Computer Science 425
Spring 2001
Take-home Final Exam
Out: 5:00pm Tuesday April 29, 2003
Due: 5PM SHARP Friday, May 2, 2003

Instructions: This exam must be entirely your own work. Do not consult with anyone else regarding this exam. If you have questions about what is being asked, contact Professor LaPaugh. You are allowed to use the following reference materials: You may use your personal notes and problem set submissions, *Database Management Systems* by Ramakrishnan and Gehrke, the solutions to odd-numbered exercises provided by the authors at http://www.cs.wisc.edu/ dbbook, the alternate text *Database System Concepts* by Silberscharz, Korth and Sudarshan, and copies of any of the material on the Web site for the course (staying within the site
http://www.cs.princeton.edu/courses/archive/spring03/cs425/...). *No other materials are allowed.* Also you are *NOT allowed to use online database interfaces.*

There are 6 problems, with point values indicated, totaling 100 points. *Be sure to give explanations for your answers.*

Problem 1 (5 points)
Consider the dynamic programming algorithm for finding the optimal left-deep plan for the evaluation of a sequence of JOINs. Pipelining the output of one JOIN to the input of the next requires that enough buffers exist to allow the result of the first JOIN to be computed and used as input to the algorithm evaluating the second JOIN without writing to disk. If the first JOIN is evaluated with a block nested loop algorithm, can the output be pipelined to the second JOIN? If so, are there restrictions on the algorithm used to evaluate the second JOIN? Explain your answers.

Problem 2 (30 points)
This problem asks you to consider computing the cost of a JOIN in the middle of a left-deep plan for the evaluation of a sequence of JOINS. Suppose the dynamic program that calculates the optimal left-deep plan produces the following information about the optimal left-deep plan for an intermediate result I_j:

- I_j can be produced sorted on attribute a at cost 15,000 disk I/Os; output can be pipelined
- I_j can be produced sorted on attribute b at cost 12,000 disk I/Os; output can be pipelined
- I_j can be produced unsorted at cost 9,000 disk I/Os; output can be pipelined
- the estimated number of disk pages required to store I_j is 6,000
- the estimated number of tuples in I_j is 60,000

All costs *exclude* writing I_j to disk.

Suppose that R is a base relation of the database (a leaf of the left-deep tree). R contains 4000 tuples with 20 tuples per disk page. There is an Alternative 2, unclustered B+ tree on attribute a of R and an Alternative 1, clustered B+ tree on attribute b. Both B+ trees are of order 40. Attribute b is the primary key of R, and there are are 500 distinct search key values in the B+ tree on attribute a.

What JOIN evaluation algorithms does the dynamic program consider to form a plan for evaluating the EQUALITY JOIN of I_j and R on shared attribute a? What is the disk I/O cost of each algorithm? What is the size estimate for the resulting relation I_{j+1} in disk pages and in tuples? What information does the dynamic program remember about the evaluation of this EQUALITY JOIN?

Problem 3 (15 points)
Let T1, T2, and T3 be three transactions and A, B, C and D be database objects. Denote the read of an object X as R(X) and the write of object X as W(X). Denote the commit of a transaction by C. Consider the following sequence of actions, listed in the order they are submitted for execution:

```
T1:R(A), T1:R(B), T2:R(D), T2:W(D), T3:R(A), T3:R(B), T3:R(C),
T2:R(C), T2:W(C), T3:W(B), T1:R(D), T1:C, T2:C, T3:C
```

Part a (5 points) Add **conservative** 2-phase locking to this sequence of actions and show the actual order of execution. Show when locks are requested, when they are granted, when they are released and when a transaction blocks waiting for a lock. Distinguish between shared locks and exclusive locks. When a transaction blocks waiting for a lock, assume execution proceeds with the the next action in the list above that belongs to an unblocked transaction.

Part b (5 points) Add **strict** but **not conservative** 2-phase locking to the original sequence of actions and show the actual order of execution. Assume there is **no** deadlock prevention. Again, show when locks are requested, when they are granted, when they are released and when a transaction blocks waiting for a lock. Distinguish between shared locks and exclusive locks. When a transaction blocks waiting for a lock, assume execution proceeds with the the next action in the list above that belongs to an unblocked transaction.

Part c (5 points) Repeat Part b but with the wait-die policy for deadlock prevention.

Problem 4 (20 points)
Consider the following sequence of actions occuring as four transactions T1, T2, T3 and T4 execute on a set of shared pages under the ARIES recovery manager protocal:

```
T1 writes page 1
T2 writes page 2
T3 writes page 1
T1 writes page 2
T2 writes page 1
begin checkpoint
end checkpoint
T1 writes page 3
T3 writes page 2
T1 writes page 4
T1 begins commit
T1 ends commit
T2 begins commit
T3 writes page 4
T4 writes page 4
T4 writes page 1
```

Part a (5 points) Write the transaction log for the sequence of actions above. Be sure to include prevLSN values and the contents of the transaction table and the dirty page table saved at the checkpoint.

Part b (8 points) Suppose the system crashes after your log of Part a is flushed to stable storage. Execute the ARIES recovery phases, showing all actions.

Part c (7 points) Suppose the system crashes again during recovery from the first crash after exactly one transaction has been aborted. (The log has been flushed to stable storage and the END for this aborted transaction has been written.) Execute the ARIES recovery phases after this second crash, again showing all actions.

Problem 5 (5 points)
Suppose the number of pages in the buffer pool for database pages of a database management system is cut in half. How, if at all, does this affect the behavior of ARIES logging and recovery for this system? Be sure to explain the reasoning behind your answer.

Problem 6 (25 points)
Consider a relational schema R with 5 attributes A, B, C, D, and E. Answer the following questions given the functional dependences

```
A->ABCDE
AB->D
BCD->E
```

Part a (5 points) For each of the 3 functional dependences above, say whether the dependency satisfies the conditions of BCNF and of 3NF. If it does satisfy the conditions of a normal form, state precisely which condition for the normal form it satsifies (e.g. AB→B would be a "trivial" functional dependence satisfying both BCNF and 3NF).

Part b (5 points) What is the lossless-join decomposition of R guided by (or, equivalently, based on) functional dependency BCD→E? Is this decomposition dependency preserving? Is the resulting set of relations in BCNF? in 3NF? Justify your answers by considering each functional dependency and showing that it does or does not satisfy the conditions of preserving dependency, of BCNF, and of 3NF.

Part c (5 points) **Add** functional dependency E→A to the original functional dependencies and repeat Part a.

Part d (5 points) Is there a lossless-join dependency-preserving BNCF decompostion of R given the functional dependencies of Part a and Part c together? Justify your answer by giving the specifics of the decomposition or showing why it cannot exist.

Part e (5 points) Do your answers to Parts c and d change if E→B is added **instead of** E→A? How and why or why not?