# 19

# SCHEMA REFINEMENT AND NORMAL FORMS

**Exercise 19.1** Briefly answer the following questions:

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 $B \rightarrow C$. 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 $A \rightarrow B$ and $B \rightarrow A$. Explain what such a pair of dependencies means (i.e., what they imply about the relationship that the relation models).

**Answer 19.1**

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 $\rightarrow$ Y, if the Y-values are determined by the X-values. More precisely, for any two tuples $r_1$ and $r_2$ in (any instance of) R

$$\pi_X(r_1) = \pi_X(r_2) \quad \Rightarrow \quad \pi_Y(r_1) = \pi_Y(r_2)$$

2. Some functional dependencies are considered trivial because they contain superfluous attributes that do not need to be listed. Consider the FD: $A \rightarrow AB$. 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 $A \rightarrow B$.

3. Consider the set of FD: $AB \rightarrow CD$ and $B \rightarrow C$. $AB$ is obviously a key for this relation since $AB \rightarrow CD$ implies $AB \rightarrow ABCD$. It is a primary key since there are no smaller subsets of keys that hold over *R(A,B,C,D)*. The FD: $B \rightarrow C$ violates 2NF since:
   - $C \in B$ is false; that is, it *is not* a trivial FD
   - $B$ *is not* a superkey
   - $C$ *is not* part of some key for $R$
   - $B$ *is* a proper subset of the key $AB$ (transitive dependency)
4. Consider the set of FD: $AB \rightarrow CD$ and $C \rightarrow D$. $AB$ is obviously a key for this relation since $AB \rightarrow CD$ implies $AB \rightarrow ABCD$. It is a primary key since there are no smaller subsets of keys that hold over *R(A,B,C,D)*. The FD: $C \rightarrow D$ violates 3NF but not 2NF since:
   - $D \in C$ is false; that is, it *is not* a trivial FD
   - $C$ *is not* a superkey
   - $D$ *is not* part of some key for $R$
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 $AB \rightarrow C$, from the semantics of a relationship set.)

**Exercise 19.2** Consider a relation $R$ with five attributes $ABCDE$. You are given the following dependencies: $A \rightarrow B$, $BC \rightarrow E$, and $ED \rightarrow A$.

1. List all keys for $R$.
2. Is $R$ in 3NF?
3. Is $R$ in BCNF?

**Answer 19.2**

1. CDE, ACD, BCD
2. R is in 3NF because B, E and A are all parts of keys.
3. R is not in BCNF because none of A, BC and ED contain a key.

The projection of the FD's of $R$ onto ABC gives us: $AB \rightarrow C$, $AC \rightarrow B$ and $BC \rightarrow A$. The projection of the FD's of $R$ onto $ACDE$ gives us: $AD \rightarrow E$ and The projection of the FD's of $R$ onto $ADG$ gives us: $AD \rightarrow G$ (by transitivity) The closure of this set of dependencies does not contain $E \rightarrow G$ nor does it contain $B \rightarrow D$. So this decomposition is not dependency preserving.

**Exercise 19.10** Suppose you are given a relation *R(A,B,C,D)*. For each of the following sets of FDs, assuming they are the only dependencies that hold for $R$, do the following: (a) Identify the candidate key(s) for $R$. (b) State whether or not the proposed decomposition of $R$ into smaller relations is a good decomposition and briefly explain why or why not.

1. $B \rightarrow C$, $D \rightarrow A$; decompose into $BC$ and $AD$.
2. $AB \rightarrow C$, $C \rightarrow A$, $C \rightarrow D$; decompose into $ACD$ and $BC$.
3. $A \rightarrow BC$, $C \rightarrow AD$; decompose into $ABC$ and $AD$.
4. $A \rightarrow B$, $B \rightarrow C$, $C \rightarrow D$; decompose into $AB$ and $ACD$.

5. $A \rightarrow B$, $B \rightarrow C$, $C \rightarrow D$; decompose into $AB$, $AD$ and $CD$.

**Answer 19.10**

1. Candidate key(s): $BD$. The decomposition into $BC$ and $AD$ is unsatisfactory because it is lossy (the join of $BC$ and $AD$ is the cartesian product which could be much bigger than $ABCD$)

2. Candidate key(s): $AB$, $BC$. The decomposition into ACD and BC is lossless since $ACD \cap BC$ (which is $C$) $\rightarrow ACD$. The projection of the FD's on ACD include C $\rightarrow$ D, C $\rightarrow$ A (so C is a key for ACD) and the projection of FD on BC produces no nontrivial dependencies. In particular this is a BCNF decomposition (check that R is not!). However, it is not dependency preserving since the dependency AB $\rightarrow$ C is not preserved. So to enforce preservation of this dependency (if we do not want to use a join) we need to add ABC which introduces redundancy. So implicitly there is some redundancy across relations (although none inside ACD and BC).

3. Candidate key(s): $A$, $C$. Since $A$ and $C$ are both candidate keys for $R$, it is already in BCNF. So from a normalization standpoint it makes no sense to decompose $R$. Further more, the decompose is not dependency-preserving since $C \rightarrow AD$ can no longer be enforced.

4. Candidate key(s): $A$. The projection of the dependencies on $AB$ are: $A \rightarrow B$ and those on ACD are: $A \rightarrow C$ and $C \rightarrow D$ (rest follow from these). The scheme $ACD$ is not even in 3NF, since $C$ is not a superkey, and $D$ is not part of a key. This is a lossless-join decomposition (since $A$ is a key), but not dependency preserving, since $B \rightarrow C$ is not preserved.

5. Candidate key(s): $A$ (just as before) This is a lossless BCNF decomposition (easy to check!) This is, however, not dependency preserving ($B$ *consider* $\rightarrow C$). So it is not free of (implied) redundancy. This is not the best decomposition ( the decomposition $AB$, $BC$, $CD$ is better.)