DB SOLUTIONS

Stanford University Computer Science Department
2000 Comprehensive Exam in Databases

WITH ANSWERS

- The exam is *open book and notes.*
- Answer all 6 questions on the exam paper itself, in the space provided for each question.
- The total number of points is 60; questions may have differing point values.
- You have 60 minutes to complete the exam (i.e., one minute per point). It is suggested you review the entire exam first, in order to plan your strategy.
- *Simplicity and clarity of solutions will count.* You may get as few as 0 points for a problem if your solution is far more complicated than necessary, or if we cannot understand your solution.

Provide your magic number here: ____________

1	2	3	4
5	6		

Problem 1: (10 points) Suppose relation $R(A, B, C, D, E)$ has functional dependencies

$$A \to B$$
$$BC \to D$$
$$BE \to C$$
$$AD \to E$$
$$CE \to A$$

a) What are all the keys of R?

Answer: AC, AD, AE, BE, and CE.

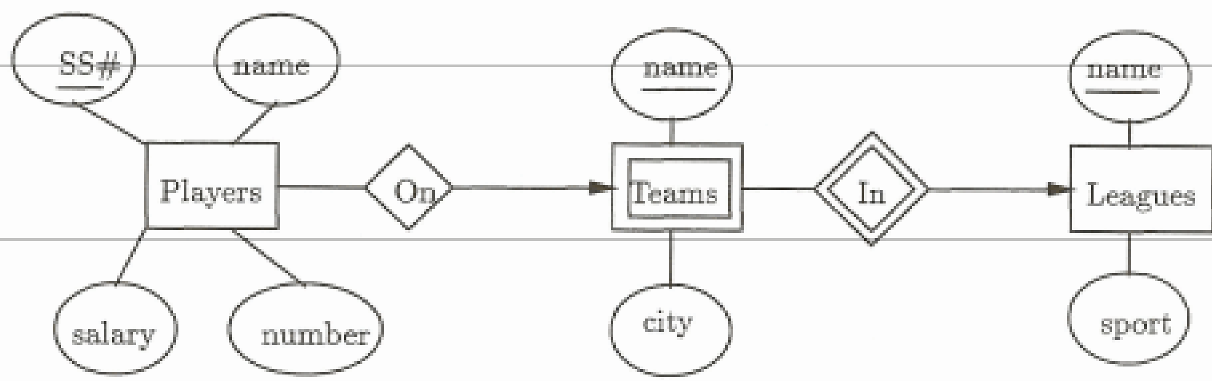
b) Of the five given FD's, which violate the BCNF condition?

Answer: The first two.

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

Answer: None.

Problem 2: (10 points) Below is an entity-relationship diagram:

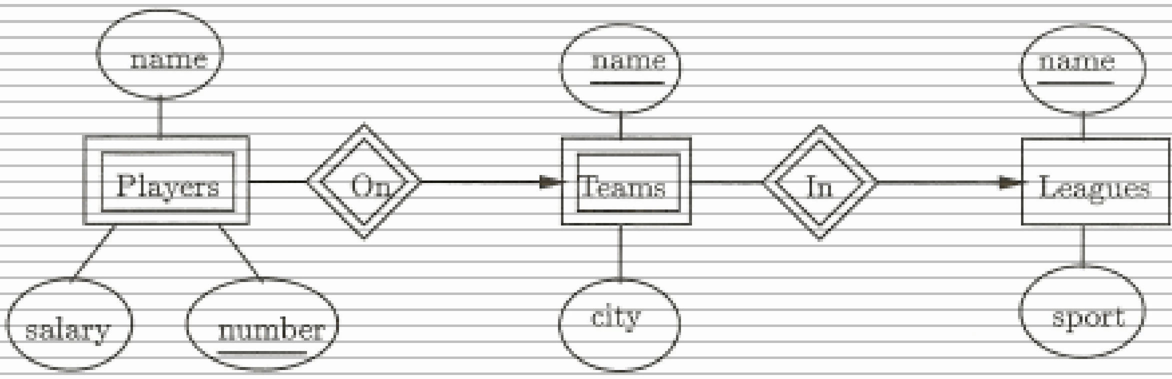


a) Choose a relational database schema that represents this E/R diagram as faithfully as possible. Do not use a relation if its information is sure to be contained in some other relation of your schema.

Answer:

```
Players(SS#, name, number, salary)
On(SS#, teamName, leagueName)
Teams(teamName, leagueName, city)
Leagues(name, sport)
```

b) Suppose that we delete the SS# attribute from *Players*. Exploit the fact that a team will not give the same number to two players in order to find a similar E/R diagram



c) Convert your E/R diagram from (b) to an appropriate relational database schema.

Answer:

```
Players(name, teamName, leagueName, number, salary)
Teams(teamName, leagueName, city)
Leagues(name, sport)
```

Problem 3: (10 points) A database system, containing of two objects A and B executes three transactions: T_1, T_2 and T_3. Initially A has a value of 10, and B a value of 20.

T_1: First writes a value of 30 for A; then changes B to 40. Finally, T_1 commits. T_1 runs with isolation level `SERIALIZABLE`.

T_2: Starts by changing A to 50, then modifies B to 60. At this point T_2 does a rollback, and all changes are undone. T_2 runs with isolation level `SERIALIZABLE`.

T_3: Is a read-only transaction that first reads A and then reads B. The isolation level of T_3 is discussed below.

We do *not* know in what order these three transactions execute.

a) Assume that T_3 runs with isolation level `SERIALIZABLE`. What are all the possible A, B values that T_3 can read? Give each answer as a pair $[X, Y]$, where X is the A value read, and Y is the B value read by T_3.

Answer: [10,20], [30,40].

b) Assume that T_3 runs with isolation level `READ COMMITTED`. What additional A, B values can T_3 read? [Do not list pairs given in part (a).]

Answer: [10,40].

c) Assume that T_3 runs with isolation level `READ UNCOMMITTED`. What additional A, B values can T_3 read? [Do not list pairs given in parts (a) or (b).]

Answer: [50,60], [50,20], [50,40], [30,20], [10,60], [30,60].

Problem 4: (5 points) Suppose we have a relation with schema $ABCD$, the functional dependency $A \rightarrow B$ 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.

Answer: Among other choices are anything with a left side A and a right side that is any subset of $\{B, C, D\}$ except for $\{C\}$ (because it is given) and $\{B, C, D\}$ (because it is trivial).

Problem 5: (20 points) Consider the following relation:

```
Advised(Advisor, Student, Year)
```

A tuple (A, S, Y) in `Advised` specifies that advisor A advised student S who graduated in year Y. Assume that `Student` is a key for this relation.

a) Consider the following SQL query, which finds all advisors who advised a student who graduated in the same year that Hector Garcia-Molina or Jennifer Widom graduated.

```
SELECT Advisor
FROM Advised
WHERE Year IN
    (SELECT Year FROM Advised
    WHERE Student = 'Hector Garcia-Molina'
        OR Student = 'Jennifer Widom')
```

Write an SQL query, *without any subqueries* and *without the keyword* `DISTINCT`, that always produces the same set of tuples as the above query.

Answer:

```
SELECT A1.Advisor
FROM Advised A1, Advised A2
    WHERE A1.Year = A2.Year
        AND (A2.Student = 'Hector Garcia-Molina'
            OR A2.Student = 'Jennifer Widom');
```

b) Are there are any circumstances in which your "equivalent" query can produce an answer different from that of the query above? Explain, if so.

Answer: If Hector and Jennifer graduated in the same year Y, then this query will return two copies of an advisor for each student he graduated in year Y, while the original query will return only one copy.

c) Using SQL3 recursion, write a query that finds all "descendants" of Jeff Ullman, i.e., all students whose advisor was Jeff Ullman, or whose advisor's advisor was Jeff Ullman, or whose advisor's advisor's advisor was Jeff Ullman, and so on.

Answer:

```
WITH RECURSIVE Descendant AS (
    (SELECT Student FROM Advised WHERE Advisor = 'Jeff Ullman')
        UNION
    (SELECT A.Student FROM Advised A, Descendant D
    WHERE A.Advisor = D.Student)
);
SELECT * FROM Descendant;
```

d) Write a SQL query that finds the advisor(s) with the longest advising span, i.e., with the longest period from their earliest advisee to their latest advisee.

Answer:

```
SELECT Advisor
FROM Advised
GROUP BY Advisor
HAVING (Max(Year) - Min(Year)) >= ALL(
    SELECT Max(Year) - Min(Year)
    FROM Advised
    GROUP BY Advisor
);
```

Problem 6: (5 points) Two relations $A(x)$ and $B(y)$ each contain integers (i.e., their tuples have one component, which is an integer). Give `CREATE TABLE` statements for A and B with `CHECK` clauses sufficient to assure that $A \cap B$ will always be empty.

Answer:

```
CREATE TABLE A(
    x INT CHECK(x NOT IN (SELECT y FROM B)
);
CREATE TABLE B(
    y INT CHECK(y NOT IN (SELECT x FROM A)
);
```