Starting Point

Go to www.wiley.com/college/gillenson to assess your knowledge of database design.
Determine where you need to concentrate your effort.

What You’ll Learn in This Chapter

▲ Entity-relationship to relational table conversion processes
▲ Associative table requirements
▲ Standard normalization forms

After Studying This Chapter, You’ll Be Able To

▲ Convert stand-alone entities to tables
▲ Identify entity relationships by type and conversion requirements
▲ Convert unary entity relationships to database objects
▲ Convert binary entity relationships to database objects
▲ Compare E-R diagrams and the resulting relational table designs
▲ Select appropriate foreign keys and their placement in relational tables
▲ Create associative tables, including intersection data where needed
▲ Recognize unnormalized data
▲ Normalize a database by applying the first, second, and third normal forms
▲ Normalize a database through non-loss decomposition
▲ Normalize the tables created from an E-R diagram
▲ Recognize the need for denormalization and selectively denormalize data
INTRODUCTION
As you’ve learned, several steps are involved in designing a database to support a business and business applications. You must collect business data in the form of business documents, employee interviews, and other data sources. You use this data to generate an entity-relationship (E-R) diagram that describes the business and its requirements. From this diagram you will create the logical and then physical design of your database.

In this chapter, we’ll focus on the next step in the process: identifying the relational tables you will create based on the entities in the E-R diagram and then going through a process known as normalization, which will help you identify duplicate data and optimize data storage. Your final result is a finished relational design from which you can identify the database objects you need to implement your physical database.

4.1 Designing Relational Tables
The conversion of E-R diagrams to relational tables is a straightforward process. Typically, each entity converts to a table. For many-to-many relationships, each associative entity also converts to a table. Attributes become table columns, at least those attributes that apply to business and application requirements. During the process, rules must be followed to ensure that foreign keys appear in their proper places in the tables.

4.1.1 Converting a Single Entity
We’ll start with the easiest example, converting a single entity to a table. Figure 4-1 shows a single entity from a representative E-R diagram.

The entity in this situation is named SALESPEOPLE. The process of creating a table based on this entity is relatively simple. You create a table, typically with

![Figure 4-1]

SALESPEOPLE entity.
the same name, with a column for each of its attributes. This is shown in Figure 4-2. Notice that Salesperson Number is underlined to indicate that it is the entity’s unique identifier and will be used as the table’s primary key.

4.1.2 Converting Binary Relationships

The process becomes more interesting and more complicated when working with related tables. The process starts out much the same, identifying tables from business entities. Selecting the identifier and the primary key takes on greater importance because they are used in defining the foreign key to establish relationships between the tables.

Adding to the complication is that there is often more than one way to represent entity relationships as relational tables. It becomes a matter of identifying your options and picking the option that best meets your business and application requirements. You may also find it necessary to go back and fine tune the tables you identify as you go along.

Converting a One-to-One Binary Relationship

We’ll start with the one-to-one binary relationship shown in Figure 4-3. As you can see, this is a simple relationship based on the SALESPERSON and
4.1.2 CONCERTING BINARY RELATIONSHIPS

OFFICE entities. Each Salesperson works in one Office. Office, in this case, identifies a single office cubicle.

There are at least three options for designing tables to represent this data. The first option is shown in Figure 4-4, with the two entities combined into one relational table. This design is possible because the one-to-one relationship means that for one salesperson, there can only be one office associated with the salesperson and conversely, for one office there can be only one salesperson. Because of this relationship, a particular salesperson and office combination could be stored as one record, as shown.

There are three reasons why Figure 4-4 is not a good data design. The first two can be determined from the diagram in Figure 4-3. First, the very fact that salesperson and office were drawn in two different entity boxes in the E-R diagram indicates that they are thought of separately in this business environment. This means that they should be kept separate in the database. Second is the modality of zero at the SALESPERSON entity. Reading Figure 4-3 from right to left, it says that an office might have no one assigned to it. In the table shown in Figure 4-4, a few or possibly many record occurrences could have values for the office number, telephone, and size attributes but have the four attributes pertaining to salespersons empty or null. A null value is an undefined value, usually used to identify that no value is provided for that attribute. Even though considered as an undefined value, it is still considered a valid value and provides useful information by the fact that the attribute is not defined. This would result in wasted storage space. It also means that the Salesperson Number cannot be declared to be the primary key of the table, because there would be records with no primary key values, which is not allowed.

Before going on, there are a couple of points about storage costs and the relationship between database design and data storage that you should be aware of. Even though the cost per byte has dropped significantly over the years, wasted space continues to be an issue deserving consideration in your database design and implementation. Inefficient design and space use can lead to inefficient indexes and could mean less than optimal performance. That's not the only performance issue. As database tables (and the database as a whole) grow, access performance tends to suffer because of the increased volume of
data involved. Also, just because the price of storage has dropped, installing additional storage when you run out of space can be time consuming and usually means down time for the database, which typically is very expensive.

The third reason is not visible from the partial E-R diagram in Figure 4-3. However, when you look at the full E-R diagram in Figure 4-5, the reason

**Figure 4-5**

![Full E-R diagram.](image)
becomes immediately evident. The salesperson entity is involved in relationships with other entities in addition to the office entity. It is related to the CUSTOMER entity through a one-to-many relationship and the PRODUCT entity through a many-to-many relationship.

Figure 4-6 is a better choice. There are separate tables for the SALESPERSON and OFFICE entities. In order to record the relationship (which salesperson is assigned to which office), the Office Number attribute is placed as a foreign key in the SALESPERSON table. This connects each salesperson with the office to which he or she is assigned. Again, look at the modalities in the E-R diagram of Figure 4-3. Each salesperson is assigned to exactly one office, as is indicated by the two “ones” adjacent to the office entity. That translates directly into each record in the SALESPERSON table of Figure 4-6 having a value (and a single value at that) for its Office Number foreign key attribute. Each unassigned office will have a record in the OFFICE table, with Office Number as the primary key. In this case, unassigned offices aren’t seen as a problem. Their office numbers will simply not appear as foreign key values in the SALESPERSON table.

The third option is shown in Figure 4-7. Instead of placing Office Number as a foreign key in the SALESPERSON table, you place Salesperson Number as a foreign key in the OFFICE table. Recall from the E-R diagram that the modality of zero adjacent to the salesperson entity says that an office might be empty. As a result, some or perhaps many records of the OFFICE table in Figure 4-7 would have no value or a null in their Salesperson Number foreign key attribute positions.

It follows that if the modalities were reversed, meaning that the zero modality was adjacent to the OFFICE entity box and the one modality was adjacent to the SALESPERSON entity box, then the design in Figure 4-7 might be preferable. Reversed modality would mean that every office must have a salesperson assigned to it, but a salesperson may or may not be assigned to an office. There are different circumstances that could result in this situation. For
example, perhaps lots of the salespersons travel most of the time and don't need offices.

While we're in a "what if" mode, what if the modality was zero on both sides and we didn't have the other relationships to consider? Then we would have to make a judgment call between the designs of Figure 4-6 and Figure 4-7. If the goal is to minimize the number of null values in the foreign key, then we have to decide whether it is more likely that a salesperson is not assigned to an office or that an office is empty.

**Converting a One-to Many Binary Relationship**

The relationship shown in Figure 4-8 is a one-to-many binary relationship. In this relationship, each occurrence of the SALESPERSON entity is related to zero or more occurrences of the CUSTOMER entity.

Figure 4-9 shows the conversion of this E-R diagram into two relational tables. This time, the conversion is relatively simple. The rule is that the unique identifier

---

**Figure 4-7**

<table>
<thead>
<tr>
<th>SALESPERSON</th>
<th>OFFICE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Salesperson Number</td>
<td>Office Number</td>
</tr>
<tr>
<td>Salesperson Name</td>
<td>Telephone</td>
</tr>
<tr>
<td>Commission Percentage</td>
<td>Salesperson Number</td>
</tr>
<tr>
<td>Year of Hire</td>
<td>Size</td>
</tr>
</tbody>
</table>

Separate SALESPERSON and OFFICE tables, second version.

---

**Figure 4-8**

SALESPERSON to CUSTOMER relationship.
4.1.2 CONCERTING BINARY RELATIONSHIPS

of the entity on the “one side” of the one-to-many relationship is placed as a foreign key in the table representing the entity on the “many side.” In this case, the Salesperson Number attribute is placed in the CUSTOMER table as a foreign key. Each salesperson has one record in the SALESPERSON table, as does each customer in the CUSTOMER table. The Salesperson Number attribute in the CUSTOMER table links the two, and since the E-R diagram tells us that every customer must have a salesperson, there are no empty attributes in the CUSTOMER table records.

This solution also fits the full E-R diagram in Figure 4-5. The same Salesperson Number attribute used to establish a relationship with the CUSTOMER table can also be used in the relationship between SALESPERSON and OFFICE.

Converting a Many-to-Many Binary Relationship

Figure 4-10 once again uses the SALESPERSON entity, this time in a many-to-many binary relationship with the PRODUCT entity.

SALESPERSON and CUSTOMER tables.

This solution also fits the full E-R diagram in Figure 4-5. The same Salesperson Number attribute used to establish a relationship with the CUSTOMER table can also be used in the relationship between SALESPERSON and OFFICE.
Most relational DBMS systems do not directly support many-to-many relationships. Instead, you must also include an associative entity, in this case SALE, to establish the relationship. The E-R diagram in Figure 4-11 includes the associative entity.

Each of the two entities converts to a table with its own attributes but with no foreign keys (regarding this relationship). The SALESPERSON table and the PRODUCT table in Figure 4-12 each contain only the attributes shown in the SALESPERSON and PRODUCT entity boxes in Figure 4-10 and Figure 4-11. The primary key of this associated SALE table is the combination of the unique identifiers of the two entities in the many-to-many relationship. The primary key...
key might also include another attribute, such as a date and time stamp, to ensure unique values. A key that is defined by multiple columns is called a composite key.

The table based on the associative entity can include additional attributes representing intersection data. This is shown as the Quantity attribute in this example. However, this is not a requirement. The associative table often includes only the data values needed to establish and maintain the many-to-many relationship.

4.1.3 Converting Unary Relationships

A unary relationship is one that is based on a single table. The problem in this case is not deciding what tables you need to create, but on how key values can be used to identify and maintain the relationship.

Converting a One-to-One Unary Relationship

In Figure 4-13 you see a one-to-one unary relationship based on the SALESPERSON entity. Each Salesperson has one backup salesperson, someone who covers when the original salesperson isn't available.

In this case, with only one entity type involved and with a one-to-one relationship, the conversion requires only one table, as shown in Figure 4-14. For a particular salesperson, the Backup Number attribute represents the salesperson number of the backup person. The Backup Number is related to the SALESPERSON table's primary key, Salesperson Number. Any value used as Backup Number must be a valid Salesperson Number.
Converting a One-to-Many Unary Relationship

Figure 4-15 shows a one-to-many relationship based on the same Salesperson entity. You can read this as “each salesperson manages zero or more salespersons,” but for the purpose of creating a related table, it’s better to read this as “each salesperson is managed by a manager.” The manager is also an occurrence of the SALESPERSON entity.

Figure 4-16 shows the conversion of this diagram into a relational database table. Some employees manage other employees. An employee’s manager is recorded in the Manager Number. The manager numbers are actually salesperson numbers because some salespersons are sales managers who manage other salespersons. This arrangement works because each employee has only one manager. For any particular SALESPERSON record, there can only be one value for the Manager Number attribute. However, if you scan down the Manager Number column, a particular value may appear several times because a person can manage several other salespersons.
4.1.3 CONverting Unary Relationships

**Converting a Many-to-Many Unary Relationship**

Though not a common situation, you can have many-to-many unary relationships. One is shown in Figure 4-17. You have a situation where one product can be constructed out of a set of other products. For example, you might have the products Cup, Saucer, and Plate as separate occurrences of the PRODUCT entity. You could also have an occurrence called Place Setting, made up of a Cup, a Saucer, and a Plate.

As Figure 4-18 indicates, this relationship requires two tables in the conversion. The PRODUCT table has no foreign keys. The COMPONENT table indicates which items go into making up which other items. This is commonly referred to as a bill of materials. COMPONENT also contains any intersection data that might exist in the many-to-many relationship. In this example, the Quantity attribute indicates how many of a particular item go into making up another item.

The fact that we wind up with two tables in this conversion is not surprising. The general rule is that in the conversion of a many-to-many relationship
of any degree—unary, binary, or ternary—the number of tables will be equal to
the number of entity types (one, two, or three, respectively) plus one more table
for the many-to-many relationship. Thus the conversion of a many-to-many
unary relationship requires two tables, a many-to-many binary relationship
requires three tables, and a many-to-many ternary relationship requires four
tables. Ternary relationships are seldom diagramed in that form, so this chapter
does not include an example of a ternary conversion.
4.2 Comparing Relational Designs

So far, we have (for the most part) taken entity relationships out of context to focus on the conversion process. You might find it easier to understand the conversion process between entities and relational tables if you could see examples based on more complete E-R diagrams. It also puts the process into more of a real-world context, letting you see how it relates to business models you might need to develop. We’re going to convert the E-R models of two fictitious companies, General Hardware Company and Good Reading Bookstores. This chapter assumes that you can read and understand the E-R diagrams used in each case.

Keep in mind that you are still working in a pencil-and-paper mode, either literally or using a design or draw program. This is where you should work through your “what if . . .” scenarios and try out different data and relationship combinations. Your only real expense when making changes now is time. However, if you need to make changes to the model after you start implementing the physical database, the situation can become very expensive. Depending on where you are in the process, it might mean having to recreate the database or reload data. It might also mean having to redesign and even recode your application.

4.2.1 Designing General Hardware

Figure 4-19 is the General Hardware E-R diagram. It is convenient to begin the database design process with an important, central E-R diagram entity, such as salesperson, that has relationships with several other entities.

Thus the relational database in Figure 4-20 includes a SALESPERSON table with the four Salesperson attributes from the SALESPERSON entity box (plus the Office Number attribute, which we will return to shortly). Looking to the
right of the SALESPEerson entity box in the E-R diagram, we see a one-to-many relationship (“sells to”) between salespersons and customers. The database then includes a CUSTOMER table with the Salesperson Number attribute as a foreign key because Salesperson is on the “one side” of the one-to-many relationship and Customer is on the “many side” of the one-to-many relationship.
Customer Employee is a dependent entity of customer, and there is a one-to-many relationship between them. Customer Employee occurrences are employees who work for a particular customer and can make purchases for that customer. Because of the one-to-many relationship, the CUSTOMER EMPLOYEE table in the database includes the Customer Number attribute as a foreign key. Furthermore, the Customer Number attribute is part of the primary key of the CUSTOMER EMPLOYEE table because Customer Employee is a dependent entity and employee numbers are unique only within a customer. For example, two different customers could each have an employee number 100, representing different unique persons.
The PRODUCT table contains the three Product entity attributes. The many-to-many relationship between the SALESPERSON and PRODUCT entities is represented by the SALES table in the database. Notice that the combination of the unique identifiers (Salesperson Number and Product Number) of the two entities in the many-to-many relationship defines the primary key of the SALES table. Finally, the OFFICE entity has its table in the database with its three attributes, which brings us to the presence of the Office Number attribute as a foreign key in the SALESPERSON table. This is needed to maintain the one-to-one binary relationship between Salesperson and Office. We put the foreign key in the SALESPERSON table rather than in the OFFICE table because the modality adjacent to SALESPERSON is zero while the modality adjacent to OFFICE is one. An office may or may not have a salesperson assigned to it, but a salesperson must be assigned to an office. The result is that every salesperson must have an associated office number; the Office Number attribute in the SALESPERSON table can’t be null. This requirement can be easily enforced, automatically helping prevent data errors (missing Office Number). If we reversed it and put the Salesperson Number attribute in the OFFICE table, many of the Salesperson Number attribute values could be null since the zero modality going from office to salesperson tells us that an office can be empty. Because the value might or might not be null, data integrity, ensuring that the data is entered and stored correctly, is harder to enforce.

One last thought: Why did the PRODUCT table end up without having any foreign keys? Simply, it is because there is no situation that requires insertion of a foreign key. It is not the “target” (it is not on the “many side”) of any one-to-many binary relationship. It is not involved in a one-to-one binary relationship that requires a foreign key to ensure relational integrity, that is, to maintain the relationship. (Maintaining the relationship between referencing and referenced tables is also called referential integrity.) Finally, the PRODUCT table is not involved in a unary relationship that requires that the primary key to be repeated in the table.

4.2.2 Designing Good Reading Bookstores

The Good Reading Bookstores E-R diagram is shown in Figure 4-21. Beginning with the central BOOK entity and looking to its left, we see that there is a one-to-many relationship between books and publishers. A Publisher publishes many books, but a BOOK is published by just one publisher.

The Good Reading Bookstore’s relational database tables of Figure 4-22 show the BOOK and PUBLISHER tables. Publisher Name is a foreign key in the BOOK table because publisher is on the “one side” of the one-to-many relationship and book is on the “many side.” Next is the AUTHOR table, which is straightforward.
Following Up on Your Design

You might discover during the conversion process that your E-R design isn’t as complete as it should be. When this happens, you may find it necessary to go back to your original data to look for missing attributes.

Consider the following situation. You are designing a database for a customer support call center. When customers call they are automatically routed to the next available customer support agent. The agent with whom the customer speaks is, in this case, completely random. Your first choice as an associative entity might be the combination of the customer and employee identities. You would use this as the primary key.

When a customer calls back, he or she might get a different customer service agent or, by luck of the draw, the same customer service agent. This would mean the same combination of customer and employee identities, and would mean a duplicate value in the primary key. This is an illegal condition. You need an additional value, such as the date and time of the call, to produce a unique primary key value.
The many-to-many binary relationship between books and authors is reflected in the WRITING table, which has no intersection data.

Finally, there is the CUSTOMER entity and the many-to-many relationship between books and customers. Correspondingly, the relational database includes a CUSTOMER table and a SALE table to handle the many-to-many relationship. Notice that the Date, Price, and Quantity attributes appear in the SALE table as intersection data. Also notice that since a customer can buy the same book on more than one day, the Date attribute must be part of the primary key to achieve uniqueness.
4.3 Normalizing Data

Data normalization is a methodology for organizing attributes into tables so that redundancy among the nonkey attributes is eliminated. Each of the resultant tables deals with a single data focus, which is just another way of saying that each resultant table will describe a single entity type or a single many-to-many relationship. Furthermore, foreign keys will appear exactly where they are needed. In other words, the output of the data normalization process is a properly structured relational database.

You should be aware that the normalization rules provided here are based on a common version of the accepted normal forms. The definitions, as they apply to relational tables, can vary somewhat depending on your reference source. If you see small differences in how they are defined in other sources, you should consider it as a difference of opinion rather than that one or the other is wrong.

4.3.1 Using Normalization Techniques

The input required by the data normalization process comes in two parts. One is a list of all the attributes that must be incorporated into the database—that is, all of the attributes in all the entities involved in the business environment under discussion, plus all of the intersection data attributes in all of the many-to-many relationships between these entities. The other input, informally, is a list of all the defining associations between the attributes, which are a means of expressing that the value of one particular attribute is associated with a single, specific value of another attribute. Formally, these defining associations are known as functional dependencies. If we know that one of these attributes has a particular value, then the other attribute must have some other value. For example, for a particular Salesperson Number, 137, there is exactly one Salesperson Name, Baker, associated with it. We know this because a Salesperson Number uniquely identifies a salesperson,
and, after all, a person can have only one last name. Informally, we might say that Salesperson Number defines Salesperson Name. If I give you a Salesperson Number, you can give me back the one and only name that goes with it. These defining associations are commonly written with a right-pointing arrow like this:

Salesperson Number → Salesperson Name

In the more formal terms of functional dependencies, the attribute on the left side is referred to as the **determinant attribute**. This is because its value determines the value of the attribute on the right side. Conversely, we also say that the attribute on the right is functionally dependent on the attribute on the left.

Data normalization is best explained with an example. In order to demonstrate the main points of the data normalization process, we will modify part of the General Hardware Company business environment and focus on the SALESPERSON and PRODUCT entities. Let's assume that salespersons are organized into departments and that each department has a manager who is not herself a salesperson. Then the list of attributes that we will consider is shown in Figure 4-23.

The list of defining associations or functional dependencies is shown in Figure 4-24. Notice a couple of fine points about the list of defining associations in Figure 4-24. The last association:

Salesperson Number, Product Number → Quantity

shows that the combination of two or more attributes may define another attribute. That is, the combination of a particular Salesperson Number and a par-
4.3.2 NORMALIZING DATA BY THE NUMBERS

The data normalization process is known as a decomposition process, sometimes called non-loss decomposition. We will line up all of the attributes to be included in the relational database and start subdividing them into groups that will eventually form the database’s tables. Thus we are going to “decompose" the original list of all the attributes into subgroups. To do so, we are going to step through the attributes and identify groups of attributes that are related to each other. The goal is to create tables that are free from data redundancies and inconsistencies. One way to do this is to identify the defining associations between the attributes. For example, if we have a sale record that includes salesperson number, product number, and quantity sold, we can say that salesperson number defines or specifies a particular quantity. Put another way, in this business context, we know how many units of a particular product a particular salesperson has sold.

Another point, which will be important in demonstrating one step of the data normalization process, is that Manager Name is defined, independently, by two different attributes, Salesperson Number and Department Number:

Salesperson Number → Manager Name
Department Number → Manager Name

Both of these defining associations are true. If you identify a salesperson by Salesperson Number, you can determine the salesperson’s manager. Also, given a department number, you can determine the department manager. What this means is that during the systems analysis process, both of these equally true defining associations were discovered and noted. By the way, the fact that we know the department that a salesperson works in:

Salesperson Number → Department Number
(and that each of these two attributes independently defines Manager Name) is also an issue in the data normalization process.

4.3.2 Normalizing Data by the Numbers

The data normalization process is known as a decomposition process, sometimes called non-loss decomposition. We will line up all of the attributes to be included in the relational database and start subdividing them into groups that will eventually form the database’s tables. Thus we are going to “decompose" the original list of all the attributes into subgroups. To do so, we are going to step

| Salesperson Number → Salesperson Name |
| Salesperson Number → Commission Percentage |
| Salesperson Number → Year of Hire |
| Salesperson Number → Department Number |
| Salesperson Number → Manager Name |
| Product Number → Product Name |
| Product Number → Unit Price |
| Department Number → Manager Name |
| Salesperson Number, Product Number → Quantity |

Salesperson entity defining associations (functional dependencies)
through a number of normal forms, which are rules for data normalization. First, we will examine what unnormalized data looks like. Then, we will work through the three main normal forms in order:

- First Normal Form
- Second Normal Form
- Third Normal Form

There are certain exception conditions that have also been described as normal forms. Exception conditions are nonstandard normal forms in addition to the three accepted standard normal forms. These include Boyce-Codd Normal Form, Fourth Normal Form, and Fifth Normal Form. They are relatively less common in practice and will not be covered here. Here are three additional points to remember:

1. Once the attributes are arranged in third normal form (and if none of the exception conditions is present), the group of tables that they comprise is, in fact, a well-structured relational database with no data redundancy.
2. A group of tables is said to be in a particular normal form if every table in the group is in that normal form.
3. The data normalization process is progressive. If a group of tables is in second normal form, it is also in first normal form. If the tables are in third normal form, they are also in second normal form.

**Understanding Unnormalized Data**

Figure 4-25 shows the salesperson and product-related attributes listed in Figure 4-24 arranged in a table with sample data. The salesperson and product data is taken from the General Hardware Company relational database. Note that salespersons 137, 204, and 361 are all in department number 73 and that their manager is Scott. Salesperson 186 is in department number 59 and his manager is Lopez.

The table in Figure 4-25 is unnormalized. The table has four records, one for each salesperson. But since each salesperson has sold several products and there is only one record for each salesperson, several attributes of each record must have multiple values. For example, the record for salesperson 137 has three product numbers, 19440, 24013, and 26722, in its Product Number attribute because salesperson 137 has sold all three of those products. Having such multivalued attributes is not permitted and so this table is unnormalized.

**Normalizing to First Normal Form**

In the first normal form, each attribute value is atomic, that is, no attribute is multivalued. The table in Figure 4-26 is the first normal form representation of
4.3.2 NORMALIZING DATA BY THE NUMBERS

Figure 4-25

SALESPERSON/PRODUCT table

<table>
<thead>
<tr>
<th>Salesperson Number</th>
<th>Product Number</th>
<th>Salesperson Name</th>
<th>Commission Percentage</th>
<th>Year of Hire</th>
<th>Department Number</th>
<th>Manager Name</th>
<th>Product Name</th>
<th>Unit Price</th>
<th>Quantity</th>
</tr>
</thead>
<tbody>
<tr>
<td>137</td>
<td>19440</td>
<td>Baker</td>
<td>10</td>
<td>1995</td>
<td>73</td>
<td>Scott</td>
<td>Hammer</td>
<td>17.50</td>
<td>473</td>
</tr>
<tr>
<td></td>
<td>24013</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Saw</td>
<td>26.25</td>
<td>170</td>
</tr>
<tr>
<td></td>
<td>26722</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Pliers</td>
<td>11.50</td>
<td>688</td>
</tr>
<tr>
<td>186</td>
<td>16386</td>
<td>Adams</td>
<td>15</td>
<td>2001</td>
<td>59</td>
<td>Lopez</td>
<td>Wrench</td>
<td>12.95</td>
<td>1745</td>
</tr>
<tr>
<td></td>
<td>19440</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Hammer</td>
<td>17.50</td>
<td>2529</td>
</tr>
<tr>
<td></td>
<td>21765</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Drill</td>
<td>32.99</td>
<td>1962</td>
</tr>
<tr>
<td></td>
<td>24013</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Saw</td>
<td>26.25</td>
<td>3071</td>
</tr>
<tr>
<td>204</td>
<td>21765</td>
<td>Dickens</td>
<td>10</td>
<td>1998</td>
<td>73</td>
<td>Scott</td>
<td>Drill</td>
<td>32.99</td>
<td>809</td>
</tr>
<tr>
<td></td>
<td>26722</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Pliers</td>
<td>11.50</td>
<td>734</td>
</tr>
<tr>
<td>361</td>
<td>16386</td>
<td>Carlyle</td>
<td>20</td>
<td>2001</td>
<td>73</td>
<td>Scott</td>
<td>Wrench</td>
<td>12.95</td>
<td>3729</td>
</tr>
<tr>
<td></td>
<td>21765</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Drill</td>
<td>32.99</td>
<td>3110</td>
</tr>
<tr>
<td></td>
<td>26722</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Pliers</td>
<td>11.50</td>
<td>2738</td>
</tr>
</tbody>
</table>

General Hardware sample unnormalized data.

the data. The attributes under consideration have been listed in one table, and a primary key has been established. In this definition of normal forms, the requirement for a primary key is not listed as part of any normal form, but is considered an assumed requirement of the initial E-R diagramming process.

As the sample data in Figure 4-27 shows, the number of records has increased compared to the unnormalized representation. Every attribute of every record has just one value. The multivalued attributes from Figure 4-25 are eliminated.

The combination of the Salesperson Number and Product Number attributes constitutes the table's primary key. The business context tells us that the combination of the two provides unique identifiers for the records of the table and that there is no single attribute that will do the job. In terms of data normalization,
DESIGNING A DATABASE

Figure 4-27

<table>
<thead>
<tr>
<th>Salesperson Number</th>
<th>Product Number</th>
<th>Salesperson Name</th>
<th>Commission Percentage</th>
<th>Year of Hire</th>
<th>Department Number</th>
<th>Manager Name</th>
<th>Product Name</th>
<th>Unit Price</th>
<th>Quantity</th>
</tr>
</thead>
<tbody>
<tr>
<td>137</td>
<td>19440</td>
<td>Baker</td>
<td>10</td>
<td>1995</td>
<td>73</td>
<td>Scott</td>
<td>Hammer</td>
<td>17.50</td>
<td>473</td>
</tr>
<tr>
<td>137</td>
<td>24013</td>
<td>Baker</td>
<td>10</td>
<td>1995</td>
<td>73</td>
<td>Scott</td>
<td>Saw</td>
<td>26.25</td>
<td>170</td>
</tr>
<tr>
<td>137</td>
<td>26722</td>
<td>Baker</td>
<td>10</td>
<td>1995</td>
<td>73</td>
<td>Scott</td>
<td>Pliers</td>
<td>11.50</td>
<td>688</td>
</tr>
<tr>
<td>186</td>
<td>16386</td>
<td>Adams</td>
<td>15</td>
<td>2001</td>
<td>59</td>
<td>Lopez</td>
<td>Wrench</td>
<td>12.95</td>
<td>1475</td>
</tr>
<tr>
<td>186</td>
<td>19440</td>
<td>Adams</td>
<td>15</td>
<td>2001</td>
<td>59</td>
<td>Lopez</td>
<td>Hammer</td>
<td>17.50</td>
<td>2529</td>
</tr>
<tr>
<td>186</td>
<td>21765</td>
<td>Adams</td>
<td>15</td>
<td>2001</td>
<td>59</td>
<td>Lopez</td>
<td>Drill</td>
<td>32.99</td>
<td>1962</td>
</tr>
<tr>
<td>186</td>
<td>24013</td>
<td>Adams</td>
<td>15</td>
<td>2001</td>
<td>59</td>
<td>Lopez</td>
<td>Saw</td>
<td>26.25</td>
<td>3071</td>
</tr>
<tr>
<td>204</td>
<td>21765</td>
<td>Dickens</td>
<td>10</td>
<td>1998</td>
<td>73</td>
<td>Scott</td>
<td>Drill</td>
<td>32.99</td>
<td>809</td>
</tr>
<tr>
<td>204</td>
<td>26722</td>
<td>Dickens</td>
<td>10</td>
<td>1998</td>
<td>73</td>
<td>Scott</td>
<td>Pliers</td>
<td>11.50</td>
<td>734</td>
</tr>
<tr>
<td>361</td>
<td>16386</td>
<td>Carlyle</td>
<td>20</td>
<td>2001</td>
<td>73</td>
<td>Scott</td>
<td>Wrench</td>
<td>12.95</td>
<td>3729</td>
</tr>
<tr>
<td>361</td>
<td>21765</td>
<td>Carlyle</td>
<td>20</td>
<td>2001</td>
<td>73</td>
<td>Scott</td>
<td>Drill</td>
<td>32.99</td>
<td>3110</td>
</tr>
<tr>
<td>361</td>
<td>26722</td>
<td>Carlyle</td>
<td>20</td>
<td>2001</td>
<td>73</td>
<td>Scott</td>
<td>Pliers</td>
<td>11.50</td>
<td>2738</td>
</tr>
</tbody>
</table>

Data normalized to the first normal form.

According to the list of defining associations or functional dependencies of Figure 4-24, every attribute in the table is either part of the primary key or is defined by one or both attributes of the primary key. This is actually a requirement of the second normal form, as we will see later. Salesperson Name, Commission Percentage, Year of Hire, Department Number, and Manager Name are each defined by Salesperson Number. Product Name and Unit Price are each defined by Product Number. Quantity is defined by the combination of Salesperson Number and Product Number.

These two different ways of approaching the primary key selection are equivalent. If the combination of a particular Salesperson Number and a particular Product Number is unique, then it identifies exactly one record of the table. And if it identifies exactly one record of the table, then that record shows the single value of each of the nonkey attributes that is associated with the unique combination of the key attributes. But that is the same thing as saying that each of the nonkey attributes is defined by or is functionally dependent on the primary key. For example, consider the first record of the table in Figure 4-27.

The combination of Salesperson Number 137 and Product Number 19440 is unique. Only one record in the table can have that combination of Salesperson Number and Product Number values. Therefore, if someone specifies those values, the only Salesperson Name that can be associated with them is Baker, the only Commission Percentage is 10, and so forth. But that has the same effect
as the concept of functional dependency. Since Salesperson Name is functionally
dependent on Salesperson Number, given a particular Salesperson Number, say
137, only one Salesperson Name can be associated with it, Baker. Since Com-
mission Percentage is functionally dependent on Salesperson Number, given a
particular Salesperson Number, say 137, there can be only one Commission Per-
centage associated with it, 10.

First normal form is merely a starting point in the normalization process.
At this point, we have a great deal of data redundancy. Three records involve
salesperson 137 (the first three records), and so there are three places in which
his name is listed as Baker, his commission percentage is listed as 10, and so
on. Similarly, two records involve product 19440 (the first and fifth records),
and this product’s name is listed twice as Hammer and its unit price is listed
twice as 17.50. Intuitively, the reason for this is that attributes of two differ-
ent kinds of entities, salespersons and products, have been mixed together in
one table.

In some references, you will see the first normal form described differently,
stated that each table within a row must be uniquely identified. Data designers
who prefer that definition roll the requirement for atomic data to the second
normal form. In that case, the requirements of the second and third normal
forms, as described in this chapter, are combined as the third normal form. In
either case, the final goal and final result remains the same.

Normalizing to Second Normal Form

Since data normalization is a decomposition process, the next step will be to
decompose the table defined in Figure 4-26 into smaller tables to eliminate data
redundancy. And since we have established that at least some of the redundancy
is due to mixing attributes about salespersons and products, it seems reasonable
to want to separate them out. Informally, we will look at each of the nonkey
attributes and decide which attributes of the key are really needed to define it.
For example, Salesperson Name really only needs Salesperson Number to define
it. Product Name needs only Product Number to define it. Quantity needs both
attributes.

More formally, second normal form does not allow partial functional
dependencies where data is dependent on part of the primary key. That is, in
a table in second normal form, every nonkey attribute must be fully func-
tionally dependent on the entire key of that table. In plain language, a nonkey
attribute cannot depend on only part of the key, the way that Salesperson Name,
Product Name, and most of the other nonkey attributes of Figure 4-26 violate
this restriction.

Figure 4-28 shows the salesperson and product attributes arranged in the
second normal form. There is a SALESPERSON table in which Salesperson
Number is the sole primary key attribute. Every nonkey attribute of the table
DESIGNING A DATABASE

is fully defined by Salesperson Number. Similarly, the PRODUCT table has Product Number as its sole primary key attribute, and the nonkey attributes of the table are dependent just on it. The QUANTITY table has the combination of Salesperson Number and Product Number as its primary key because its nonkey attribute, Quantity, requires both of them taken together to define it.

Figure 4-29 shows the sample salesperson and product data arranged in the second normal form structure. Much of the initial data redundancy has been eliminated. Now, salesperson 137’s name is listed as Baker, his commission percentage listed as 10, and so forth only once in the SALESPERSON table. Product 19440’s name is listed as Hammer, and its unit price is listed as 17.50 only once in the PRODUCT table.

Second normal form is thus a great improvement over first normal form, but has all of the redundancy been eliminated? In general, that depends on the particular list of attributes and defining associations. It is possible that second normal form can be completely free of data redundancy. In such a case, the second normal form representation is identical to the third normal form representation. This is not the case in our example.

A close look at the sample data in Figure 4-29 reveals that the second normal form structure has not eliminated all of the data redundancy. At the right-hand end of the SALESPERSON table, the fact that Scott is the manager of department 73 is repeated three times. This constitutes redundant data. All of the nonkey attributes are fully functionally dependent on Salesperson Number, but that is not the source of the problem. It’s true that Salesperson Number defines
4.3.2 NORMALIZING DATA BY THE NUMBERS

Figure 4-29

SALESPERSON table

<table>
<thead>
<tr>
<th>Salesperson Number</th>
<th>Salesperson Name</th>
<th>Commission Percentage</th>
<th>Year of Hire</th>
<th>Department Number</th>
<th>Manager Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>137</td>
<td>Baker</td>
<td>10</td>
<td>1995</td>
<td>73</td>
<td>Scott</td>
</tr>
<tr>
<td>186</td>
<td>Adams</td>
<td>15</td>
<td>2001</td>
<td>59</td>
<td>Lopez</td>
</tr>
<tr>
<td>204</td>
<td>Dickens</td>
<td>10</td>
<td>1998</td>
<td>73</td>
<td>Scott</td>
</tr>
<tr>
<td>361</td>
<td>Carlyle</td>
<td>20</td>
<td>2001</td>
<td>73</td>
<td>Scott</td>
</tr>
</tbody>
</table>

PRODUCT table

<table>
<thead>
<tr>
<th>Product Number</th>
<th>Product Name</th>
<th>Unit Price</th>
</tr>
</thead>
<tbody>
<tr>
<td>16386</td>
<td>Wrench</td>
<td>12.95</td>
</tr>
<tr>
<td>19440</td>
<td>Hammer</td>
<td>17.50</td>
</tr>
<tr>
<td>21765</td>
<td>Drill</td>
<td>32.99</td>
</tr>
<tr>
<td>24013</td>
<td>Saw</td>
<td>26.25</td>
</tr>
<tr>
<td>26722</td>
<td>Pliers</td>
<td>11.50</td>
</tr>
</tbody>
</table>

QUANTITY table

<table>
<thead>
<tr>
<th>Salesperson Number</th>
<th>Product Number</th>
<th>Quantity</th>
</tr>
</thead>
<tbody>
<tr>
<td>137</td>
<td>19440</td>
<td>473</td>
</tr>
<tr>
<td>137</td>
<td>24013</td>
<td>170</td>
</tr>
<tr>
<td>137</td>
<td>26722</td>
<td>528</td>
</tr>
<tr>
<td>186</td>
<td>16386</td>
<td>1745</td>
</tr>
<tr>
<td>186</td>
<td>19440</td>
<td>2529</td>
</tr>
<tr>
<td>186</td>
<td>21765</td>
<td>1962</td>
</tr>
<tr>
<td>186</td>
<td>24013</td>
<td>3071</td>
</tr>
<tr>
<td>204</td>
<td>21765</td>
<td>809</td>
</tr>
<tr>
<td>204</td>
<td>26722</td>
<td>734</td>
</tr>
<tr>
<td>361</td>
<td>16386</td>
<td>3729</td>
</tr>
<tr>
<td>361</td>
<td>21765</td>
<td>3110</td>
</tr>
<tr>
<td>361</td>
<td>26722</td>
<td>2738</td>
</tr>
</tbody>
</table>

Second normal form sample data.

both Department Number and Manager Name. Focusing in on a particular salesperson, you should know the salesperson’s department and manager’s name. But, as indicated in the next-to-the-last defining association of Figure 4-24, one of those two attributes defines the other: given a department number, you can tell who the manager of that department is. In the SALESPERSON table, one of the nonkey attributes, Department Number, defines another one of the nonkey attributes, Manager Name. This is what is causing the problem.
Normalizing to Third Normal Form

In third normal form, nonkey attributes are not allowed to define other nonkey attributes. Stated more formally, third normal form does not allow transitive dependencies in which one nonkey attribute is functionally dependent on another.

In Figure 4-29, you see that in the SALESPERSON table, Department Number, and Manager Name are both nonkey attributes. Department Number defines Manager Name. Figure 4-30 shows the third normal form representation of the attributes. Note that the SALESPERSON table in Figure 4-28 has been further decomposed into the SALESPERSON and DEPARTMENT tables in Figure 4-30. The Department Number and Department Manager attributes, which were the cause of redundant data in the second normal form, were split off to form the DEPARTMENT table, but a copy of the Department Number attribute (the primary key attribute of the new DEPARTMENT table) was left behind in the SALESPERSON table. If this had not been done, there no longer would have been a way to indicate each salesperson's department, and you would have lost that relationship. Keep in mind, the goal is non-loss decomposition. In other words, no data is lost in the process.

The sample data for the third normal form structure in Figure 4-30 is shown in Figure 4-31. Now, the fact that Scott is the manager of department 73 is shown...
**4.3.2 NORMALIZING DATA BY THE NUMBERS**

**Figure 4-31**

**SALESPEOPLE table**

<table>
<thead>
<tr>
<th>Salesperson Number</th>
<th>Salesperson Name</th>
<th>Commission Percentage</th>
<th>Year of Hire</th>
<th>Department Number</th>
</tr>
</thead>
<tbody>
<tr>
<td>137</td>
<td>Baker</td>
<td>10</td>
<td>1995</td>
<td>73</td>
</tr>
<tr>
<td>186</td>
<td>Adams</td>
<td>15</td>
<td>2001</td>
<td>59</td>
</tr>
<tr>
<td>204</td>
<td>Dickens</td>
<td>10</td>
<td>1998</td>
<td>73</td>
</tr>
<tr>
<td>361</td>
<td>Carlyle</td>
<td>20</td>
<td>2001</td>
<td>73</td>
</tr>
</tbody>
</table>

**DEPARTMENT table**

<table>
<thead>
<tr>
<th>Department Number</th>
<th>Manager Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>59</td>
<td>Lopez</td>
</tr>
<tr>
<td>73</td>
<td>Scott</td>
</tr>
</tbody>
</table>

**PRODUCT table**

<table>
<thead>
<tr>
<th>Product Number</th>
<th>Product Name</th>
<th>Unit Price</th>
</tr>
</thead>
<tbody>
<tr>
<td>16386</td>
<td>Wrench</td>
<td>12.95</td>
</tr>
<tr>
<td>19440</td>
<td>Hammer</td>
<td>17.50</td>
</tr>
<tr>
<td>21765</td>
<td>Drill</td>
<td>32.99</td>
</tr>
<tr>
<td>24013</td>
<td>Saw</td>
<td>26.25</td>
</tr>
<tr>
<td>26722</td>
<td>Pliers</td>
<td>11.50</td>
</tr>
</tbody>
</table>

**QUANTITY table**

<table>
<thead>
<tr>
<th>Salesperson Number</th>
<th>Product Number</th>
<th>Quantity</th>
</tr>
</thead>
<tbody>
<tr>
<td>137</td>
<td>19440</td>
<td>473</td>
</tr>
<tr>
<td>137</td>
<td>24013</td>
<td>170</td>
</tr>
<tr>
<td>137</td>
<td>26722</td>
<td>688</td>
</tr>
<tr>
<td>186</td>
<td>16386</td>
<td>1745</td>
</tr>
<tr>
<td>186</td>
<td>19440</td>
<td>2529</td>
</tr>
<tr>
<td>186</td>
<td>21765</td>
<td>1962</td>
</tr>
<tr>
<td>186</td>
<td>24013</td>
<td>3071</td>
</tr>
<tr>
<td>204</td>
<td>21765</td>
<td>809</td>
</tr>
<tr>
<td>204</td>
<td>26722</td>
<td>734</td>
</tr>
<tr>
<td>361</td>
<td>16386</td>
<td>3729</td>
</tr>
<tr>
<td>361</td>
<td>21765</td>
<td>3110</td>
</tr>
<tr>
<td>361</td>
<td>26722</td>
<td>2738</td>
</tr>
</tbody>
</table>

Sample data in the third normal form.
only once, in the second record of the DEPARTMENT table. Notice that the
Department Number attribute in the SALESPERSON table continues to indicate
the salesperson's department.

4.3.3 Shortening the Process
As you can see, the normalization process tends to result in more and smaller
tables created through decomposition. When you look back over the design
process, you'll see that most of the tables identified through the normalization
process had already been identified through data modeling. In fact, you should
typically start as we did in the beginning of this chapter, by defining tables based
on the E-R diagram.

Consider the tables defined by the E-R diagram your first draft of the data-
base design. Once you have your relational tables identified, apply the three nor-
mal forms to each of the tables. The usual goal is normalizing to the third nor-
mal form, sometimes indicated as 3NF. Check each table and verify that each
adheres to the requirements for normalization to 3NF. This will save you the
time and effort of building a "supertable" (not a formal term, just a convenient
description) containing every possible attribute in a relationship. Remember that
when looking at normalization through full deconstruction, we focused on one
primary entity. We would have to go through this process with each entity that
brought with it additional relationships. It's easy to see how this could become
a daunting task.

If you start with your E-R diagram, you will still find normalization
requirements. Start with the first normal form and work your way through.
For example, the customer table might have an attribute of Address. This vio-
lates the first normal form because this attribute is multivalued. For addresses
in the United States, it includes a street address or PO Box (and possibly both),
City, State, and ZIP Code. You would need to break each of these down into
individual attributes to meet the first normal form. Continue the process
through the second and third normal forms, creating additional related tables
as necessary to meet normalization requirements.

You will often find that this is the least complicated and most efficient way
to handle the normalization process. Even though some tables require additional
normalization, others will already be at 3NF and not require any additional
effort.

4.3.4 Denormalizing Data
The smaller tables created by the normalization process are typically the most
efficient design for data entry and data modification. One reason for this is
that you have eliminated duplicate data, reducing the amount of data that
must be written to the database. However, there are two potential problems.
One problem is that this process can be taken to the extreme. Consider customer addresses for example. It's likely that you will have several customers in the same state. You could create a separate STATE table to contain this information and create a relationship between the CUSTOMER and STATE tables, but it's probably less work and requires less storage space to just store the 2-character state abbreviation with each customer.

The other problem is that a design that makes for efficient data entry does not always make for efficient data retrieval. During normalization, you tend to break a table down into smaller, related tables. There is a good possibility that at least some of your queries will require you to recombine that data into a single result. This is done through a query process known as joining, where you combine the data from two tables based on a linking column or columns. Typically, you will combine two related tables based on the foreign key, but that's not the only possibility. This can be a resource-intensive process, becoming more intensive as more tables are added.

This can be an issue in any database, but tends especially to be a problem in decision support databases where the majority of the database activity relates

---

**FOR EXAMPLE**

**Finding New Tables**

It's fairly common to “discover” new tables during the normalization process. Your E-R diagram includes an ORDER entity. For each order, you have the customer placing the order, the employee writing up the order, the order date, order number, and other information that applies to the order as a whole. You also have information about individual line items, such as the item ID, quantity, and selling price. There's no reason, usually, to store the extended total (quantity times selling price) because that can be calculated whenever you need it.

If you create an ORDER table, for it to be properly normalized, you will need a row for each line item in the order. That means that the customer, employee, order date, and any other general information about the order are also repeated for each line item. This could result in a significant amount of data and wasted space. A better solution is to have two tables. One, call it ORDERHEAD, contains the information that applies to the order as a whole. The other, call it ORDERITEM, contains the information for each line item. You would use the Order Number as the identifier in ORDERHEAD, and also use it as the foreign key in ORDERITEM to maintain the relationship between the two tables.
to data retrieval. If you regularly need the joined data, you could find it more efficient in the long run to denormalize the data, combine two or more normalized data tables into one less normalized table. For example, you might need to draw on data from three or four different tables to generate employee paychecks, including columns from an EMPLOYEE table, a TIMESHEET table, a PAYRATE table, and other tables in a single report. You might find it better to create a separate table named EMPLOYEELYPAY that contains all of this information. Keep in mind, however, that if you also keep all of the other tables, you are introducing duplicate data into the database. Whether or not the performance increase is worth the additional overhead will have to be evaluated on a case-by-case basis.

Why the concern about performance? The more operations the database has to perform, the greater the load on resources, which can result in performance loss. If you create a new EMPLOYEELYPAY table while also keeping the same data in the EMPLOYEE, TIMESHEET, and PAYRATE tables, you are forcing the database server to make additional updates anytime you add or modify data. If you change a rate in the PAYRATE table or hours in the TIMESHEET table, for example, you will also have to update records in the EMPLOYEELYPAY table to reflect these changes. Introduce too many situations where duplicate updates are needed and performance eventually suffers.

**SELF-CHECK**

- List and describe the three normal forms.
- Explain how normalizing to the third normal form can result in additional relational tables.
- Explain the meaning of the term non-loss decomposition.

**SUMMARY**

This chapter discussed the process of creating a relational database design. You saw how to convert simple entities, unary relationships, and binary relationships to relational tables. This included choosing the foreign keys needed to establish and maintain the relationships. You compared examples in which E-R diagrams were converted to relational tables. You also learned about the normalization process and how apply the three normal forms.
## KEY TERMS

<table>
<thead>
<tr>
<th>Term</th>
<th>Term</th>
</tr>
</thead>
<tbody>
<tr>
<td>Composite key</td>
<td>Non-loss decomposition</td>
</tr>
<tr>
<td>Data integrity</td>
<td>Normal forms</td>
</tr>
<tr>
<td>Data normalization</td>
<td>Null value</td>
</tr>
<tr>
<td>Decomposition process</td>
<td>Partial functional dependency</td>
</tr>
<tr>
<td>Defining association</td>
<td>Referential integrity</td>
</tr>
<tr>
<td>Determinant attribute</td>
<td>Relational integrity</td>
</tr>
<tr>
<td>Exception conditions</td>
<td>Second normal form</td>
</tr>
<tr>
<td>First normal form</td>
<td>Third normal form (3NF)</td>
</tr>
<tr>
<td>Functional dependency</td>
<td>Transitive dependency</td>
</tr>
<tr>
<td>Joining</td>
<td></td>
</tr>
</tbody>
</table>
ASSESS YOUR UNDERSTANDING

Go to www.wiley.com/college/gillenson to evaluate your knowledge of database design.

Measure your learning by comparing pre-test and post-test results.

Summary Questions

1. There is always a one-to-one relationship between entities in an E-R diagram and the tables in a database design. True or False?

2. Which statement best describes the process of converting a one-to-one binary relationship to related tables?
   (a) You should convert the entities into a single related table.
   (b) There will typically be only one way to define the tables.
   (c) There will typically be more than one way to define the tables.

3. You are converting a many-to-many unary relationship. How many relational tables will result from this conversion?
   (a) one
   (b) two
   (c) three
   (d) four

4. There is a many-to-many relationship between the Customer and Issue entities. Each combination of Customer and Issue occurrences is unique. Which should you use as the associative table primary key?
   (a) the Customer identifier
   (b) the Issue identifier
   (c) the combination of the Customer and Issue identifiers
   (d) you don’t have enough information to determine the best primary key

5. Customer relates to Order as a one-to-many relationship. How should you define the relationship between the CUSTOMER and ORDER tables?
   (a) Add the CUSTOMER primary key to the ORDER table as a foreign key.
   (b) Add the ORDER primary key to the CUSTOMER table as a foreign key.
   (c) Create a new value to use as a foreign key in both the CUSTOMER and ORDER tables.
   (d) Create an associative table.

6. An associative table is needed when converting a one-to-many or many-to-many relationship. True or False?

7. When converting related entities, it is necessary to consider all relationships in which an entity is included. True or False?
8. Manager-to-Employee is a one-to-many unary relationship. You create an EMPLOYEE table. How any additional tables do you need?
   (a) none
   (b) one
   (c) two

9. Data integrity ensures that data is entered and stored correctly. True or False?

10. The usual goal in normalization is normalizing to the third normal form. True or False?

11. What statement best describes the second normal form?
   (a) Each attribute value is atomic, that is, no attribute is multivalued.
   (b) Transitive dependencies are not allowed.
   (c) Every nonkey attribute must be fully functionally dependent on the entire key.

12. If tables are in the second normal form, they may or may not also be in the first normal form. True or false?

13. Which normal form works to eliminate partial functional dependencies?
   (a) first
   (b) second
   (c) third
   (d) fourth

14. Which term is used to refer to the value on the left in the following association?
    Salesperson number → Salesperson name
   (a) independent attribute
   (b) associative attribute
   (c) dependent attribute
   (d) determinant attribute

15. In normalization, what is an exception condition?
    (a) normalization forms beyond the three standard normal forms
    (b) data that cannot be normalized
    (c) the requirement to add a table to support relationships
    (d) a table that is self-normalizing

16. Saying that a table is normalized to the Boyce-Codd normal form is the same as saying that a table is normalized to 3NF. True or False?

17. Combining data from different normalized tables into a query result is referred to as joining. True or False?

18. Which column list violates the third normal form?
(a) vendor name (primary key), alternate vendor, PO Box, city, state, postal code

(b) employee number (primary key), last name, first name, department number, department name, e-mail address, cubicle location, manager ID

(c) product SKU (primary key), description, warehouse location, quantity on hand, quantity on order, vendor

(d) order number (primary key), line item number (primary key), product SKU, quantity, selling price

**Applying This Chapter**

1. You can order multiple products from any vendor. A product can be ordered from multiple vendors. The identifier for the PRODUCT entity is SKU. The identifier for the VENDOR entity is Vendor Number. What kind of relationship does this describe? List the tables that would result from conversion to relational tables. Include the primary key for each of the tables.

2. You are designing relational tables for the E-R diagram shown in Figure 4-32. What foreign keys would be added during conversion (list by table)? What tables not currently identified by an entity would be added (if any)? What are the primary key attributes in the first normal form for this model?
3. You are designing relational tables based on the functional dependencies in Figure 4-33. List the tables and primary keys represented by this.

**Figure 4-33**

<table>
<thead>
<tr>
<th>Functional Dependencies</th>
<th>Table and Primary Keys</th>
</tr>
</thead>
<tbody>
<tr>
<td>Salesperson Number → Salesperson Name</td>
<td>Salesperson Name</td>
</tr>
<tr>
<td>Salesperson Number → Year of Hire</td>
<td>Year of Hire</td>
</tr>
<tr>
<td>Salesperson Number → Department Number</td>
<td>Department Number</td>
</tr>
<tr>
<td>Salesperson Number → Manager Name</td>
<td>Manager Name</td>
</tr>
<tr>
<td>Customer Number → Customer Name</td>
<td>Customer Name</td>
</tr>
<tr>
<td>Customer Number → Salesperson Number</td>
<td>Customer Number, Employee Number → Employee Name</td>
</tr>
<tr>
<td>Customer Number → HQ City</td>
<td>HQ City</td>
</tr>
<tr>
<td>Customer Number, Employee Number → Title</td>
<td>Customer Number, Employee Number → Product Name</td>
</tr>
<tr>
<td>Product Number → Product Name</td>
<td>Product Name</td>
</tr>
<tr>
<td>Product Number → Unit Price</td>
<td>Unit Price</td>
</tr>
<tr>
<td>Department Number → Manager Name</td>
<td>Department Number</td>
</tr>
<tr>
<td>Salesperson Number, Product Number → Quantity</td>
<td>Salesperson Number, Product Number</td>
</tr>
<tr>
<td>Office Number → Telephone</td>
<td>Office Number</td>
</tr>
<tr>
<td>Office Number → Salesperson Number</td>
<td>Office Number</td>
</tr>
<tr>
<td>Office Number → Size</td>
<td>Office Number</td>
</tr>
</tbody>
</table>

Sample functional dependencies for General Hardware Company.
YOU TRY IT

Creating and Normalizing a Relational Table Design

Figure 4-34 is the E-R diagram for Lucky Rent-A Car.

1. Complete Table 4-1 with the information requested.

Table 4-1: Lucky Rent-a-Car

<table>
<thead>
<tr>
<th>Table</th>
<th>Primary Key Column(s)</th>
<th>Foreign Key Column(s)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

2. For each of the relational tables in Table 4-1, list the attributes based on the E-R diagram. Indicate any attributes that violate the first normal form.
Figure 4-34

Lucky Rent-a-Car E-R diagram.
Non-loss Decomposition
Figure 4-35 is the E-R diagram for General Hardware Company.

Figure 4-35

General Hardware Company E-R diagram.
Figure 4-36

<table>
<thead>
<tr>
<th>Salesperson Number</th>
<th>Customer Number</th>
<th>Employee Number</th>
<th>Product Number</th>
<th>Office Number</th>
<th>Salesperson Name</th>
<th>Commission Percentage</th>
<th>Year of Hire</th>
<th>Department Number</th>
<th>Manager Name</th>
<th>Customer Name</th>
<th>HQ City</th>
<th>Employee Name</th>
<th>Title</th>
<th>Product Name</th>
<th>Unit Price</th>
<th>Quantity</th>
<th>Telephone</th>
<th>Size</th>
</tr>
</thead>
</table>

General Hardware Company attributes.

Figure 4-36 represents the attributes of the General Hardware Company taken to the first normal form. Continue the process and apply the second and third normal forms. Show the result below.