Name:________________________ **Student ID:**________________________

FINAL EXAM

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

Exam Rules:

1) Closed book and notes, 90 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.

1. **Relational Operators** (25 Points)
    a. (5 Points) Briefly describe two alternatives for evaluating a projection with duplicate elimination, such as $\pi_A(R)$. Name one factor that the DBMS makes in choosing between these two options (one or two sentences).

    b. (10 Points) Suppose that you are given relations R with 4000 Pages and S with 20000 pages. Suppose the amount of memory available is M = 2001. Compute the disk IO cost of a nested-loop join for R and S.

c. (10 Points) Suppose that you are given relations R with 4000 Pages and S with 20000 pages. Suppose the amount of memory available is M = 2001. Compute the disk IO cost of a sort merge join for R and S.

This cost is the cost of sorting R + cost of sorting S
+ cost of merging R & S

The cost of sorting R is $\left(\left\lceil \log_{2000} \left\lceil \frac{4000}{2001} \right\rceil \right\rceil + 1\right) * 8000$

The cost of sorting S is $\left(\left\lceil \log_{2000} \left\lceil \frac{20000}{2001} \right\rceil \right\rceil + 1\right) * 40000$

The cost of merging the sorted R and S varies. In the worst-case scenario, it is 4000 * 20000. In the best-case scenario, it is (4000 + 20000). [This does not assume some optimization trick such as finding out that the smallest value of R is greater than the biggest value of S.]

2. (15 Points) **Normalization**

   a. (10 Points). Given the relation R(A,B,C,D,E) with functional dependencies A-> BC, C -> D, decompose it into BCNF. Please show the decomposition process.

b. (5 Points) Given R(A,B,C) with constraint A -> C, give an example of a decomposition of R that is lossless and a decomposition of R that is lossy. Does BCNF ensure the lossless join property (that is, the lossless decomposition)?

3. Relational algebra (15 points). Consider the two relations **Company(cname, city, president)**
Product(pname, maker, price)
where cname is the name of the company, and maker is the name of the company that makes the product.

(a) (10 points) Write a **relational algebra expression** that returns the names of all expensive products made by companies in Madison, where a product is expensive if there is at least one company that makes that product at the price of at least $100.

$$\Pi_{pname}\left(\sigma_{(price \geq 100) \wedge (city = Madison)} \; Product \bowtie_{maker = cname} Company\right)$$

(b) (5 points) Write a **relational algrebra expression** that returns the names of all products that are made by at least two different companies.

renaming operation

$$\Pi_{pname}\left(\sigma_{(maker1 \neq maker2)}\left(\rho_{\#2 = maker1, \; \#5 = maker2}\left(Product \bowtie_{(pname = pname)} Product\right)\right)\right)$$

or

$$\Pi_{\#1}\left(\sigma_{\#2 \neq \#5}\left(Product \bowtie_{(pname = pname)} Product\right)\right)$$

refers to position of attributes

4. Query optimization (30 points).

(a) (15 points) Suppose a SQL query must join three relations A, B, and C. A join order specifies how to join the three relations. For example, the join order (A join B) join C specifies that first we join A and B (with A being the outer join), and then we join the result with C in a pipeline fashion (with C being the inner join). Current RDBMSs consider only certain join orders, not all possible ones. For the above three relations, write down all join orders that current RDBMSs would consider. Explain briefly why they consider only those and not the remaining join orders.

(b) (15 points) Consider again the two relations **Company(cname, city, president)**
Product(pname, maker, price)

Suppose that the size of relation **Company** is 20,000 pages, with 100 tuples per page, and that the size of relation **Product** is 100,000 pages, with 200 tuples per page. Suppose further that each company makes no more than 20 products, and that **cname** is a key in **Company**. Finally, suppose that **price** is distributed uniformly between 1 and 100 in relation **Product**.

Consider the following plan: first we do a selection on **Product** with condition (price >= 20) AND (price <= 60), that is, price is between 20 and 60. We materialize the output of this selection on disk, then join this output with relation **Company** using a sort-merge join, on the condition (cname = maker). Then we perform on the fly the selection (city = Madison) followed by the projection of attributes cname and president.

Assume the memory size is 15,000 pages. Compute the cost of the above plan. Showing the formula for the cost (instead of computing the actual number) is fine.

Price is distributed uniformly between 1 and 100, so the output size after the selection on Product is $100\,000 * \frac{60-20}{100} = 40000$.

So the total cost of the plan is:

The cost of reading Product: 100 000

The cost of writing the output of $\sigma_{\text{price in } [20,60]}$ Product to disk: 40000

The cost of sorting these 40 000 pages

The cost of sorting Company.

The cost of merging the two sorted relations. Here, for each tuple in Company, it will join with at most 20 tuples in the output of Product (under a page of data). So the total cost of merging is 20000 + 40000.

5. Transaction management (15 points).
(a) (10 points) Briefly describe the ACID property of transaction. Who (the programmer or the system) is responsible for ensuring which property?

(b) (5 points) Briefly explain the idea of checkpointing. Does the database have to freeze (that is, stop accepting new transactions) during checkpointing?