CHAPTER 10

DATA NORMALIZATION METHOD

CHAPTER OBJECTIVES

- Understand how the normalization method creates the relational data model
- Examine this informal approach and its potential pitfalls
- Note the significance of the normalization approach
- Study the systematic steps that lead to a normalized relational data model
- Learn the fundamental normal forms in depth
- Review the higher normal forms

As you studied the model transformation method in Chapter 9, you might have had thoughts on the necessity of that method. You might have wondered why you need to create a semantic data model first and then bother to transform that model into a relational data model. If you already know that your target database system is going to be a relational database system, why not create a relational data model itself from the information requirements? These are valid thoughts. Even though you learned the merits of the model transformation method in Chapter 9, is it not a longer route for logical design?

In this chapter, we will pursue these thoughts. We will attempt to put together a relational data model from the information requirements. We will see what happens and whether the resultant model readily becomes a relational data model. If not, we will explore what should be done to make the initial outcome of this method become a good relational model.

Database Design and Development: An Essential Guide for IT Professionals by Paulraj Ponniah
ISBN 0-471-21877-4 Copyright © 2003 by John Wiley and Sons, Inc.

INFORMAL DESIGN

In a sense, this attempt to create a relational model seems to be an informal design technique. Creating a semantic data model first is a rigorous and systematic approach. On the other hand, if you want to create relational tables straight away, you seem to bypass standard and proven techniques. Therefore, first try to understand exactly what we mean by an informal design method.

Let us describe this method first. Then let us review steps that can formalize this methodology. Although the goal is to come up with relational tables in the initial attempt, you will note that the initial attempt does not always produce a good relational data model. Therefore, we need specific steps to make the initial data model a true relational model.

As you know very well, a relational model consists of relations or two-dimensional tables with columns and rows. Because our desired outcome is a true relational data model, let us quickly review its fundamental properties:

- Relationships are established through foreign keys.
- Each row in a relation is unique.
- Each attribute value in each row is atomic or single-valued.
- The order of the columns in a relation is immaterial.
- The sequence of the rows in a relation is immaterial.
- Relations must conform to entity integrity and referential integrity rules.
- Each relation must conform to the functional dependency rule.

Forming Relations from Requirements

The attempt in this method is simply to come up with relations or tables from the information requirements. Figure 10-1 explains this seemingly informal approach in a simple manner.

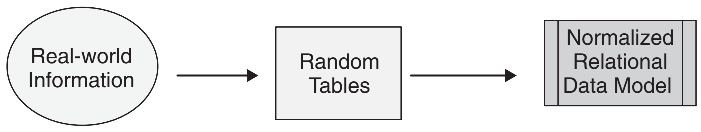
Note the objectives of the method. When you create a relational data model with this method, you must come up with tables that conform to the relational rules and possess the correct properties of a relational data model.

What are the steps in the creation of a proper relational model? Create an initial data model by putting together a set of initial tables. Examine this initial set of tables and then apply procedures to make this initial set into a proper set of relational tables. As you will learn in the later sections, this application of procedures to rectify problems found in the initial set of tables is known as normalization.

Of course, an obvious question is why you should go through normalization procedures. Are you not able to produce a proper set of relational tables from the information requirements in the initial attempt itself? Let us explore the reasons.

Pitfalls of Informal Design

Let us consider a very simple set of information requirements. Using these information requirements, we will attempt to create an initial relational data model and then examine the model. Creating an initial relational data model by this approach simply means coming up with an initial set of relational tables. Study the following



Objectives:

- Create well-structured relations
- Avoid data redundancies
- Ensure that the data model conforms to relational rules
- Ensure data manipulation will not have problems or anomalies

Steps:

- Create random tables or relations
- Normalize the tables

Figure 10-1 Informal design of relational data model.

statement of information requirements for which you need to create a relational data model:

Assignment of employees to projects

Employees work in departments. Information about the employees such as name, salary, position, and bonus amount must be represented in the data model. The model should include names of the departments and their managers. Project numbers and project descriptions are available. The model must represent the start date, end date, and hours worked on a project for each employee. New employees are not assigned to a project before they finish training.

Examine the information requirements. Clearly, your data model must represent information about the employees and their project assignments. Also, some information about the departments must be included. Compared to other real-world information requirements, the information about employee-project assignments being modeled here is very simple. With this set of information requirements, you need to come up with two-dimensional tables. Let us say that you are able to put the data in the form of tables and also express the relationships within the tables. If you are able to do this, then you are proceeding toward creating the relational data model.

Looking at the simplicity of the information requirements, it appears that all of the data can be put in just one table. Let us create that single table and inspect the data content. Figure 10-2 represents this single table showing sample data values.

Inspect the PROJECT-ASSIGNMENT table carefully. To uniquely identify each row, you have to assign EmpId and ProjNo together as the primary key. At first glance, you note that the table contains all the necessary data to completely represent the data content of the information requirements. The table contains columns and rows. Review each column. It represents an attribute, and the column name represents the name of the attribute. Now look at the rows. Each row represents one employee, a single instance of the entity represented by the table. So far, the table looks like it qualifies to be part of a relational data model.

PROJECT-ASSIGNMENT

<u>EmpId</u>	Name	Salary	Position	Bonus	DptNo	DeptName	Manager	<u>ProjNo</u>	ProjDesc	ChrgCD	Start	End	Hrs
100	Simpson	35000	Analyst	5000	3	Design	Ross	23	DB design	D100	Apr-02	Jul-02	200
140	Beeton	28000	Technician	3000	2	Operations	Martin	14	Network cabling	N140	Sep-02	Oct-02	120
160	Davis	30000	Technician	3000	4	Tech Suprt	Lucero	14	Network cabling	S160	Sep-02	Nov-02	150
								36	Network testing	S160	Nov-02	Dec-02	100
190	Berger	45000	DBA	6000	1	DB Suprt	Rawlins	45	Physical design	D190	Aug-02	Nov-02	300
								48	Space allocation	S190	Nov-02	Dec-02	80
100	Simpson	35000	Analyst	5000	3	Design	Ross	25	Reports	C100	Oct-02	Nov-02	100
110	Covino	34000	Analyst	5000	5	Analysis	Williams	31	Forms	D110	Mar-02	May-02	120
								25	Reports	D110	May-02	Jul-02	150
120	Brown	35000	Analyst	5000	5	Analysis	Williams	11	Order entry	D120	Jul-02	Sep-02	300
180	Smith	30000	Programmer	4000	6	Programming	Goldner	31	Forms	C180	Sep-02	Nov-02	250
								25	Reports	C180	Nov-02	Dec-02	200
200	Rogers	32000	Programmer	4000	6	Programming	Goldner	11	Order entry	D200	Sep-02	Oct-02	200
								12	Inventory Control	P200	Oct-02	Dec-02	200
								13	Invoicing	P200	Nov-02	Dec-02	100
100	Simpson	35000	Analyst	5000	3	Design	Ross	31	Forms	D100	Aug-02	Oct-02	150
130	Clemens	38000	Analyst	5000	3	Design	Ross	23	DB design	D130	Apr-02	Jun-02	200

Figure 10-2 Table created from information requirements.

Before proceeding further, let us have a brief explanation of the column named ChrgCD. When an employee is assigned to a project, a charge code is given for that assignment. The charge code depends on the type of work done by the employee in that assignment irrespective of his or her position or title. For example, when Simpson, an analyst, does design work in a project, a charge code of D100 is given for that assignment; when he does coding work in another project, a charge code of C100 is given for this assignment. Charge codes indicate the type of work done by an employee in the various projects.

Next, observe the projects for Davis, Berger, Covino, Smith, and Rogers. Each of these employees has been assigned to multiple projects. The resulting relational database must contain information about these multiple assignments. However, looking at the rows for these employees, these rows contain multiple values for some attributes. In other words, not all columns contain atomic or single-valued attribute values. This is a violation of the attribute atomicity requirement in a relational data model. Therefore, the random PROJECT-ASSIGNMENT table we created quickly cannot be part of true relational data model.

Let us now examine the table further and see how it will hold up when we try to manipulate the data contents. As indicated in Figure 10-1, a proper relational data model must avoid data redundancies and also ensure that data manipulation will not cause problems. When we attempt to use the data model for data manipulation, you will find that we run into three types of problems or anomalies as noted below:

Update anomaly. Occurs while updating values of attributes in the database.

Deletion anomaly. Occurs while deleting rows from a relation.

Addition anomaly. Occurs while adding (inserting) new rows in a relation.

We will discuss these anomalies in the next subsections. Try to understand the nature of these problems and how our PROJECT-ASSIGNMENT table has such problems and therefore cannot be correct. Unless we remove these anomalies, our table cannot be part of a true relational model.

Update Anomaly

If a relational two-dimensional table does not conform to relational rules, you find that problems arise when you try to do updates to data in a database based on such a table. Our data model at this point consists of the randomly created PROJECT-ASSIGNMENT table. Let us try to do an update to the data in the PROJECT-ASSIGNMENT table and see what happens.

After the database is populated, users find that the name "Simpson" is recorded incorrectly and that it should be changed to the correct name "Samson." How is the correction accomplished? The correction will have to be made wherever the name "Simpson" exists in the database. Now look at the example of data content shown in Figure 10-2.

Even in this extremely limited set of rows in the table, you have to make the correction in three rows. Imagine a database of 500 or 5000 employees. Even this is not a large database. It is not unusual to store data about many thousands of employees in a typical database. Now go back to the correction. In a large database covering a large number of rows for employees, the number of rows for PROJECT-ASSIGNMENT is expected to be high. Therefore, it is very likely that when you make the correction to the name, you will miss some rows that need to be changed. So, what is the effect of the update anomaly in this case?

Update anomaly

Results in data inconsistency because of possible partial update instead of the proper complete update.

Deletion Anomaly

Again, if the relational two-dimensional table does not conform to relational rules, you find that problems arise when you try to delete rows from a database based on such a table. Let us try to delete some data from the PROJECT-ASSIGNMENT table and see what happens.

Here is the situation. Employee Beeton leaves your organization. Therefore, it is no longer necessary keep any information about Beeton in your database. You are authorized to delete all data about Beeton from the database. Now inspect the sample database contents shown in Figure 10-2.

How is the deletion of data about Beeton carried out? Luckily, you have to delete just one row, namely, the second row in the PROJECT-ASSIGNMENT table, to get rid of all data about Beeton in the database. Now, consider another aspect of this operation. What happens when you delete this row? Data such as Beeton's EmpId, Name, Salary, Position, and his project assignment are deleted. This is fine, because this is what you intended to do.

Now examine the row as shown in the figure. When you delete this row, you not only remove data about Beeton, you also delete data about Department 2. And looking at the entire contents of the table, you notice that this is only row that has information about Department 2. By deleting this row, you also delete all data about Department 2 from the database. However, this is not your intention. Data about Department 2 have to be preserved in the database for possible future uses. But if you delete the second row, data about Department 2 are also (unintentionally) lost. Let us express the effect of deletion anomaly.

Deletion anomaly

Results in unintended loss of data because of possible deletion of data other than what must be deleted.

Addition Anomaly

We have considered the effects of updates and deletions in a two-dimensional table that is put together in a random fashion from information requirements. You have noted that these operations cause anomalies or problems. Now let us try to perform one more common data operation on this table. Try to add new data to the database.

This is the situation. A new employee, Potter, has joined your organization. As usual, the human resources department has already assigned a unique EmpId to Potter. So you need to add data about Potter to the database. However, Potter is still in training and therefore is not assigned to a project yet. You have data about Potter such as his salary, bonus, and the department in which is hired. You can add all of these data to the database.

Begin to create a row for Potter in the database. You are ready to create a row in our PROJECT-ASSIGNMENT table for Potter. You can enter the name, department, and so on. But what about the unique primary key for this row? As you know, the primary key for this table consists of EmpId and ProjNo together. However, you are unable to assign a value for ProjNo for this row because he is not assigned to a project yet. So you can have a null value for ProjNo until Potter is assigned to a project. But can you really do this? If you place a null value in the ProjNo column, you will be violating the entity integrity rule that states no part of the primary key may be null. You are faced with a problem—an anomaly concerning added new data. Data about Potter cannot be added to the database until he is assigned to a project. Even though he is already an employee, data about Potter will be missing in the database. This is the effect of addition anomaly.

Addition anomaly

Results in inability to add data to the database because of the absence of some data presently unavailable.

NORMALIZATION APPROACH

Let us review our discussion so far. We inspected the information requirements about employees, departments, projects, and project assignments. Our intention was to create a relational data model directly from the study of the information requirements. This meant creating a data model consisting of two-dimensional tables or relations that normally make up a relational data model. Because of the simplicity of the information requirements, we were able to represent all the data in a single random table. So far, this is the relational data model for us. If it has to be a good relational model, it must conform to relational rules.

You have observed that the random table PROJECT-ASSIGNMENT violates some relational rules at the outset. More importantly, when you attempt to update data, delete data, or add data, our initial data model has serious problems. You have noted the problems of update, deletion, and addition anomalies. So what is the next step? Do you simply abandon the initial data model and look for other methods? Your goal is to create a good relational model even while you attempt to do this directly from information requirements.

It turns out that by adopting a systematic methodology you can, indeed, regularize the initial data model created by the first attempt. This methodology is based on Dr. Codd's approach to normalizing the initial tables created in a random manner directly from information requirements. Before getting into the actual methodology of normalization, let us consider its merits and note how the methodology is used.

Purpose and Merits

Normalization methodology resolves the three types of anomalies encountered when data manipulation operations are performed on a database based on an improper relational data model. Therefore, after applying the principles of normalization to the initial data model, the three types of anomalies will be eliminated. The normalization process standardizes or "normalizes" the table structures. You come up with revised table structures. It is a systematic, step-by-step methodology.

Normalization

- Creates well-structured relations
- Removes data redundancies
- Ensures that the initial data model is properly transformed into a relational data model conforming to relational rules
- Guarantees that data manipulation will not result in anomalies or other problems

How to Apply This Method

As mentioned above, this normalization process is a step-by-step approach. It is not completed in one large task. The process breaks down the problem and applies remedies, one at a time. The initial data model is refined and standardized in a clear and systematic manner, one step at a time.

At each step, the methodology consists of examining the data model, removing one type of problem, and changing it to a better normal form. You take the initial data model created directly from information requirements in a random fashion. This initial model, at best, consists of two-dimensional tables representing the entire data content, nothing more or less. As we have seen, such an initial data model is subject to data manipulation problems.

You apply the principles of the first step. In this step, you are examining the initial data model for only one type of nonconformance and seeking to remove one type of irregularity. Once this one type of irregularity is resolved, your data model becomes better and is rendered into a first normal form of table structures. Then you look for another type of irregularity in the second step and remove this type of irregularity from the data model resulting from the step. After this next step, your data model becomes still better and becomes a data model in the second normal form. The process continues through a reasonable number of steps until the resulting data model becomes truly relational.

Steps and Tasks

The first few steps of the normalization methodology transform the initial data model into a workable relational data model that is free from the common types of irregularities. These first steps produce the normal forms of relations that are fundamental to creating a good relational data model. After these initial steps, in some cases, further irregularities still exist. When you remove the additional irregularities, the resulting relations become higher normal form relations.

We will discuss the fundamental and higher normal forms in the next section. However, let us just list the steps here:

Fundamental normal forms

Step 1

Examine initial data model and remove first type of irregularity.

Data model becomes a set of first normal form relations.

Step 2

Inspect data model resulting from previous step and remove second type of irregularity.

Data model becomes a set of second normal form relations.

Step 3

Inspect data model resulting from previous step and remove third type of irregularity.

Data model becomes a set of third normal form relations.

Step 4

Inspect resulting data model and remove another type of irregularity.

Data model becomes a set of Boyce-Codd normal form relations.

Higher normal forms

Step 1

Examine data model in the final fundamental normal form and remove fourth type of irregularity.

Data model becomes a set of fourth normal form relations.

Step 2

Inspect data model resulting from previous step and remove fifth type of irregularity.

Data model becomes a set of fifth normal form relations.

Step 3

Inspect data model resulting from previous step and remove domain-key irregularity.

Data model becomes a set of domain-key normal form relations.

In practice, only a few initial data models need to go through all the above steps. Generally, a set of third normal form relations will form a good relational data model. You may want to go one step further to make the model a set of Boyce-Codd normal form relations. Only very infrequently would you need to go to higher normal forms.

FUNDAMENTAL NORMAL FORMS

As explained above, normalization is a process of rectifying potential problems in two-dimensional tables created at random. This process is a step-by-step method, each step addressing and remedying one specific type of potential problem. As we proceed with the normalization process, you will clearly understand why this step-by-step approach works so well. By taking a step-by-step approach, you will not overlook any type of anomaly. And, when the process is completed, you will have resolved every type of potential problem.

In the last section, we have noted the four steps that make up the portion of the normalization process transforming the initial data model into the fundamental normal forms. After the third step, the initial data model becomes a third normal form relational data model. As already mentioned, for most practical purposes, a third normal form data model is an adequate relational data model. You need not go further. Occasionally, you may have to proceed to the fourth step and refine the data model further and make it a set of Boyce-Codd normal form relations.

First Normal Form

Refer back to Figure 10-2 showing the PROJECT-ASSIGNMENT relation created as the initial data model. You have already observed that the rows for Davis, Berger, Covino, Smith, and Rogers contain multiple values for attributes in six different columns. You know that this violates the requirement in a relational model that states each row must have atomic values for each of the attributes.

This step in normalization process addresses the problem of repeating groups of attribute values for single rows. If a relation has such repeating groups, we say that the relation is not in the first normal form. The objective of this step is to transform the data model into a model in the first normal form. Here is what must be done to make this transformation.

Transformation to First Normal Form (1NF)

Remove repeating groups of attributes and create rows without repeating groups.

Figure 10-3 shows the result of the transformation to first normal form.

Carefully inspect the PROJECT-ASSIGNMENT relation shown in the figure. Each row has a set of unique values in the columns. The composite primary key consisting of EmpId and ProjNo uniquely identifies each row. No single row has multiple values for any of its attributes. This step has rectified the problem of multiple values for the same attribute in a single row.

PROJECT-ASSIGNMENT

EmpId	Name	Salary	Position	Bonus	DptNo	DeptName	Manager	ProjNo	ProjDesc	ChrgCD	Start	End	Hrs
100	Simpson	35000	Analyst	5000	3	Design	Ross	23	DB design	D100	Apr-02	Jul-02	200
140	Beeton	28000	Technician	3000	2	Operations	Martin	14	Network cabling	N140	Sep-02	Oct-02	120
160	Davis	30000	Technician	3000	4	Tech Suprt	Lucero	14	Network cabling	S160	Sep-02	Nov-02	150
160	Davis	30000	Technician	3000	4	Tech Suprt	Lucero	36	Network testing	S160	Nov-02	Dec-02	100
190	Berger	45000	DBA	6000	1	DB Suprt	Rawlins	45	Physical design	D190	Aug-02	Nov-02	300
190	Berger	45000	DBA	6000	1	DB Suprt	Rawlins	48	Space allocation	S190	Nov-02	Dec-02	80
100	Simpson	35000	Analyst	5000	3	Design	Ross	25	Reports	C100	Oct-02	Nov-02	100
110	Covino	34000	Analyst	5000	5	Analysis	Williams	31	Forms	D110	Mar-02	May-02	120
110	Covino	34000	Analyst	5000	5	Analysis	Williams	25	Reports	D110	May-02	Jul-02	150
120	Brown	35000	Analyst	5000	5	Analysis	Williams	11	Order entry	D120	Jul-02	Sep-02	300
180	Smith	30000	Programmer	4000	6	Programming	Goldner	31	Forms	C180	Sep-02	Nov-02	250
180	Smith	30000	Programmer	4000	6	Programming	Goldner	25	Reports	C180	Nov-02	Dec-02	200
200	Rogers	32000	Programmer	4000	6	Programming	Goldner	11	Order entry	D200	Sep-02	Oct-02	200
200	Rogers	32000	Programmer	4000	6	Programming	Goldner	12	Inventory Control	P200	Oct-02	Dec-02	200
200	Rogers	32000	Programmer	4000	6	Programming	Goldner	13	Invoicing	P200	Nov-02	Dec-02	100
100	Simpson	35000	Analyst	5000	3	Design	Ross	31	Forms	D100	Aug-02	Oct-02	150
130	Clemens	38000	Analyst	5000	3	Design	Ross	23	DB design	D130	Apr-02	Jun-02	200

Figure 10-3 Data model in first normal form.

Let us examine whether the transformation step has rectified the other types of update, deletion, and addition anomalies encountered before the model was transformed into first normal form. Compare the PROJECT-ASSIGNMENT relation shown in Figure 10-3 to the earlier version in Figure 10-2. Apply the tests to the transformed version of the relation contained in Figure 10-3.

Update: Correction of name "Simpson" to "Samson"

The correction has to be made in multiple rows. Update anomaly still persists.

Deletion: Deletion of data about Beeton

This deletion will unintentionally delete data about Department 2. Deletion anomaly still persists.

Addition: Addition of data about new employee Potter

Cannot add new employee Potter to the database until he is assigned to a project. Addition anomaly still persists.

So, you note that, although this step has resolved the problem of multivalued attributes, data manipulation problems still remain. Nevertheless, this step has removed a major deficiency from the initial data model. We have to proceed to the next steps and examine the effect of data manipulation operations.

Second Normal Form

Recall the discussion on functional dependencies in Chapter 8 covering the properties and rules of the relational data model. If the value of one attribute determines the value of a second attribute in a relation, we say that the second attribute is functionally dependent on the first attribute. The discussion on functional dependencies in Chapter 8 concluded with a functional dependency rule. Let us repeat the functional dependency rule:

Each data item in a tuple of a relation is uniquely and functionally determined by the primary key, by the whole primary key, and only by the primary key.

Examine the dependencies of data items in the PROJECT-ASSIGNMENT relation in Figure 10-3. You know that this relation is in the first normal form, having gone through the process of removal of repeating groups of attributes. Let us inspect the dependency of each attribute on the whole primary key consisting of EmpId and ProjNo. Only the following attributes depend on the whole primary key: ChrgCD, Start, End, and Hrs. The remaining nonkey attributes do not appear to be functionally dependent on the whole primary key. They seem to functionally depend on one or another part of the primary key.

This step in the normalization process deals specifically with this type of problem. Once this type of problem is resolved, the data model becomes transformed to a data model in the second normal form. In other words, the condition for a second normal form data model is as follows:

If a data model is in the second normal form, no nonkey attributes may be dependent on part of the primary key.

Therefore, if there are partial key dependencies in a data model, this step resolves this type of dependencies. Here is what must be done to make this transformation.

Transformation to Second Normal Form (2NF)

Remove partial key dependencies.

If you look at the other attributes in the PROJECT-ASSIGNMENT relation in Figure 10-3, you will note that the following attributes depend on just EmpId, a part of the primary key: Name, Salary, Position, Bonus, DptNo, DeptName, and Manager. The attribute ProjDesc depends on ProjNo, another part of the primary key. These are partial key dependencies. This step resolves partial key dependencies. Now look at Figure 10-4 that shows the resolution of partial key dependencies. The relations shown in this figure are in the second normal form.

Note how the resolution is done. The original relation has been decomposed into three separate relations. In each relation, to make sure that each row is unique, duplicate rows are eliminated. For example, multiple duplicate rows for employee Simpson have been replaced by a single row in the EMPLOYEE relation.

Decomposition is an underlying technique for normalization. If you go carefully through each of the three relations, you will be satisfied that none of these has any partial key dependencies. So this step has rectified the problem of partial key dependencies. But what about the types of anomalies encountered during data manipulation?

Let us examine whether the transformation step has rectified the types of update, deletion, and addition anomalies encountered before the model was transformed

EMPLOYEE-PROJECT

EmpId	ProjNo	ChrgCD	Start	End	Hrs
100	23	D100	Apr-02	Jul-02	200
140	14	N140	Sep-02	Oct-02	120
160	14	S160	Sep-02	Nov-02	150
160	36	S160	Nov-02	Dec-02	100
190	45	D190	Aug-02	Nov-02	300
190	48	S190	Nov-02	Dec-02	80
100	25	C100	Oct-02	Nov-02	100
110	31	D110	Mar-02	May-02	120
110	25	D110	May-02	Jul-02	150
120	11	D120	Jul-02	Sep-02	300
180	31	C180	Sep-02	Nov-02	250
180	25	C180	Nov-02	Dec-02	200
200	11	D200	Sep-02	Oct-02	200
200	12	P200	Nov-02	Dec-02	200
200	13	P200	Nov-02	Dec-02	100
100	31	D100	Aug-02	Oct-02	150
130	23	D130	Apr-02	Jun-02	200

PROJECT

ProjNo	ProjDesc
23	DB design
14	Network cabling
36	Network testing
45	Physical design
48	Space allocation
25	Reports
31	Forms
11	Order entry
12	Inventory Control
13	Invoicing

EMPLOYEE

EmpId	Name	Salary	Position	Bonus	DptNo	DeptName	Manager
100	Simpson	35000	Analyst	5000	3	Design	Ross
140	Beeton	28000	Technician	3000	2	Operations	Martin
160	Davis	30000	Technician	3000	4	Tech Suprt	Lucero
190	Berger	45000	DBA	6000	1	DB Suprt	Rawlins
110	Covino	34000	Analyst	5000	5	Analysis	Williams
120	Brown	35000	Analyst	5000	5	Analysis	Williams
180	Smith	30000	Programmer	4000	6	Programming	Goldner
200	Rogers	32000	Programmer	4000	6	Programming	Goldner
130	Clemens	38000	Analyst	5000	3	Design	Ross

Figure 10-4 Data model in second normal form.

into the first normal form. Compare the relations shown in Figure 10-4 to the previous version in Figure 10-3. Apply the tests to the transformed version of the relations contained in Figure 10-4.

Update: Correction of name "Simpson" to "Samson"

The correction has to be made only in one row in EMPLOYEE relation. The update anomaly has disappeared.

Deletion: Deletion of data about Beeton

This deletion will unintentionally delete data about Department 2. Deletion anomaly still persists.

Addition: Addition of data about new employee Potter

You can now add new employee Potter to the database in EMPLOYEE relation. Addition anomaly has disappeared.

So you note that, although this step has resolved the problem of partial key dependencies, still some data manipulation problems remain. Nevertheless, this step has removed a major deficiency from the data model. We have to proceed to the next steps and examine the effect of data manipulation operations.

Third Normal Form

After transformation to the second normal form, you note that a particular type of functional dependencies is removed from the preliminary data model and that the data model is closer to becoming a correct and true relational data model. In the previous step, we removed partial key dependencies. Let us examine the resulting data model to see whether any more irregular functional dependencies still exist. Remember the goal is to make each relation in the data model into a form where each data item in a tuple is functionally dependent only on the full primary key and nothing but the full primary key.

Refer to the three relations shown in Figure 10-4. Let us inspect these relations one by one. The attribute ProjDesc functionally depends on the primary key ProjNo. So this relation PROJECT is correct. Next look at the relation EMPLOYEE-PROJECT. In this relation, each of the attributes ChrgCD, Start, End, and Hrs depends on full primary key EmpId, ProjNo.

Now examine the relation EMPLOYEE carefully. What about the attributes Position and Bonus? Bonus depends on the position. Bonus for an Analyst is different from that for a Technician. Therefore, in that relation, the attribute Bonus is functionally dependent on another attribute Position, not on the primary key. Look further. How about the attributes DeptName and Manager? Do they depend on the primary key EmpId? Not really. These two attributes functionally depend on another attribute in the relation, namely, DptNo.

So what is the conclusion from your observation? In the relation EMPLOYEE, only the two attributes Name and Salary depend on the primary key EmpId. The other attributes do not depend on the primary key. Bonus depends on Position; DeptName and Manager depend on DptNo.

This step in normalization process deals with this type of problem. Once this type of problem is resolved, the data model is transformed to a data model in the third normal form. In other words, the condition for a third normal form data model is as follows:

> If a data model is in the third normal form, no nonkey attributes may be dependent on another nonkey attribute.

In the relation EMPLOYEE, dependency of the attribute DeptName on the primary key EmpId is not direct. The dependency is passed over to the primary key through another nonkey attribute DptNo. This passing over of the dependency means that the dependency on the primary key is a transitive dependency—passed over through another nonkey attribute, DptNo. Therefore, this type of problematic dependency is also called a transitive dependency in a relation. If there are transitive dependencies in a data model, this step resolves this type of dependencies. Here is what must be done to make this transformation.

Transformation to Third Normal Form (3NF)

Remove transitive dependencies.

Figure 10-5 shows the resolution of transitive dependencies. The relations shown in the figure are all in the third normal form.

Note how the resolution is done. The EMPLOYEE relation is further decomposed into two additional relations, POSITION and DEPARTMENT. In each relation, to ensure that each row is unique, duplicate rows are eliminated. For example, multiple duplicate rows for the position Analyst in the EMPLOYEE relation have been replaced by a single row in the POSITION relation.

EMPLOYEE

EmpId	Name	Salary	PosnId	DptNo
100	Simpson	35000	ANLY	3
140	Beeton	28000	TECH	2
160	Davis	30000	TECH	4
190	Berger	45000	DBAM	1
110	Covino	34000	ANLY	5
120	Brown	35000	ANLY	5
180	Smith	30000	PGMR	6
200	Rogers	32000	PGMR	6
130	Clemens	38000	ANLY	3

EMPLOYEE-PROJECT

EmpId	ProjNo	ChrgCD	Start	End	Hrs
100	23	D100	Apr-02	Jul-02	200
140	14	N140	Sep-02	Oct-02	120
160	14	S160	Sep-02	Nov-02	150
160	36	S160	Nov-02	Dec-02	100
190	45	D190	Aug-02	Nov-02	300
190	48	S190	Nov-02	Dec-02	80
100	25	C100	Oct-02	Nov-02	100
110	31	D110	Mar-02	May-02	120
110	25	D110	May-02	Jul-02	150
120	11	D120	Jul-02	Sep-02	300
180	31	C180	Sep-02	Nov-02	250
180	25	C180	Nov-02	Dec-02	200
200	11	D200	Sep-02	Oct-02	200
200	12	P200	Nov-02	Dec-02	200
200	13	P200	Nov-02	Dec-02	100
100	31	D100	Aug-02	Oct-02	150
130	23	D130	Apr-02	Jun-02	200

PROJECT

ProjNo	ProjDesc
23	DB design
14	Network cabling
36	Network testing
45	Physical design
48	Space allocation
25	Reports
31	Forms
11	Order entry
12	Inventory Control
13	Invoicing

DEPARTMENT

DptNo	DeptName	Manager
3	Design	Ross
2	Operations	Martin
4	Tech Suprt	Lucero
1	DB Suprt	Rawlins
5	Analysis	Williams
6	Programming	Goldner

POSITION

PosnId	Position	Bonus
ANLY	Analyst	5000
TECH	Technician	3000
DBAM	DBA	6000
PGMR	Programmer	4000

Figure 10-5 Data model in third normal form.

Again, you will note that decomposition is a basic technique for normalization. If you carefully go through each of the relations, you will be satisfied that none of these has any transitive dependencies—one nonkey attribute depending on some other nonkey attribute. So this step has rectified the problem of transitive dependencies. But what about the types of anomalies encountered during data manipulation?

Let us examine whether the transformation step has rectified the other types of update, deletion, and addition anomalies encountered before the model was transformed into first normal form. Compare the relations shown in Figure 10-5 to the previous version in Figure 10-4. Apply the tests to the transformed version of the relation contained in Figure 10-5.

Update: Correction of name "Simpson" to "Samson"

The correction has to be made only in one row in EMPLOYEE relation. Update anomaly has disappeared.

Deletion: Deletion of data about Beeton

Removal of Beeton and his assignments from EMPLOYEE and EMPLOYEE-PROJECT relations does not affect the data about Department 2 in DEPARTMENT relation. Deletion anomaly has disappeared from the data model.

Addition: Addition of data about new employee Potter

You can now add new employee Potter to the database in EMPLOYEE relation. Addition anomaly has disappeared.

So you note that this step has resolved the problem of transitive dependencies and the data manipulation problems, at least the ones we have considered. Before we declare that the resultant data model is free from all types of data dependency problems, let us examine the model one more time.

Boyce-Codd Normal Form

Consider the EMPLOYEE-PROJECT relation in Figure 10-5. Think about the ChrgCD attribute. A particular charge code indicates the specific employee's role in an assignment. Also, each project may be associated with several charge codes depending on the employees and their roles in the project. The charge code is not for the project assignment. The attribute ChrgCD does not depend on the full primary key or on a partial primary key. The dependency is the other way around.

In the EMPLOYEE-PROJECT relation, EmpId depends on ChrgCD and not the other way around. Note how this is different from partial key dependency. Here a partial key attribute is dependent on a nonkey attribute. This kind of dependency also violates the functional dependency rule for the relational data model.

This step in normalization process deals with this type of problem. Once this type of problem is resolved, the data model is transformed to a data model in the Boyce-Codd normal form. In other words, the condition for a Boyce-Codd normal form data model is as follows:

> If a data model is in the Boyce-Codd normal form, no partial key attribute may be dependent on another nonkey attribute.

Here is what must be done to make this transformation.

> **Transformation to Third Normal Form (3NF)**
>
> Remove anomalies from dependencies of key components.

Figures 10-6 and 10-7 show the resolution of the remaining dependencies. The relations shown in both figures are all in the Boyce-Codd normal form.

Note how the resolution is done. The EMPLOYEE-PROJECT relation is decomposed into two additional relations, CHRG-EMP and PROJ-CHRG. Note that duplicate rows are eliminated when the additional relations are formed.

Again, note that decomposition is a basic technique for normalization. The final set of relations in Figures 10-6 and 10-7 is free from all types of problems resulting from invalid functional dependencies.

HIGHER NORMAL FORMS

Once you transform an initial data model into a data model conforming to the principles of the fundamental normal forms, most of the discrepancies are removed. For all practical purposes, your resultant data model is a good relational data model. It will satisfy all the primary constraints of a relational data model. The major problems with functional dependencies are resolved.

EMPLOYEE

EmpId	Name	Salary	PosnId	DptNo
100	Simpson	35000	ANLY	3
140	Beeton	28000	TECH	2
160	Davis	30000	TECH	4
190	Berger	45000	DBAM	1
110	Covino	34000	ANLY	5
120	Brown	35000	ANLY	5
180	Smith	30000	PGMR	6
200	Rogers	32000	PGMR	6
130	Clemens	38000	ANLY	3

PROJECT

ProjNo	ProjDesc
23	DB design
14	Network cabling
36	Network testing
45	Physical design
48	Space allocation
25	Reports
31	Forms
11	Order entry
12	Inventory Control
13	Invoicing

DEPARTMENT

DptNo	DeptName	Manager
3	Design	Ross
2	Operations	Martin
4	Tech Suprt	Lucero
1	DB Suprt	Rawlins
5	Analysis	Williams
6	Programming	Goldner

POSITION

PosnId	Position	Bonus
ANLY	Analyst	5000
TECH	Technician	3000
DBAM	DBA	6000
PGMR	Programmer	4000

Figure 10-6 Data model in Boyce-Codd normal form – part 1.

EMPLOYEE-PROJECT

EmpId	ProjNo	Start	End	Hrs
100	23	Apr-02	Jul-02	200
140	14	Sep-02	Oct-02	120
160	14	Sep-02	Nov-02	150
160	36	Nov-02	Dec-02	100
190	45	Aug-02	Nov-02	300
190	48	Nov-02	Dec-02	80
100	25	Oct-02	Nov-02	100
110	31	Mar-02	May-02	120
110	25	May-02	Jul-02	150
120	11	Jul-02	Sep-02	300
180	31	Sep-02	Nov-02	250
180	25	Nov-02	Dec-02	200
200	11	Sep-02	Oct-02	200
200	12	Nov-02	Dec-02	200
200	13	Nov-02	Dec-02	100
100	31	Aug-02	Oct-02	150
130	23	Apr-02	Jun-02	200

CHRG-EMP

ChrgCD	EmpId
C100	100
C180	180
D100	100
D110	110
D120	120
D130	130
D190	190
D200	200
N140	140
P200	200
S160	160
S190	190

PROJ-CHRG

ProjNo	ChrgCD
23	D100
14	N140
14	S160
36	S160
45	D190
48	S190
25	C100
31	D110
25	D110
11	D120
31	C180
25	C180
11	D200
12	P200
13	P200
31	D100
23	D130

Figure 10-7 Data model in Boyce-Codd normal form – part 2.

We want to examine the resultant data model further and check whether any other types of discrepancies are likely to be present. Occasionally, you may have to take additional steps and go to higher normal forms. Let us consider the nature of higher normal forms and study the remedies necessary to reach these higher normal forms.

Initial table

Executive	Department	Committee
Jones	Administration	Planning
	Finance	Technology
	Info. Technology	
Cooper	Marketing	R & D
	Personnel	Recruitment
	Production	

Initial relation

<u>Executive</u>	<u>Department</u>	<u>Committee</u>
Jones	Administration	Planning
Jones	Finance	Planning
Jones	Info. Technology	Planning
Jones	Administration	Technology
Jones	Finance	Technology
Jones	Info. Technology	Technology
Cooper	Marketing	R & D
Cooper	Personnel	R & D
Cooper	Production	R & D
Cooper	Marketing	Recruitment
Cooper	Personnel	Recruitment
Cooper	Production	Recruitment

Figure 10-8 Multivalued dependencies.

Fourth Normal Form

Before we discuss the fourth normal form for a data model, we need to define the concept of multivalued dependencies. Consider the following assumptions about the responsibilities and participation of company executives:

- Each executive may have direct responsibility for several departments.
- Each executive may be a member of several management committees.
- The departments and committees related to a particular executive are independent of each other.

Figure 10-8 contains data in the form of an initial data model to illustrate these assumptions. The first part of the figure shows the basic table and the second part the transformed relation.

Note that for each value of Executive attribute, there are multiple values for Department attribute and multiple values for Committee attribute. Note also that the values of Department attribute for an executive are independent of the values of Committee attribute. This type of dependency is known as multivalued dependency. A multivalued dependency exists in a relation consisting of at least three attributes A, B, and C such that for each value of A, there is a defined set of values for B and another defined set of values for C, and furthermore, the set of values for B is independent of the set of values for C.

Now observe the relation shown in the second part of Figure 10-8. Because the relation indicating the relationship between the attributes just contains the primary key, the relation is even in the Boyce-Codd normal form. However, by going through the rows of this relation, you can easily see that the three types of anomalies—update, deletion, and addition—are present in the relation.

This step in normalization process deals with this type of problem. Once this type of problem is resolved, the data model is transformed to a data model in the fourth normal form. In other words, the condition for a fourth normal form data model is as follows:

RESPONSIBILITY

Executive	Department
Jones	Administration
Jones	Finance
Jones	Info. Technology
Cooper	Marketing
Cooper	Personnel
Cooper	Production

MEMBERSHIP

Executive	Committee
Jones	Planning
Jones	Technology
Cooper	R & D
Cooper	Recruitment

Figure 10-9 Data model in fourth normal form.

If a data model is in the fourth normal form, no multivalued dependencies exist.

Here is what must be done to make this transformation.

Transformation to Fourth Normal Form (4NF)

Remove multivalued dependencies.

Figure 10-9 shows the resolution of the multivalued dependencies. The two relations are in the fourth normal form.

When you examine the two relations in Figure 10-9, you can easily establish that these relations are free from update, deletion, or addition anomalies.

Fifth Normal Form

When you transform a data model into second, third, and Boyce-Codd normal forms, you are able to remove anomalies resulting from functional dependencies. After going through the steps and arriving at a data model in the Boyce-Codd normal form, the data model is free from functional dependencies. When you proceed further and transform the data model into a fourth normal form relation, you are able to remove anomalies resulting from multivalued dependencies.

A further step transforming the data model into fifth normal form removes anomalies arising from what are known as join dependencies. What is the definition of join dependency? Go back and look at the figures showing the steps for the earlier normal forms. In each transformation step, the original relation is decomposed into smaller relations. When you inspect the smaller relations, you note that the original relation may be reconstructed from the decomposed smaller relations. However, if a relation has join dependencies and if we are able to decompose the relation into smaller relations, it will not be possible to put the decomposed relations together and recreate the original relation. The smaller relations cannot be joined together to come up with the original relation. The original relation is important because that relation was obtained directly from information requirements. Therefore, in whatever ways you may decompose the original relation to normalize it, you should be able to go back to the original relation from the decomposed relations.

Figure 10-10 shows a relation that has join dependency. Note the columns in the relation shown in the figure.

This relation describes the materials supplied by suppliers to various buildings that are being constructed. Building B45 gets sheet rock from supplier S67 and

BUILDING-MATERIAL-SUPPLIER

BuildingID	Material	SupplierNo
B45	Sheet Rock	S67
B45	Ceiling Paint	S72
B51	Sheet Rock	S72
B51	Shower Stall	S67
B93	Ceiling Paint	S75

Figure 10-10 Relation with join dependency.

ceiling paint from supplier S72. Suppose you have a constraint that suppliers may supply only certain materials to specific buildings even though a supplier may be able to supply all materials. In this example, supplier S72 can supply sheet rock to building B45, but to this building B45, only supplier S67 is designated to supply sheet rock. This constraint imposes a join dependency on the relation. However, the way the relation is composed, it does support the join dependency constraint. For example, there is no restriction to adding a row (B45, Sheet Rock, S72). Such a row would violate the join constraint and not be a true representation of the information requirements.

This step in normalization process deals with this type of problem. Once this type of problem is resolved, the data model is transformed to a data model in fifth normal form. In other words, the condition for a fifth normal form data model is as follows:

If a data model is in the fifth normal form, no join dependencies exist.

Here is what must be done to make this transformation.

Transformation to Fifth Normal Form (5NF)

Remove join dependencies.

Figure 10-11 shows the resolution of the join dependencies. The three relations are in the fifth normal form.

Note something important in the relations shown in the figure. If you join any two of the three relations, the result will produce incorrect information, not the true real-world information with the join dependency. To arrive at the correct original real-world information with the join dependency constraint, you have to join all three relations.

Domain-Key Normal Form

This normal form is the ultimate goal of good design of a proper relational data model. If a data model is in the domain-key normal form (DKNF), it satisfies the conditions of all the normal forms discussed so far. The objective of DKNF is to make one relation represent just one subject and to have all the business rules expressed in terms of domain constraints and key relationships. In other words, all rules could be expressly defined by the relational rules themselves.

BUILDING-MATERIAL

BuildingID	Material
B45	Sheet Rock
B45	Ceiling Paint
B51	Sheet Rock
B51	Shower Stall
B93	Ceiling Paint

MATERIAL-SUPPLIER

Material	SupplierNo
Sheet Rock	S67
Ceiling Paint	S72
Sheet Rock	S72
Shower Stall	S67
Ceiling Paint	S75

BUILDING-SUPPLIER

BuildingID	SupplierNo
B45	S67
B45	S72
B51	S72
B51	S67
B93	S75

Figure 10-11 Data model in fifth normal form.

EMPLOYEE (EmpID, EmpName, SkillType, TrainerID)

TRAINER (TrainerID, TrainerName, Location, SubjectArea)

Business rule:

An employee can have many trainers, but only a specific trainer for each skill type. A trainer can train only in his or her subject area.

Figure 10-12 Relations not in DKNF.

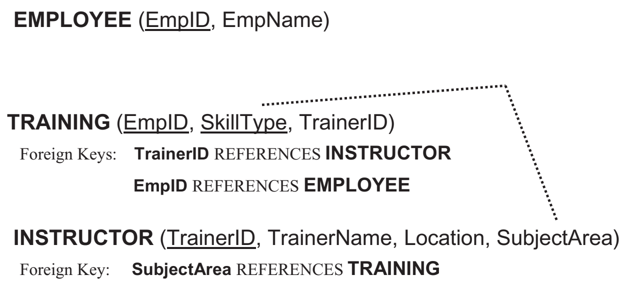
Domain constraints impose rules on the values for attributes—they indicate restrictions on the data values. In DKNF, every other rule must be expressed clearly in terms of keys and relationships without any hidden relationships. Consider the relations shown in Figure 10-12 and also note the accompanying business rule.

How do you know if the relations are in DKNF? You cannot know this until you are aware of the business rule. From the business rule, you understand that an employee can have multiple skill types. Therefore, the primary key EmpId of the EMPLOYEE relation cannot be unique. Furthermore, trainer is related to skill type, and this is a hidden relationship in the relation. There must also be an explicit relationship between skill type and subject area.

Figure 10-13 resolves these discrepancies and expresses the business rule and the relationships correctly. The resultant data model is in DKNF.

NORMALIZATION SUMMARY

Let us go back and review the normalization approach covered so far. Compare this approach with the method of creating a semantic data model first and then transforming the semantic data model into a relational data model. Consider the merits and disadvantages of either method. Also, think about the circumstances and con-



EMPLOYEE (<u>EmpID</u>, EmpName)

TRAINING (<u>EmpID</u>, <u>SkillType</u>, TrainerID)
Foreign Keys: **TrainerID** REFERENCES **INSTRUCTOR**
EmpID REFERENCES **EMPLOYEE**

INSTRUCTOR (<u>TrainerID</u>, TrainerName, Location, SubjectArea)
Foreign Key: **SubjectArea** REFERENCES **TRAINING**

Figure 10-13 Data model in domain-key normal form.

ditions under which one method is preferable to the other. You notice that both methods finally produce a true and correct relational data model. In the final relational data model, every single relation or table represents just one object set or entity type. In each relation, every attribute is functionally dependent on the full primary key and only on the full primary key.

As you know, the data model transformation method is a more straightforward approach. Systematically, you create partial semantic data models by applying standard techniques. You then integrate all the partial data models to produce the consolidated semantic model. After this step, you transform the consolidated semantic model into a final relational data model. Although straightforward, the data model transformation method might take longer to come up with the final relational data model.

On the other hand, the normalization approach starts out with an intuitive initial data model. If you cannot begin with an intuitive initial data model that reflects the real-world information requirements completely, then this method will not work. That is why this normalization approach is difficult when the real-world information requirements are large and complex. If you are able to start with a good initial data model, then it is a matter of rendering the initial data model into a successive series of normal forms. Each step brings you closer to the true relational data model. Observe, however, that each step in the normalization process is defined well. In each step, you know exactly the type of problem you have look for and correct. For example, to refine the data model and make it a first normal form data model, you remove repeating groups of attributes. To refine the data model and make it a second normal data model, you look for partial key dependencies and rectify this problem.

Review of the Steps

When we discussed the normalization steps, we grouped the steps into two major sets. The first set of steps deal with the refinement of the data model into the fundamental normal forms. The second set of steps relate to higher normal forms. As mentioned above, if you complete the first set of steps, then for a vast majority of cases your resulting data model will be truly relational. You need not proceed to the second set of steps to produce higher normal forms.

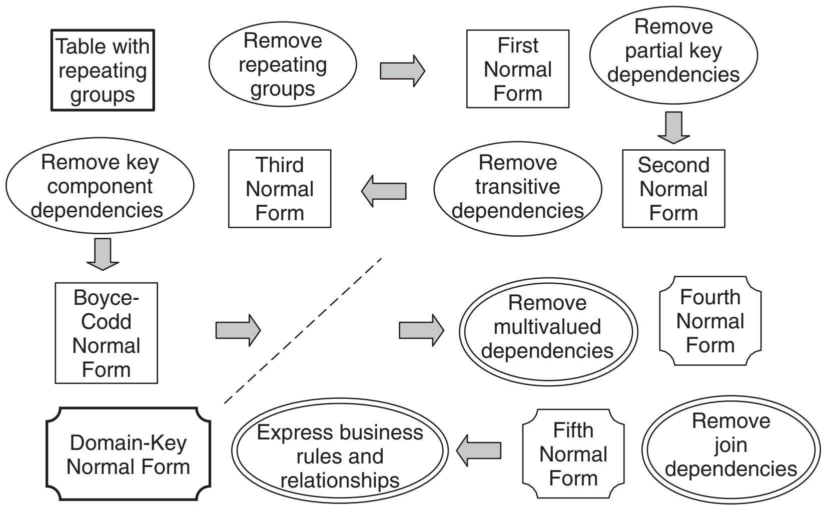


Table with repeating groups
Remove repeating groups
First Normal Form
Remove partial key dependencies
Second Normal Form
Remove transitive dependencies
Third Normal Form
Remove key component dependencies
Boyce-Codd Normal Form
Remove multivalued dependencies
Fourth Normal Form
Remove join dependencies
Fifth Normal Form
Express business rules and relationships
Domain-Key Normal Form

Figure 10-14 Normalization summary.

What exactly do you accomplish in the first set of steps refining the data model into the fundamental normal forms? In the relational data model, for every relation, each attribute must functionally depend only on the full primary key. There should not be any other type of functional dependencies. Update, deletion, and addition anomalies are caused by incorrect functional dependencies within a relation. Once you complete the first set of steps to produce the fundamental normal forms, all problems of invalid functional dependencies are removed.

The second set of normalization steps considers other types of dependencies. Such dependency problems are rare in practice. For that reason, the fundamental normal forms are more important.

Figure 10-14 summarizes the normalization steps. Note the two sets of steps producing the two types of normal forms—fundamental normal forms and higher normal forms. In each step, you tackle one and only one type of problem. The result in each step is a progression toward the final true relational data model.

Critical Issues

Before we end our discussion of the normalization approach, it is worthwhile to list a few critical issues to be concerned about. Give careful consideration to the following points.

- Remember that the normalization method begins with an intuitive data model.
- If the scope of the real-world information is too large or complex and you are not able to create a complete initial data model intuitively, then do not pursue this method.
- Each normalization step focuses on only one type of problem. Make sure you look for the appropriate type of problem in each step.
- The normalization process follows a predefined sequence of steps. Follow the steps in sequence for best results.

- Decomposition of a relation into more than one relation or table is the general technique in each step. Make sure that you do not lose any information when decomposing a table into multiple tables.
- Furthermore, at each step, you must be able to recreate the original set of relations from the set of decomposed relations. This will ensure that you did not lose any information in the normalization step.
- At the end of each step, it is a good practice to verify that the update, deletion, and addition anomalies are eliminated in each of the resulting relations.

Normalization Example

In Chapter 7, we discussed the information requirements for the teaching aspects of a university and developed an entity-relationship diagram as shown in Figure 7-22. The diagram represents the semantic data model. Later, in Chapter 9, we transformed that semantic data model into a relational data model as shown in Figure 9-22. This was the data model transformation method for producing a true relational data model.

Let us take the same example of information requirements for a university and adopt the normalization approach to create a true relational model. First, create an initial set of random tables intuitively from the information requirements. Figure 10-15 shows this initial data model.

Information Requirements

SocSecNo	
StudntName	
ClassNo	
RoomNo	
CourseDesc	
Credits	
FacltyName	
Speczn	
Department	
Phone	
TextbookTitle	
Author	Possibility of multiple authors
Price	
ExamName	Multiple exams for each student in a class
StudntScore	
Score	
MinScore	
Major	

Initial Set of Random Tables

SocSecNo	StudntName	ClassNo	RoomNo	CourseDesc	Credits	FacltyName	Speczn	Department	Phone

BookTitle	Author	Price	ExamName	Score	Grade	Major

Figure 10-15 University information: initial set of random tables.

SocSecNo	StudntName	ClassNo	RoomNo	CourseDesc	Credits	FacltyName	Speczn	Department	Phone	Major

ISBN	Title	Price

ISBN	Author

TypeID	ExamName	MinScore	SocSecNo	ClassNo	Score

SocSecNo	StudntName	Major

ClassNo	RoomNo	CourseDesc	Credits	FacltyName	Speczn	Department	Phone

---

ClassNo	CourseNo	RoomNo	FacultyID

CourseNo	CourseDesc	Credits	ISBN

FacultyID	FacltyName	Speczn	Department	Phone

TypeID	ExamName	MinScore

CourseNo	ClassNo	TypeID	SocSecNo	Score

Figure 10-16 University information: normalization steps.

Next, examine each table and go through the normalization steps one by one. Figure 10-16 illustrates the normalization steps. Note carefully the outcome from each step.

Figure 10-17 presents the final relational data model as a result of the normalization process. Compare this with the relational data model in Figure 9-22 arrived at through the data model transformation method.

CHAPTER SUMMARY

- The normalization method of creating a relational data model is an informal design methodology.
- In this method, you intuitively create an initial data model from information requirements and normalize it into a relational data model.
- An initial data model created through intuition contains potential anomalies relating to updates, deletes, and addition of data.
- The normalization approach consists of systematically examining the initial data model, removing the causes for anomalies, and transforming the data model into a true relational data model. The approach consists of definite, sequential steps.
- Each step removes one type of irregularity in the initial data model and transforms the model into a distinct normal form.
- The first four steps transform the initial data model into fundamental normal forms. Transformation into these fundamental normal forms removes

STUDENT

SocSecNo	StudntName	Major

TEXTBOOK

ISBN	Title	Price

AUTHOR

ISBN	Author

COURSE

CourseNo	CourseDesc	Credits	ISBN

CLASS

ClassNo	CourseNo	RoomNo	FacultyID

FACULTY

FacultyID	FacltyName	Speczn	Department	Phone

EXAMTYPE

TypeID	ExamName	MinScore

GRADE

CourseNo	ClassNo	TypeID	SocSecNo	Score

Figure 10-17 University information: final relational data model.

irregularities resulting from incorrect functional dependencies in the original relations.

- In practice, after the initial data model goes through the first four steps for fundamental normal forms, the resultant data model becomes truly relational and suitable for completing the logical design. No further normalization steps are usually necessary.
- The next three steps remove irregularities resulting from other types of incorrect dependencies in the original relations. These steps transform the data model into higher normal forms.
- The normalization method of creating a relational data model is difficult when the real-world information you are trying to model is large and complex.

REVIEW QUESTIONS

1. "The method of creating a relational data model seems to be an informal design technique." Discuss briefly.
2. Describe briefly the process of creating an initial data model from real-world information requirements. Why is this initial data model potentially incorrect?
3. "An update anomaly occurs when values of attributes are updated in a database." Explain with an example.

4. What is an addition anomaly in a database? Give an example.
5. Normalization is a systematic, step-by-step methodology. Describe how it is such a methodology.
6. When is a relation not in the second normal form? Give an example. How do you transform it into second normal form relations?
7. What are transitive dependencies in a relation? How do you remove transitive dependencies?
8. What is the Boyce-Codd normal form (BCNF)? Under what condition is a relation in BCNF?
9. What are multivalued dependencies? Explain with an example.
10. What is your understanding of the domain-key normal form? Why do you think this normal form is the ultimate goal of good design of a proper relational data model?

EXERCISES

1. Indicate whether true or false:
   A. Informal design of a relational data model starts with random tables created intuitively.
   B. Deletion anomaly prevents all deletion of data from a database.
   C. Each normalization step removes only one type of irregularity.
   D. When partial key dependencies are removed, a relation is in third normal form.
   E. In most cases, the first four normalization steps for fundamental normal forms are sufficient to produce a truly relational data model.
   F. A relation has transitive dependencies when every attribute is not functionally dependent on the full primary key.
   G. A relation in Boyce-Codd normal form is practically free from incorrect functional dependencies.
   H. If a data model is in fourth normal form, no multivalued dependencies exist.
   I. Join dependency in a relation is the same as multivalued dependency.
   J. Decomposition of a relation into more than one relation is the general technique in each step of the normalization process.
2. Describe the circumstances under which the normalization method is preferable to the data model transformation method for creating a relational data model. If you are the database designer for a florist business, which method will you adopt? Why?
3. List the fundamental normal forms. Describe the type of irregularity each step removes in arriving at the fundamental normal forms.
4. You are asked to create a small relational database to manage customer orders for a small business manufacturing standard vacuum cleaner parts. Create and initial data model through intuition for the Customer Order database. Normalize the initial data model and produce the final relational data model.

5. You are the senior database specialist in the database project of a large home improvement store responsible for implementing a relational database. Make assumptions and list the objects and relationships you need to include in your data model. Review the scope and complexity of the task and choose the appropriate method for creating the relational data model. Explain why you would choose either the data model transformation method or the normalization method.