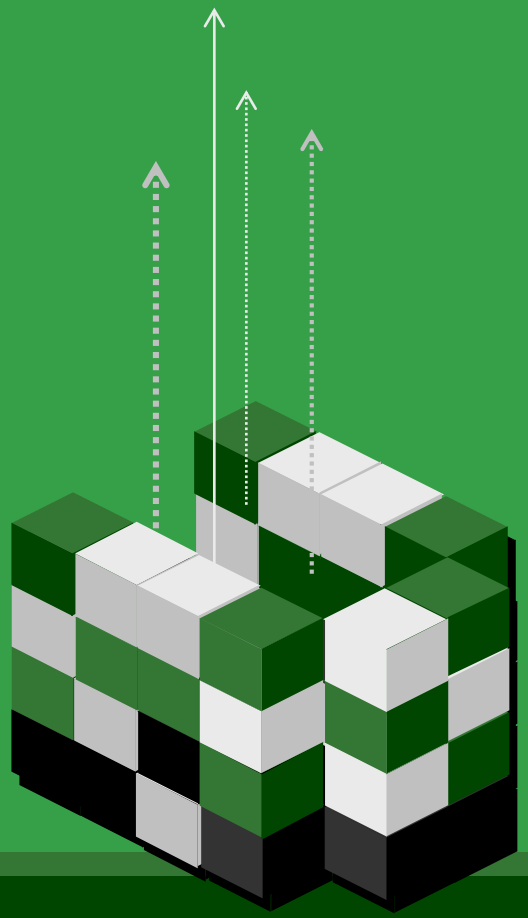
chaper 3

The Relational Model And Normalization



2、Normal Forms

All relations are not equal. Some are easy to process, and others are problematic. Relations are categorized into normal forms based on the kinds of problems that they have. Knowledge of these normal forms will help you create appropriate database designs. To understand normal forms, we need first to define modification anomalies.

- **Modification Anomalies**

Consider the EQUIPMENT_REPAIR relation in Figure 3-10, which stores data about manufacturing equipment and equipment repairs. Suppose we delete the data for repair number 2100. When we delete this row (the second one in Figure 3-10), we remove not only data about the repair, but also data about the machine itself. We will no longer know, for example, that the machine was a Lathe and that its AcquisitionPrice was 4750.00. When we delete one row, the structure of this table forces us to lose facts about two different things, a machine and a repair. This condition is called a deletion anomaly.

Now suppose we want to enter the first repair for a piece of equipment. To enter repair data, we need to know not just RepairNumber, RepairDate, and RepairCost, but also ItemNumber, EquipmentType, and AcquisitionCost. If we work in the repair department, this is a problem, because we are unlikely to know the value of AcquisitionCost. The structure of this table forces us to enter facts about two entities when we just want to enter facts about one. This condition is called an insertion anomaly.

2、Normal Forms

➢**Modification Anomalies**

Finally, suppose we want to change existing data. If we alter a value of RepairNumber, RepairDate, or RepairCost, there is no problem. But if we alter a value of ItemNumber, EquipmentType, or AcquisitionCost, we may create a data inconsistency. To see why, suppose we update the last row of the table in Figure 3-10 using the data (100, 'Drill Press', 5500, 2500, '08/17/09', 275).

Figure 3-11 shows the table after this erroneous update. The drill press has two different AcquisitionCosts. Clearly, this is an error. Equipment cannot be acquired at two different costs. If there were, say, 10,000 rows in the table, however, it might be very difficult to detect this error. This condition is called an update anomaly.

Figure3-10 The EQUIPMENT_REPAIR Table

	ItemNumber	EquipmentType	AcquisitionCost	RepairNumber	RepairDate	RepairCost
1	100	Drill Press	3500.00	2000	2011-05-05 ...	375.00
2	200	Lathe	4750.00	2100	2011-05-07 ...	255.00
3	100	Drill Press	3500.00	2200	2011-06-19 ...	178.00
4	300	Mill	27300.00	2300	2011-06-19 ...	1875.00
5	100	Drill Press	3500.00	2400	2011-07-05 ...	0.00
6	100	Drill Press	3500.00	2500	2011-08-17 ...	275.00

2、Normal Forms

➢**Modification Anomalies**

Figure 3-11 The EQUIPMENT_REPAIR Table After an Incorrect Update

	ItemNumber	EquipmentType	AcquisitionCost	RepairNumber	RepairDate	RepairCost
1	100	Drill Press	3500.00	2000	2011-05-05 ...	375.00
2	200	Lathe	4750.00	2100	2011-05-07 ...	255.00
3	100	Drill Press	3500.00	2200	2011-06-19 ...	178.00
4	300	Mill	27300.00	2300	2011-06-19 ...	1875.00
5	100	Drill Press	3500.00	2400	2011-07-05 ...	0.00
6	100	Drill Press	5500.00	2500	2011-08-17 ...	275.00

By the way Notice that the EQUIPMENT_REPAIR table in Figures 3-10 and 3-11 duplicates data. For example, the AcquisitionCost of the same item of equipment appears several times. Any table that duplicates data is susceptible to update anomalies like the one in Figure 3-11. A table that has such inconsistencies is said to have data integrity problems.

As you will learn in Chapter 4, to improve query speed we sometimes design a table to have duplicated data. Be aware, however, that any time we design a table this way we open the door to data integrity problems.

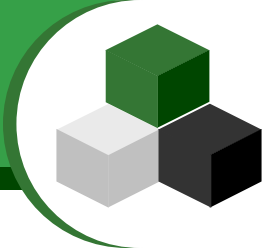
2、Normal Forms

➢**A Short History of Normal Forms**

When Codd defined the relational model, he noticed that some tables had modification anomalies. In his second paper,[2] he defined first normal form, second normal form, and third normal form. He defined first normal form (1NF) as the set of conditions for a relation shown in Figure 3-4. Any table meeting the conditions in Figure 3-4 is therefore a relation in 1NF. Codd also noted that some tables (or, interchangeably in this book, relations) in 1NF had modification anomalies. He found that he could remove some of those anomalies by applying certain conditions. A relation that met those conditions, which we will discuss later in this chapter, was said to be in second normal form (2NF). He also observed, however, that relations in 2NF could also have anomalies, and so he defined third normal form (3NF), which is a set of conditions that removes even more anomalies, and which we will also discuss later in this chapter. As time went by, other researchers found still other ways that anomalies can occur, and the conditions for Boyce-Codd Normal Form (BCNF) were defined.

These normal forms are defined so that a relation in BCNF is in 3NF, a relation in 3NF is in 2NF, and a relation in 2NF is in 1NF. Thus, if you put a relation into BCNF, it is automatically in the lesser normal forms.

2、Normal Forms



➢**A Short History of Normal Forms**

Normal forms 2NF through BCNF concern anomalies that arise from functional dependencies. Other sources of anomalies were found later. They led to the definition of fourth normal form (4NF) and fifth normal form (5NF), both of which we will discuss later in this chapter. So it went, with researchers chipping away at modification anomalies, each one improving on the prior normal form.

In 1982, Fagin published a paper that took a different tack.[3] Instead of looking for just another normal form, Fagin asked, “What conditions need to exist for a relation to have no anomalies?” In that paper, he defined domain/key normal form (DK/NF). Fagin ended the search for normal forms by showing that a relation in DK/NF has no modification anomalies and, further, that a relation that has no modification anomalies is in DK/NF. DK/NF is discussed in more detail later in this chapter.

2、Normal Forms

➢Normalization Categories

As shown in Figure 3-12, normalization theory can be divided into three major categories. Some anomalies arise from functional dependencies, some arise from multivalued dependencies, and some arise from data constraints and odd conditions.

BCNF, 3NF, and 2NF, are all concerned with anomalies that are caused by functional dependencies. A relation that is in BCNF has no modification anomalies from functional dependencies. It is also automatically in 2NF and 3NF, and, therefore, we will focus on transforming relations into BCNF. However, it is instructive to work through the progression of normal forms from 1NF to BCNF in order to understand how each normal form deals with specific anomalies, and we will do this later in this chapter 4.

As shown in the second row of Figure 3-12, some anomalies arise because of another kind of dependency called a multivalued dependency. Those anomalies can be eliminated by placing each multivalued dependency in a relation of its own, a condition known as 4NF. You will see how to do that in the last section of this chapter.

The third source of anomalies is esoteric. These problems involve specific, rare, and even strange data constraints. Accordingly, we will not discuss them in this text.

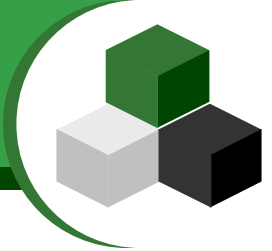
2、Normal Forms

➢**Normalization Categories**

Figure 3-12 Summary of Normalization Theory

Source of Anomaly	Normal Forms	Design Principles
Functional dependencies	1NF, 2NF, 3NF. BCNF	BCNF: Design tables so that every determinant is a candidate key.
Multivalued dependencies	4NF	4NF: Move each multivalued dependency to a table of its own.
Data constraints and oddities	5NF, DK/NF	DK/NF: Make every constraint a logical consequence of candidate keys and domains.

2、Normal Forms



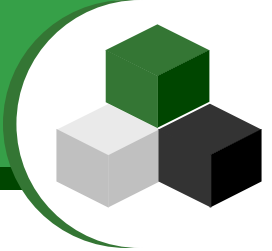
➢From First Normal Form to Boyce-Codd Normal

Any table that meets the definition of a relation in Figure 3-4 is defined as being in 1NF. This means that the following must hold: The cells of a table must be a single value, and neither repeating groups nor arrays are allowed as values; all entries in a column must be of the same data type; each column must have a unique name, but the order of the columns in the table is not significant; no two rows in a table may be identical, but the order of the rows is not significant.

Second Normal Form

When Codd discovered anomalies in 1NF tables, he defined 2NF to eliminate some of these anomalies. A relation is 2NF if and only if it is in 1NF and all non-key attributes are determined by the entire primary key. This means that if the primary key is a composite primary key, then no non-key attribute can be determined by an attribute or set of attributes that make up only part of the key. Thus, if you have a relation R (A, B, N, O, P) with the composite key (A, B), then none of the non-key attributes N, O, or P can be determined by just A or just B.

2、Normal Forms



➢From First Normal Form to Boyce-Codd Normal

Note that the only way a non-key attribute can be dependent on part of the primary key is if there is a composite primary key. This means that relations with single-attribute primary keys are automatically in 2NF.

For example, consider the STUDENT_ACTIVITY relation:

STUDENT_ACTIVITY (StudentID, Activity, ActivityFee)

The STUDENT_ACTIVITY relation is shown with sample data in Figure 3-13. Note that STUDENT_ACTIVITY has the composite primary key (StudentID, Activity), which allows us to determine the fee a particular student will have to pay for a particular activity. However, because fees are determined by activities, Fee is also functionally dependent on just Activity itself, and we can say that Fee is partially dependent on the key of the table. The set of functional dependencies is therefore:

(StudentID, Activity) : (ActivityFee)

(Activity) : (ActivityFee)

2、Normal Forms

➢**From First Normal Form to Boyce-Codd Normal**

Figure 3-13 **The STUDENT_ACTIVITY Relation**

STUDENT_ACTIVITY

	StudentID	Activity	ActivityFee
1	100	Golf	65.00
2	100	Skiing	200.00
3	200	Skiing	200.00
4	200	Swimming	50.00
5	300	Skiing	200.00
6	300	Swimming	50.00
7	400	Golf	65.00
8	400	Swimming	50.00

2、Normal Forms

➢From First Normal Form to Boyce-Codd Normal

Thus, there is a non-key attribute determined by part of the composite primary key, and the STUDENT_ACTIVITY relation is not in 2NF. What do we do in this case? We will have to move the columns of the functional dependency based on the partial primary key attribute into a separate relation while leaving the determinant in the original relation as a foreign key. We will end up with two relations:

STUDENT_ACTIVITY (StudentID, Activity)

ACTIVITY_FEE (Activity, ActivityFee)

The Activity column in STUDENT_ACTIVITY becomes a foreign key. The new relations are shown in Figure 3-14. Now, are the two new relations in 2NF? Yes. STUDENT_ACTIVITY still has a composite primary key, but now has no attributes that are dependent on only a part of this composite key. ACTIVITY _FEE has a set of attributes (just one each in this case) that are dependent on the entire primary key.

2、Normal Forms

➢**From First Normal Form to Boyce-Codd Normal**

Figure 3-14

The 2NF STUDENT_ACTIVITY and ACTIVITY_FEE Relations

STUDENT_ACTIVITY

	StudentID	Activity
1	100	Golf
2	100	Skiing
3	200	Skiing
4	200	Swimming
5	300	Skiing
6	300	Swimming
7	400	Golf
8	400	Swimming

ACTIVITY_FEE

	Activity	ActivityFee
1	Golf	65.00
2	Skiing	200.00
3	Swimming	50.00

2、Normal Forms

➢From First Normal Form to Boyce-Codd Normal

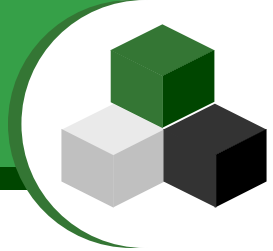
Third Normal Form

However, the conditions necessary for 2NF do not eliminate all anomalies. To deal with additional anomalies, Codd defined 3NF. A relation is in 3NF if and only if it is in 2NF and there are no non-key attributes determined by another non-key attribute. The technical name for a nonkey attribute determined by another non-key attribute is transitive dependency. We can therefore restate the definition of 3NF: A relation is in 3NF if and only if it is in 2NF and it has no transitive dependencies. Thus, in order for our relation R (A, B, N, O, P) to be in 3NF, none of the non-key attributes N, O, or P can be determined by N, O, or P.

For example, consider the relation STUDENT_HOUSING (StudentID, Building, Fee) shown in Figure 3-15. The STUDENT_HOUSING schema is:

STUDENT_HOUSING (StudentID, Building, HousingFee)

2、Normal Forms



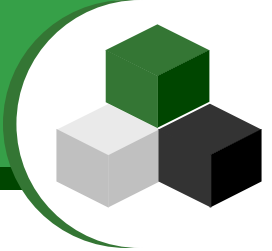
➢**From First Normal Form to Boyce-Codd Normal**

Figure 3-15 **The STUDENT_HOUSING Relation**

STUDENT_HOUSING

	StudentID	Building	BuildingFee
1	100	Randolph	3200.00
2	200	Ingersoll	3400.00
3	300	Randolph	3200.00
4	400	Randolph	3200.00
5	500	Pitkin	3500.00
6	600	Ingersoll	3400.00
7	700	Ingersoll	3400.00
8	800	Pitkin	3500.00

2、Normal Forms



➢From First Normal Form to Boyce-Codd Normal

Third Normal Form

Here, we have a single-attribute primary key, StudentID, so the relation is in 2NF because there is no possibility of a non-key attribute being dependent on only part of the primary key. Furthermore, if we know the student, we can determine the building where he or she is residing, so:

(StudentID) : Building

However, the building fee is independent of which student is housed in the building, and, in fact, the same fee is charged for every room in a building. Therefore, Building determines HousingFee:

(Building) : (HousingFee)

Thus, a non-key attribute (HousingFee) is functionally determined by another non-key attribute (Building), and the relation is not in 3NF.

2、Normal Forms

➢From First Normal Form to Boyce-Codd Normal

Third Normal Form

To put the relation into 3NF, we will have to move the columns of the functional dependency into a separate relation while leaving the determinant in the original relation as a foreign key. We will end up with two relations:

STUDENT_HOUSING (StudentID, Building)

BUILDING_FEE (Building, HousingFee)

The Building column in STUDENT_HOUSING becomes a foreign key. The two relations are now in 3NF (work through the logic yourself to make sure you understand 3NF), and are shown in Figure 3-16.

Figure 3-16 **The 3NF STUDENT_HOUSING and**

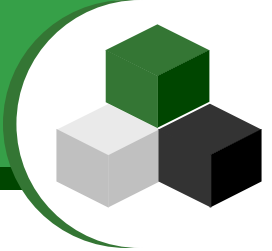
STUDENT_HOUSING

	StudentID	Building
1	100	Randolph
2	200	Ingersoll
3	300	Randolph
4	400	Randolph
5	500	Pitkin
6	600	Ingersoll
7	700	Ingersoll
8	800	Pitkin

HOUSING_FEE

	Building	BuildingFee
1	Ingersoll	3400.00
2	Pitkin	3500.00
3	Randolph	3200.00

2、Normal Forms



➢From First Normal Form to Boyce-Codd Normal

Boyce-Codd Normal Form

Some database designers normalize their relations to 3NF. Unfortunately, there are still anomalies due to functional dependences in 3NF. Together with Raymond Boyce, Codd defined BCNF to fix this situation. A relation is in BCNF if and only if it is in 3NF and every determinant is a candidate key.

For example, consider the relation STUDENT_ADVISIOR shown in 3-17, where a student (StudentID) can have one or more majors (Major), a major can have one or more faculty advisors (AdvisorName), and a faculty member advises in only one major area. Note that the figure shows two students (StudentIDs 700 and 800) with double majors (both students show Majors of Math and Psychology), and two Subjects (Math and Psychology) with two Advisors.

2、Normal Forms

➢**From First Normal Form to Boyce-Codd Normal**

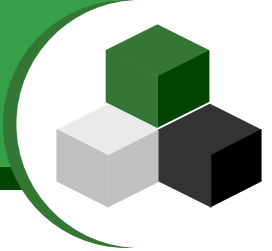
Boyce-Codd Normal Form

Figure 3-17 **The STUDENT_ADVISOR Relation**

STUDENT_ADVISOR

	StudentID	Subject	AdvisorName
1	100	Math	Cauchy
2	200	Psychology	Jung
3	300	Math	Riemann
4	400	Math	Cauchy
5	500	Psychology	Perls
6	600	English	Austin
7	700	Psychology	Perls
8	700	Math	Riemann
9	800	Math	Cauchy
10	800	Psychology	Jung

2、Normal Forms



➢From First Normal Form to Boyce-Codd Normal

Boyce-Codd Normal Form

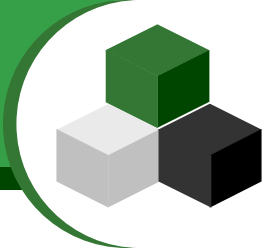
Because students can have several majors, StudentID does not determine Major. Moreover, because students can have several advisers, StudentID does not determine AdvisorName. Therefore, StudentID by itself cannot be a key. However, the composite key (StudentID, Major) determines AdvisorName, and the composite key (StudentID, AdvisorName) determines Major. This gives us (StudentID, Major) and (StudentId, AdvisorName) as two candidate keys. We can select either of these as the primary key for the relation. Thus, two STUDENT_ADVISOR schemas with different candidate keys are possible:

STUDENT_ADVISOR (StudentID, Major, AdvisorName)

and

STUDENT_ADVISOR (StudentID, Major, AdvisorName)

2、Normal Forms



➢From First Normal Form to Boyce-Codd Normal

Boyce-Codd Normal Form

Note that STUDENT_ADVISOR is in 2NF because it has no non-key attributes in the sense that every attribute is a part of at least one candidate key. This is a subtle condition, based on the fact that technically the definition of 2NF states that no non-prime attribute can be partially dependent on a candidate key, where a non-prime attribute is an attribute that is not contained in any candidate key. Furthermore, STUDENT_ADVISOR is in 3NF because there are no transitive dependencies in the relation.

The two candidate keys for this relation are overlapping candidate keys, because they share the attribute StudentID. When a table in 3NF has overlapping candidate keys, it can still have modification anomalies based on functional dependencies. In the STUDENT_ADVISOR relation, there will be modification anomalies because there is one other functional dependency in the relation. Because a faculty member can be an advisor for only one major area, AdvisorName determines Major. Therefore, AdvisorName is a determinant, but not a candidate key.

2、Normal Forms

➢From First Normal Form to Boyce-Codd Normal

Boyce-Codd Normal Form

Suppose that we have a student (StudentID = 300) majoring in psychology (Major = Psychology) with faculty advisor Perls (AdvisorName = Perls). Further, assume that this row is the only one in the table with the AdvisorName value of Perls. If we delete this row, we will lose all data about Perls. This is a deletion anomaly. Similarly, we cannot insert the data to represent the Economics advisor Keynes until a student majors in Economics. This is an insertion anomaly. Situations like this led to the development of BCNF.

What do we do with the STUDENT_ADVISOR relation? As before, we move the functional dependency creating the problem to another relation while leaving the determinant in the original relation as a foreign key. In this case, we will create the relations:

STUDENT_ADVISOR (StudentID, AdvisorName)
ADVISOR_SUBJECT (AdvisorName, Major)

The AdvisorName column in STUDENT_ADVISOR is the foreign key, and the two final relations are shown in Figure 3-18.

2、Normal Forms

➢**From First Normal Form to Boyce-Codd Normal**

Boyce-Codd Normal Form

Figure 3-18 **The BCNF STUDENT_ADVISOR and ADVISOR_SUBJECT Relations**

STUDENT_ADVISOR

	StudentID	AdvisorName
1	100	Cauchy
2	200	Jung
3	300	Riemann
4	400	Cauchy
5	500	Perls
6	600	Austin
7	700	Perls
8	700	Riemann
9	800	Cauchy
10	800	Jung

ADVISOR_SUBJECT

	AdvisorName	Subject
1	Austin	English
2	Cauchy	Math
3	Jung	Psychology
4	Perls	Psychology
5	Riemann	Math

LOGO

Thank You!