Stanford University Computer Science Department
1998 Comprehensive Exam in Databases

SOLUTION

- The exam is open book and notes.
- There are 6 problems on the exam, with a varying number of points for each problem and subproblem for a total of 60 points. You should look through the entire exam before getting started, in order to plan your strategy. You have 60 minutes to complete the exam.
- Please write your solutions in the spaces provided on the exam. Make sure your solutions are neat and clearly marked.
- *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: ____________

Problem	Points	Maximum
1		10
2		6
3		16
4		8
5		10
6		10
Total		60

1. **(Database Design – 10 points)**

Suppose we have a relation $R(A, B, C, D)$ with functional dependencies $AD \rightarrow C$, $B \rightarrow A$, and $C \rightarrow D$.

(a) List all minimal keys of R.
Answer: BC, BD

(b) Which of the given functional dependencies violate Boyce-Codd Normal Form?
Answer: all three

(c) Which of the given functional dependencies violate Third Normal Form?
Answer: $B \rightarrow A$

2. **(Multivalued and Functional Dependencies – 6 points)**

Suppose we have a relation $R(A, B, C)$ and the following instance of R.

A	B	C
2	40	500
2	50	400
1	20	200
1	30	300
2	40	400
2	50	500

(a) Given this instance, specify two nontrivial multivalued dependencies that cannot hold for R.
Answer: $A \twoheadrightarrow B$, $A \twoheadrightarrow C$.

(b) Can you deduce from this instance any nontrivial functional or multivalued dependencies that are always guaranteed to hold for R?
Answer: No

3. **(Basic Relational Algebra, SQL, Constraints – 16 points)**

Suppose we have the following schema:

```
Depts(dept#, deptName)              // dept# is key
Courses(course#, dept#, units)      // <course#,dept#> is key
```

Relation Depts associates a department number (e.g., 230) with a department name (e.g., "CS"). Relation Courses contains course information, e.g., the tuple ⟨145, 230, 3⟩ states that course 145 of department 230 is taken for 3 units.

(a) Specify a relational algebra expression that finds all courses (identified by course# and dept#) in a department named "CS" with fewer than 3 units.
Answer: $\Pi_{\text{course\#,dept\#}}(\sigma_{\text{deptName}=\text{“}CS\text{”} \wedge \text{units}<3}(\text{Depts} \bowtie \text{Courses}))$

(b) Write the previous query in standard SQL2.
Answer:

```
SELECT course#, d.dept#
FROM Depts d, Courses c
WHERE d.dept# = c.dept# AND deptName = 'CS' AND units < 3
```

(c) Suppose there is a referential integrity constraint from Courses.dept# to Depts.dept#. Specify a SQL2 assertion that ensures that this referential integrity constraint is not violated. Your answer should be of the form:
CREATE ASSERTION ⟨name⟩ CHECK (⟨condition⟩).
Answer:

```
CREATE ASSERTION RefInt CHECK (
   NOT EXISTS (
      SELECT *
      FROM Courses
      WHERE dept# NOT IN (SELECT dept# FROM Depts) ) )
```

(d) Suppose we happen to know that the following relational algebra assertion always holds:

$$\Pi_{\texttt{dept\#}}(\texttt{Depts}) \subseteq \Pi_{\texttt{dept\#}}(\sigma_{\texttt{deptName}=\text{"}CS\text{"}}(\texttt{Depts}))$$

Also assume that the referential integrity constraint from `Courses.dept#` to `Depts.dept#` specified in part (c) holds. Can you simplify your relational algebra query from part (a) based on these two constraints?

Answer: $\Pi_{\texttt{course\#,dept\#}}(\sigma_{\texttt{units}<3}(\texttt{Courses}))$

4. **(Advanced SQL, Relational Algebra – 8 points)**

Consider again the schema from Problem 3:

```
Depts(dept#, deptName)              // dept# is key
Courses(course#, dept#, units)      // <course#,dept#> is key
```

(a) Write a query in SQL2 that finds the number of courses offered by each department. The schema of the query result should be `(dept#, num-of-courses)`. Do not assume that any of the constraints from Problem 3 hold.
Answer:

```
(SELECT dept#, COUNT(*)
 FROM Courses
 GROUP BY dept#)
UNION
(SELECT dept#, 0
 FROM Depts
 WHERE dept# NOT IN (SELECT dept# FROM Courses))
```

(b) Can you write the same query in relational algebra? If so, specify the relational algebra expression.
Answer: No

5. **(Transactions – 10 points)**

Continue with the schema from Problems 3 and 4:

```
Depts(dept#, deptName)                 // dept# is key
Courses(course#, dept#, units)         // <course#,dept#> is key
```

The following code segment is intended to ensure that any insertion transaction to `Courses` does not violate the referential integrity constraint from `Courses.dept#` to `Depts.dept#` upon completion. Complete the code segment by filling out the underlined portions. Do *not* write code to implement the comments.

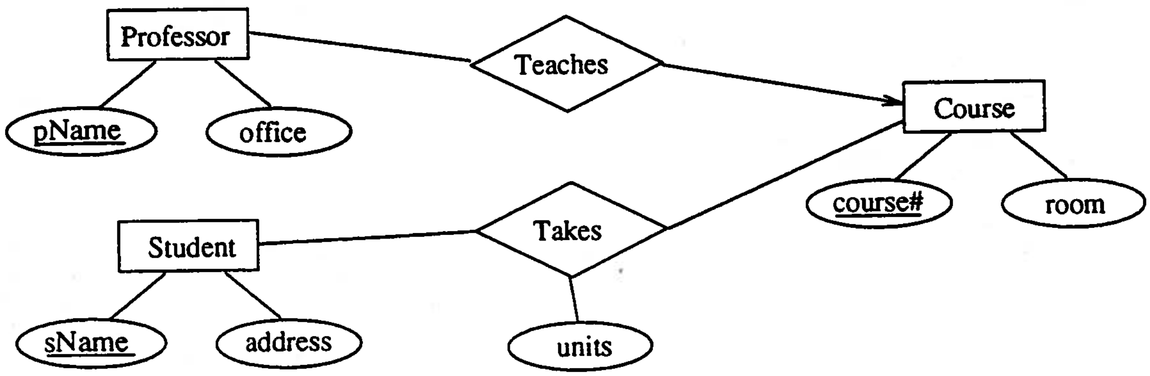
```
1)  EXEC SQL BEGIN DECLARE SECTION;
2)    int courseN, deptN, nUnits; /* variables for Courses attributes */
3)    int checkCount; /* variable for checking constraint */
3)  EXEC SQL END DECLARE SECTION;

4)  void insertToCourses() {
5)    /* C code to prompt the user to enter values for new Courses tuple.
6)       The values entered for the tuple attributes are stored in
7)       courseN, deptN, and nUnits. */

8)    EXEC SQL SELECT ______________________ INTO :checkCount
9)             FROM Depts
10)            WHERE Depts.dept# = :deptN;

11)   if (______________________) {
12)      EXEC SQL INSERT INTO Courses
13)               VALUES (:courseN, :deptN, :nUnits);
14)      ______________________
15)   }

16)   else
17)      ______________________
18) }
```

Answer:

```
Line 8)  COUNT(dept#)
Line 11) checkCount > 0
Line 14) EXEC SQL COMMIT;
Line 17) EXEC SQL ROLLBACK;
```

6. **(Entity-Relationship Design – 10 points)**

The following Entity-Relationship diagram has entity sets *Professor*, *Student*, and *Course*. Relationship *Teaches* relates professors to the course(s) they teach, and relationship *Takes* relates students to the course(s) they take, including the number of units enrolled. (Note that this design is not related to the Depts-Courses relational schema used in Problems 3–5.)



(a) Based on the above diagram, can a professor teach more than one course?
Answer: No

Can a student take more than one course?
Answer: Yes

(b) Specify a relational schema corresponding to the above diagram. Your schema should be based on a straightforward mapping in which there is one relation for each entity set and one for each relationship. Underline a minimal key for each relation.
Answer:
Professor(<u>*pName*</u>, *office*)
Student(<u>*sName*</u>, *addr*)
Course(<u>*course#*</u>, *room*)
Teaches(<u>*pName*</u>, *course#*)
Takes(<u>*sName*</u>, <u>*course#*</u>, *units*),

(c) Suppose we combine *Teaches* and *Takes* into a single ternary relationship. Identify a real-world situation that can be modeled by the two binary relationships *Teaches* and *Takes*, but that cannot be modeled by the ternary relationship.
Answer: A student may take a course that no professor is teaching. Alternatively, a professor may teach a course that no student is taking.