CS145 Final Examination

Wednesday, December 10, 1997, 8:30–11:30 AM

Directions

The exam is *open book*; any written materials may be used. Answer all 11 questions on the exam paper itself. The total number of points is 180, i.e., one point per minute. When doing a design in any of the models we've discussed, make the design as succinct as possible, and avoid redundancy where possible. Likewise, when asked to write queries or other database operations, you should make them as succinct as possible. All questions about SQL refer to the SQL2 or SQL3 standard, *not* to Oracle SQL.

Do not forget to **sign the pledge** below.

I acknowledge and accept the honor code.

____

Print your name here: ____

1	2	3	4
5	6	7	8
9	10	11	

Problem 1: (20 points) *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 and (if you ignore the Suez Canal) to Africa. 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; the latter is a "strait" for our purposes). You should not assume that a strait is adjacent to only one continent or to only one island.

In the space below, design an Entity/Relationship diagram suitable for representing the above information. Do not forget to indicate keys and the multiplicity of relationships.

Problem 2: (30 points) In this and the following exercises we shall write queries in various languages about our running "beer" example database:

```
Beers(name, manf)
Bars(name, addr, license)
Drinkers(name, addr, phone)
Likes(drinker, beer)
Sells(bar, beer, price)
Frequents(drinker, bar)
```

This question is devoted to SQL queries, database modifications, and declarations. Write the following in **standard SQL2**, being as succinct as possible:

a) Find the name and address of all drinkers who frequent Joe's Bar and like some beer that Joe's Bar sells. Do not print any drinker more than once.

b) Delete from `Drinkers` all drinkers in the 650 area code. You may assume that phone numbers are represented by character strings of the form `'(xxx) yyy-zzzz'`, where `xxx` is the area code.

c) Find for each price (that appears in `Sells`) the number of bars that serve at least one beer at that price.

d) Declare an assertion that says no drinker can frequent a bar with the same address as the drinker.

e) Write a declaration for the `Bars` relation, including the fact that `name` is a key, the default address is `'who knows?'`, and the `license` value can only be one of the strings `'beer'` and `'full'`. Use appropriate data types for the attributes.

f) Insert into `Bars` (with default values for `addr` and `license`) all those bars that are mentioned in `Frequents` but not in `Bars`.

Problem 3: (15 points) Using the same beer schema as Problem 2:

```
Beers(name, manf)
Bars(name, addr, license)
Drinkers(name, addr, phone)
Likes(drinker, beer)
Sells(bar, beer, price)
Frequents(drinker, bar)
```

write the following in **relational algebra**. You may, if you wish, break complex expressions into steps by defining temporary relations corresponding to nodes of the expression tree, as we did in the class notes.

a) Find all pairs of drinkers (i.e., their names) that have the same address. Produce the pair in only one order; e.g., if you produce (a, b), don't also produce (b, a).

b) Find all the bars mentioned in both `Sells` and `Frequents`, but not in `Bars`.

c) Find all the bars that serve only beers that drinker "Sally" likes.

Problem 4: (15 points) Again using the beer schema from Problems 2 and 3:

```
Beers(name, manf)
Bars(name, addr, license)
Drinkers(name, addr, phone)
Likes(drinker, beer)
Sells(bar, beer, price)
Frequents(drinker, bar)
```

write the following queries in **Datalog**. They are the same queries as in the previous problem.

a) Find all pairs of drinkers (i.e., their names) that have the same address. Produce the pair in only one order; e.g., if you produce (a, b), don't also produce (b, a).

b) Find all the bars mentioned in both `Sells` and `Frequents`, but not in `Bars`.

c) Find all the bars that serve only beers that drinker "Sally" likes.

Problem 5: (20 points) Suppose we have a relation R with attributes $ABCDE$ and functional dependencies $A \to B$, $C \to D$, $BD \to E$, and $E \to C$.

a) What are all the keys for R?

---

b) Which of the given FD's violate BCNF?

---

c) Which of the given FD's violate 3NF?

---

d) Suppose we decompose R into several relations, one of which has schema ACE. What nontrivial FD's hold for ACE? You need not mention a FD if it follows from another that you mention by adding attributes to the left side, and you need not mention FD's that have more than one attribute on the right.

Problem 6: (10 points) Suppose X is a bag, and $\cup$, $\cap$, and $-$ are bag union, intersection, and difference, respectively. For each of the following "laws," tell whether it holds (Y) or does not hold (N) for bags. **Score = 2 $\times$ (number right $-$ number wrong).**

a) $X \cup X = X$. ____________

b) $X \cap X = X$. ____________

c) $X - X = \emptyset$. ____________

d) $(X \cup X) - X = (X - X) \cup X$. ____________

e) $X \cup (X \cap X) = X \cup (X - X)$. ____________

Problem 7: (10 points) Suppose we have a relation with schema $ABCD$, the functional dependency $A \to B$ and the multivalued dependency $A \twoheadrightarrow C$. Give five other *nontrivial* multivalued dependencies that R must satisfy (i.e., MVD's that follow logically from the two given dependencies). Note that for a MVD to be *nontrivial*, its left and right sides must have no attribute in common, and there must be some attribute that is in neither the left nor right side.

Problem 8: (20 points) Network diagrams involve "logical record types" (LRT's) and "links." We can represent a network diagram by two predicates:

1. $LRT(x)$ means that x is the name of a logical record type of the diagram.
2. $Link(x, y, z)$ means that the link named y goes from LRT x to LRT z (i.e., y is a many-one relationship from x to z.

Note: this question is about representing network *diagrams* themselves as predicates or relations; it is *not* about converting network diagrams to relational database schemas.

a) Write a recursive Datalog program to compute $ManyOne(x, y)$ to be the set of pairs of LRT's x and y such that there is a many-one relationship from x to y that is formed by composing one or more links in the "forward" direction. More precisely, $ManyOne(a_1, a_k)$ is true exactly when there is a sequence of LRT's $a_1, a_2, \ldots, a_k$, with $k \geq 2$, such that for all $i = 1, 2, \ldots, k-1$, there is a link from a_i to a_{i+1}.

b) Write a recursive Datalog program to compute $SomeCon(x, y)$ to be the set of pairs of LRT's x and y such that there is some connection from x to y that is formed by composing one or more links in either direction. More precisely, $SomeCon(a_1, a_k)$ is true exactly when there is a sequence of LRT's $a_1, a_2, \ldots, a_k$, with $k \geq 2$, such that for all $i = 1, 2, \ldots, k-1$, there is either a link from a_i to a_{i+1} or a link from a_{i+1} to a_i, or both.

c) Write a Datalog program to compute $ManyMany(x, y)$ to be the set of pairs x and y such that there is a connection from x to y but no many-one relationship *in either direction* (i.e., from x to y or from y to x). You may use the IDB predicates from parts (a) and (b), even if you were unable to write these programs.

d) Give the strata of all the IDB predicates involved in your answer to (c).

Is your answer to (c) stratified? ________

Problem 9: (10 points) Below is an SQL3 recursive definition of a relation $R(x, y)$. It uses an EDB relation $Arc(a, b)$, which you may think of as the arcs of a graph, but what the query does is unimportant.

```
(1) WITH
(2)     RECURSIVE R(x,y) AS
(3)         (SELECT a, b FROM Arc WHERE b > 10)
(4)             EXCEPT
(5)         (SELECT y, x FROM R WHERE x < ALL
(6)             (SELECT a FROM Arc WHERE b < 20)
(7)         )
    /* some query about R */
```

The stratum graph for this query has nodes for R and for the three subqueries. For the subqueries, use the names:

$SQ1$ = the subquery of line 3
$SQ2$ = the subquery of line 6
$SQ3$ = the subquery of lines 5–7

a) Draw the stratum graph below. Do not forget to indicate which arcs are negative because the dependence they represent is not monotone.

b) Is the query stratified? Briefly justify your answer.

Problem 10: (15 points) Suppose we have the relation R defined by:

```
CREATE TABLE R(
    a INT,
    b INT
);
```

Write a program, in C with embedded standard SQL2, to print, for each tuple of R, the larger of its a and b components. You should print the integers one to a line. It is not necessary to indicate whether a value comes from the a or b column.

Problem 11: (15 points) This question is based on the following ODL schema:

```
interface Auto (extent Autos key serial#) {
    attribute integer serial#;
    attribute string brand;
    relationship Owner ownedBy inverse Owner::owns;
}

interface Owner (extent Owners) {
    attribute string name;
    attribute string addr;
    relationship Set<Auto> owns inverse Auto::ownedBy;
}
```

Write the following queries in OQL:

a) Find the names and addresses of the owners of Chevrolets ("Chevrolet" is a brand of auto).

b) Find the brands of automobiles owned by all owners whose name is "Sally." Do not produce the same brand more than once.

Part (c) is on the next page.

c) Find the (serial numbers of the) pairs of autos with the same owner. You should not produce a pair in both orders; e.g., if you produce (a, b) do not also produce (b, a). However, if n autos are owned by one person you can list all $\binom{n}{2}$ pairs in one order each.