CS145 Final Examination
Thursday, December 14, 1995, 3:30–6:30PM

Directions

The exam is *open book*; any written materials may be used. Answer all 16 questions on the exam paper itself. The total number of points is 225. In general, when doing a design in any of the models we've discussed, **make the design as succinct as possible. Avoid redundancy where possible.** All questions about SQL refer to the SQL2 standard, *not* to Sybase 10.0 or any other SQL implementation.

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

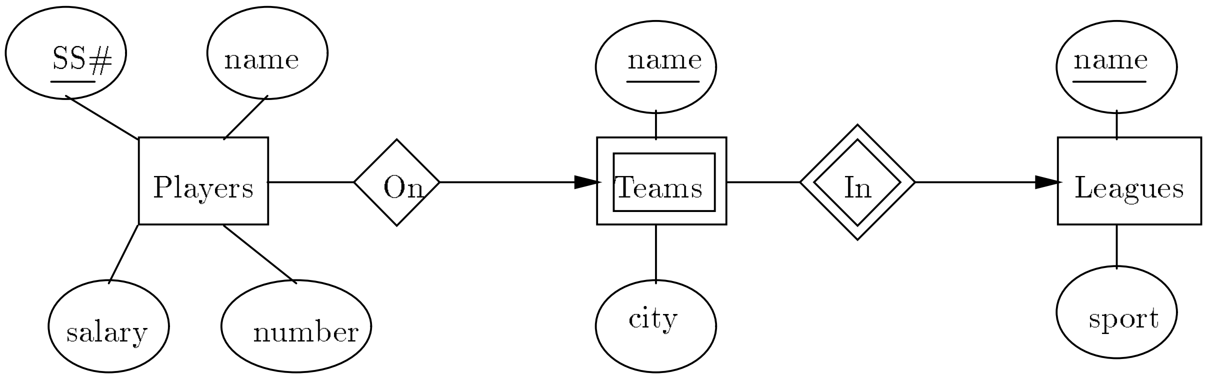
I acknowledge and accept the honor code.

____

Print your name here: ____

1	2	3	4
5	6	7	8
9	10	11	12
13	14	15	16

Problem 1: (15 points) In the E/R diagram below,[1]

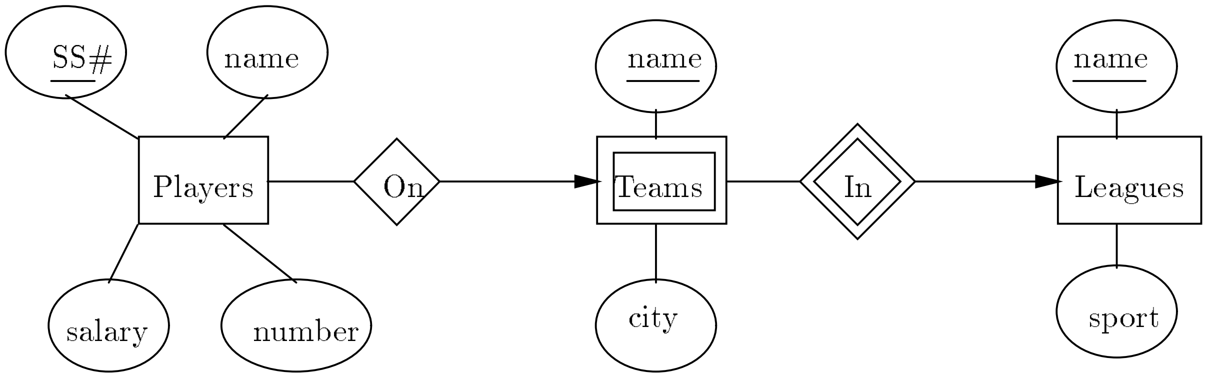


Teams is a weak entity set; its key is both its name and the name of the unique league to which it belongs.

[1] We use the convention of putting double lines around weak entity sets and their relationships that lead to (parts of) their keys. However, that is just to remind you of what is "weak." You can ignore the double lines. The diagrams then conform to the notation used in class.

Below, give an ODL description of the same database. You need not define an extent for a class. You must define keys and give inverses for relationships.

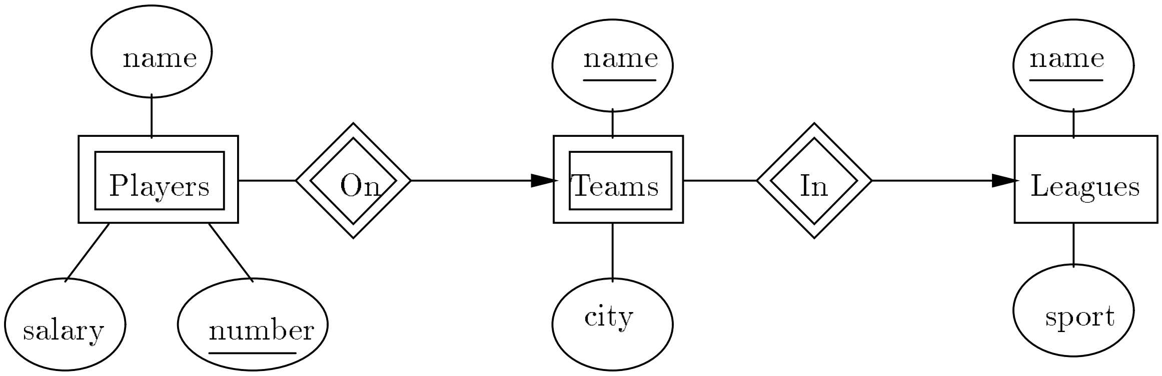
Problem 2: (10 points) For the E/R diagram of Problem 1 (which we reproduce below):



a) Give a relational schema of the same database. You *must* follow the process outlined in Handout #8 and in class for converting an E/R diagram with weak entity sets to relation schemas. Indicate the keys of your relations by underlining attributes.

b) Simplify your answer to (a) by eliminating redundant attributes and redundant relation schemas. Again indicate keys by underscore.

Problem 3: (10 points) Now, it is decided to modify the database of the previous two questions by eliminating the SS# as an attribute of players. Therefore, we do not have a convenient key for *Players*. The logical choice for a key is to take the number of the player and a key for the team of the player; no team would assign the same number to two players. That makes player a weak entity set, depending on *Teams* for part of its key. The situation is shown below:



Below, give an appropriate relational database schema for the E/R diagram above. You do not have to follow any particular algorithm to produce your answer, but your answer should be as succinct as possible and, of course, faithful to the database we described above. Again, indicate keys for your relations by underscore.

Problem 4: (25 points) Suppose relation $R(A, B, C, D, E)$ has functional dependencies

$$AB \rightarrow C$$
$$D \rightarrow A$$
$$AE \rightarrow B$$
$$CD \rightarrow E$$
$$BE \rightarrow D$$

a) Find all the keys of R (15 pts.).

---

b) Of the five given FD's, which violate the BCNF condition?[2]

---

c) Of the five given FD's, which violate the 3NF condition?

Problem 5: (15 points) Consider the following database schema:

```
Paintings(artist, work, date, museum)
Museums(name, city, curator)
```

Intuitively, `Paintings` contains tuples (a, w, d, m) such that artist a painted work w on date d, and the work is hanging in museum m. `Museums` contains tuple (n, c, x) if n is the name of a museum in city c, and the curator of museum m is x. Write the following queries in *Relational Algebra*. As in all questions on this exam, remember that simplicity counts. Also, **you may assume the relations have no duplicates and no nulls**.

a) On what date did Leonardo paint the Mona Lisa?

---

b) In what city is the Mona Lisa?

---

c) Find all works that were painted by the curator of the museum in which they hang.

---

[2] Note that the BCNF condition applies not only to given FD's but to all FD's that follow from them. However, we are not asking you to examine all FD's — just these five.

Problem 6: (15 points) Using the same database schema as in Problem 5:

```
Paintings(artist, work, date, museum)
Museums(name, city, curator)
```

write *Datalog* queries to answer the following questions. You may define IDB predicates if you like and may use more than one rule. In each part, put the answer in a predicate called *answer*. We'll assume that the uses of this predicate in each part have nothing to do with each other.

a) Find all the museums in New York City.

b) Find all the works that are either painted by Monet or hang in a museum in Paris.

c) Find all the museums that have at least three different works by the same artist.

Problem 7: (20 points) This question is based on the database schema

```
Paintings(artist, work, date, museum)
Museums(name, city, curator)
Prices(work, value)
```

which is like the schema of the two previous problems, but with the addition of a relation `Prices`. This relation contains tuple (w, v) if the value of work w is v. Again, **you may assume no duplicates and no nulls**. Write the following queries in SQL.

a) Find the museums that have works valued at more than $10,000,000.

b) Print a table in which the columns are headed `Painter` and `Painting`, with one row for each work held by the Rijksmuseum, which is located in the city of Amsterdam.

c) Print a table (any column headers are OK) giving for each museum the average value of its holdings.

d) Do the same as (c), but only for those museums that hold more than 100 works.

Problem 8: (5 points) Here is a query based on the schema of Problem 7; it uses a subquery.

```
SELECT artist
FROM Paintings
WHERE museum IN
    (SELECT name
     FROM Museums
     WHERE city = 'London'
    )
```

Write an equivalent SQL query without any subqueries.

Problem 9: (10 points) Based on the running example of a database schema

```
Paintings(artist, work, date, museum)
Museums(name, city, curator)
Prices(work, value)
```

Write SQL statements that define (create) the following views.

a) The set of (m, a) pairs, where a is the average value of the paintings held by museum m.

b) The set of (a, w, m, c) such that work w by artist a is held by museum m, which is located in city c. However, if there is no information in our database about the city in which museum m is located, then we want the tuple $(a, w, m, \texttt{NULL})$ in our view, instead.

Problem 10: (15 points) The following SQL queries are based on our running database schema:

```
Paintings(artist, work, date, museum)
Museums(name, city, curator)
Prices(work, value)
```

Write for each of them an equivalent query in relational algebra.

a)

```
SELECT DISTINCT work, date
FROM Paintings
WHERE artist = 'Rembrandt'
```

b)

```
SELECT DISTINCT work
FROM Prices
WHERE value = (
    SELECT MAX(value)
    FROM Prices
)
```

c)

```
SELECT DISTINCT work
FROM Prices
WHERE value >= ANY (
    SELECT value
    FROM Prices
)
```

Problem 11: (15 points) Suppose we wish to make SQL declarations of the database schema in our running example:

```
Paintings(artist, work, date, museum)
Museums(name, city, curator)
Prices(work, value)
```

We might declare the `Paintings` relation by

```
CREATE TABLE Paintings (
    artist CHAR(30),
    work CHAR(30),
    date DATE,
    museum CHAR(30),
/*********************************/
)
```

with similar declarations for `Museums` and `Prices`. We want to include some constraints in the declaration for `Paintings`, however. Give text that could be substituted for the line of *'s to declare the following constraints as simply as possible:

a) `artist` and `work` together form a key for `Paintings`.

---

b) If a value appears in the `museum` column of `Paintings`, then it must also appear in the `name` column of `Museums`. You may assume for this problem that `name` is a key for `Museums`.

---

c) Every work that appears in the `work` column of `Paintings` must have a value of at least $100,000.

Problem 12: (10 points) Define SQL assertions that say, of the database schema of Problem 11:

a) The average value of the works hanging in each museum must be at least \$100,000.

b) Museums in San Francisco must not hang paintings by Andrew Wyeth.

Problem 13: (20 points) In this problem you will fill in some of the C and embedded SQL statements needed to write a function that computes the sum of the squares of the values of all the works in the `Prices` relation. Follow the comments as a guide to what is expected. You may use the shorthand `NOT_FOUND` for the condition that `SQLSTATE` returns the code `'02000'`, indicating that no tuple was found in response to a fetch.[3]

```
float sumSqVals() {
    float sumSq; /* accumulates the sum of squares */
    EXEC SQL BEGIN DECLARE SECTION;
        float val; /* to hold the value component of a tuple */
        char SQLSTATE[5];
    EXEC SQL END DECLARE SECTION;
    /* Declare and open a cursor on the unary relation consisting
        of the value components of the Prices tuples */
```

```
    /* examine each value in turn and add its square to sumSq */
```

```
    /* on completing examination of all values, close the cursor */
```

```
    return(sumSq);
}
```

[3] The precise C statement on line (13) of Fig. 5.22 of the reader was bungled. `NOT_FOUND` actually stands for `!(strcmp(SQLSTATE, '02000')`.

Problem 14: (15 points) Suppose there is a function `p(w,a)` that updates the relation `Prices(work, value)` by adding amount `a` to the value of work `w`. Function `p(w,a)` works by performing the following four steps in order. In what follows, `w` and `a` are parameters of `p`, holding the work and amount, respectively. Variable `w` is also declared in an `SQL DECLARE` section so it can be shared between the C code of `p` and the embedded SQL. Variables `v` and `b` are local to function `p`, and they are declared in the `SQL DECLARE` section so they may be shared.

1. Execute the following single-row-select query (see Sect. 5.10.6 of the reader):

```
EXEC SQL SELECT price INTO :v
    FROM Prices
    WHERE work = :w
```

2. Compute `b=v+a;`.
3. Execute the following deletion of the old tuple for work `w`.

```
EXEC SQL DELETE FROM Prices
    WHERE work = :w AND value = :v
```

4. Execute the following insertion of the new value for work `w`:

```
EXEC SQL INSERT INTO Prices(work, value)
    VALUES(:w, :b)
```

Let P be a process that executes `p("Sunflowers", 1000000)` and let Q be a process that executes `p("Sunflowers", 2000000)`. Suppose that initially, the only tuple in `Prices` with first component `"Sunflowers"` is `("Sunflowers", 40000000)`.[4] We use P_i and Q_i to refer to the ith step (above) of process P and Q, respectively, for $i = 1, 2, 3, 4$.

a) If P and Q execute as transactions, what tuple(s) with first component `"Sunflowers"` exist in `Prices` after both have executed?

---

b) If P and Q execute their steps in the following interleaved order:

$$P_1, P_2, Q_1, Q_2, P_3, P_4, Q_3, Q_4$$

what tuple(s) with first component `"Sunflowers"` exist in `Prices` after all eight steps have executed?

---

c) Give an example of an order of the eight steps that will result in a run-time error (i.e., one of the steps cannot be executed properly).

---

[4] If you can't follow all those 0's, the original price is 40 million, and there are two increments of 1 million and 2 million.

Problem 15: (10 points) Suppose p is the privilege `SELECT ON Museums`. Suppose A, B, C, D, and E are users, and A is the owner of relation `Museums`. Finally, suppose the following sequence of grants are executed by the following users:

User	Action
A	GRANT p TO B, C WITH GRANT OPTION
B	GRANT p TO D WITH GRANT OPTION
D	GRANT p TO C, E WITH GRANT OPTION
C	GRANT p TO E

a) Below, draw the grant diagram for the privileges regarding property p. Since all nodes represent p and some user, you need not show p in the nodes.

b) Now, suppose B executes `REVOKE` p `FROM` D `CASCADE`. Show the resulting grant diagram with nodes corresponding to revoked privileges removed.

Problem 16: (15 points) Suppose we wish to represent grant diagrams by a relational database. We shall assume a "node ID" (NID), i.e., a serial number for nodes in grant diagrams; NID may be used as an attribute to represent nodes. Suggest below a database schema that represents all the information found in a grant diagram. Your attributes should either have self-explanatory names or be accompanied by a short comment explaining what they represent. Indicate keys by underlining. *Note*: We are not asking you to represent a particular grant diagram, but rather to show how relations could be used to represent the same information as the graphs used in the course reader and in Problem 15.