CHAPTER 7

Relational-Database Design

Solutions to Practice Exercises

7.1 A decomposition $\{R_1,\ R_2\}$ is a lossless-join decomposition if $R_1 \cap R_2 \rightarrow R_1$ or $R_1 \cap R_2 \rightarrow R_2$. Let $R_1 = (A,\ B,\ C)$, $R_2 = (A,\ D,\ E)$, and $R_1 \cap R_2 = A$. Since A is a candidate key (see Practice Exercise 7.6), Therefore $R_1 \cap R_2 \rightarrow R_1$.

7.2 The nontrivial functional dependencies are: $A \rightarrow B$ and $C \rightarrow B$, and a dependency they logically imply: $AC \rightarrow B$. There are 19 trivial functional dependencies of the form $\alpha \rightarrow \beta$, where $\beta \subseteq \alpha$. *C* does not functionally determine *A* because the first and third tuples have the same *C* but different *A* values. The same tuples also show *B* does not functionally determine *A*. Likewise, *A* does not functionally determine *C* because the first two tuples have the same *A* value and different *C* values. The same tuples also show *B* does not functionally determine *C*.

7.3 Let $Pk(r)$ denote the primary key attribute of relation r.

- The functional dependencies Pk(*account*) $\rightarrow$ Pk (*customer*) and Pk(*customer*) $\rightarrow$ Pk(*account*) indicate a one-to-one relationship because any two tuples with the same value for account must have the same value for customer, and any two tuples agreeing on customer must have the same value for account.
- The functional dependency Pk(*account*) $\rightarrow$ Pk(*customer*) indicates a many-to-one relationship since any account value which is repeated will have the same customer value, but many account values may have the same customer value.

7.4 To prove that:

$$\text{if } \alpha \rightarrow \beta \text{ and } \alpha \rightarrow \gamma \text{ then } \alpha \rightarrow \beta\gamma$$

Following the hint, we derive:

$\alpha \rightarrow \beta$	given
$\alpha\alpha \rightarrow \alpha\beta$	augmentation rule
$\alpha \rightarrow \alpha\beta$	union of identical sets
$\alpha \rightarrow \gamma$	given
$\alpha\beta \rightarrow \gamma\beta$	augmentation rule
$\alpha \rightarrow \beta\gamma$	transitivity rule and set union commutativity

7.5 Proof using Armstrong's axioms of the Pseudotransitivity Rule: if $\alpha \rightarrow \beta$ and $\gamma\beta \rightarrow \delta$, then $\alpha\gamma \rightarrow \delta$.

$\alpha \rightarrow \beta$	given
$\alpha\gamma \rightarrow \gamma\beta$	augmentation rule and set union commutativity
$\gamma\beta \rightarrow \delta$	given
$\alpha\gamma \rightarrow \delta$	transitivity rule

7.6 Note: It is not reasonable to expect students to enumerate all of F^+. Some shorthand representation of the result should be acceptable as long as the nontrivial members of F^+ are found.

Starting with $A \rightarrow BC$, we can conclude: $A \rightarrow B$ and $A \rightarrow C$.

Since $A \rightarrow B$ and $B \rightarrow D$, $A \rightarrow D$	(decomposition, transitive)
Since $A \rightarrow CD$ and $CD \rightarrow E$, $A \rightarrow E$	(union, decomposition, transitive)
Since $A \rightarrow A$, we have	(reflexive)
$A \rightarrow ABCDE$ from the above steps	(union)
Since $E \rightarrow A$, $E \rightarrow ABCDE$	(transitive)
Since $CD \rightarrow E$, $CD \rightarrow ABCDE$	(transitive)
Since $B \rightarrow D$ and $BC \rightarrow CD$, $BC \rightarrow ABCDE$	(augmentative, transitive)
Also, $C \rightarrow C$, $D \rightarrow D$, $BD \rightarrow D$, etc.	

Therefore, any functional dependency with A, E, BC, or CD on the left hand side of the arrow is in F^+, no matter which other attributes appear in the FD. Allow * to represent any set of attributes in R, then F^+ is $BD \rightarrow B$, $BD \rightarrow D$, $C \rightarrow C$, $D \rightarrow D$, $BD \rightarrow BD$, $B \rightarrow D$, $B \rightarrow B$, $B \rightarrow BD$, and all FDs of the form $A* \rightarrow \alpha$, $BC* \rightarrow \alpha$, $CD* \rightarrow \alpha$, $E* \rightarrow \alpha$ where α is any subset of $\{A, B, C, D, E\}$. The candidate keys are A, BC, CD, and E.

7.7 The given set of FDs F is:

$$
\begin{aligned}
A &\rightarrow BC \\
CD &\rightarrow E \\
B &\rightarrow D \\
E &\rightarrow A
\end{aligned}
$$

The left side of each FD in F is unique. Also none of the attributes in the left side or right side of any of the FDs is extraneous. Therefore the canonical cover F_c is equal to F.

7.8 The algorithm is correct because:

- If A is added to $result$ then there is a proof that $\alpha \rightarrow A$. To see this, observe that $\alpha \rightarrow \alpha$ trivially so α is correctly part of $result$. If $A \notin \alpha$ is added to $result$ there must be some FD $\beta \rightarrow \gamma$ such that $A \in \gamma$ and β is already a subset of $result$. (Otherwise $fdcount$ would be nonzero and the **if** condition would be false.) A full proof can be given by induction on the depth of recursion for an execution of **addin**, but such a proof can be expected only from students with a good mathematical background.
- If $A \in \alpha^+$, then A is eventually added to $result$. We prove this by induction on the length of the proof of $\alpha \rightarrow A$ using Armstrong's axioms. First observe that if procedure **addin** is called with some argument β, all the attributes in β will be added to $result$. Also if a particular FD's *fdcount* becomes 0, all the attributes in its tail will definitely be added to $result$. The base case of the proof, $A \in \alpha \Rightarrow A \in \alpha^+$, is obviously true because the first call to **addin** has the argument α. The inductive hypotheses is that if $\alpha \rightarrow A$ can be proved in n steps or less then $A \in result$. If there is a proof in $n+1$ steps that $\alpha \rightarrow A$, then the last step was an application of either reflexivity, augmentation or transitivity on a fact $\alpha \rightarrow \beta$ proved in n or fewer steps. If reflexivity or augmentation was used in the $(n+1)^{st}$ step, A must have been in $result$ by the end of the n^{th} step itself. Otherwise, by the inductive hypothesis $\beta \subseteq result$. Therefore the dependency used in proving $\beta \rightarrow \gamma$, $A \in \gamma$ will have $fdcount$ set to 0 by the end of the n^{th} step. Hence A will be added to $result$.

To see that this algorithm is more efficient than the one presented in the chapter note that we scan each FD once in the main program. The resulting array $appears$ has size proportional to the size of the given FDs. The recursive calls to **addin** result in processing linear in the size of $appears$. Hence the algorithm has time complexity which is linear in the size of the given FDs. On the other hand, the algorithm given in the text has quadratic time complexity, as it may perform the loop as many times as the number of FDs, in each loop scanning all of them once.

7.9 **a.** The query is given below. Its result is non-empty if and only if $b \rightarrow c$ does not hold on r.

select *b*
from *r*
group by *b*
having count(distinct *c*) > 1

b.

```
create assertion b-to-c check
    (not exists
        (select b
         from r
         group by b
         having count(distinct c) > 1
        )
    )
```

7.10 Consider some tuple t in u.

Note that $r_i = \Pi_{R_i}(u)$ implies that $t[R_i] \in r_i$, $1 \leq i \leq n$. Thus,

$$t[R_1] \bowtie t[R_2] \bowtie \ldots \bowtie t[R_n] \in r_1 \bowtie r_2 \bowtie \ldots \bowtie r_n$$

By the definition of natural join,

$$t[R_1] \bowtie t[R_2] \bowtie \ldots \bowtie t[R_n] = \Pi_\alpha (\sigma_\beta (t[R_1] \times t[R_2] \times \ldots \times t[R_n]))$$

where the condition β is satisfied if values of attributes with the same name in a tuple are equal and where $\alpha = U$. The cartesian product of single tuples generates one tuple. The selection process is satisfied because all attributes with the same name must have the same value since they are projections from the same tuple. Finally, the projection clause removes duplicate attribute names.

By the definition of decomposition, $U = R_1 \cup R_2 \cup \ldots \cup R_n$, which means that all attributes of t are in $t[R_1] \bowtie t[R_2] \bowtie \ldots \bowtie t[R_n]$. That is, t is equal to the result of this join.

Since t is any arbitrary tuple in u,

$$u \subseteq r_1 \bowtie r_2 \bowtie \ldots \bowtie r_n$$

7.11 The dependency $B \rightarrow D$ is not preserved. F_1, the restriction of F to (A, B, C) is $A \rightarrow ABC$, $A \rightarrow AB$, $A \rightarrow AC$, $A \rightarrow BC$, $A \rightarrow B$, $A \rightarrow C$, $A \rightarrow A$, $B \rightarrow B$, $C \rightarrow C$, $AB \rightarrow AC$, $AB \rightarrow ABC$, $AB \rightarrow BC$, $AB \rightarrow AB$, $AB \rightarrow A$, $AB \rightarrow B$, $AB \rightarrow C$, AC (same as AB), BC (same as AB), ABC (same as AB). F_2, the restriction of F to (C, D, E) is $A \rightarrow ADE$, $A \rightarrow AD$, $A \rightarrow AE$, $A \rightarrow DE$, $A \rightarrow A$, $A \rightarrow D$, $A \rightarrow E$, $D \rightarrow D$, E (same as A), AD, AE, DE, ADE (same as A). $(F_1 \cup F_2)^+$ is easily seen not to contain $B \rightarrow D$ since the only FD in $F_1 \cup F_2$ with B as the left side is $B \rightarrow B$, a trivial FD. We shall see in Practice Exercise 7.13 that $B \rightarrow D$ is indeed in F^+. Thus $B \rightarrow D$ is not preserved. Note that $CD \rightarrow ABCDE$ is also not preserved.

A simpler argument is as follows: F_1 contains no dependencies with D on the right side of the arrow. F_2 contains no dependencies with B on the left side of the arrow. Therefore for $B \rightarrow D$ to be preserved there must be an FD $B \rightarrow \alpha$ in F_1^+ and $\alpha \rightarrow D$ in F_2^+ (so $B \rightarrow D$ would follow by transitivity). Since the intersection of the two schemes is A, $\alpha = A$. Observe that $B \rightarrow A$ is not in F_1^+ since $B^+ = BD$.

7.12 Let F be a set of functional dependencies that hold on a schema R. Let $\sigma = \{R_1, R_2, \ldots, R_n\}$ be a dependency-preserving 3NF decomposition of R. Let X be a candidate key for R.

Consider a legal instance r of R. Let $j = \Pi_X(r) \bowtie \Pi_{R_1}(r) \bowtie \Pi_{R_2}(r) \ldots \bowtie \Pi_{R_n}(r)$. We want to prove that $r = j$.

We claim that if t_1 and t_2 are two tuples in j such that $t_1[X] = t_2[X]$, then $t_1 = t_2$. To prove this claim, we use the following inductive argument –
Let $F' = F_1 \cup F_2 \cup \ldots \cup F_n$, where each F_i is the restriction of F to the schema R_i in σ. Consider the use of the algorithm given in Figure 7.9 to compute the closure of X under F'. We use induction on the number of times that the for loop in this algorithm is executed.

- *Basis* : In the first step of the algorithm, $result$ is assigned to X, and hence given that $t_1[X] = t_2[X]$, we know that $t_1[result] = t_2[result]$ is true.
- *Induction Step* : Let $t_1[result] = t_2[result]$ be true at the end of the k th execution of the for loop.
  Suppose the functional dependency considered in the $k + 1$ th execution of the for loop is $\beta \rightarrow \gamma$, and that $\beta \subseteq result$. $\beta \subseteq result$ implies that $t_1[\beta] = t_2[\beta]$ is true. The facts that $\beta \rightarrow \gamma$ holds for some attribute set R_i in σ, and that $t_1[R_i]$ and $t_2[R_i]$ are in $\Pi_{R_i}(r)$ imply that $t_1[\gamma] = t_2[\gamma]$ is also true. Since γ is now added to $result$ by the algorithm, we know that $t_1[result] = t_2[result]$ is true at the end of the $k + 1$ th execution of the for loop.

Since σ is dependency-preserving and X is a key for R, all attributes in R are in $result$ when the algorithm terminates. Thus, $t_1[R] = t_2[R]$ is true, that is, $t_1 = t_2$ – as claimed earlier.

Our claim implies that the size of $\Pi_X(j)$ is equal to the size of j. Note also that $\Pi_X(j) = \Pi_X(r) = r$ (since X is a key for R). Thus we have proved that the size of j equals that of r. Using the result of Practice Exercise 7.10, we know that $r \subseteq j$. Hence we conclude that $r = j$.

Note that since X is trivially in 3NF, $\sigma \cup \{X\}$ is a dependency-preserving lossless-join decomposition into 3NF.

7.13 Given the relation $R' = (A, B, C, D)$ the set of functional dependencies $F' = A \rightarrow B, C \rightarrow D, B \rightarrow C$ allows three distinct BCNF decompositions.

$$R_1 = \{(A, B), (C, D), (B, C)\}$$

is in BCNF as is

$$R_2 = \{(A, B), (C, D), (A, C)\}$$

$$R_2 = \{(A, B), (C, D), (A, C)\}$$

$$R_3 = \{(B, C), (A, D), (A, B)\}$$

7.14 Suppose R is in 3NF according to the textbook definition. We show that it is in 3NF according to the definition in the exercise. Let A be a nonprime attribute in

R that is transitively dependent on a key α for R. Then there exists $\beta \subseteq R$ such that $\beta \rightarrow A$, $\alpha \rightarrow \beta$, $A \notin \alpha$, $A \notin \beta$, and $\beta \rightarrow \alpha$ does not hold. But then $\beta \rightarrow A$ violates the textbook definition of 3NF since

- $A \notin \beta$ implies $\beta \rightarrow A$ is nontrivial
- Since $\beta \rightarrow \alpha$ does not hold, β is not a superkey
- A is not any candidate key, since A is nonprime

Now we show that if R is in 3NF according to the exercise definition, it is in 3NF according to the textbook definition. Suppose R is not in 3NF according the the textbook definition. Then there is an FD $\alpha \rightarrow \beta$ that fails all three conditions. Thus

- $\alpha \rightarrow \beta$ is nontrivial.
- α is not a superkey for R.
- Some A in $\beta - \alpha$ is not in any candidate key.

This implies that A is nonprime and $\alpha \rightarrow A$. Let γ be a candidate key for R. Then $\gamma \rightarrow \alpha$, $\alpha \rightarrow \gamma$ does not hold (since α is not a superkey), $A \notin \alpha$, and $A \notin \gamma$ (since A is nonprime). Thus A is transitively dependent on γ, violating the exercise definition.

7.15 Referring to the definitions in Practice Exercise 7.14, a relation schema R is said to be in 3NF if there is no non-prime attribute A in R for which A is transitively dependent on a key for R.

We can also rewrite the definition of 2NF given here as :
"A relation schema R is in 2NF if no non-prime attribute A is partially dependent on any candidate key for R."

To prove that every 3NF schema is in 2NF, it suffices to show that if a non-prime attribute A is partially dependent on a candidate key α, then A is also transitively dependent on the key α.

Let A be a non-prime attribute in R. Let α be a candidate key for R. Suppose A is partially dependent on α.

- From the definition of a partial dependency, we know that for some proper subset β of α, $\beta \rightarrow A$.
- Since $\beta \subset \alpha$, $\alpha \rightarrow \beta$. Also, $\beta \rightarrow \alpha$ does not hold, since α is a candidate key.
- Finally, since A is non-prime, it cannot be in either β or α.

Thus we conclude that $\alpha \rightarrow A$ is a transitive dependency. Hence we have proved that every 3NF schema is also in 2NF.

7.16 The relation schema $R = (A, B, C, D, E)$ and the set of dependencies

$$A \rightarrow\rightarrow BC$$
$$B \rightarrow\rightarrow CD$$
$$E \rightarrow\rightarrow AD$$

constitute a BCNF decomposition, however it is clearly not in 4NF. (It is BCNF because all FDs are trivial).