Stanford University Computer Science Department
2003 Comprehensive Exam in Databases
SAMPLE SOLUTION

1. **(14 pts.)** Relation $R(A, B, C, D, E)$ has functional dependencies $AB \rightarrow C$, $BC \rightarrow D$, $CD \rightarrow E$, and $DE \rightarrow A$.

   (a) **(6 pts.)** What are all the keys of R?

   Answer: AB, BC, and BDE. Note that B must be in any key, since it doesn't appear on the right of any FD. That fact makes the search for keys fairly easy.

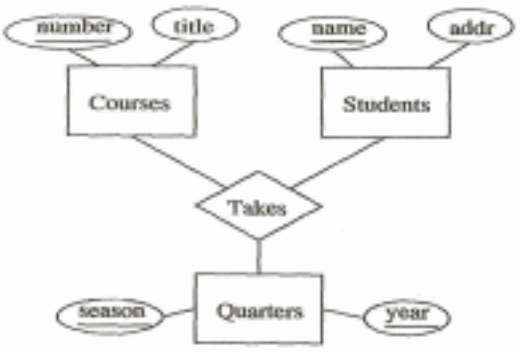
   (b) **(2 pts.)** Which of the given functional dependencies are Boyce-Codd Normal Form (BCNF) violations? Which are Third Normal Form (3NF) violations?

   Answer: $CD \rightarrow E$ and $DE \rightarrow A$ violate BCNF. None violate 3NF, because all attributes are prime.

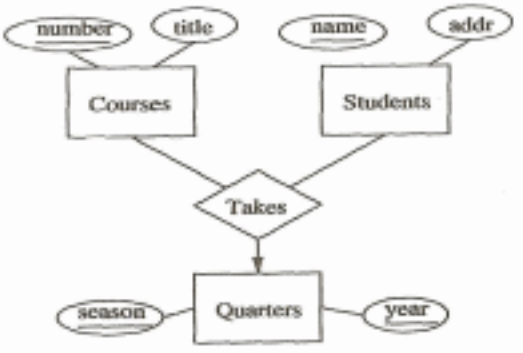
   (c) **(6 pts.)** Find a lossless-join decomposition (a decomposition from which the original relation R can be recovered) of R into BCNF relations. Show your steps and your reasoning—which violating functional dependencies cause which decompositions in which order.

   Answer: Suppose we use $CD \rightarrow E$ to decompose. Since $CD^+ = ACDE$, one of the schemes is $R1(A, C, D, E)$ and the other is $R2(B, C, D)$. The latter is in BCNF, since BC is the only key, and $BC \rightarrow D$ the only projected FD. However, $R1$ is not in BCNF. For example, $DE \rightarrow A$ is a projected FD, but $DE^+ = ADE$, so DE is not a superkey for $R1$. Thus, we decompose $R1$ into $R3(A, D, E)$ and $R4(C, D, E)$. The constituents of the decomposition are $R2$, $R3$, and $R4$.

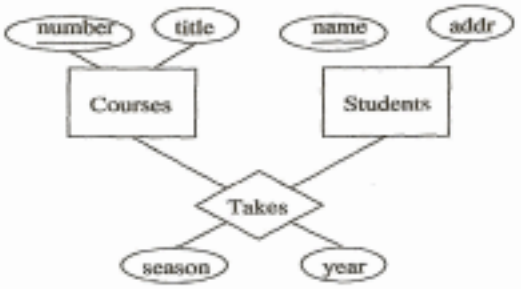
2. **(16 pts.)** Consider the following three entity-relationship (E/R) diagrams that represent students, courses, and the quarters (e.g., Autumn, 2003) that the student took the course.



(a)



(b)



(c)

(a) **(6 pts.)** Explain the differences among the three diagrams (a), (b), and (c). What constraints does each imply? Based on constraints, do any two of them capture exactly the same real-world data? Explain the differences in terms of students, courses, and quarters (e.g., "a student can take only one course in a given quarter").

Answer: (a) is an unconstrained relationship among students, courses, and quarters. In particular, students can take a course several times. (b) adds the constraint that a student can take a given course in only one quarter; i.e., no repeating courses is allowed. (c) is the same as (b). In fact, (c) is really a shorthand for the diagram (b). Many people did not realize that the "no repeating courses" constraint is implied by (c), but you can reason as follows. Since the meaning of *Takes* is a relationship set, a student-course pair can appear in this set only once. Whatever season and year are attributes of this pair are the one and only one quarter in which that student took that course.

(b) **(2 pts.)** If in E/R diagram (b) we added an arrow on the line from *Takes* to *Students*, what additional constraint would be implied by that diagram?

Answer: Only one student can take any given course in any given quarter.

(c) **(8 pts.)** Convert the structure of E/R diagram (b), *without the addition mentioned in part (b) of this problem*, to an ODL (Object Definition Language) schema. You

need not specify extents, but you should specify keys when appropriate. Try to keep it simple, not introducing classes that you don't really need. You may use the following abbreviations: `att` for "attribute," `rel` for "relationship," `inv` for "inverse."

Answer: In general, we need a connecting class to represent student-course-quarter triples. However, since quarters are really just data, we are better off making its data part of the connecting class, which we'll call *Takes*.

```
class Student (key name) {
    attribute string name;
    attribute string addr;
    relationship Set<Takes> courses
        inverse Takes::theStudent;
}

class Course (key number) {
    attribute integer number;
    attribute string title;
    relationship Set<Takes> students
        inverse Takes::theCourse;
}

class Takes (key (theStudent, theCourse)) {
    attribute string season;
    attribute integer year;
    relationship Student theStudent
        inverse Student::courses;
    relationship Course theCourse
        inverse Course::students;
}
```

3. **(14 pts.)** Consider a relation $R(A, B)$ that contains $r > 0$ tuples, and a relation $S(B, C)$ that contains $s > 0$ tuples. You may assume that neither R nor S contains duplicate tuples.

   (a) **(10 pts.)** For each of the expressions in (set-based) relational algebra in the following table, state the minimum and maximum number of tuples that could possibly be in the result of the expression. (The actual number will depend on the actual data.) Your statements should be based on r and s to the extent possible.

   Answer:

Expression	Minimum #tuples	Maximum #tuples
$\sigma_P(R)$ P a predicate	0	r
$\Pi_A(R)$ A an attribute list	1	r
$R \bowtie S$	0	$r \cdot s$
$R \cup \rho_{S(A,B)}(S)$	$\max(r, s)$	$r + s$
$\Pi_B(R) - (\Pi_B(R) - \Pi_B(S))$	0	$\min(r, s)$
$R \overset{\circ}{\bowtie}_L S$ (left outerjoin)	r	$r \cdot s$
$R \overset{\circ}{\bowtie} S$ (full outerjoin)	$\max(r, s)$	$r \cdot s$

(b) **(4 pts.)** Continue to assume neither R nor S contains duplicate tuples, but now use bag (multiset) instead of set-based relational algebra. Do any of your entries in the table above change? If so, state which ones and give the new entries.

Answer: The minimum for $\Pi_A(R)$ changes to r. The minimum for $R \cup \rho_{S(A,B)}(S)$ changes to $r + s$.

4. **(10 pts.)** Consider a table `CompScore(name, score)` containing scores on the database comp. For simplicity assume that both names and scores are unique in the table, and further assume the table has an odd number of rows. Use SQL to find the name of the student with the median score. Assume you do *not* have a built-in `median` operation.

Answer:

```
Select name
From   CompScore C1
Where  (Select Count(*) From CompScore C2
        Where C2.score < C1.score) =
       (Select Count(*) From CompScore C2
        Where C2.score > C1.score)
```

5. **(6 pts.)** Consider a database with three tables containing one tuple each:

   $R(A, B)$ contains tuple $\langle 1, 2 \rangle$
   $S(C, D)$ contains tuple $\langle 2, 3 \rangle$
   $T(E, F)$ contains tuple $\langle 2, 4 \rangle$

   Specify a minimal set of SQL referential-integrity constraints over the three-table schema such that when the single tuple in relation R is deleted, all three tables are made empty automatically.

   Answer:

```
S.C References R.B On Delete Cascade
T.E References R.B On Delete Cascade
```