Name:_____ **Student ID:**_____

FINAL EXAM

CS 564 Introduction to Database Management Systems
Department of Computer Sciences
University of Wisconsin, Madison

Exam Rules:

1) Open book and notes, 60 minutes

2) Please write down your name and student ID number NOW.

3) Please wait until being told to start reading and working on the exam.

4) Calculators are allowed.

1. **(5 points)** What is the difference between logical and physical operators? Give an example of each.

2. (15 points) Given two relations R and S with size B(R) = 10000 and B(S) = 8000:
(a) **(5 points)** Suppose the amount of memory available is M = 2001. Compute the disk IO cost of a nested-loop join for R and S.

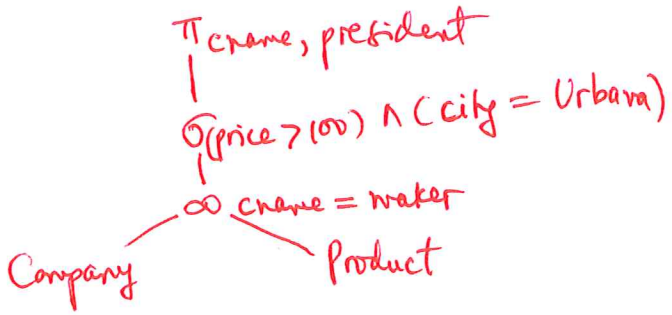
(b) **(10 points)** Suppose the amount of memory available is M=6000. Is it possible to perform a nested-loop join for R and S with cost no more than 25000 disk IO? Explain your answer.

3. (30 points; 6 points each) Consider the two relations: **Company(cname, city, president)**
Product(pname, maker, price).

Assume B(Company) = B(Product) = 10000 blocks. Assume further that each block can accommodate 5 records, and that the memory has 101 buffers.

Consider the SQL query: **SELECT Company.cname, Company.president**
FROM Company, Product
WHERE (cname=maker) AND [(price > 100) AND (city = "Urbana")]

(a) Draw a logical query plan tree for the above SQL query. This plan tree should join Company and Product, then perform a selection on price and city, then project out cname and president.



(b) For the logical query plan that you draw in Part a, consider the following physical query plan. First, perform a nested loop join of Company and Product, with Company being the outer relation. Next, pipeline the result of the join into the selection operation. Finally, pipeline the result of the selection operation into the projection operation. Compute the total cost of this physical query plan.

This is just the cost of the nested loop join, which is

$$\left\lceil \frac{10000}{101-2} \right\rceil * 10000 + 10000$$

(c) Assume that there is an index on attribute cname of relation Company, and that cname is a key for that relation. Can we replace the nested loop join operation in the physical query plan mentioned in Part b with an index based join operation? If so, which relation should be the outer relation in this index based join operation, and what should be the cost of this new physical query plan (which uses the index based join operation)?

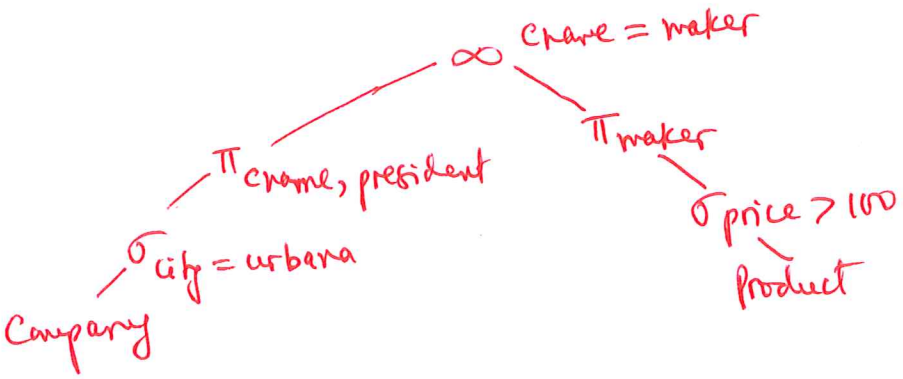
Product

Yes

cost is the cost of the index based join. Assuming a hash index, then for each product, we need on average 1.2 page I/O to use the index to look up the address of the page on which the matching Company tuple resides. So the total cost is

$10000 + (10000 \times 5) \times (1.2 + 1)$

(d) Consider the logical query plan that you have drawn in Part a. Draw a new logical query plan that pushes selections and projections down as far as possible.



(e) Outline at least one scenario where pushing selections down is not desirable. Explain why.

Suppose we join Product with Company with an index on cname. Then pushing city = Urbana to below the join is not OK, because we cannot use the index on Company to do the join.

4. (20 points) Compute the optimal plan for R ⋈ S ⋈ T ⋈ U using the dynamic programming algorithm, assuming the following:
B(R) = 500, B(S) = 200, B(T) = 600, B(U) = 400
The size of a join is estimated as: B(A ⋈ B) = 0.01 * B(A) * B(B)
The cost of a join is estimated to be the cost of the subplans plus the size of the intermediate results (the same as the cost model we have covered in the lecture).
Please draw the table for dynamic programming, to show how you compute the optimal plan.

5. (10 points) Briefly describe the ACID properties.

See notes.

6. (20 points) A database has four elements A, B, C, and D. It has just crashed. Assume we maintain an undo log whose content after the crash is

```
<start T1>
<T1,A,3>
<start T2>
<T2,B,1>
<start ckpt(T1,T2)>
<T1,A,4>
<start T3>
<commit T1>
<T2,B,2>
<T3,C,5>
<commit T2>
<end ckpt>
<start T4>
<T4,D,8>
<start ckpt(T3,T4)>
<start T5>
<T4,D,9>
<T5,A,10>
<commit T5>
<start T6>
<T6,B,12>
```

Assume further that after the crash the four elements have value: A=1, B=2, C=3, and D=4.
Recover the database. Clearly indicate which portion of the log you would need to inspect, and which transactions have to be executed again. Show the values of A, B, C, and D after the recovery.