# CS4604 Final Exam

December 10, 1999

Please enter the following information:

- **Name:**

- **ID:**

GOOD LUCK and Happy Y2K!
Do not write below this line

---

| Problem | Max Score | Score |
|---|---|---|
| 1 | 30 | |
| 2 | 10 | |
| 3 | 10 | |
| 4 | 10 | |
| 5 | 10 | |
| 6 | 10 | |
| 7 | 10 | |
| 8 | 4 | |
| 9 | 6 | |
| 10 | 15 (XC) | |
| Total | 100 | |

1. (30 points) Short answer questions:

   (a) Which of the three relational algebra operations ($\cup, \cap, -$) can be expressed in terms of the others?

   (b) In SQL, stored relations are called *tables* and virtual relations are called *views*. There is a third kind of relation, one that is neither. What is it referred to as?

   (c) Assume that the FD $A \rightarrow B$ holds in a relation $R(A, B)$. Express this FD as a constraint in relational algebra.

   (d) What does the *impedance mismatch* between database systems and programming languages refer to?

   (e) RDBMSs are *declarative*. What does this mean?

   (f) What does 3NF guarantee that BCNF and 4NF do not?

   (g) Since indices speedup queries, why isn't placing indices on all attributes of all relations a good idea?

   (h) If relation R has $m$ tuples and relation $S$ has $n$ tuples, what is the minimum number of tuples that $R \cap S$ can have (assuming set-theoretic semantics)?

   (i) Examples were given in class about interesting web queries that are not effectively addressed by current search engines. Mention any one of them.

   (j) What is the correct greeting when the instructor enters the class and wishes "Good Morning"?

2. (7 points) Design an E/R diagram for the following situation: Land masses are either islands or continents. All land masses have a name and an area; the name is the key. Some continents are connected to each other, e.g., Asia is connected to Europe. No island is connected to any other island or to a continent. Bodies of water are either oceans or straits. A body of water has a name (the key) and an area. Islands may be either located in one ocean (e.g., Hawaii is in the Pacific Ocean) or separated from a continent by a strait (e.g., Honshu is separated from Asia by the 'Sea of Japan'). You should not assume that a strait is adjacent to only one continent or to only one island.

   (3 points) After drawing your diagram, if there are any aspects of the above english description that cannot be modeled (or perhaps should be enforced by integrity constraints or other domain-specific constraints), point them out.

3. • (5 points) You are given the relational schema $R(A, B, C, D, E)$ with FDs $AB \to C$, $DE \to C$ and $B \to D$. Indicate all BCNF violations. Do not forget to consider FDs that are not in the given set, but follow from them. However, it is not necessary to give violations that have more than one attribute on the right hand side. Finally, decompose the relations, as necessary, into collections of relations that are in BCNF.

• (5 points) Suppose we have a relation $R(A, B, C)$ with MD $A \to\to B$. If we know that tuples $(a, b1, c1)$, $(a, b2, c2)$, $(a, b2, c3)$, $(a, b3, c3)$ are in $R$, what other tuples do we know must also be in $R$?

4. (10 points) Consider the relation `Student(ID,name,address,gpa)`. Write a query to find the name that appears most often in the `Student` relation. Your query should be written in relational algebra.

5. (a) (4 points) Consider the relation `Car(maker,model,year,fuel)`. Write a query in Datalog to find those `Car`s that are not both made in 1999 and manufactured by *GM*.

(b) (3 points) Does the following SQL query compute $R \cap (S \cup T)$? Why/Why not? Give reasons.

```
SELECT R.A
FROM R,S,T
WHERE R.A = S.A OR R.A = T.A;
```

(c) (3 points) Explain why a database tuning specialist might rewrite the query:

```
SELECT ...
FROM R1, R2, ...
WHERE A=B-C
```

to

```
SELECT ...
FROM R1, R2, ...
WHERE A+C=B
```

6. (10 points) Do the following two SQL queries produce the same output? If yes, explain why. If no, provide examples of the `S` and `P` relations. Just saying 'Yes' or 'No' is not worth any points.

```
SELECT *
FROM Students AS S
WHERE S.name IN
    (SELECT DISTINCT P.name
     FROM Players AS P);
```

```
SELECT S.*
FROM Students AS S, Players AS P
WHERE S.name = P.name;
```

7. (10 points) Consider the relational schema modeling people:

   `HeardOf(name1,name2)`

   which indicates that the person with name `name1` has heard of the person with name `name2`. `HeardOf` is not symmetric (for example, you have heard of the President but he might not know you). A *celebrity* is defined as somebody who has been heard of by everybody else other than himself/herself. Write a query in Datalog to find all the celebrities in the `HeardOf` relation. You may assume that the `HeardOf` relation does not list a person as being heard of by himself/herself.

8. (4 points) Consider the following schema:

```
Ships(name,class,launched)
Battles(name,date)
Outcomes(ship,battle,result)
```

where relation `Ships` records the name of a ship, the name of its class, and the year in which the ship was launched. Relation `Battles` gives the name and date of battles involving ships, and relation `Outcomes` gives the result (‘sunk’, ‘damaged’, or ‘ok’) for each ship in each battle. Write an SQL query to find those ships that ‘lived to fight another day’; i.e., they were damaged in one battle, but later fought in another. Datalog and relational algebra notations are not permissible.

9. (6 points) Consider the two relations $R(A, B)$ and $S(B, C)$. Assume that $R$ has the tuples: {(1,2),(1,3)}. Assume that $S$ has the tuples: {(2,3),(4,6)}. We know that $R \bowtie S$ is: {(1,2,3)}.

   SQL has an operator called `OUTERJOIN` that functions just like a join, except that it doesn't discard the tuples that don't match. It just pads them with `NULL` values for the extra columns. For example, `R OUTERJOIN S` would be: {(1,2,3),(1,3,NULL),(NULL,4,6)}. Notice that it only pads `NULL`s for those tuples of either relation that do not join with at least one tuple of the other relation.

   Express the outerjoin functionality in SQL without using the `OUTERJOIN` operator. You can use the above relational schema to illustrate your idea.

10. (Extra credit question: 15 points) Consider a bank database where we run the following SQL query:

    ```
    SELECT SUM(balance)
    FROM Accounts;
    ```

    which computes the sum of the balances of all accounts in the relation `Accounts`. Now, assume that we calculated the sum to be \$1,000,000.00. If we increase the account of some particular individual by \$100, it is pretty straightforward to find the new `SUM` without having to calculate it again laboriously by looking at all the tuples. The new `SUM` is just going to be the old `SUM` + the extra amount (\$100). So, this brings the final aggregate to \$1,000,100.00. Similarly, if the account of some particular individual is decreased, we can recompute the new `SUM` by a corresponding subtraction.

    Can we recompute the results of the MAX and AVG aggregate operations, in a similar incremental fashion, from the old values? If yes, how? If no, what 'extra information' (short of seeing the whole data again, that is) do you need to be able to do this effectively? Or can it not be done at all, even with more information?