many subqueries each iteration needs to consider. (Note: we are asking for logical queries, **NOT** physical plans) (*5 points*)

(c) Now, if in the dynamic programming process, you do not want to consider cartesian products at all. With this assumption, give your answers for (b) again here. (*5 points*)

Problem 8 (*15 points*) Failure Recovery

Consider the following log sequence.

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

(a) Assume the given log sequence is an *undo* log. Suppose we want to start checkpointing right after logID 11. In the space below, indicate where and what the start checkpointing record would look like. Then, indicate where and what the earliest end checkpoint record would look like.

(b) Continue from (a). Suppose the system crashes right after logID 16. What is the portion of the log we would need to inspect and which transactions need to be undone?

Note: For the questions below, assume the given log sequence is a *redo* log.

(c) Suppose we want to start checkpointing right after logID 4. In the space below, indicate where and what the start checkpointing record would look like. Also, indicate where and what the earliest end checkpoint record would look like.

(d) Continue from (c). Suppose the system crashes right after logID 16. If ⟨END CKPT⟩ is written onto the log, indicate the portion of the log we would need to inspect and which transactions need to be redone.

(e) Now, suppose we start checkpointing right after logID 4 and the system crashes right after logID 6. If ⟨END CKPT⟩ is *not* written onto the log, indicate the portion of the log we would need to inspect and which transactions need to be redone.