Third Normal Form

Third normal form is designed to handle situations like the one you just read about in the preceding section. In terms of entities, the item relation does contain two entities: the merchandise item and the distributor. That alone should convince you that the relation needs to be broken down into two smaller relations, both of which are now in third normal form:

image

The theoretical definition of third normal form says:

▪ The relation is in second normal form.

▪ There are no transitive dependencies.

The functional dependencies found in the original relation are an example of a transitive dependency.

TRANSITIVE DEPENDENCIES

A transitive dependency exists when you have the following functional dependency pattern:

image

This is precisely the case with the original items relation. The only reason that the warehouse phone number is functionally dependent on the item number is because the distributor is functionally dependent on the item number, and the phone number is functionally dependent on the distributor. The functional dependencies are really:

image

Note: Transitive dependencies take their name from the transitive property in mathematics, which states that if a > b and b > c, then a > c.

There are two determinants in the original items relation, each of which should be the primary key of its own relation. However, it is not merely the presence of the second determinant that creates the transitive dependency. What really matters is that the second determinant is not a candidate key (could be used as a primary key) for the relation.

Consider, for example, this relation:

image

The item number is an arbitrary number that Antique Opticals assigns to each merchandise item. The UPC is an industry-wide code that is unique to each item as well. The functional dependencies in this relation are:

image

Is there a transitive dependency here? No, because the second determinant is a candidate key. (Antique Opticals could just as easily have used the UPC as the primary key.) There are no insertion, deletion, or modification anomalies in this relation; it describes only one entity—the merchandise item.

A transitive dependency therefore exists only when the determinant that is not the primary key is not a candidate key for the relation. For example, in the items table we have been using as an example, the distributor is a determinant, but not a candidate key for the table. (There can be more than one item coming from a single distributor.)

When you have a transitive dependency in a 2NF relation, you should break the relation into two smaller relations, each of which has one of the determinants in the transitive dependency as its primary key. The attributes determined by the determinant become nonkey attributes in each relation. This removes the transitive dependency—and its associated anomalies—and places the relation in third normal form.

Note: A second normal form relation that has no transitive dependencies is, of course, automatically in third normal form.

Boyce–Codd Normal Form

For most relations, third normal form is a good design objective. Relations in that state are free of most anomalies. However, occasionally you run into relations that exhibit special characteristics where anomalies still occur. Boyce–Codd normal form (BCNF), fourth normal form (4NF), and fifth normal form (5NF) were created to handle such special situations.

Note: If your relations are in third normal form and do not exhibit the special characteristics that BCNF, 4NF, and 5NF were designed to handle, then they are automatically in 5NF.

The easiest way to understand BCNF is to start with an example. Assume that Antique Opticals decides to add a relation to its database to handle employee work scheduling. Each employee works one or two 4-hour shifts a day at the store. During each shift, an employee is assigned to one station (a place in the store, such as the front desk or the stockroom). Only one employee works a station during the given shift.

A relation to handle the schedule might be designed as follows:

image

Given the rules for the scheduling (one person per station per shift), there are two possible primary keys for this relation: employee\_ID + date + shift or date + shift + station. The functional dependencies in the relation are:

image

Keep in mind that this holds true only because there is only one person working each station during each shift.

Note: There is very little difference between the two candidate keys as far as the choice of a primary key is concerned. In cases like this, you can choose either one.

This schedule relation exhibits overlapping candidate keys. (Both candidate keys have date and shift in common.) BCNF was designed to deal with relations that exhibit this characteristic.

To be in BCNF, a relation must meet the following rules:

▪ The relation is in third normal form.

▪ All determinants are candidate keys.

BCNF is considered to be a more general way of looking at 3NF because it includes those relations with the overlapping candidate keys. The sample schedule relation we have been considering does meet the criteria for BCNF because the two determinants are indeed candidate keys.

Fourth Normal Form

Like BCNF, fourth normal form was designed to handle relations that exhibit a special characteristic that does not arise too often. In this case, the special characteristic is something known as a multivalued dependency.

As an example consider the following relation:

image

A given movie can have more than one star; it can also have more than one producer. The same star can appear in more than one movie; a producer can also work on more than one movie (for example, see the instance in Figure 7.5). The relation must therefore include all columns in its key.

image

FIGURE 7.5 A relation with a multivalued dependency.

Because there are no nonkey attributes, this relation is in BCNF. Nonetheless, the relation exhibits anomalies:

▪ You cannot insert the stars of a movie without knowing at least one producer.

▪ You cannot insert the producer of a movie without knowing at least one star.

▪ If you delete the only producer from a movie, you lose information about the stars.

▪ If you delete the only star from a movie, you lose information about its producers.

▪ Each producer’s name is duplicated for every star in the movie. By the same token, each star’s name is duplicated for each producer of the movie. This unnecessary duplicated data forms the basis of a modification anomaly.

There are at least two unrelated entities in this relation: one that handles the relationship between a movie and its stars and another that handles the relationship between a movie and its producers. In a practical sense, that is the cause of the anomalies. (Arguably, there are also movie, star, and producer entities involved.)

However, in a theoretical sense, the anomalies are caused by the presence of a multivalued dependency in the same relation, which must be eliminated to get to fourth normal form. The rules for fourth normal form are:

▪ The relation is in BCNF.

▪ There are no multivalued dependencies.

MULTIVALUED DEPENDENCIES

A multivalued dependency exists when for each value of attribute A, there exists a finite set of values of attribute B that are associated with it, and a finite set of values of attribute C that are also associated with it. Attributes B and C are independent of each other.

In the example we have been using, there is just such a dependency. First, for each movie title, there is a group of actors (the stars) who are associated with the movie. For each title, there is also a group of producers who are associated with it. However, the actors and the producers are independent of one another. As you can see in the ER diagram in Figure 7.6, the producers and stars have no direct relationship (despite the relationships being M:M).

image

FIGURE 7.6 An ER diagram of the multivalued dependency.

Note: At this point, do not let semantics get in the way of database theory. Yes, it is true that producers fund the movies in which the actors are starring, but in terms of database relationships, there is no direct connection between the two.

The multivalued dependency can be written:

image

and read “title multidetermines star and title multidetermines producer.”

Note: To be strictly accurate, a functional dependency is a special case of a multivalued dependency where what is being determined is one value rather than a group of values.

To eliminate the multivalued dependency and bring this relation into fourth normal form, you split the relation placing each part of the dependency in its own relation:

image

With this design, you can independently insert and remove stars and producers without affecting the other. Star and producer names also appear only once for each movie with which they are involved.

Another way to look at this is to notice that movie\_stars and movie\_producers are actually composite entities. When we add them to the ER diagram, they resolve the M:M relationships (Figure 7.7).

image

FIGURE 7.7 Creating fourth normal form with composite entities.

Fifth Normal Form

Fifth normal form—also known as projection-join normal form—is designed to handle a general case of a multivalued dependency known as a join dependency.

Consider the following relation that is in fourth normal form, but not fifth:

image

This relation represents various series of discs, such as Star Trek or Rambo. Customers place orders for a series; when a customer orders a series, he or she must take all items in that series. Determining fifth normal form becomes relevant only when this type of rule is in place. If customers could request selected titles from a series, then the relation would be fine. Because it would be all-key and would have no multivalued dependencies, it would automatically fall through the normal form rules to 5NF.

To make the problems with this table under the preceding rule clearer, consider the instance of the relation in Figure 7.8.

image

FIGURE 7.8 A relation in 4NF but not 5NF.

Because this table has no multivalued dependencies, it is automatically in fourth normal form. However, there is a great deal of unnecessary duplicated data in this relation—the item numbers are repeated for every customer that orders a given series. The series name is also repeated for every item in the series and for every customer ordering that series. This relation will therefore be prone to modification anomalies.

There is also a more subtle issue: under the rules of this relation, if customer 2180 orders the first Harry Potter movie and indicates that he or she would like more movies in the series, then the only way to put that choice in the table is to add rows for all five Harry Potter movies. You may be forced to add rows that you don’t want to add and introduce data that are not accurate.

Note: There is no official term for the preceding anomaly. It is precisely the opposite of the insertion anomalies described earlier in this chapter, although it does involve a problem with inserting data.

By the same token, if a customer doesn’t want one item in a series, then you must remove all the rows for that customer for that series from the table. If the customer still wants the remaining items in the series, then you have a deletion anomaly.

As you might guess, you can solve the problem by breaking the table into two smaller tables, eliminating the unnecessary duplicated data, and the insertion and deletion anomalies:

image

The official definition for 5NF is as follows:

▪ The relation is in fourth normal form.

▪ All join dependencies are implied by the candidate keys.

A join dependency occurs when a table can be put together correctly by joining two or more tables, all of which contain only attributes from the original table. The original selections relation does have a join dependency, because it can be created by joining the series subscription and series content relations. The join is valid only because of the rule that requires a customer to order all items in a series.

A join dependency is implied by candidate keys when all possible projections from the original relation that form a join dependency each contain a candidate key for the original relation. For example, the following projections can be made from the selections relation:

image

We can regenerate the selections relation by combining any two of the preceding relations. Therefore, the join dependencies are A + B, A + C, B + C, and A + B + C. Like other relational algebra operations, the join theoretically removes duplicate rows, so although the raw result of the join contains extra rows, they will be removed from the result, producing the original table.

Note: One of the problems with 5NF is that as the number of columns in a table increases, the number of possible projections increases exponentially. It can therefore be very difficult to determine 5NF for a large relation.

However, each of the projections does not contain a candidate key for the selections relation. All three columns from the original relation are required for a candidate key. Therefore, the relation is not in 5NF. When we break down the selections relation into series\_selections and series\_content, we eliminate the join dependencies, ensuring that the relations are in 5NF.

Sixth Normal Form

Normalization theory has been very stable for more than 45 years. However, in the late 1990s, C. J. Date, one of the foremost experts in database theory, proposed a sixth normal form, particularly to handle situations in which there is temporal data. This is technically not a project-join normal form, as were all of those discussed earlier in this chapter.

Consider the following relation:

image

The intent of this relation is to maintain a history of a customer’s locations and when they were valid (starting date to ending date). Depending on the circumstances, there may be a great deal of duplicated data in this relation (for example, if only the phone number changed) or very little (for example, if there is a move to a new state with a new phone number). Nonetheless, there is only one functional dependency in the relation:

image

There are no transitive dependencies, no overlapping candidate keys, no multivalued dependencies, and all join dependencies are implied by the candidate key(s). The relation is therefore in fifth normal form.

Sixth normal form was created to handle the situation where temporal data vary independently to avoid unnecessary duplication. The result is tables that cannot be decomposed any further; in most cases, the tables include the primary key and a single non-key attribute. The sixth normal form tables for the sample customer relation would be as follows:

image

The resulting tables eliminate the possibility of redundant data, but introduce some time consuming joins to find a customer’s current address or to assemble a history for a customer. For this reason alone, you may decide that it makes sense to leave the relation in 5NF and not decompose it further.

Going to sixth normal form may also introduce the need for a circular inclusion constraint. There is little point in including a street address for a customer unless a city, state, and zip code exist for the same date interval. The circular inclusion constraint would therefore require that if a row for any given interval and any given customer ID exists in any of street\_addresses, cities, states, or zip\_codes, matching rows must exist in all of those tables. Today’s relational DBMSs do not support circular inclusion constraints nor are they included in the current SQL standard. If such a constraint is necessary, it will need to be enforced through application code.

For Further Reading

There are many books available that deal with the theory of relational databases. You can find useful supplementary information in the following:

Date CJ. Database Design and Relational Theory: Formal Forms and All That Jazz. O’Reilly Media; 2012.

Date, C. J. On DN/NF Normal Form. http://www.dbdebunk.com/page/page/621935.htm

Date CJ, Darwen H, Lorentzos N. Temporal Data and the Relational Model. Morgan Kaufmann; 2002.

Earp R. Database Design Using Entity-Relationship Diagrams. Taylor & Francis; 2007.

Halpin T, Morgan T. Information Modeling and Relational Databases. third ed. Morgan Kaufman; 2008.

Hillyer, M. An Introduction to Database Normalization. http://dev.mysql.com/tech-resources/articles/intro-to-normalization.html

Olivé A. Conceptual Modeling of Information Systems. Springer; 2007.

Pratt PJ, Adamski JJ. Concepts of Database Management. sixth ed. Course Technology; 2007.

Ritchie C. Database Principles and Design. third ed. Cengage Learning Business Press; 2008.

CopyAdd HighlightAdd Note

Support Sign Out

© 2020 O'Reilly Media, Inc. Terms of Service / Privacy Policy