Chapter 6

DATABASES AND NORMAL FORMS

Before delving into relational databases, let us review keys and superkeys in light of what we know about FDs.

Recall: Given a relation scheme R, a key on R is a subset K of R such that for any permissible relation $r(R)$, there are no two distinct tuples t_1 and t_2 in r such that $t_1(K) = t_2(K)$, and no proper subset K' of K has this property. Remember that for some permissible relations on R, K' could be a key, but we are concerned with all permissible relations on R. A superkey is any set of attributes containing a key.

Example 6.1 In the relation *leaves* in Table 6.1 we have a list of departures from airports. At first, {FROM, DEPARTS} might seem to be a key for *leaves*, but when we consider that there can be two flights from the same city at the same time (say we add the tuple ⟨234 Denver 9:30p O'Hare⟩), we see that {FROM, DEPARTS, TO} is actually the key we want.

Table 6.1 The relation *leaves*.

leaves(FLIGHT	FROM	DEPARTS	TO)
16	JFK	9:10a	O'Hare
142	Denver	10:32a	O'Hare
146	Denver	9:30p	JFK
197	Atlanta	1:15p	Houston

In terms of FDs, a key for a scheme R is a subset K of R such that any permissible relation $r(R)$ satisfies $K \rightarrow R$, but no proper subset of K has this property. If K is a key for r, then there are no distinct tuples t_1 and t_2 in r such that t_1 and t_2 have the same K-values. Therefore, when testing the FD $K \rightarrow R$, if we ever have $t_1(K) = t_2(K)$, we must have $t_1 = t_2$, which is to say $t_1(R) = t_2(R)$. A superkey is a subset K of R such that $K \rightarrow R$, but there is no requirement for minimality.

6.1 DATABASES AND DATABASE SCHEMES

In what follows, we shall assume that a relation scheme R is composed of two parts, S and $\mathbf{K}$, where S is a set of attributes and $\mathbf{K}$ is a set of designated keys. (Remember that a designated key can be a superkey.) We write this situation as $R = (S, \mathbf{K})$. We shall sometimes still use R in place of S when discussing sets of attributes, such as using $X \subseteq R$ for $X \subseteq S$.

Definition 6.1 Let $\mathbf{U}$ be a set of attributes, each with an associated domain. A *relational database scheme* $\mathbf{R}$ *over* $\mathbf{U}$ is a collection of relation schemes $\{R_1, R_2, \ldots, R_p\}$, where $R_i = (S_i, \mathbf{K}_i)$, $1 \leq i \leq p$,

$$\bigcup_{i=1}^{p} S_i = \mathbf{U},$$

and $S_i \neq S_j$ if $i \neq j$.
A *relational database* d *on database scheme* $\mathbf{R}$ is a collection of relations $\{r_1, r_2, \ldots, r_p\}$ such that for each relation scheme $R = (S, \mathbf{K})$ in $\mathbf{R}$ there is a relation r in d such that r is a relation on S that satisfies every key in $\mathbf{K}$. We abuse set notation and always assume that r_i is the relation on S_i.

Example 6.2 Table 6.2 shows a database $d = \{$*flies, times*$\}$ on the database scheme $\mathbf{R} = \{$(PILOT FLIGHT DATE, {PILOT DATE}), (FLIGHT DEPARTS, {FLIGHT})$\}$. Relations *flies* and *times* are projections of the relation *assign* in Table 4.1.

Definition 6.2 A relation scheme $R = (S, \mathbf{K})$ *embodies* the FD $K \rightarrow R$ if K is a designated key in $\mathbf{K}$.

Notice that we are already using R for S.

Definition 6.3 A database scheme $\mathbf{R} = \{R_1, R_2, \ldots, R_p\}$ *represents* the set of FDs $G = \{X \rightarrow Y \mid \text{some } R_i \text{ in } \mathbf{R} \text{ embodies } X \rightarrow Y\}$. $\mathbf{R}$ *completely characterizes* a set of FDs F if $F \equiv G$.

Example 6.3 The database scheme $\mathbf{R}$ in Example 6.2 represents the set of FDs $G = \{$PILOT DATE $\rightarrow$ PILOT FLIGHT DATE, FLIGHT $\rightarrow$ FLIGHT DEPARTS$\}$. $\mathbf{R}$ completely characterizes the set $F = \{$PILOT DATE $\rightarrow$ FLIGHT DEPARTS, FLIGHT $\rightarrow$ DEPARTS$\}$.

Table 6.2 The relations *flies* and *times*, comprising a database.

flies(PILOT	FLIGHT	DATE)	*times*(FLIGHT	DEPARTS)
Cushing	83	9 Aug	83	10:15a
Cushing	116	10 Aug	116	1:25p
Clark	281	8 Aug	281	5:50a
Clark	301	12 Aug	301	6:35p
Clark	83	11 Aug	412	1:25p
Chin	83	13 Aug		
Chin	116	12 Aug		
Copley	281	9 Aug		
Copley	281	13 Aug		
Copley	412	15 Aug		

We may wish to place constraints upon relations in our databases other than those imposed by the designated keys of the relation schemes. In some such cases we specify a set of FDs F that the relations in the database must satisfy. Not every FD in F will apply to every relation in the database. How could the FD $A\ B \rightarrow C$ apply to a relation $r(A\ C)$? We must modify the definition of satisfies to suit databases.

Definition 6.4 Let R be a relation scheme. The FD $X \rightarrow Y$ *applies* to R if $X \rightarrow Y$ is an FD over R.

Definition 6.5 Let $d = \{r_1, r_2, \ldots, r_p\}$ be a database on scheme $\mathbf{R} = \{R_1, R_2, \ldots, R_p\}$ over $\mathbf{U}$. Let F be a set of FDs over $\mathbf{U}$. Database d *satisfies* F if for every FD $X \rightarrow Y$ in F^+ and every R_i in $\mathbf{R}$, if $X \rightarrow Y$ applies to R_i, then r_i satisfies $X \rightarrow Y$. Let G be the set of all FDs in F^+ that apply to some scheme R_i in $\mathbf{R}$. Any FD in G^+ is said to be *enforceable* on $\mathbf{R}$; any FD in $F^+ - G^+$ is *unenforceable* on $\mathbf{R}$. The set F is *enforceable* on $\mathbf{R}$ if every FD in F^+ is enforceable on $\mathbf{R}$, that is, $G \equiv F$.

To show that a set F is enforceable on $\mathbf{R}$, it is sufficient to find a set $F' \equiv F$ such that every FD in F' applies to some scheme R in $\mathbf{R}$.

Definition 6.6 A database d on scheme $\mathbf{R}$ *obeys* a set of FDs F if F is enforceable on $\mathbf{R}$ and d satisfies F^+.

If F' is a set of FDs as described above, d obeys F if d satisfies F'.

Example 6.4 Let $\mathbf{R} = \{R_1, R_2, R_3\}$, where $R_1 = A\,B\,C$, $R_2 = B\,C\,D$, and $R_3 = D\,E$. Let $F = \{A \to B\,C,\ C \to A,\ A \to D,\ D \to E,\ A \to E\}$. FDs $A \to D$ and $A \to E$ do not apply to any relation scheme in $\mathbf{R}$. However, F is enforceable on $\mathbf{R}$, since $G = \{A \to B\,C,\ C \to A,\ C \to D,\ D \to E\}$ is equivalent to F and every FD in G applies to some relation scheme in $\mathbf{R}$. The set $F' = \{A \to D\}$ is not enforceable on $\mathbf{R}$.

6.2 NORMAL FORMS FOR DATABASES

We are about to define three normal forms for databases. A normal form is a restriction on the database scheme that presumably precludes certain undesirable properties from the database. Normal forms are first defined for a relation scheme in a database scheme, and are then extended to apply to the database scheme as a whole.

6.2.1 First Normal Form

A relation scheme R is in *first normal form* (1NF) if the values in $dom(A)$ are atomic for every attribute A in R. That is, the values in the domain are not lists or sets of values or composite values. A database scheme $\mathbf{R}$ is in *first normal form* if every relation scheme in $\mathbf{R}$ is in first normal form. All the examples we have seen thus far have been in 1NF. The definition of atomic is hazy; a value that is atomic in one application could be non-atomic in another. For a general guideline, a value is non-atomic if the application deals with only a part of the value.

Example 6.5 Shown below is a relation *born* that lists birthdays.

born(NAME	BIRTHDATE)
Allen	June 7, 1949
Alfred	March 21, 1933
Alphonse	April 30, 1959
Alice	December 12, 1963

If we are ever interested only in the month or year a person was born, the relation *born* is not in 1NF, since we are dealing with part of a value. To be in 1NF in this case, the attribute BIRTHDATE should be broken up as shown below.

born

NAME	BMONTH	BDAY	BYEAR
Allen	June	7	1949
Alfred	March	21	1933
Alphonse	April	30	1959
Alice	December	12	1963

Example 6.6 The relation *gender,* shown below, is not in 1NF because it contains values that are sets of atomic values.

gender

NAME	SEX
{John, Jean, Ivan}	male
{Mary, Marie}	female

To be in 1NF, *gender* should be stored like this:

gender

NAME	SEX
John	male
Jean	male
Ivan	male
Mary	female
Marie	female

What is the advantage of 1NF? It may not be possible to express FDs in as great detail as we would like without 1NF. Suppose we want to add an attribute SIGN that gives the astrological sign of the person to the first relation *born* in Example 6.5. SIGN is functionally dependent only on the month and day of birth, and not on the year of birth. However, the best we can do the first relation of Example 6.5 is BIRTHDATE $\rightarrow$ SIGN, which would allow two people born on the same day in different years to have different signs. In the second relation *born,* there is no problem. We can write the FD BMONTH BDAY $\rightarrow$ SIGN, which prevents the problem with the previous FD.

Updates can also be a problem if a scheme is not in 1NF. Suppose we want to process the update CH(*gender*; Jean, male; SEX = female). The result of this update is ambiguous in the first relation gender in Example 6.6. Do we move Jean from one set to the other or do we change male to female? The update is unambiguous when applied to the second version of *gender* in Example 6.6.

6.2.2 Anomalies and Data Redundancy

Second and third normal forms arise from trying to avoid anomalies when updating relations and data redundancy in relations. An update anomaly is an unwanted side effect that results from changing a relation.

Consider the relation *assign* in Table 6.3.

Table 6.3 The relation *assign*

assign (FLIGHT	DAY	PILOT	GATE)
112	6 June	Bosley	7
112	7 June	Brooks	7
203	9 June	Bosley	12

FLIGHT DAY is a key for *assign*, and the relation must also satisfy the FD FLIGHT → GATE. We would like to update the relation by specifying values for the key and then giving values for the remaining attributes. However, if we perform

$$\text{CH}(\textit{assign};\ 112,\ 6\ \text{June};\ \text{PILOT} = \text{Bosley},\ \text{GATE} = 8),$$

the relation will violate the FD FLIGHT → GATE. To avoid violating the FD, every time an update is made, we have to scan the relation and update the gate number every place the flight number appears. We wanted to change only one tuple. Furthermore, the flight number-gate number information is duplicated in the relation, thus making the data redundant.

We are better off, with respect to updates and redundancy, if we represent the same information as a database of two relations, *passign* and *gassign*, as shown below.

passign (FLIGHT	DAY	PILOT)
112	6 June	Bosley
112	7 June	Brooks
203	9 June	Bosley

gassign (FLIGHT	GATE)
112	7
203	12

We can reconstruct the original relation *assign* by taking *passign* $\bowtie$ *gassign*. The update anomaly no longer exists, since only one tuple needs to be altered to change a gate assignment. We have also removed some data redundancy, since flight number-gate number pairs are only recorded once.

6.2.3 Second Normal Form

Definition 6.7 Given a set of FDs F and an FD $X \rightarrow Y$ in F^+, Y is *partially dependent* upon X under F if $X \rightarrow Y$ is not left-reduced. That is, there is a proper subset X' of X such that $X' \rightarrow Y$ is in F^+. If $X \rightarrow Y$ is left-reduced, then Y is *fully dependent* on X.

Example 6.7 Let F = {FLIGHT DAY → PILOT GATE, FLIGHT → GATE}. GATE is partially dependent upon FLIGHT DAY, while PILOT is fully dependent upon FLIGHT DAY.

Definition 6.8 Given a relation scheme R, an attribute A in R, and a set of FDs F over R, attribute A is *prime* in R with respect to F if A is contained in some key of R. Otherwise A is *nonprime* in R.

Caveat Lector: Do not confuse the keys in the definition of prime with designated keys for R, as the latter may actually be superkeys. Also, there may be keys for R that are not designated.

Example 6.8 Let R = FLIGHT DAY PILOT GATE and let F be as in Example 6.7. FLIGHT and DAY are prime, PILOT and GATE are nonprime. (We allow the possibility that a pilot can have two flights in one day, so PILOT DAY is not a key.)

Definition 6.9 A relation scheme R is in *second normal form* (2NF) with respect to a set of FDs F if it is in 1NF and every nonprime attribute is fully dependent on every key of R. A database scheme **R** is in *second normal form* with respect to F if every relation scheme R in **R** is in 2NF with respect to F.

Example 6.9 Let R and F be as in Example 6.8 and let $\mathbf{R} = \{R\}$. **R** is not in 2NF because in R, GATE is partially dependent on FLIGHT DAY. If we let **R** = {FLIGHT DAY PILOT, FLIGHT GATE}, then **R** is in 2NF. FLIGHT is now a key for the relation scheme FLIGHT GATE.

6.2.4 Third Normal Form

Consider the relation *assign* in Table 6.4. It is similar to the relation *assign* in Table 6.3. We again assume *assign* has the key FLIGHT DAY and further satisfies the FDs PILOT-ID → NAME and NAME → PILOT-ID.

Table 6.4 The relation *assign*

assign(FLIGHT	DAY	PILOT-ID	NAME)
112	6 June	31174	Bosley
112	7 June	30046	Brooks
203	9 June	31174	Bosley

If we make the update

CH(*assign*; 112, 6 June; PILOT-ID = 31039, NAME = Bosley),

we violate the FD NAME → PILOT-ID. We also have redundant pilot identification-pilot name pairs. The cause of the problem here is not a partially dependent nonprime attribute, although the solution is the same. We represent the relation as a database, as shown below.

passign(FLIGHT	DAY	PILOT-ID)
112	6 June	31174
112	7 June	30046
203	9 June	31174

ident(PILOT-ID	NAME)
31174	Bosley
30046	Brooks

We can still retrieve the original relation through a join.

Definition 6.10 Given a relation scheme R, a subset X of R, an attribute A in R, and a set of FDs F, A is *transitively dependent* upon X in R if there is a subset Y of R with $X \to Y$, $Y \not\to X$, and $Y \to A$ under F and $A \notin X\,Y$.

Example 6.10 Let R = FLIGHT DAY PILOT-ID NAME and let F = {FLIGHT DAY → PILOT-ID NAME, PILOT-ID → NAME, NAME → PILOT-ID}. NAME is transitively dependent upon FLIGHT DAY, since FLIGHT DAY → PILOT-ID, PILOT-ID $\not\to$ FLIGHT DAY, and PILOT-ID → NAME. PILOT-ID fills the role of Y in the definition.

Definition 6.11 A relation scheme R is in *third normal form* (3NF) with respect to a set of FDs F if it is in 1NF and no nonprime attribute in R is transitively dependent upon a key of R. A database scheme **R** is in *third normal form* with respect to F if every relation scheme R in **R** is in third normal form with respect to F.

Example 6.11 Let R and F be as in Example 6.10 and let **R** = $\{R\}$. **R** is not in 3NF with respect to F because of the transitive dependency of NAME on FLIGHT DAY. If $\mathbf{R} = \{$FLIGHT DAY PILOT-ID, PILOT-ID NAME$\}$, then $\mathbf{R}$ is in 3NF with respect to F.

Lemma 6.1 Any relation scheme R that is in 3NF with respect to F is in 2NF with respect to F.

Proof We show that a partial dependency implies a transitive dependency. Suppose nonprime attribute A in R is partially dependent upon key $K \subseteq R$. Then there is a proper subset K' of K such that $F \models K' \rightarrow A$. $K' \not\rightarrow K$, since then K' would be a key for R, and keys cannot properly contain keys. Also, $A \notin K$, since K is a key and A is nonprime. So $K \rightarrow K'$, $K' \not\rightarrow K$, $K' \rightarrow A$, and $A \notin K\,K' = K$. Therefore A is transitively dependent upon K.

6.3 NORMALIZATION THROUGH DECOMPOSITION

It is always possible to start with any relation scheme R that is not in 3NF with respect to a set of FDs F and decompose it into a database scheme that is in 3NF with respect to F. Decomposing a relation scheme means breaking the relation scheme into a pair of relation schemes R_1 and R_2 (possibly intersecting) such that any relation $r(R)$ that satisfies F decomposes losslessly onto R_1 and R_2. That is, $\pi_{R_1}(r) \bowtie \pi_{R_2}(r) = r$. We may have to repeat the decomposition process on R_1 and R_2 if either of them is not in 3NF. We keep decomposing until all the relations we have are in 3NF with respect to F.

Suppose we have a transitive dependency upon a key in R. We have a key $K \subseteq R$, a set $Y \subseteq R$, and a nonprime attribute A in R with $K \rightarrow Y$, $Y \not\rightarrow K$, $Y \rightarrow A$ under F and $A \notin K\,Y$. Let $R_1 = R - A$ and $R_2 = YA$. If we have designated keys for our relation scheme, say $R = (S, \mathbf{K})$, then let $\mathbf{K}$ be the set of designated keys for R_1 and let $\{Y\}$ be the set of designated keys for R_2. It is possible that some designated key K in $\mathbf{K}$ contains A. If so, K is a superkey for R. Let $K' = K - A$. K' is still a superkey for R, since A cannot be part of any key for R. Replace K by K' in $\mathbf{K}$.

We have removed one transitive dependency from R, and for any $r(R)$ satisfying F, $r = \pi_{R_1}(r) \bowtie \pi_{R_2}(r)$ (see Exercise 6.4).

We can decompose again if any transitive dependencies remain in R_1 or R_2. We do not go on decomposing relation schemes forever, though. Every time we decompose a relation scheme, the two resultant relation schemes are smaller, and there can be no transitive dependencies in a relation scheme with only two attributes (see Exercise 6.5).

The decomposition process can be speeded up by checking if there are any other nonprime attributes in $R - (K\,Y)$ that are dependent upon Y. If such attributes exist, they are also transitively dependent upon K and can be removed at the same time. Suppose $A_1, A_2, \ldots, A_m$ are in $R - (K\ Y)$ and are dependent on Y. We then let $R_1 = R - (A_1 A_2 \cdots A_m)$ and $R_2 = Y A_1 A_2 \cdots A_m$. Again, any relation $r(R)$ that satisfies F decomposes losslessly onto R_1 and R_2.

Example 6.12 Let $R =$

FLIGHT FROM TO DEPARTS ARRIVES DURATION PLANE-TYPE
FIRST-CLASS COACH TOTAL-SEATS #MEALS,

where FIRST-CLASS and COACH are the number of seats available in each section. Let the set of designated keys be

K = {FLIGHT, FROM TO DEPARTS, FROM TO ARRIVES}.

We are assuming there cannot be two flights from the same source to the same destination arriving or departing at the same time. Let all the designated keys be actual keys, and suppose we also have the following FDs in our set F.

PLANE-TYPE → FIRST-CLASS COACH TOTAL-SEATS
DEPARTS DURATION → #MEALS
ARRIVES DURATION → #MEALS
FIRST-CLASS COACH → TOTAL-SEATS
FIRST-CLASS TOTAL-SEATS → COACH
COACH TOTAL-SEATS → FIRST-CLASS

It might seem that ARRIVES DEPARTS → DURATION should also be an FD, but the times for ARRIVES and DEPARTS are assumed to be local times, so DURATION depends also on the time zones for the source and destination cities.

We first remove the transitive dependency of #MEALS upon FLIGHT via DEPARTS DURATION. We get the relation scheme

R_1 = FLIGHT FROM TO DEPARTS ARRIVES DURATION
PLANE-TYPE FIRST-CLASS COACH TOTAL-SEATS

with designated keys

$\mathbf{K}_1$ = {FLIGHT, FROM TO DEPARTS, FROM TO ARRIVES} and the relation scheme

$$R_2 = \text{DEPARTS DURATION \#MEALS}$$

with designated key

$$\mathbf{K}_2 = \{\text{DEPARTS DURATION}\}.$$

Scheme R_2 is in 3NF; scheme R_1 is not, since FIRST-CLASS, COACH, and TOTAL-SEATS are all transitively dependent upon FLIGHT via PLANE-TYPE. We decompose R_1 into schemes

$$R_{11} = \text{FLIGHT FROM TO DEPARTS ARRIVES DURATION PLANE-TYPE}$$

with designated keys

$$\mathbf{K}_{11} = \{\text{FLIGHT, FROM TO DEPARTS, FROM TO ARRIVES}\}$$

and

$$R_{12} = \text{PLANE-TYPE FIRST-CLASS COACH TOTAL-SEATS}$$

with designated key

$$\mathbf{K}_{12} = \{\text{PLANE-TYPE}\}.$$

Relation scheme R_{11} is now in 3NF with respect to F, but R_{12} is not, since TOTAL-SEATS is transitively dependent upon PLANE-TYPE via FIRST-CLASS COACH. We decompose R_{12} into

$$R_{121} = \text{PLANE-TYPE FIRST-CLASS COACH}$$

with designated key

$$\mathbf{K}_{121} = \{\text{PLANE-TYPE}\}$$

and relation scheme

$$R_{122} = \text{FIRST-CLASS COACH TOTAL-SEATS}$$

with designated key

$$\mathbf{K}_{122} = \{\text{FIRST-CLASS COACH}\}.$$

We have now decomposed R to the point where every relation scheme is in 3NF with respect to F. Hence, the database scheme

$$\mathbf{R} = \{R_{11}, R_{121}, R_{122}, R_2\}$$

is in 3NF. Furthermore, we can faithfully represent any relation $r(R)$ that satisfies the FDs in F by its projections onto the relation schemes in $\mathbf{R}$, since

$$r = (\pi_{R_{11}}(r) \bowtie (\pi_{R_{121}}(r) \bowtie \pi_{R_{122}}(r))) \bowtie \pi_{R_2}(r).$$

The parentheses are not necessary, since join is commutative and associative. They only serve to point out the stages by which r was losslessly decomposed. If the order of taking the joins is changed, however, the intermediate results may not be meaningful. For example,

$$\pi_{R_{122}}(r) \bowtie \pi_{R_2}(r) \neq \pi_{R_{122}R_2}(r).$$

If the evaluation is done according to the parentheses, every intermediate result is a projection of r.

Database scheme $\mathbf{R}$ is not unique. There are points at which we had a choice of ways to decompose a relation to remove a transitively dependent attribute. At the first step, we could have chosen

$$R_2 = \text{ARRIVES DURATION \#MEALS},$$

since #MEALS is also transitively dependent upon FLIGHT via ARRIVES DURATION. There are three choices for decomposing R_{12} at the third step. (What are they?) Some keys for relation schemes are not picked up as designated keys, such as FIRST-CLASS TOTAL-SEATS and COACH TOTAL-SEATS for R_{122}.

6.4 SHORTCOMINGS OF NORMALIZATION THROUGH DECOMPOSITION

Several problems can arise when normalizing a relation scheme by decomposition. First, the time complexity of the process is probably not polynomially bounded. There can be an exponential number of keys for a relation scheme in terms of the size of the relation scheme and the governing set of FDs (see Exercise 6.8). Also, deciding if an attribute is nonprime in a scheme is an NP-complete problem.

A second problem is that we may end up producing more relation schemes than we really need for 3NF.

Example 6.13 Let relation scheme $R = A\ B\ C\ D\ E$ and let $F = \{A\ B \rightarrow C\ D\ E, A\ C \rightarrow B\ D\ E, B \rightarrow C, C \rightarrow B, C \rightarrow D, B \rightarrow E\}$. The keys for R under F are $A\ B$ and $A\ C$. Using the transitive dependency of D on $A\ B$ via C, we decompose to

$$\begin{array}{ll} R_1 = A\ B\ C\ E & \mathbf{K}_1 = \{A\ B,\ A\ C\} \\ R_2 = C\ D & \mathbf{K}_2 = \{C\}. \end{array}$$

We then use the transitive dependency of E on $A\ B$ via B in R_1 to get

$$\begin{array}{ll} R_{11} = A\ B\ C & \mathbf{K}_{11} = \{A\ B,\ A\ C\} \\ R_{12} = B\ E & \mathbf{K}_{12} = \{B\}. \end{array}$$

The final 3NF database scheme is

$$\mathbf{R} = \{R_{11}, R_{12}, R_2\}.$$

There is a 3NF decomposition of R into only two relation schemes, namely

$$\begin{array}{ll} R_1 = A\ B\ C & \mathbf{K}_1 = \{A\ B,\ A\ C\} \\ R_2 = B\ D\ E & \mathbf{K}_2 = \{B\}. \end{array}$$

A third problem is that we can introduce partial dependencies into a relation scheme through decomposition. The partial dependencies can cause more relation schemes to appear in the final database scheme than are actually needed.

Example 6.14 Let relation scheme $R = A\ B\ C\ D$ and let $F = \{A \rightarrow B\ C\ D, C \rightarrow D\}$. A is the only key for R under F. D is transitively dependent upon A via $B\ C$. Decomposing, we get

$$\begin{array}{ll} R_1 = A\ B\ C & \mathbf{K}_1 = \{A\} \\ R_2 = B\ C\ D & \mathbf{K}_2 = \{B\ C\}. \end{array}$$

$B\ C$ is an actual key of R_2, but D is partially dependent upon it. Hence D is transitively dependent upon $B\ C$. Scheme R_2 must be decomposed into

$R_{21} = B\ C$	$\mathbf{K}_{21} = \{B\ C\}$
$R_{22} = C\ D$	$\mathbf{K}_{22} = \{C\}$.

R_1, R_{21} and R_{22} form a 3NF database scheme for R. However, the two relation schemes R_1 and R_{22} also form a 3NF database scheme for R.

This problem can be avoided if we are careful that the intermediary set of attributes in the transitive dependency we decompose with is minimal. In Example 6.14 above, we had D transitively dependent on A via $B\ C$, but $B\ C$ is not minimal. D is transitively dependent upon A via C only.

A fourth problem is that we may create a database scheme on which the set of FDs involved is not enforceable.

Example 6.15 Let relation scheme $R = A\ B\ C\ D\ E$ and let $F = \{A \rightarrow B\ C\ D\ E,\ C\ D \rightarrow E,\ E\ C \rightarrow B\}$. If we eliminate the transitive dependency of E upon A via $C\ D$, we get

$R_1 = A\ B\ C\ D$	$\mathbf{K}_1 = \{A\}$
$R_2 = C\ D\ E$	$\mathbf{K}_2 = \{C\ D\}$.

F is not enforceable on the database scheme $\mathbf{R} = \{R_1, R_2\}$, since $E\ C \rightarrow B$ is not implied by the FDs in F^+ that apply to R_1 or R_2. (This statement must be checked by generating F^+.)

Finally, we can produce relation schemes with "hidden" transitive dependencies through decomposition.

Example 6.16 Let relation scheme $R = A\ B\ C\ D$ and let $F = \{\mathrm{A} \rightarrow B,\ B \rightarrow C\}$. $A\ D$ is a key for F and B is partially dependent on $A\ D$. Decomposing, we get

$R_1 = A\ C\ D$	$\mathbf{K}_1 = \{A\ D\}$
$R_2 = A\ B$	$\mathbf{K}_2 = \{A\}$.

Although R_1 and R_2 are both technically in 3NF, there is still a "hidden" transitive dependency of C on $A\ D$ in R_1.

6.5 NORMALIZATION THROUGH SYNTHESIS

In this section we present another means for achieving third normal form, which avoids the problems associated with decomposition cited in the previous section.

The basic problem we address is finding a 3NF database scheme to represent a relation scheme that is not in 3NF. We assume our input to be a relation scheme R and a set F of FDs over R. With this input, we wish to create a database scheme $\mathbf{R} = \{R_1, R_2, \ldots, R_p\}$ over R with the following four properties:

1. F is completely characterized by $\mathbf{R}$. That is,

$$F \equiv \{K \rightarrow R_i \mid R_i \text{ is in } \mathbf{R} \text{ and } K \text{ is a designated key of } R_i\}.$$

2. Every relation scheme R_i in $\mathbf{R}$ is in 3NF with respect to F.
3. There is no database scheme with fewer relation schemes than $\mathbf{R}$ satisfying properties 1 and 2.
4. For any relation $r(R)$ that satisfies F,

$$r = \pi_{R_1}(r) \bowtie \pi_{R_2}(r) \bowtie \cdots \bowtie \pi_{R_p}(r).$$

We call any database scheme R satisfying properties 1 to 3 above a *complete* database scheme for F.

Let us discuss the reasons for these requirements. Property 1 ensures that F is enforceable on $\mathbf{R}$ and that we know how to enforce it without computing F^+. Property 1 also guarantees that the only FDs we must enforce to make $\mathbf{R}$ obey F are the ones that derive from designated keys. The reasons behind property 2 have already been discussed. Property 3 prevents data redundancy. Property 4 allows us to represent a relation on scheme R faithfully by its projections onto the schemes in $\mathbf{R}$.

We first develop an algorithm using annular covers that produces a complete database scheme for F. We call this algorithm a *synthesis* algorithm, since it constructs the database scheme directly from the FDs in F. We then point out some other useful properties the database schemes synthesized by our algorithm possess. We shall then modify the algorithm so that its output satisfies property 4 and also show how to make some further refinements to the algorithm.

The synthesis algorithm we develop will have polynomial time complexity, since the hardest part of the algorithm is computing a reduced, minimum annular cover for the input set of FDs. Therefore, we avoid the first problem mentioned in the previous section. The second and third problems are avoided by property 3. The fourth problem mentioned is prevented by property 1 and the fifth problem is avoided by properties 1 and 3 together.

6.5.1 Preliminary Results for the Synthesis Algorithm

Lemma 6.2 If **R** is a database scheme representing the set of FDs G, then there are at least $|\bar{E}_G|$ relation schemes in **R**. That is, there are at least as many relation schemes in **R** as there are equivalence classes in $\bar{E}_G$.

Proof All the FDs embodied by a single relation scheme R in **R** must have equivalent left sides. If K_1 and K_2 are designated keys for R, then $K_1 \rightarrow R$ and $K_2 \rightarrow R$, hence $K_1 \rightarrow K_2$ and $K_2 \rightarrow K_1$. Therefore, each relation scheme in R can embody FDs from at most one equivalence class in $\bar{E}_G$. To represent all the FDs in G, we need at least $|\bar{E}_G|$ relation schemes in **R**.

Corollary Let F be a set of FDs. Any database scheme **R** that completely characterizes F must have at least $|\bar{E}_{F'}|$ relation schemes, where F' is a nonredundant cover for F.

Proof By Lemma 6.2, we know if G is the set of FDs represented by **R**, then $|\bar{E}_G| \geq |\bar{E}_{F'}|$, since $G \equiv F \equiv F'$ and F' is nonredundant.

6.5.2 Developing the Synthesis Algorithm

The corollary to Lemma 6.2 suggests a way to synthesize a complete database scheme for a set F of FDs. We find a nonredundant cover F' for F and compute the equivalence classes in $\bar{E}_{F'}$. For each $E_{F'}(X)$ in $\bar{E}_{F'}$, we construct a relation scheme R consisting of all attributes appearing in any FDs of $E_{F'}(X)$. We let $e_{F'}(X)$ be the set of designated keys for R. The database scheme **R** consists of all relation schemes so synthesized. **R** certainly has the minimum possible number of relation schemes, by the corollary to Lemma 6.2. It is also possible to show that **R** completely characterizes F (see Exercise 6.11). However, the relation schemes may not be in 3NF with respect to F, as the next example shows.

Example 6.17 Let $F = F' = \{A \rightarrow B\,C,\ B \rightarrow C\}$ and let $R = A\,B\,C$. The procedure outlined above gives relation scheme

$$R_1 = A\ B\ C \quad \text{with designated key } \mathbf{K}_1 = \{A\}$$

and relation scheme

$$R_2 = B\ C \quad \text{with designated key } \mathbf{K}_2 = \{B\}.$$

Relation R_1 is not in 3NF with respect to F. (Why?)

The problem in Example 6.17 arises from F not being reduced. The next example shows that even with a reduced set of FDs, we are not guaranteed a 3NF database scheme.

Example 6.18 Let F be the FDs

$B_1 B_2 \to A$	$D_1 D_2 \to B_1 B_2$
$B_1 \to C_1$	$B_2 \to C_2$
$D_1 \to A$	$D_2 \to A$
$A B_1 C_2 \to D_2$	$A B_2 C_1 \to D_1$

and let $R = A B_1 B_2 C_1 C_2 D_1 D_2$. This is the same set of FDs as in Example 5.21. The only two FDs that are in an equivalence class together are $B_1 B_2 \to A$ and $D_1 D_2 \to B_1 B_2$. The relation scheme produced from this equivalence class is

$$R = A B_1 B_2 D_1 D_2, \quad \text{with designated keys } \mathbf{K} = \{B_1 B_2, D_1 D_2\}.$$

R is not in 3NF, since A is transitively dependent upon $B_1 B_2$ via D_1.

Finding a minimum cover for F does not help the situation, as the set F in Example 6.18 is minimum. Attribute A is the problem here just as it was in Example 5.21. If we use annular covers, as before, our problems are solved. A synthesis algorithm using annular covers is given as Algorithm 6.1.

Algorithm 6.1 SYNTHESIZE
Input: A set of FDs F over $\mathbf{U}$.
Output: A complete database scheme for F.

```
SYNTHESIZE($F$)
  1. Find a reduced, minimum annular cover $G$ for $F$.
  2. For each CFD $(X_1, X_2, \ldots, X_k) \to Y$ in $G$, construct a relation scheme
     $R = X_1 X_2 \cdots X_k Y$ with designated keys $\mathbf{K} = \{X_1, X_2, \ldots, X_k\}$.
  3. Return the set of relation schemes constructed in step 2.
```

Example 6.19 Let F be the set of FDs

$$\begin{array}{lll}
A \rightarrow B_1\, B_2\, C_1\, C_2\, D\, E\, I_1\, I_2\, I_3\, J & & \\
B_1\, B_2\, C_1 \rightarrow A\, C_2\, D\, E\, I_1\, I_2\, I_3\, J & & \\
B_1\, B_2\, C_2 \rightarrow A\, C_1\, D\, E\, I_1\, I_2\, I_3\, J & & \\
E \rightarrow I_1\, I_2\, I_3 & & \\
C_1\, D \rightarrow J & C_2\, D \rightarrow J & \\
I_1\, I_2 \rightarrow I_3 & I_2\, I_3 \rightarrow I_1 & I_1\, I_3 \rightarrow I_2
\end{array}$$

and let $R = A\, B_1\, B_2\, C_1\, C_2\, D\, E\, I_1\, I_2\, I_3\, J$.

We are using the same FDs as in Example 6.12; we have only renamed the attributes. Set F is minimum, but not reduced. Reducing F we get $F' =$

$$\begin{array}{lll}
\{A \rightarrow B_1\, B_2\, C_1\, C_2\, D\, E & & E \rightarrow I_1\, I_2 \\
\;B_1\, B_2\, C_1 \rightarrow A & & B_1\, B_2\, C_2 \rightarrow A \\
\;C_1\, D \rightarrow J & & C_2\, D \rightarrow J \\
\;I_1\, I_2 \rightarrow I_3 & I_2\, I_3 \rightarrow I_1 & I_1\, I_3 \rightarrow I_2\}.
\end{array}$$

Converting to an annular cover and making the obvious reductions we get $G =$

$$\begin{array}{ll}
\{(A,\, B_1\, B_2\, C_1,\, B_1\, B_2\, C_2) \rightarrow D\, E & \\
\;(E) \rightarrow I_1\, I_2 & \\
\;(C_1\, D) \rightarrow J & (C_2\, D) \rightarrow J \\
\;(I_1\, I_2,\, I_2\, I_3,\, I_1\, I_3)\}. &
\end{array}$$

Converting each CFD to a relation scheme with designated keys we get

$$\begin{array}{ll}
R_1 = A\, B_1\, B_2\, C_1\, C_2\, D\, E & \mathbf{K}_1 = \{A,\, B_1\, B_2\, C_1,\, B_1\, B_2\, C_2\} \\
R_2 = E\, I_1\, I_2 & \mathbf{K}_2 = \{E\} \\
R_3 = C_1\, D\, J & \mathbf{K}_3 = \{C_1\, D\} \\
R_4 = C_2\, D\, J & \mathbf{K}_4 = \{C_2\, D\} \\
R_5 = I_1\, I_2\, I_3 & \mathbf{K}_5 = \{I_1\, I_2,\, I_2\, I_3,\, I_1\, I_3\}.
\end{array}$$

The final database scheme is $\mathbf{R} = \{R_1,\, R_2,\, R_3,\, R_4,\, R_5\}$.

6.5.3 Correctness and Other Properties of the Synthesis Algorithm

Lemma 6.3 Let $\mathbf{R}$ be a database scheme produced by SYNTHESIZE from a set of FDs F. For any relation scheme R_i in $\mathbf{R}$, every designated key of R_i is a key.

Proof Let $(X_1, X_2, \ldots, X_k) \rightarrow Y$ be the CFD from which R_i was synthesized. Let K be a designated key for R_i that is not a key. $K = X_j$ for some X_j in the left side of the CFD. Since K is not a key, X_j contains a shiftable attribute. Hence, $(X_1, X_2, \ldots, X_k) \rightarrow Y$ is not reduced, a contradiction.

Theorem 6.1 SYNTHESIZE constructs a complete database scheme for a set of FDs F in $O(n^2)$ time on inputs of length n.

Proof First, we show the time complexity of SYNTHESIZE is $O(n^2)$. The time spent in steps 2 and 3 of the algorithm is dominated by the time spent in step 1. From the observations in Section 5.8, we know that step 1 can be implemented to run in $O(n^2)$ time.

Let **R** be the result of SYNTHESIZE(F). **R** completely characterizes F, since the set of embodied FDs for any relation scheme R in **R** is a characteristic set for the CFD from which R was synthesized (see Exercise 5.24). From Lemma 5.10 and the corollary to Lemma 6.2, we see that **R** has a minimum number of relation schemes among all database schemes that completely characterize F.

All that remains to be proved is that all the relation schemes in **R** are in 3NF with respect to F. Consider a relation R_i in **R**, with designated keys $K_i = \{X_1, X_2, \ldots, X_i\}$, that was synthesized from the CFD $(X_1, X_2, \ldots, X_k) \rightarrow Y$. If attribute A is nonprime in R_i, then A is in Y, since every X_j in K_i is a key for R_i, by Lemma 6.3. Let X be a key for R_i (not necessarily a designated key). Suppose there is a subset Z of R_i such that F implies $X \rightarrow Z$ and $Z \rightarrow A$, F does not imply $Z \rightarrow X$, and $A \notin X\,Z$. Form F' from the set G of CFDs in SYNTHESIZE by taking natural characteristic sets for every CFD in G. Consider an F'-based DDAG H for $Z \rightarrow A$. $U(H)$ can contain no FDs from $E_{F'}(X)$, which is the natural characteristic set for $(X_1, X_2, \ldots, X_k) \rightarrow Y$. If it did, we would have $Z \rightarrow X$ under F. Thus, if we remove A from Y, we still can prove $Z \rightarrow A$ from F', hence $(X_1, X_2, \ldots, X_k) \rightarrow Y$ is not reduced.

The proof of Theorem 6.1 uses both conditions for a reduced CFD: no shiftable attributes on the left side and no extraneous attributes on the right side. We cannot simplify SYNTHESIZE by leaving out either part of the reduction step. The proof of Theorem 6.1 actually demonstrates a slightly stronger result than the statement of the theorem, namely, that no attribute in a relation scheme produced by the synthesis algorithm is transitively dependent upon a key unless it is contained in a designated key. The next example shows it is possible to have a 3NF database scheme where this stronger condition does not hold.

Example 6.20 Let $R = A\,B\,C\,D\,E$ and let $F = \{A \to B,\ B \to A\,E,\ A\,C \to D\}$. Consider the database scheme **R** for F consisting of the relation schemes

$$\begin{array}{ll} R_1 = A\,B\,C\,D & \text{with designated key } \mathbf{K}_1 = \{A\,C\}, \text{ and} \\ R_2 = A\,B\,E & \text{with designated keys } \mathbf{K}_2 = \{A,\ B\}. \end{array}$$

R_1 is in 3NF since B is prime ($B\ C$ is a key), although B is partially dependent upon $A\ C$.

There are two other properties we can prove about database schemes produced by SYNTHESIZE.

Lemma 6.4 Let **R** be the database scheme constructed by SYNTHESIZE from a set of FDs F. There is no complete database scheme for F with fewer designated keys.

Proof Left to the reader (see Exercise 6.15).

Consider the 3NF database scheme **R** consisting of relation schemes

$$\begin{array}{l} R_1 = \underline{A\,B}\,D \\ R_2 = \underline{A}\,C \\ R_3 = \underline{B\,C}\,D \end{array}$$

where the underlined attributes are designated and actual keys and there are no other FDs. Notice that $A\ B \to B\ C$, $B\ C \not\to A\ B$, and $B\ C \to D$. R_1 is in 3NF, since $B\ C \not\subseteq R_1$. However, D can be removed from R_1 without changing the closure of the set of FDs represented by **R**.

Definition 6.12 Let R be a relation scheme in database scheme **R** over **U** and let F be a set of FDs. Let $X \subseteq R$ and $A \in R$. A is *externally dependent* upon X under F if there is a subset Y of **U** such that Y is not a subset of R and $X \to Y$, $Y \not\to X$, and $Y \to A$ under F and $A \notin X\ Y$.

An external dependency is what we were calling a hidden transitive dependency in Example 6.16. External dependencies cannot always be avoided in 3NF database schemes.

Example 6.21 Let **R** be the database scheme composed of the relation schemes

$$\begin{array}{ll} R_1 = A\,B & \text{with designated key } \mathbf{K}_1 = \{A\}, \text{ and} \\ R_2 = B\,C & \text{with designated keys } \mathbf{K}_2 = \{B,\ C\}. \end{array}$$

Let F be the set of FDs represented by **R**. B is externally dependent upon A under F, since $A \rightarrow C$, $C \not\rightarrow A$, and $C \rightarrow B$, but B cannot be removed from R_1 without changing the closure of the set of FDs **R** represents.

Let **R** be a 3NF database scheme over **U**. Let G be the set of FDs that **R** represents and let R be a relation scheme in **R**. Suppose attribute A in R is externally dependent upon key K of R via a set $Y \subseteq \mathbf{U}$, where $Y \not\subseteq R$. Unless $K' \rightarrow A$ is necessary to derive $K \rightarrow Y$, for some designated key K' of R, A can be removed from R without changing the closure of G, since G still would imply $K \rightarrow Y$ and $Y \rightarrow A$. $Y \rightarrow A$ still holds because any G-based DDAG H for $Y \rightarrow A$ cannot use any FD in $E_G(K)$, or else $Y \rightarrow K$.

Definition 6.13 Let **R** be a 3NF database scheme over **U** and let G be the set of FDs that **R** represents. Let R be a relation scheme in **R**. Attribute A in R is *removable* in R if removing A from R does not change the closure of G. (Removing A implies removing any designated key of R containing A.)

Complete 3NF database schemes derived by decomposition do not contain removable attributes (see Exercise 6.17). The same holds for synthesized database schemes.

Lemma 6.5 If **R** is the database scheme produced by SYNTHESIZE from a set of FDs F, then no relation scheme R in **R** contains a removable attribute.

Proof Suppose R contains a removable attribute A. Let $(X_1, X_2, \ldots, X_k) \rightarrow Y$ be the CFD from which R was synthesized. A cannot be in any X_i in the left side of the CFD. If A were in some X_i, since A is removable, either X_i is not a key, contradicting Lemma 6.3, or X_i can be removed completely from $(X_1, X_2, \ldots, X_k) \rightarrow Y$, contradicting Lemma 6.4. We conclude A is in Y. Therefore, $(X_1, X_2, \ldots, X_k) \rightarrow Y$ is not reduced, since A is extraneous in Y. Hence, A does not appear in $(X_1, X_2, \ldots, X_k) \rightarrow Y$, contradicting the construction of R.

6.5.4 Refinements of the Synthesis Algorithm

Although synthesis solves the problems associated with decomposition listed in Section 6.4, there is one shortcoming of synthesis that is not shared by decomposition. In a 3NF database scheme **R**, obtained from a single relation scheme R and a set of FDs F by decomposition, we know that any relation $r(R)$ satisfying F decomposes losslessly onto the relation schemes in **R**. The same is not true if **R** is obtained by synthesis.

Example 6.22 Let $F = \{A \rightarrow C, B \rightarrow C\}$. SYNTHESIZE($F$) produces the relation schemes

$$R_1 = \underline{A}\, C \quad \text{and} \quad R_2 = \underline{B}\, C,$$

where the underlined attributes are the designated keys. However, the relation r shown below does not decompose losslessly onto R_1 and R_2.

$r(A$	B	$C)$
a	b	c
a'	b'	c

Definition 6.14 A database scheme **R** over **U** has the *lossless join property* with respect to a set of FDs F if any relation $r(\mathbf{U})$ that satisfies F decomposes losslessly onto the relation schemes of **R**.

Property 4 at the beginning of Section 6.5 says that the database scheme **R** we synthesize must have the lossless join property with respect to the set of FDs **R** represents. Example 6.22 shows that a database scheme produced by synthesis does not necessarily have the lossless join property. There is a related problem involving attributes in **R** that are not mentioned in the dependencies of F. They do not show up in the database scheme synthesized from F. However, a minor modification of SYNTHESIZE will solve both problems.

Definition 6.15 Let **R** be a database scheme over **U** and let G be the set of FDs that **R** represents. A subset X of **U** is a *universal key* for R if $G \models X \rightarrow \mathbf{U}$. We make no requirement for the minimality of X.

We shall see in Chapter 8 that if some relation scheme in a database scheme R contains a universal key, then **R** has the lossless join property with respect to the set of represented FDs, and conversely. The problem with the synthesis algorithm is that there may be no relation scheme in **R** containing a universal key. Such is the case in Example 6.22. The problem can be remedied by adding a relation scheme consisting solely of the attributes in some universal key. This addition technically violates the minimality constraint on R, but we shall look the other way during such transgressions.

The modification to SYNTHESIZE is to add the FD $\mathbf{U} \rightarrow C$ to F as the first step, where C is an attribute not contained in **U**. In finding the annular cover G for F, $\mathbf{U} \rightarrow C$ will not be eliminated as redundant, since no other FD in F has C on the right side. $\mathbf{U} \rightarrow C$ may be combined with the FD $X \rightarrow Y$

when finding a minimum cover for F, but then $X \leftrightarrow \mathbf{U}$ and X must be a universal key. During the reduction stage of finding G, $\mathbf{U}$ may be reduced to $\mathbf{U}'$ by removing shiftable attributes (or X reduced to X' in the same way), but $\mathbf{U}' \rightarrow \mathbf{U}$ holds, so $\mathbf{U}'$ is a universal key.

Some relation scheme R_i is synthesized from the CFD containing $\mathbf{U}'$ as a left set, and hence $\mathbf{U}$ will have a universal key in one of its relation schemes and the lossless join property with respect to F. At the end of the algorithm, attribute C can be removed from the relation scheme in which it appears. (It will only appear in one.)

Example 6.23 Let $F = \{A \rightarrow C, B \rightarrow C\}$. We add $A\ B\ C \rightarrow D$ to F, convert to an annular cover, and reduce, to get $G = \{(A) \rightarrow C, (B) \rightarrow C, (A\ B) \rightarrow D\}$. The CFD $(A\ B) \rightarrow D$ is used to synthesize the relation scheme $R_1 = A\ B\ D$ with designated key $\mathbf{K}_1 = \{A\ B\}$. R_1 contains the universal key $A\ B$. D can be removed from R_1.

6.6 AVOIDABLE ATTRIBUTES

We have seen that SYNTHESIZE(F) produces a complete database scheme $\mathbf{R}$ for F with no removable attributes. It may still be possible to remove an attribute from a relation scheme R in $\mathbf{R}$ by changing the set of designated keys.

Example 6.24 Let $F = \{A \rightarrow B, B \rightarrow A, A\ C \rightarrow D\ E, B\ D \rightarrow C\}$. SYNTHESIZE($F$) produces a database scheme $\mathbf{R}$ containing the relation schemes $R_1 = A\ B$ with designated keys $\mathbf{K}_1 = \{A, B\}$ and $R_2 = A\ B\ C\ D\ E$ with designated keys $\mathbf{K}_2 = \{A\ C, B\ D\}$. B is not removable from R_2, since it belongs to a designated key. If K_2 is changed to $\{A\ C, A\ D\}$, $\mathbf{R}$ still completely characterizes F and B becomes removable.

Definition 6.16 Let $\mathbf{R}$ be a complete database scheme for a set of FDs F. Let R_i be a relation scheme in $\mathbf{R}$, and let B be an attribute in R_i. B is *avoidable* in R_i if changing the designated keys of R_i makes B removable in R_i.

An avoidable attribute in a database scheme produced by SYNTHESIZE must belong to a designated key of some relation scheme, otherwise it is removable. It might seem as if finding an alternative set of designated keys for a relation scheme R that preserves the enforceability of F is as hard as finding all the keys of R. If R was generated by SYNTHESIZE, it is possible to find a set of alternative keys, if one exists, without generating every key or set of keys for R. The designated keys in R correspond to the left sides of FDs in a single equivalence class in a minimum cover of F, and we know a great deal about the structure of minimum covers.

Suppose relation scheme R in database scheme $\mathbf{R}$ was synthesized from the CFD $(X_1, X_2, \ldots, X_k) \rightarrow Y$ in a minimum, reduced annular cover G for F. Thus, the set of designated keys for R is $\mathbf{K} = \{X_1, X_2, \ldots, X_k\}$. $X_1, X_2, \ldots, X_k$ are left sides of FDs in a single equivalence class $E_{F'}(X)$ for some minimum cover F' for F. Therefore, we know that any alternative set of keys $\mathbf{K}' = \{Z_1, Z_2, \ldots, Z_m\}$ must have $m \geq k$, if the enforceability of F is to be preserved. If $m > k$, then $Z_i \overset{\cdot}{\rightarrow} Z_j$ for some i and j. Z_i can be removed from $\mathbf{K}'$ without changing the closure of the set of FDs represented by $\mathbf{R}$. Therefore we shall assume that any alternative set of keys we seek will have exactly k members.

Let $\mathbf{K}' = \{Z_1, Z_2, \ldots, Z_k\}$ be an alternative set of designated keys for R. We know direct determination induces a one-to-one correspondence between the elements of $\mathbf{K}$ and $\mathbf{K}'$, since they both can be used as left sides for an equivalence class in some minimum cover for F. Assume $Z_1, Z_2, \ldots, Z_k$ are numbered so $X_i \overset{\cdot}{\rightarrow} Z_i$ and $Z_i \overset{\cdot}{\rightarrow} X_i$, $1 \leq i \leq k$, under F.

Let us try to remember where we are headed. We are trying to detect if some attribute B in R is avoidable by finding a replacement set of designated keys for R that does not use B. We assume that $\mathbf{K}'$ above is such a set. We may further assume that $X_i = Z_i$ if $B \notin X_i$. If not, since $Z_i \overset{\cdot}{\rightarrow} X_i$, we can replace Z_i by X_i in $\mathbf{K}'$ without reintroducing B and still keep $\mathbf{R}$ a complete database scheme for F.

We have narrowed the problem down considerably. Starting with the set $\mathbf{K} = \{X_1, X_2, \ldots, X_k\}$ of designated keys for R, for each X_i in $\mathbf{K}$ containing B, we are looking for a replacement key Z_i not containing B with $X_i \overset{\cdot}{\rightarrow} Z_i$ and $Z_i \overset{\cdot}{\rightarrow} X_i$ under F. Note that Z_i must be contained in R, or we cannot use it as a replacement key for X_i. AVOID in Algorithm 6.2 finds an alternative set of designated keys for R not containing B, if such exists. AVOID assumes a procedure DCLOSURE(X, F) that returns the maximum set X' such that $X \overset{\cdot}{\rightarrow} X'$ under F.

Algorithm 6.2 AVOID

Input: A relation scheme R produced by SYNTHESIZE(F), with designated keys $\mathbf{K}$, and an attribute B in R, and the set F of FDs.

Output: An alternative set of designated keys for R not containing B, if such a set exists; $\emptyset$ otherwise.

```
AVOID($R$, $B$, $F$)
  **begin**
  $\mathbf{K}' := \mathbf{K}$; fail := *false*;
  **for** each $X_i$ in $\mathbf{K}$ **do**
    **if** $B \in X_i$ **then begin**
      $M :=$ DCLOSURE($X_i$, $F$);
      $M' := (M \cap R) - B$;
      **if** DCLOSURE($M'$, $F$) $\supseteq X_i$ **then**
        replace $X_i$ in $\mathbf{K}'$ by a minimal subset $Z$ of $M'$ such that $Z \dot{\rightarrow} X_i$
        **else** fail := *true*
      **end**;
  **if** fail = *false* **then**
    **return**($\mathbf{K}'$)
    **else return**($\emptyset$)
  **end**.
```

Example 6.25 Let F, R_1, and R_2 be as in Example 6.24. AVOID(R_2, B, F) will only consider BD in $\mathbf{K}_2$ for replacement. M will be ABD and M' will be $A\ D$. $A\ D \dot{\rightarrow} B\ D$ under F, so $A\ D$ can replace $B\ D$ in $\mathbf{K}_2$.

We shall not spend more time with AVOID to derive its time complexity, which is polynomial, or prove its correctness (see Exercise 6.20). We can use AVOID to eliminate avoidable attributes from a relation scheme R in a database scheme $\mathbf{R}$ produced by SYNTHESIZE(F). For each attribute B in R, we run AVOID(R, B, F). If AVOID returns something other than $\emptyset$, we replace the set of designated keys for R by the alternative set provided by AVOID. We know that if such a set of keys exists, B must be removable when the set of keys is used. $R - B \rightarrow B$ can be derived from the new set of FDs represented by $\mathbf{R}$, since there must be a new designated key K for R with $K \dot{\rightarrow} B$. (K is one of the replacement keys found by AVOID. See Exercises 6.21 and 6.22.)

A complete database scheme with no avoidable attributes is in *LTK normal form* (for Ling, Tompa, and Kameda). Exercise 6.23 gives another characterization of LTK normal form.

6.7 BOYCE-CODD NORMAL FORM

We saw that our synthesis algorithm yielded relation schemes that were in a form slightly stronger than 3NF in that any attribute in a designated key was not transitively dependent upon any key. We now ask the question, is it possible to remove all transitive dependencies? The answer is a qualified yes. Starting with a relation scheme R with FDs F, we can find a database scheme $\mathbf{R}$ for R with no transitive dependencies, but F may be unenforceable on $\mathbf{R}$.

First, let us see why we might want to remove a prime attribute that is transitively dependent upon a key.

Example 6.26 Consider a relation *billto* on the relation scheme AIRPORT COMPANY OFFICE. The meaning of a tuple $\langle a\ c\ f\rangle$ in *billto* is that if someone from company c charges a ticket at airport a, the bill should be sent to office f of the company. We thus have the two FDs AIRPORT COMPANY $\rightarrow$ OFFICE and OFFICE $\rightarrow$ COMPANY. AIRPORT OFFICE is a key for the relation scheme, and COMPANY is partially, hence transitively, dependent upon AIRPORT OFFICE. Although COMPANY is a prime attribute, it is still desirable to remove it from the relation scheme, since there is a duplication of COMPANY-OFFICE pairs.

Definition 6.17 A relation scheme R is in *Boyce-Codd normal form* (BCNF) with respect to a set of FDs F if it is in 1NF and no attribute in R is transitively dependent upon any key of R. A database scheme $\mathbf{R}$ is in *Boyce-Codd normal form* with respect to a set of FDs F if every relation scheme R in $\mathbf{R}$ is in Boyce-Codd normal form with respect to F.

BCNF implies 3NF (see Exercise 6.24). The relation scheme AIRPORT COMPANY OFFICE in Example 6.26 is not in BCNF. The following is an alternative definition of BCNF (see Exercise 6.25).

Definition 6.18 A relation scheme R is in *Boyce-Codd normal form* with respect to a set of FDs F if for every subset Y of R and for every attribute $A \in R - Y$, if $Y \rightarrow A$, then $Y \rightarrow R$ under F. That is, if Y non-trivially determines any attribute of R, then Y is a superkey for R.

We can always use decomposition to find a BCNF database scheme for a relation scheme that is not in BCNF. If $Y \rightarrow A$ under F for relation scheme R with $Y \subseteq R$ and $A \in R - Y$, and if Y is not a key for R, then decompose R into $R_1 = R - A$ and $R_2 = YA$. The designated keys for R_1 and R_2 are produced in the same manner as for 3NF database schemes.

Example 6.27 Let relation scheme R = AIRPORT COMPANY OFFICE with the FDs given in Example 6.26. Using the FD OFFICE $\rightarrow$ COMPANY, we decompose R into

R_1 = AIRPORT OFFICE with designated key $\mathbf{K}_1$ = {AIRPORT OFFICE}

and

R_2 = COMPANY OFFICE with designated key $\mathbf{K}_2$ = {OFFICE}.

Unfortunately, the synthesis approach does not guarantee BCNF.

Example 6.28 Let $R = A\ B\ C\ D\ E$ and let $F = \{A \to B\ C,\ B\ C \to A,\ B\ C\ D \to E,\ E \to C\}$. The annular cover for F produced by SYNTHESIZE is

$$G = \{(A,\ B\ C),\ (B\ C\ D) \to E,\ (E) \to C\}.$$

The second CFD in G yields the relation scheme

$$R_2 = B\ C\ D\ E \quad \text{with designated key } \mathbf{K}_2 = \{B\ C\ D\}.$$

R_2 is not in BCNF since $E \to C$ and E is not a key of R_2. Choosing an equivalent annular cover

$$G' = \{(A,\ B\ C),\ (A\ D) \to E,\ (E) \to C\}$$

will produce a database scheme in BCNF.

6.7.1 Problems with Boyce-Codd Normal Form

We have seen that given a set of FDs F over R, it is possible to find a 3NF database scheme that completely characterizes F. The same is not true for BCNF. Exhaustive consideration of Example 6.26 will show there is no BCNF database scheme completely characterizing the given set of FDs. We are faced with a choice of BCNF or enforceable FDs.

Not only is it possible for a set of FDs not to have a complete BCNF database scheme, it is NP-complete to decide if a given database scheme is in BCNF.

6.8 EXERCISES

6.1* Assume we restrict database schemes over **U** so that for any two relation schemes R_1 and R_2 in a database scheme, $R_1 \not\subseteq R_2$. Let **U** contain

n attributes. How many database schemes of p relation schemes are there over $\mathbf{U}$ (ignoring keys)?

6.2 Let database scheme $\mathbf{R} = \{A\,B\,C,\ A\,D\,E,\ C\,E\}$. Prove that the set of FDs $F = \{A\,B \rightarrow C,\ C \rightarrow E,\ E \rightarrow C,\ C \rightarrow D,\ A\,B \rightarrow E\}$ is enforceable on $\mathbf{R}$.

6.3 Give an example of a relation in 3NF that has some prime attribute transitively dependent upon a key.

6.4 Let R_1 and R_2 be relation schemes with $R_1 \cap R_2 = X$. Show that for any relation $r(R_1\,R_2)$ that satisfies $X \rightarrow R_2$,

$$r = \pi_{R_1}(r) \bowtie \pi_{R_2}(r).$$

6.5 Let F be a set of FDs containing no FDs of the form $\emptyset \rightarrow Y$. Show that any relation scheme R with two attributes can have no transitive dependencies under F.

6.6 Let R = STUDENT# NAME BIRTHDAY AGE ADVISOR DEPARTMENT SEMESTER COURSE GRADE, with key STUDENT# SEMESTER COURSE and FDs STUDENT# $\rightarrow$ NAME BIRTHDAY AGE ADVISOR DEPARTMENT, BIRTHDAY $\rightarrow$ AGE, and ADVISOR $\rightarrow$ DEPARTMENT. Find a 3NF database scheme for R.

6.7 Find a relation $r(R)$ such that r decomposes losslessly onto some set of three relation schemes, but r does not decompose losslessly onto any pair of relation schemes. In both cases assume no relation scheme is the same as R.

6.8 (a) Show that a set of $2n$ FDs can induce 2^n keys on a relation scheme with $2n$ attributes.

(b) Try to generalize the result of part a) to sets of m FDs on relation schemes with n attributes.

6.9 Let $\mathbf{R} = \{R_1, R_2, \ldots, R_p\}$ be a database scheme over $\mathbf{U}$ where $R_1 \subseteq R_2$. Show that if a relation $r(\mathbf{U})$ decomposes losslessly onto the schemes in $\mathbf{R}$, then r decomposes losslessly onto the relation schemes in $\mathbf{R}' = \{R_2, R_3, \ldots, R_p\}$, and conversely.

6.10 Give a set of FDs F and a database scheme $\mathbf{R}$ completely characterizing F, such that $\mathbf{R}$ has more than $|\bar{E}_F|$ relation schemes, yet no relation scheme can be removed from $\mathbf{R}$ and still have $\mathbf{R}$ completely characterize F.

6.11 Show that the method for producing a database scheme $\mathbf{R}$ from a set of FDs F by using a nonredundant cover F' for F (as outlined at the beginning of Section 6.5.2) guarantees that $\mathbf{R}$ completely characterizes F.

6.12 Compare the database schemes produced by decomposition and synthesis in Examples 6.12 and 6.19.

6.13 Show that SYNTHESIZE may not work correctly if some CFD in the annular cover G contains either shiftable or extraneous attributes.

6.14 Use both synthesis and decomposition to obtain a 3NF database scheme for the set of FDs in Example 5.13.

6.15 Prove Lemma 6.4.

6.16 Give an example of a relation scheme R in a database scheme $\mathbf{R}$ with an external dependency $K \rightarrow Y$, $Y \not\rightarrow K$, $Y \rightarrow A$, where K is a key of R, $A \not\rightarrow Y$, but A is not removable in R.

6.17* Prove that any complete database scheme $\mathbf{R}$ for a set of FDs F produced by decomposition from a single relation scheme R does not contain any removable attributes.

Definition 6.19 Let F be a set of FDs, let $\mathbf{R} = \{R_1, R_2, \ldots, R_p\}$ be a database scheme, and let G be the set of FDs in F^+ that apply to relation schemes in $\mathbf{R}$. F is *indirectly enforceable* on $\mathbf{R}$ if $\mathbf{R}$ has the lossless join property with respect to G.

6.18* Prove that for any database scheme $\mathbf{R}$ produced by decomposition from a relation scheme R and a set of FDs F, F is indirectly enforceable on $\mathbf{R}$.

6.19 Consider the algorithm SYNTHESIZE as modified in Section 6.5.4. Show that adding $R \rightarrow C$ to F does not produce an extra designated key unless it produces an extra relation scheme. Show that $R \rightarrow C$ does not produce an extra relation scheme if there is an FD $X \rightarrow Y$ in F with $X \rightarrow R$ under F.

6.20 Find the complexity of Algorithm 6.2 in Section 6.6. You may assume that information calculated by SYNTHESIZE is available to AVOID.

6.21 Let R be a relation scheme in a complete database scheme $\mathbf{R}$. Show that if B is not avoidable in R, then B is not avoidable in any relation scheme R' obtained by changing the designated keys of R and removing removable attributes.

6.22 Give an algorithm to put a complete database scheme produced by SYNTHESIZE into LTK normal form.

6.23* Let $\mathbf{R}$ be a complete BCNF database scheme for the set of FDs F and let R be a relation scheme in $\mathbf{R}$. Let B be an attribute in R. Let $\mathbf{R}'$ be the database scheme obtained by removing B from R in $\mathbf{R}$. Let F' be the subset of F^+ that applies to some relation scheme in $\mathbf{R}'$. Prove that B is avoidable in $\mathbf{R}$ if and only if $F \equiv F'$.

6.24 Show that BCNF implies 3NF.

6.25 Prove the equivalence of the two definitions of BCNF in Section 6.7.

6.26 Exhibit a database scheme that is in BCNF but not LTK normal form and vice versa.

6.9 BIBLIOGRAPHY AND COMMENTS

Second and third normal forms were introduced by Codd [1971b, 1972a], who showed how to achieve them through normalization. Kent [1973] provides an introduction to normal forms. Early proposals for synthesis algorithms were given by Delobel and Casey [1973] and Wang and Wedekind [1975], but they contained some imperfections. Bernstein [1976b] was the first to give a synthesis algorithm for a complete database scheme for a set of FDs. Osborn [1977], Dayal and Bernstein [1978a], and Biskup, Dayal, and Bernstein [1979] discuss 3NF database schemes that meet the lossless join condition. Other algorithms for 3NF schemes are given by Beeri and Bernstein [1979] and Pichat and Delobel [1979]. Avoidable attributes were introduced by Ling, Tompa, and Kameda [1981], who propose an algorithm for their removal. BCNF was introduced by Codd [1974].

Fagin [1977] compares decomposition and synthesis for achieving normal form schemes. Beeri, Bernstein, and Goodman [1978] discuss the goals of normalization. Heath [1971] attempts to classify update anomalies by their severity. They also point out that there can be many normalized schemes for a given set of dependencies, and there is no clear criterion for which is the "best". Bernstein and Goodman [1980b] raise questions about how well BCNF avoids update anomalies.

Lucchesi and Osborn [1978] show that several problems connected with normalization, such as finding nonprime attributes and minimum keys, are NP-complete. Beeri and Bernstein [1979] use those results to show that determining if a given database scheme is in BCNF and determining whether a set of FDs has a complete BCNF scheme are NP-complete problems. Osborn [1979a] and LeDoux and Parker [1980] give algorithms for determining if a set of FDs has a complete BCNF scheme, although both can use exponential time in the worst case. Tsou and Fischer [1980] give a polynomial-time algorithm for finding a BCNF database scheme for a set of FDs with the lossless join property, but the scheme may not be complete. Jou [1980] and Tsou [1980] both examine complexity issues related to normalization.