of relationships. The first concept is a foundation of database design, and the second will help you understand key considerations made during design.

NORMALIZATION

Normalization is the process of converting poorly structured tables into two or more well-structured tables. A table is such a simple construct that you may wonder how one could possibly be poorly structured. In truth, there are many ways that tables can be malformed—so many, in fact, that researchers have published hundreds of papers on this topic alone.

Consider the *Employee* table in Figure CE5-7. It lists employee names, hire dates, email addresses, and the name and number of the department in which the employee works. This table seems innocent enough. But consider what happens when the Accounting department changes its name to Accounting and Finance. Because department names are duplicated in this table, every row that has a value of "Accounting" must be changed to "Accounting and Finance."

Data Integrity Problems

Suppose the Accounting name change is correctly made in two rows, but not in the third. The result is shown in Figure CE5-7b. This table has what is called a **data integrity problem**: Two rows indicate that the name of Department 100 is Accounting and Finance, and another row indicates that the name of Department 100 is Accounting.

This problem is easy to spot in this small table. But consider a table in a large database that has more than 300,000 rows. Once a table that large develops serious data integrity problems, months of labor will be required to remove them.

Data integrity problems are serious. A table that has data integrity problems will produce incorrect and inconsistent information. Users will lose confidence in the information, and the system will develop a poor reputation. Information systems with poor reputations become heavy burdens to the organizations that use them.

Normalizing for Data Integrity

The data integrity problem can occur only if data are duplicated. Because of this, one easy way to eliminate the problem is to eliminate the duplicated data. We can do this by transforming the table design in Figure CE5-7a into two tables, as shown in Figure CE5-8. Here the name of the department is stored just once; therefore, no data inconsistencies can occur.

Figure CE5-7
A Poorly Designed *Employee* Table

Employee

Name	HireDate	Email	DeptNo	DeptName
Jones	Feb 1, 2010	Jones@ourcompany.com	100	Accounting
Smith	Dec 3, 2007	Smith@ourcompany.com	200	Marketing
Chau	March 7, 2007	Chau@ourcompany.com	100	Accounting
Greene	July 17, 2010	Greene@ourcompany.com	100	Accounting

a. Table Before Update

Employee

Name	HireDate	Email	DeptNo	DeptName
Jones	Feb 1, 2010	Jones@ourcompany.com	100	Accounting and Finance
Smith	Dec 3, 2007	Smith@ourcompany.com	200	Marketing
Chau	March 7, 2007	Chau@ourcompany.com	100	Accounting and Finance
Greene	July 17, 2010	Greene@ourcompany.com	100	Accounting

b. Table with Incomplete Update

Figure CE5-8
Two Normalized Tables

Employee

Name	HireDate	Email	DeptNo
Jones	Feb 1, 2010	Jones@ourcompany.com	100
Smith	Dec 3, 2011	Smith@ourcompany.com	200
Chau	March 7, 2007	Chau@ourcompany.com	100
Greene	July 17, 2010	Greene@ourcompany.com	100

Department

DeptNo	DeptName
100	Accounting
200	Marketing
300	Information Systems

Of course, to produce an employee report that includes the department name, the two tables in Figure CE5-8 will need to be joined back together. Because such joining of tables is common, DBMS products have been programmed to perform it efficiently, but it still requires work. From this example, you can see a trade-off in database design: Normalized tables eliminate data duplication, but they can be slower to process. Dealing with such trade-offs is an important consideration in database design.

The general goal of normalization is to construct tables such that every table has a *single* topic or theme. In good writing, every paragraph should have a single theme. This is true of databases as well; every table should have a single theme. The problem with the table design in Figure CE5-7 is that it has two independent themes: employees and departments. The way to correct the problem is to split the table into two tables, each with its own theme. In this case, we create an *Employee* table and a *Department* table, as shown in Figure CE5-8.

As mentioned, there are dozens of ways that tables can be poorly formed. Database practitioners classify tables into various **normal forms** according to the kinds of problems they have. Transforming a table into a normal form to remove duplicated data and other problems is called *normalizing* the table.[1] Thus, when you hear a database designer say, "Those tables are not normalized," she does not mean that the tables have irregular, not-normal data. Instead, she means that the tables have a format that could cause data integrity problems.

Summary of Normalization

As a future user of databases, you do not need to know the details of normalization. Instead, understand the general principle that every normalized (well-formed) table has one and only one theme. Further, tables that are not normalized are subject to data integrity problems.

Be aware, too, that normalization is just one criterion for evaluating database designs. Because normalized designs can be slower to process, database designers sometimes choose to accept non-normalized tables. The best design depends on the users' requirements.

REPRESENTING RELATIONSHIPS

Figure CE5-9 shows the steps involved in transforming a data model into a relational database design. First, the database designer creates a table for each entity. The identifier of the entity becomes the key of the table. Each attribute of the entity becomes a column of the table. Next, the resulting tables are normalized so that each table has a single theme. Once that has been done, the next step is to represent the relationship among those tables.

For example, consider the E-R diagram in Figure CE5-10a. The *Adviser* entity has a 1:N relationship to the *Student* entity. To create the database design, we construct a table for *Adviser* and a