APPENDIX 7.A: SECOND AND THIRD NORMAL FORMS

Second (2NF) and third (3NF) normal forms were proposed before BCNF. Conceptually, BCNF makes 2NF and 3NF obsolete although many books still reference the earlier forms. BCNF is a superior definition because it is simpler and covers two special cases not covered by 3NF. The special cases are rare, but the simplified definition makes BCNF superior to the combination of 2NF/3NF. This appendix covers 2NF and 3NF as reference material in case you see these normal forms mentioned in books and websites.

The definitions of 2NF and 3NF distinguish between key and nonkey columns.[8] A column is a key column if it is part of a candidate key or a candidate key by itself. Recall that a candidate key is a minimal set of column(s) that has unique values in a table. Minimality means that none of the columns can be removed without losing the uniqueness property. Essentially, candidate keys do not have extra columns. Nonkey columns are any other columns. In the big university database table shown in Table 7A-1, the combination of (*StdNo*, *OfferNo*) is the only candidate key. Other columns such as *StdCity* and *StdClass* are nonkey columns.

2NF and 3NF involve the relationship between key and nonkey columns and the relationship between nonkey columns. The basic goal of 2NF and 3NF is to produce tables in which every key determines the nonkey columns and nonkey columns do not determine other nonkey columns. An easy way to remember the definitions of both 2NF and 3NF is shown in the following definition.

Combined Definition of 2NF and 3NF: a table is in 3NF if each nonkey column depends on all candidate keys, whole candidate keys, and nothing but candidate keys.[9]

Second Normal Form

To understand this definition, let us break it down to the 2NF and 3NF parts. The **2NF definition** uses the first part of the definition as shown in the following definition.

2NF Definition
a table is in 2NF if each nonkey column depends on whole candidate keys, not on a subset of any candidate key.

To see if a table is in 2NF, you should look for FDs that violate the definition. An FD in which part of a key determines a nonkey column violates 2NF. If all candidate keys contain only one column, the table is in 2NF. Looking at the FDs in Table 7A-2, you can easily detect violations of 2NF. For example, *StdCity* is a nonkey column but *StdNo*, not the entire primary key (combination of *StdNo* and *OfferNo*), determines it.

TABLE 7A-1
Sample Data for the Big University Database Table

StdNo	StdCity	StdClass	OfferNo	OffTerm	OffYear	EnrGrade	CourseNo	CrsDesc
S1	SEATTLE	JUN	O1	FALL	2017	3.5	C1	DB
S1	SEATTLE	JUN	O2	FALL	2017	3.3	C2	VB
S2	BOTHELL	JUN	O3	SPRING	2018	3.1	C3	OO
S2	BOTHELL	JUN	O2	FALL	2017	3.4	C2	VB

TABLE 7A-2
List of FDs for the Big University Database Table

StdNo → StdCity, StdClass
OfferNo → OffTerm, OffYear, CourseNo, CrsDesc
CourseNo → CrsDesc
StdNo, OfferNo → EnrGrade

[8] In some academic literature, key columns are known as prime, and nonkey columns as nonprime.

[9] You can remember this definition by its analogy to the traditional justice oath: "Do you swear to tell the truth, the whole truth, and nothing but the truth, ...".

2NF Violation: an FD in which part of key determines a nonkey violates 2NF. A table with only single column candidate keys cannot violate 2NF.

The only FDs that satisfy the 2NF definition are *StdNo*, *OfferNo* → *EnrGrade* and *CourseNo* → *CrsDesc*.

To place the table into 2NF, you split the original table into smaller tables that satisfy the 2NF definition. In each smaller table, the entire primary key (not part of the primary key) should determine the nonkey columns. The splitting process involves the project operator of relational algebra. For the university database table, three projection operations split it so that the underlined primary key determines the nonkey columns in each table below.

UnivTable1 (StdNo, StdCity, StdClass)
UnivTable2 (OfferNo, OffTerm, OffYear, CourseNo, CrsDesc)
UnivTable3 (StdNo, OfferNo, EnrGrade)

The splitting process should preserve the original table in two ways. First, the original table should be recoverable using natural join operations on the smaller tables. Second, the FDs in the original table should be derivable from the FDs in the smaller tables. Technically, the splitting process is known as a nonloss, dependency-preserving decomposition. Some of the references at the end of this chapter explain the theory underlying the splitting process.

After splitting the original table into smaller tables, you should add referential integrity constraints to connect the tables. Whenever a table is split, the splitting column becomes a foreign key in the table in which it is not a primary key. For example, *StdNo* is a foreign key in *UnivTable3* because the original university table was split on this column. Therefore, you should define a referential integrity constraint stating that *UnivTable3.StdNo* refers to *UnivTable1.StdNo*. The *UnivTable3* table is repeated below with its referential integrity constraints.

UnivTable3 (StdNo, OfferNo, EnrGrade)
FOREIGN KEY (StdNo) REFERENCES UnivTable1
FOREIGN KEY (OfferNo) REFERENCES UnivTable2

Third Normal Form

UnivTable2 still has modification anomalies. For example, you cannot add a new course unless the *OfferNo* column value is known. To eliminate the modification anomalies, the definition of 3NF should be applied.

3NF Definition: a table is in 3NF if it is in 2NF and each nonkey column depends only on candidate keys, not on other nonkey columns.

An FD in which a nonkey column determines another nonkey column violates 3NF. In *UnivTable2* above, the FD (*CourseNo* → *CrsDesc*) violates 3NF because both columns, *CourseNo* and *CrsDesc* are nonkey. To fix the violation, split *UnivTable2* into two tables, as shown below, and add a foreign key constraint.

UnivTable2-1 (OfferNo, OffTerm, OffYear, CourseNo)
FOREIGN KEY (CourseNo) REFERENCES UnivTable2-2
UnivTable2-2 (CourseNo, CrsDesc)

Transitive Dependency
an FD derived by the law of transitivity. Transitive FDs should not be recorded as input to the normalization process.

An equivalent way to define 3NF is that 3NF prohibits **transitive dependencies**. A transitive dependency is a functional dependency derived by the law of transitivity. The law of transitivity says that if an object A is related to B and B is related to C, then you can conclude that A is related to C. For example, the < operator obeys the

transitive law for real numbers: A < B and B < C implies that A < C. Functional dependencies, like the < operator, obey the law of transitivity: A → B, B → C, then A → C. In Figure 7.2, *OfferNo* → *CrsDesc* is a transitive dependency derived from *OfferNo* → *CourseNo* and *CourseNo* → *CrsDesc*.

Because transitive dependencies are easy to overlook, the preferred definition of 3NF does not use transitive dependencies. In addition, you learned in Section 7.2.3 to omit derived dependencies such as transitive dependencies in the list of FDs for a table.

Combined Example of 2NF and 3NF

The big patient table as depicted in Table 7A-3 provides another example for applying your knowledge of 2NF and 3NF. The big patient table contains facts about patients, health care providers, patient visits to a clinic, and diagnoses made by health care providers. The big patient table contains a combined primary key consisting of the combination of *VisitNo* and *ProvNo* (provider number). Like the big university database table depicted in Table 7A-1, the big patient table reflects a poor table design with many redundancies. Table 7A-4 lists the associated FDs. You should verify that the sample rows in Table 7A-3 do not falsify the FDs.

As previously discussed, FDs that violate 2NF involve part of a key determining a nonkey. Many of the FDs in Table 7A-4 violate the 2NF definition because the combination of *VisitNo* and *ProvNo* is the primary key. Thus, the FDs with only *VisitNo* or *ProvNo* in the LHS violate 2NF. To alleviate the 2NF violations, split the big patient table so that the violating FDs are associated with separate tables. In the revised list of tables, *PatientTable1* and *PatientTable2* contain the violating FDs. *PatientTable3* retains the remaining columns.

PatientTable1 (ProvNo, ProvSpecialty)
PatientTable2 (VisitNo, VisitDate, PatNo, PatAge, PatCity, PatZip)
PatientTable3 (VisitNo, ProvNo, Diagnosis)
FOREIGN KEY (VisitNo) REFERENCES PatientTable2
FOREIGN KEY (ProvNo) REFERENCES PatientTable1

PatientTable1 and *PatientTable3* are in 3NF because there are no nonkey columns that determine other nonkey columns. However, *PatientTable2* violates 3NF because the FDs *PatNo* → *PatZip, PatAge* and *PatZip* → *PatCity* involve nonkey columns that determine other nonkey columns. To alleviate the 3NF violations, split *PatientTable2* into three tables as shown in the revised table list. In the revised list of tables, *PatientTable2-1*

TABLE 7A-3
Sample Data for the Big Patient Table

VisitNo	VisitDate	PatNo	PatAge	PatCity	PatZip	ProvNo	ProvSpecialty	Diagnosis
V10020	1/13/2018	P1	35	DENVER	80217	D1	INTERNIST	EAR INFECTION
V10020	1/13/2018	P1	35	DENVER	80217	D2	NURSE PRACTITIONER	INFLUENZA
V93030	1/20/2018	P3	17	ENGLEWOOD	80113	D2	NURSE PRACTITIONER	PREGNANCY
V82110	1/18/2018	P2	60	BOULDER	85932	D3	CARDIOLOGIST	MURMUR

TABLE 7A-4
List of FDs for the Big Patient Table

PatNo → PatAge, PatCity, PatZip
PatZip → PatCity
ProvNo → ProvSpecialty
VisitNo → PatNo, VisitDate, PatAge, PatCity, PatZip
VisitNo, ProvNo → Diagnosis

and *PatientTable2-2* contain the violating FDs, while *PatientTable2-3* retains the remaining columns.

PatientTable2-1 (PatNo, PatAge, PatZip)
FOREIGN KEY (PatZip) REFERENCES PatientTable2-2
PatientTable2-2 (PatZip, PatCity)
PatientTable2-3 (VisitNo, PatNo, VisitDate)
FOREIGN KEY (PatNo) REFERENCES PatientTable2-1

Using 2NF and 3NF requires two normalization steps. The normalization process can be performed in one step using BCNF and the simple synthesis procedure, as presented in section 7.2.