# DBA 372: Database Management Systems

## Assignment 4

**1. Define the term *functional dependency*.**

**2. Why are some functional dependencies called *trivial*?**

**3. Give a set of FDs for the relation schema *R(A,B,C,D)* with primary key *AB* under which *R* is in 1NF but not in 2NF.**

**4. Give a set of FDs for the relation schema *R(A,B,C,D)* with primary key *AB* under which *R* is in 2NF but not in 3NF.**

**5. Consider the relation schema *R(A,B,C)*, which has the FD *B → C*. If *A* is a candidate key for *R*, is it possible for *R* to be in BCNF? If so, under what conditions? If not, explain why not.**

**6. Suppose we have a relation schema *R(A,B,C)* representing a relationship between two entity sets with keys *A* and *B*, respectively, and suppose that *R* has (among others) the FDs *A → B* and *B → A*. Explain what such a pair of dependencies means (i.e., what they imply about the relationship that the relation models).**

**7. Consider the relation shown in Figure 1.**

**1. List all the functional dependencies that this relation instance satisfies.**

**2. Assume that the value of attribute Z of the last record in the relation is changed**

**from z3 to z2. Now list all the functional dependencies that this relation instance**

**satisfies**

****

Figure

1. Let R be a relational schema and let X and Y be two subsets of the set of all attributes of R. We say Y is functionally dependent on X, written X *→* Y, if the Y-values are determined by the X-values.
2. Some functional dependencies are considered trivial because they contain superfluous attributes that do not need to be listed. Consider the FD: *A → AB*. By reflexivity, *A* always implies *A*, so that the *A* on the right hand side is not necessary and can be dropped. The proper form, without the trivial dependency would then be *A → B*.
3. Consider the set of FD: *AB → CD* and *B → C*. *AB* is obviously a key for this relation since *AB → CD* implies *AB → ABCD*. It is a primary key since there are no smaller subsets of keys that hold over *R(A,B,C,D)*. The FD: *B → C* violates 2NF since: *B is* a proper subset of the key *AB*
4. Consider the set of FD: *AB → CD* and *C → D*. *AB* is obviously a key for this relation since *AB → CD* implies *AB → ABCD*. It is a primary key since there are no smaller subsets of keys that hold over *R(A,B,C,D)*. The FD: *C → D* violates 3NF but not 2NF since: *C is not* a candidate key
5. The only way *R* could be in BCNF is if *B* includes a key, *i.e. B* is a key for R.
6. It means that the relationship is one to one. That is, each A entity corresponds to at most one *B* entity and vice-versa. (In addition, we have the dependency *AB → C*, from the semantics of a relationship set.).
7. 1.The following functional dependencies hold over R: Z → Y, X → Y, and XZ → Y

2. Same as part 1. Functional dependency set is unchanged.

**Consider the following collection of relations and dependencies. Assume that each relation is obtained through decomposition from a relation with attributes *ABCDEFGHI* and that all the known dependencies over relation *ABCDEFGHI* are listed for each question. (The questions are independent of each other, obviously, since the given dependencies over *ABCDEFGHI* are different.) For each (sub)relation:**

**(a) State the strongest normal form that the relation is in. (b) If it is not in BCNF, decompose it into a collection of BCNF relations.**

**1. *R1(A,C,B,D,E)*, *A → B*, *C → D***

**2. *R2(A,B,F), AC → E*, *B → F***

**3. *R3(A,D), D → G, G → H***

**4. *R4(D,C,H,G)*, *A → I, I → A***

**5. *R5(A,I,C,E)***

1. 1NF. BCNF decomposition: AB, CD, ACE.

2. 1NF. BCNF decomposition: AB, BF

3. BCNF.

4. BCNF.

5. BCNF.

**Suppose you are given a relation *R* with four attributes *ABCD*. For each of the following sets of FDs, assuming those are the only dependencies that hold for *R*, do the following: (a) Identify the candidate key(s) for *R*. (b) Identify the best normal form that *R* satisfies (1NF, 2NF, 3NF, or BCNF). (c) If *R* is not in BCNF, decompose it into a set of BCNF relations that preserve the dependencies.**

**1. *C → D, C → A, B → C***

**2. *B → C, D → A***

**3. *ABC → D, D → A***

**4. *A → B, BC → D, A → C***

**5. *AB → C, AB → D, C → A, D → B***

1. (a) Candidate keys: *B*

(b) *R* is in 2NF but not 3NF.

 (c) *C → D* and *C → A* both cause violations of BCNF. One way to obtain a (lossless) join preserving decomposition is to decompose R into *AC*, *BC*, and *CD*.

2. (a) Candidate keys: *BD*

(b) *R* is in 1NF but not 2NF.

(c) Both *B → C* and *D → A* cause BCNF violations. The decomposition: *AD*, *BC*, *BD* (obtained by first decomposing to *AD*, *BCD*) is BCNF and lossless and join-preserving.

3. (a) Candidate keys: *ABC*, *BCD*

(b) *R* is in 3NF but not BCNF.

(c) *ABCD* is not in BCNF since *D → A* and *D* is not a key. However if we split up *R* as *AD*, *BCD* we cannot preserve the dependency *ABC → D*. So there is no BCNF decomposition.

4. (a) Candidate keys: *A*

(b) *R* is in 2NF but not 3NF (because of the FD: *BC → D*).

(c) *BC → D* violates BCNF since *BC* does not contain a key. So we split up *R* as in: *BCD*, *ABC*.

5. (a) Candidate keys: *AB*, *BC*, *CD*, *AD*

(b) R is in 3NF but not BCNF (because of the FD: C *→* A).

(c) *C → A* and *D → B* both cause violations. So decompose into: *AC*, *BCD* but this does not preserve *AB → C* and *AB → D*, and *BCD* is still notBCNF because *D → B*. So we need to decompose further into: *AC*, *BD*, *CD*. However, when we attempt to revive the lost functional dependenciesby adding *ABC* and *ABD*, we that these relations are not in BCNF form.Therefore, there is no BCNF decomposition.