CS145 Midterm Examination

Thursday, November 4, 2004, 2:45 – 4PM

Directions

- The exam is *open book*; any written materials may be used, as well as on-line or electronically stored materials. However, it is prohibited to access Oracle or another DBMS for the purpose of entering data and/or queries.
- Answer all 5 questions on the exam paper itself.
- The total number of points is 75 (i.e., one point per minute of work).
- When SQL is called for, you *must* use standard SQL-99, not the Oracle version.
- You will be graded not only on correctness of queries and other database code, but on the simplicity of your answers. You may get as little as 0 points for an answer that "works" but is excessively complicated.
- Do not forget to **sign the pledge** below.

I acknowledge and accept the honor code.

____

Print your name here: ____

1	2	3	4
5			

Problem 1: (12 points) In the space below, draw an E/R diagram that best represents the following stipulations. Indicate keys, many-one relationships, weak entity sets, and other features of E/R diagrams covered in class, as appropriate. At the bottom, you may add a few lines of notes explaining the points that you think are least obvious.

i. There are hospitals, with a name and address.
ii. There are rooms in hospitals, with a room number.
iii. There are people, with a name and social-security number.
iv. Some people are doctors, and doctors have a specialty; some people are patients, and patients have an age.
v. An operation involves a patient and a doctor, and takes place in a hospital room. An operation has a date and time. For full credit, **do not** make operations an entity set; rather, make it a relationship.

Problem 2: (9 points) Consider the relation schema $R(A, B, C, D, E, F)$ with functional dependencies $AC \to B$, $BD \to F$, and $F \to CE$.

a) (3 pts.) Find all the keys of R.

b) (4 pts.) Choose one FD of R that violates BCNF (and state which). Decompose R into two relations $R1$, and $R2$, using this FD.

FD: ______ $R1 =$ ______ $R2 =$ ______ Is $R1$ in BCNF? ____ Is $R2$? ____

c) (2 pts.) Suppose we project R onto $S(A, C, D, E)$. Give one nontrivial FD that holds in S.

Problem 3: (6 points) Consider each of the following proposed rules regarding dependencies. For each, if it is false, give a counterexample. If it is true, give a brief argument why it is true, e.g., by applying the closure to derive a FD from given FD's. You may assume the relation to which they apply is $R(A, B, C, D)$.

a) If $A \twoheadrightarrow B$ then $A \to B$.

b) If $A \to B$ and $BC \to D$, then $AC \to D$.

c) If $A \to B$ and $B \to C$ then $A \twoheadrightarrow BD$.

Problem 4: (20 points) Consider the following four relations to store data about animals in a zoo:

Animals(<u>idNumber</u>, type, cageNumber)
Cages(<u>cageNumber</u>, maxAnimals)
TypeKeepers(<u>name</u>, <u>type</u>)
CageKeepers(<u>name</u>, <u>cageNumber</u>)

idNumber is a key for *Animals*, and *cageNumber* is a key for *Cages*. *Cages.maxAnimals* specifies the maximum number of animals allowed in that cage. *TypeKeepers* lists people responsible for caring for a particular type of animal, while *CageKeepers* lists people responsible for caring for a certain cage. For each of these two, both attributes form the key; i.e. it is possible for the same person to care for multiple types/cages, and it is possible for the same type/cage to be cared for by several people.

In the following, you may express queries either as a sequence of assignment statements or as a single complex expression. You should assume all relations are sets, not bags.

a) (6 pts.) Consider the following sequence of relational algebra statements:

```
Temp1 := Animals ⋈ CageKeepers
Temp2 := Animals ⋈ TypeKeepers
FinalAnswer := Temp1 ∩ Temp2
```

Given the following sets of data for each of the relations, show (in the space on the next page) the relation that will be stored into *FinalAnswer* (show the schema and all the tuples that will be in the relation).

Animals:

idNumber	*type*	*cageNumber*
1	Zebra	10
2	Monkey	20
3	Monkey	10
4	Kangaroo	30
5	Zebra	40
6	Monkey	40

CageKeepers:

name	*cageNumber*
Ann	10
Chris	20
Chris	30
Bob	40
Bob	10

TypeKeepers:

name	*type*
Ann	Zebra
Bob	Monkey
Ann	Kangaroo
Chris	Kangaroo

b) (2 pts.) State briefly what the query from (a) is asking.

c) (6 pts.) Using (some of) the same relations:

Animals(<u>idNumber</u>, type, cageNumber)
Cages(<u>cageNumber</u>, maxAnimals)
TypeKeepers(<u>name</u>, <u>type</u>)
CageKeepers(<u>name</u>, <u>cageNumber</u>)

write an expression to find all pairs of animals of two different types in the same cage. For each pair of animals satisfying this requirement, your expression should return each animal's *idNumber* and *type*, as well as the *cageNumber* of the cage they are in. For example, if Animal 1 of type Zebra and Animal 2 of type Monkey are both in cage 10, then the result of your expression should include a tuple (1, Zebra, 2, Monkey, 10) (the attributes could be in a different order). The same animal pair should not appear twice in your result.

d) (6 pts.) Using (some of) the same relations:

Animals(idNumber, type, cageNumber)
Cages(cageNumber, maxAnimals)
TypeKeepers(name, type)
CageKeepers(name, cageNumber)

write an expression to find all cages with more animals than the allowed maximum for that cage. Your result should be a list of all cages that have more than their maximum. For example, if there are 5 animals listed as being in cage 10, but cage 10 has a maximum of 3, then (10) would appear in the result relation. Note: you may use the extended relational algebra for this query.

[Problem 5 begins on the next page.]

Problem 5: (28 points) The relations below represent information about the 15 members of the UN Security Council and the resolutions (*res*) that they voted on.

Members(name, status)
Votes(res, country, vote)

The status of a country is either 'permanent' or 'elected'. A vote can be either 'yes', 'no', or 'abstain'.

You are asked below to write some fairly complex declarations and queries. **Please** think these through, perhaps using scratch paper, before you write your answers in the spaces. Reasonable responses will fit in the space allotted, provided you put SELECT and FROM clauses on one line. We don't insist that you print your answers, but illegible scrawl will be assumed incorrect.

a) (4 pts.) Complete the SQL schema for *Members* and *Votes*, which is started below:

```
CREATE TABLE Members (
    name CHAR(50)

    status CHAR(10)


)
CREATE TABLE Votes (
    res INT

    country CHAR(50)

    vote CHAR(10)


)
```

Include in your schema:

i. The keys indicated by underlines in the informal schema above.

ii. The constraint that any country mentioned in *Votes* appears in the *name* column of *Members*.

iii. The constraint that no *vote* can be `NULL`.

iv. The constraint that *vote* can be only one of the three strings `yes`, `no`, and `abstain`.

Do not forget to place commas where needed.

b) (8 pts.) In order for a resolution to pass, there must be at least 9 'yes' votes from among the 15 members of the council. In addition, there must be no veto; that is, none of the five permanent members of the council vote 'no' on that resolution. We wish to write a query on the relations:

Members(name, status)
Votes(res, country, vote)

to find the set of resolutions that passed. Enter subqueries in the spaces below that will make the query work:

```
SELECT res FROM Votes vv
WHERE 9 <= ( /* SQ1 */
```

```
)
AND NOT EXISTS ( /* SQ2 */
```

```
)
```

c) (2 pts.) Explain informally what your SQ1 from (b) does.

d) (2 pts.) Explain informally what your SQ2 from (b) does.

e) (8 pts.) We also wish to write, for the relations

Members(<u>name</u>, status)
Votes(<u>res</u>, <u>country</u>, vote)

a query to list each permanent member of the council, along with the number of vetos ('no' votes) they have cast. We must not forget to consider the possibility that a permanent member cast no vetos. Below is the beginning of a query to do so; you have to complete it correctly.

```
(SELECT name, COUNT(vote) AS numVetos
 FROM Members JOIN Votes ON name = country
 WHERE status = 'permanent' AND vote = 'no'
 GROUP BY name)
```

f) (4 pts.) Explain informally what each subquery you wrote in (e) does.