INTRODUCTION

Over the past two decades, data have become strategic assets for most organizations. Databases store, manipulate, and retrieve data in nearly every type of organization, including business, health care, education, government, libraries, and many scientific fields. Individuals with various personal devices and employees using enterprise-wide distributed applications depend on database technology. Customers and other remote users access databases through diverse technologies, such as automated teller machines, Web browsers, smartphones, and intelligent living and office environments. Most Web-based applications depend on a database foundation.

Following this period of rapid growth, will the demand for databases and database technology level off? Very likely not! In the highly competitive environment of today, there is every indication that database technology will assume even greater importance. Managers seek to use knowledge derived from databases for competitive advantage. For example, detailed sales databases can be mined to determine customer buying patterns as a basis for advertising and marketing campaigns. Organizations embed procedures called *alerts* in databases to warn of unusual conditions, such as impending stock shortages or opportunities to sell additional products, and to trigger appropriate actions. Analytics in its various forms—including big data analytics—depends on databases and other data management technologies.

Although the future of databases is assured, much work remains to be done. Many organizations have a proliferation of incompatible databases that were developed to meet immediate needs rather than based on a planned strategy or a well-managed evolution. Enormous amounts of data are trapped in older, "legacy" systems, and the data are often of poor quality. New skills are required to design and manage data warehouses and other repositories of data and to fully leverage all the data that are being captured in the organization. There is a shortage of skills in areas such as database analysis, database design, database application development, and business analytics. You will learn about these and other important issues in this textbook to equip you for the jobs of the future.

A course in database management has emerged as one of the most important courses in the information systems curriculum today. Further, many schools have added additional elective courses in data warehousing, data mining, and other aspects of business analytics to provide in-depth coverage of these important topics. As information systems professionals, you must be prepared to analyze database requirements and design and to implement databases within the context of information systems development. You also must be prepared to consult with end users and show them how they can use databases (or data warehouses) to build decision models and systems for competitive advantage. The widespread use of databases attached to Web sites that return dynamic information to users of these sites requires that you understand not only how to link databases to the Web-based applications but also how to secure those databases so that their contents can be viewed but not compromised by outside users.

In this chapter, you will learn about the basic concepts of databases and database management systems (DBMSs). You will review traditional file management systems and some of their shortcomings that led to the database approach. Next, you will consider the benefits, costs, and risks of using the database approach. We review the range of technologies used to build, use, and manage databases; describe the types of applications that use databases (personal, multi-tier, and enterprise); and describe how databases have evolved over the past five decades. The chapter also presents a framework that will help you understand traditional and emerging data management approaches and technologies in a joint context.

Because a database is one part of an information system, this chapter also examines how the database development process fits into the overall information systems development process. The chapter emphasizes the need to coordinate database development with all the other activities in the development of a complete

information system. It includes highlights from a hypothetical database development process at Pine Valley Furniture Company. Using this example, the chapter introduces tools for developing databases on personal computers and the process of extracting data from enterprise databases for use in stand-alone applications.

There are several reasons for discussing database development at this point. First, although you may have used the basic capabilities of a database management system, such as Microsoft Access, you may not yet have developed an understanding of how these databases were developed. Using simple examples, this chapter briefly illustrates what you will be able to do after you complete a database course using this text. Thus, this chapter helps you develop a vision and context for each topic developed in detail in subsequent chapters.

Second, many students learn best from a text full of concrete examples. Although all of the chapters in this text contain numerous examples, illustrations, and actual database designs and code, each chapter concentrates on a specific aspect of database management. This chapter will help you understand, with minimal technical details, how all of these individual aspects of database management are related and how database development tasks and skills relate to what you are learning in other information systems courses.

Finally, many instructors want you to begin the initial steps of a database development group or individual project early in your database course. This chapter gives you an idea of how to structure a database development project sufficient to begin a course exercise. Obviously, because this is only the first chapter, many of the examples and notations you will encounter in this chapter are much simpler than those required for your project, for other course assignments, or in a real organization.

One note of caution: You will not learn how to design or develop databases just from this chapter. Sorry! You will discover that the content of this chapter is introductory and simplified. Many of the notations used in this chapter are not exactly like the ones you will learn in subsequent chapters. Our purpose in this chapter is to give you a general understanding of the key steps and types of skills, not to teach you specific techniques. You will, however, learn fundamental concepts and definitions and develop an intuition and motivation for the skills and knowledge presented in later chapters.

BASIC CONCEPTS AND DEFINITIONS

Database
An organized collection of logically related data.

A **database** is an organized collection of logically related data. Not many words in the definition, but have you looked at the size of this book? There is a lot to do to fulfill this definition.

A database may be of any size and complexity. For example, a salesperson may maintain a small database of customer contacts—consisting of a few megabytes of data—on her laptop computer. A large corporation may build a large database consisting of several terabytes of data (a *terabyte* is a trillion bytes) on a large mainframe computer that is used for decision support applications. Very large data warehouses contain more than a petabyte of data. (A *petabyte* is a quadrillion bytes.) The assumption throughout the text is that all databases are computer based.

Data

Historically, the term *data* referred to facts concerning objects and events that could be recorded and stored on computer media. For example, in a salesperson's database, the data would include facts such as customer name, address, and telephone number. This type of data is called *structured* data. The most important structured data types are numeric, character, and dates. Structured data are stored in tabular form (in tables, relations, arrays, spreadsheets, and so forth) and are most commonly found in traditional databases and data warehouses.

The traditional definition of data now needs to be expanded to reflect a new reality: Databases today are used to store objects such as documents, e-mails, tweets, Facebook

posts, GPS information, maps, photographic images, sound, and video segments in addition to structured data. For example, the salesperson's database might include a photo image of the customer contact. It might also include a sound recording or video clip about the most recent product. This type of data is referred to as *unstructured* data, or as multimedia data. Today, structured and unstructured data are often combined in the same database to create a true multimedia environment. For example, an automobile repair shop can combine structured data (describing customers and automobiles) with multimedia data (photo images of the damaged autos and scanned images of insurance claim forms). One of the defining elements of "big data" technologies, which will also be covered later in this book, is that they provide capabilities to deal with highly heterogeneous data.

An expanded definition of **data** that includes structured and unstructured types is "a stored representation of objects and events that have meaning and importance in the user's environment."

Data
Stored representations of objects and events that have meaning and importance in the user's environment.

Data versus Information

The terms *data* and *information* are closely related and in fact are often used interchangeably. However, it is useful to distinguish between data and information. **Information** is data that have been processed in such a way that the knowledge of the person who uses the data is increased. For example, consider the following list of facts:

Information
Data that have been processed in such a way as to increase the knowledge of the person who uses the data.

Baker, Kenneth D.	324917628
Doyle, Joan E.	476193248
Finkle, Clive R.	548429344
Lewis, John C.	551742186
McFerran, Debra R.	409723145

These facts satisfy our definition of data, but most people would agree that the data are useless in their present form. Even if you guessed that this is a list of people's names paired with their Social Security numbers, the data remain useless because you would have no idea what the entries mean. Notice what happens when you see the same data in a context, as shown in Figure 1-1a.

By adding a few additional data items and providing some structure, you are able to recognize a class roster for a particular course. This is useful information to some users, such as the course instructor and the registrar's office. Of course, as general awareness of the importance of strong data security has increased, few organizations use Social Security numbers as identifiers any longer. Instead, most organizations use an internally generated number for identification purposes.

Another way to convert data into information is to summarize them or otherwise process and present them for human interpretation. For example, Figure 1-1b shows

Class Roster

Course: MGT 500 Business Policy Semester: Spring 2018

Section: 2

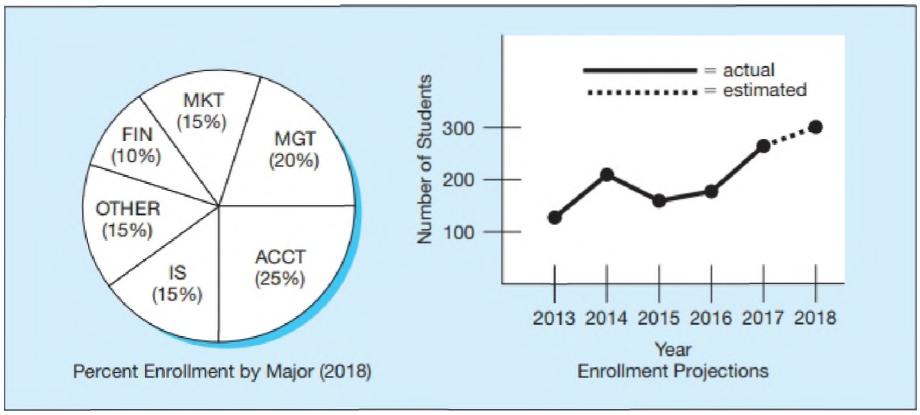
Name	ID	Major	GPA
Baker, Kenneth D.	324917628	MGT	2.9
Doyle, Joan E.	476193248	MKT	3.4
Finkle, Clive R.	548429344	PRM	2.8
Lewis, John C.	551742186	MGT	3.7
McFerran, Debra R.	409723145	IS	2.9
Sisneros, Michael	392416582	ACCT	3.3

FIGURE 1-1 Converting data to information

(a) Data in context

FIGURE 1-1 *(continued)*

(b) Summarized data



summarized student enrollment data presented as graphical information. This information could be used as a basis for deciding whether to add new courses or to hire new faculty members.

In practice, according to our definitions, databases today may contain data, information, or both. For example, a database may contain an image of the class roster document shown in Figure 1-1a. Also, data are often preprocessed and stored in summarized form in databases that are used for decision support. Throughout this text, we use the term *database* without distinguishing its contents as data or information.

Metadata

Metadata

Data that describe the properties or characteristics of end-user data and the context of those data.

As discussed earlier, data become useful only when placed in some context. The primary mechanism for providing context for data is metadata. **Metadata** are data that describe the properties or characteristics of end-user data and the context of that data. Some of the properties that are typically described include data names, definitions, length (or size), and allowable values. Metadata describing data context include the source of the data, where the data are stored, ownership (or stewardship), and usage. Although it may seem circular, many people think of metadata as "data about data."

Some sample metadata for the Class Roster (Figure 1-1a) are listed in Table 1-1. For each data item that appears in the Class Roster, the metadata show the data item name, the data type, length, minimum and maximum allowable values (where appropriate), a brief description of each data item, and the source of the data (sometimes called the *system of record*). Notice the distinction between data and metadata. Metadata are once removed from data. That is, metadata describe the properties of data but are separate from that data. Thus, the metadata shown in Table 1-1 do not include any sample data from the Class Roster of Figure 1-1a. Metadata enable database designers and users to understand what data exist, what the data mean, and how to distinguish between data items

TABLE 1-1 Example Metadata for Class Roster

Data Item			Metadata			
Name	Type	Length	Min	Max	Description	Source
Course	Alphanumeric	30			Course ID and name	Academic Unit
Section	Integer	1	1	9	Section number	Registrar
Semester	Alphanumeric	10			Semester and year	Registrar
Name	Alphanumeric	30			Student name	Student IS
ID	Integer	9			Student ID (SSN)	Student IS
Major	Alphanumeric	4			Student major	Student IS
GPA	Decimal	3	0.0	4.0	Student grade point average	Academic Unit

that at first glance look similar. Managing metadata is at least as crucial as managing the associated data because data without clear meaning can be confusing, misinterpreted, or erroneous. Typically, much of the metadata are stored as part of the database and may be retrieved using the same approaches that are used to retrieve data or information.

Data can be stored in files (think Excel sheets) or in databases. In the following sections, you will learn about the progression from file processing systems to databases and the advantages and disadvantages of each.

TRADITIONAL FILE PROCESSING SYSTEMS

When computer-based data processing was first available, there were no databases. To be useful for business applications, computers had to store, manipulate, and retrieve large files of data. Computer file processing systems were developed for this purpose. Although these systems have evolved over time, their basic structure and purpose have changed little over several decades.

As business applications became more complex, it became evident that traditional file processing systems had a number of shortcomings and limitations (described next). As a result, these systems have been replaced by database processing systems in most business applications today. Nevertheless, you should have at least some familiarity with file processing systems since understanding the problems and limitations inherent in file processing systems can help you avoid these same problems when designing database systems. It should be noted that Excel files, in general, fall into the same category as file systems and suffer from the same drawbacks listed below. Informal use of Excel for management of data is believed to continue to be quite widespread, although valid research results regarding this are difficult to find.

File Processing Systems at Pine Valley Furniture Company

Early computer applications at Pine Valley Furniture used the traditional file processing approach. This approach to information systems design met the data processing needs of individual departments rather than the overall information needs of the organization. The information systems group typically responded to users' requests for new systems by developing (or acquiring) new computer programs for individual applications, such as inventory control, accounts receivable, or human resource management. No overall map, plan, or model guided application growth.

Three of the computer applications based on the file processing approach are shown in Figure 1-2. The systems illustrated are Order Filling, Invoicing, and Payroll.

FIGURE 1-2 Old file processing systems at Pine Valley Furniture Company

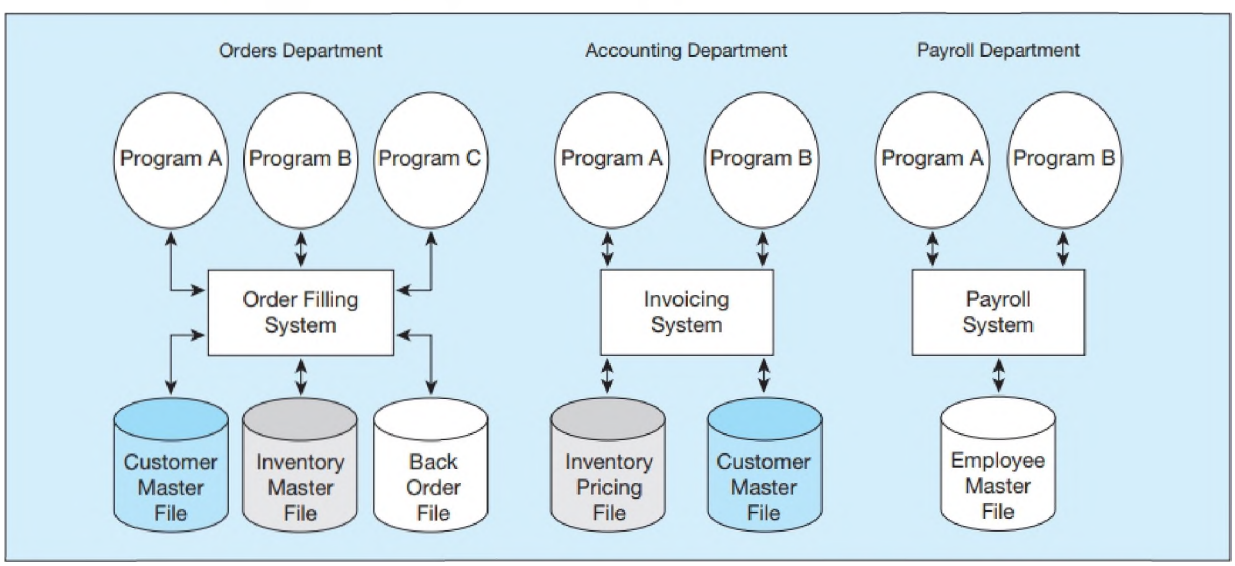


TABLE 1-2 Disadvantages of File Processing Systems
Program-data dependence
Duplication of data
Limited data sharing
Lengthy development times
Excessive program maintenance

The figure also shows the major data files associated with each application. A *file* is a collection of related records. For example, the Order Filling System has three files: Customer Master, Inventory Master, and Back Order. Notice that there is duplication of some of the files used by the three applications, which is typical of file processing systems.

Disadvantages of File Processing Systems

Several disadvantages associated with conventional file processing systems are listed in Table 1-2 and described briefly next. It is important to understand these issues because if you do not follow the database management practices described in this book, some of these disadvantages can also become issues for databases as well.

Database application
An application program (or set of related programs) that is used to perform a series of database activities (create, read, update, and delete) on behalf of database users.

PROGRAM-DATA DEPENDENCE File descriptions are stored within each **database application** program that accesses a given file. For example, in the Invoicing System in Figure 1-2, Program A accesses the Inventory Pricing File and the Customer Master File. Because the program contains a detailed file description for these files, any change to a file structure requires changes to the file descriptions for all programs that access the file.

Notice in Figure 1-2 that the Customer Master File is used in the Order Filling System and the Invoicing System. Suppose it is decided to change the customer address field length in the records in this file from 30 to 40 characters. The file descriptions in each program that is affected (up to five programs) would have to be modified. It is often difficult even to locate all programs affected by such changes. Worse, errors are often introduced when making such changes.

DUPLICATION OF DATA Because applications are often developed independently in file processing systems, unplanned duplicate data files are the rule rather than the exception. For example, in Figure 1-2, the Order Filling System contains an Inventory Master File, whereas the Invoicing System contains an Inventory Pricing File. These files contain data describing Pine Valley Furniture Company's products, such as product description, unit price, and quantity on hand. This duplication is wasteful because it requires additional storage space and increased effort to keep all files up to date. Data formats may be inconsistent, data values may not agree, or both. Reliable metadata are very difficult to establish in file processing systems. For example, the same data item may have different names in different files, or, conversely, the same name may be used for different data items in different files.

LIMITED DATA SHARING With the traditional file processing approach, each application has its own private files, and users have little opportunity to share data outside their own applications. Notice in Figure 1-2, for example, that users in the Accounting Department have access to the Invoicing System and its files, but they probably do not have access to the Order Filling System or to the Payroll System and their files. Managers often find that a requested report requires a major programming effort because data must be drawn from several incompatible files in separate systems. When different organizational units own these different files, additional management barriers must be overcome.

LENGTHY DEVELOPMENT TIMES With traditional file processing systems, each new application requires that the developer essentially start from scratch by designing

new file formats and descriptions and then writing the file access logic for each new program. The lengthy development times required are inconsistent with today's fast-paced business environment, in which time to market (or time to production for an information system) is a key business success factor.

EXCESSIVE PROGRAM MAINTENANCE The preceding factors all combined to create a heavy program maintenance load in organizations that relied on traditional file processing systems. In fact, as much as 80 percent of the total information system's development budget might be devoted to program maintenance in such organizations. This, in turn, means that resources (time, people, and money) are not being spent on developing new applications.

As discussed above, these disadvantages are true also in situations when individuals and organizational units maintain important organizational data in Excel spreadsheets. Further, it is important to note that many of the disadvantages of file processing you have learned about can also be limitations of databases if an organization does not properly apply the database approach. For example, if an organization develops many separately managed databases (say, one for each division or business function) with little or no coordination of the metadata, uncontrolled data duplication, limited data sharing, lengthy development time, and excessive program maintenance can occur. Thus, the database approach, which is explained in the next section, is as much a way to manage organizational data as it is a set of technologies for defining, creating, maintaining, and using these data.

THE DATABASE APPROACH

So, how do you overcome the flaws of file processing? No, you do not call Ghostbusters, but you can do something better: You should follow the database approach. You will first learn some core concepts that are fundamental in understanding the database approach to managing data. You will then discover how the database approach can overcome the limitations of the file processing approach.

Data Models

Designing a database properly is fundamental to establishing a database that meets the needs of the users. **Data models** capture the nature of and relationships among data and are used at different levels of abstraction as a database is conceptualized and designed. The effectiveness and efficiency of a database is directly associated with the structure of the database. Various graphical systems exist that convey this structure and are used to produce data models that can be understood by end users, systems analysts, and database designers. Chapters 2 and 3 are devoted to developing your understanding of data modeling, as is Chapter 14, on the book's Web site, which addresses a different approach using object-oriented data modeling. A typical data model is made up of entities, attributes, and relationships, and the most common data modeling representation is the entity-relationship model. A brief description is presented next. More details will be forthcoming in Chapters 2 and 3.

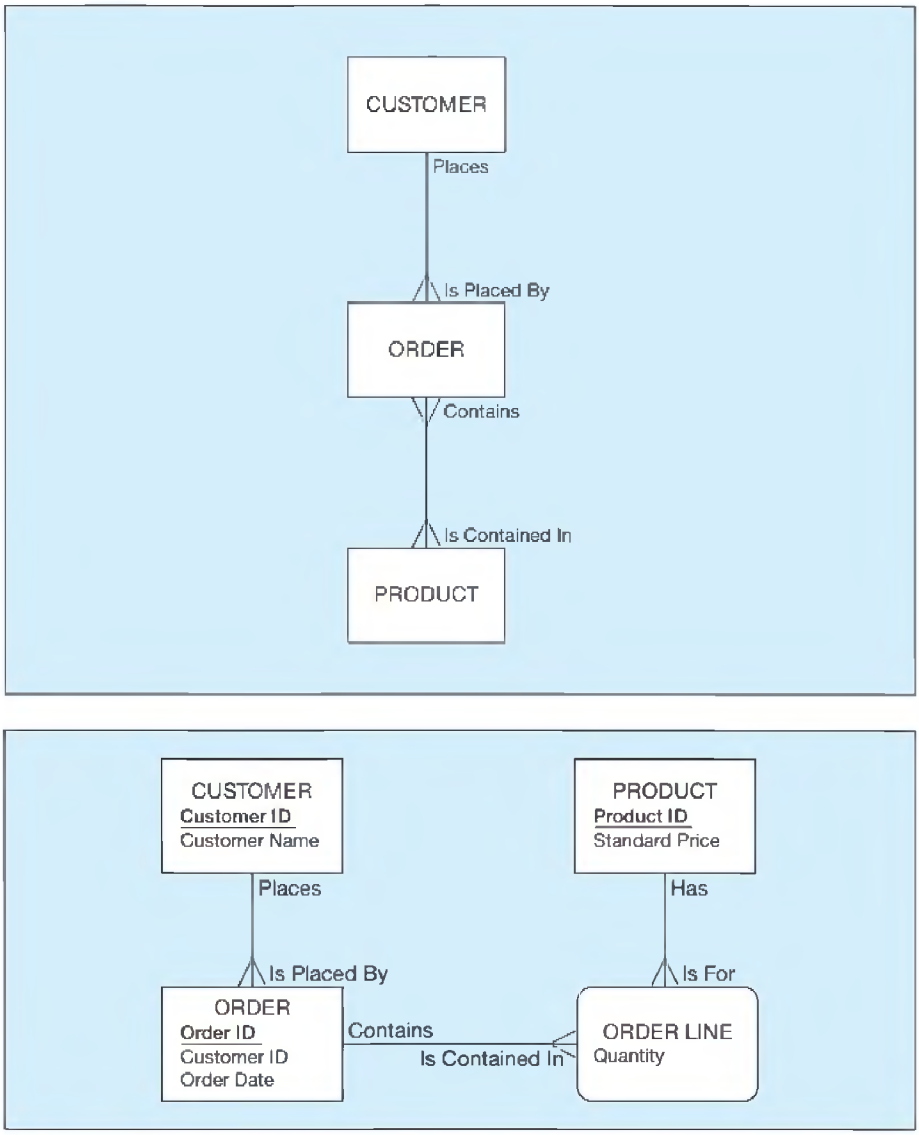
Data model
Graphical systems used to capture the nature and relationships among data.

ENTITIES Customers and orders are objects about which a business maintains information. They are referred to as "entities." An **entity** is like a noun in that it describes a person, a place, an object, an event, or a concept in the business environment for which information must be recorded and retained. CUSTOMER and ORDER are entities in Figure 1-3a. The data you are interested in capturing about the entity (e.g., Customer Name) is called an *attribute*. Data are recorded for many customers. Each customer's information is referred to as an *instance* of CUSTOMER.

Entity
A person, a place, an object, an event, or a concept in the user environment about which the organization wishes to maintain data.

RELATIONSHIPS A well-structured database establishes the *relationships* between entities that exist in organizational data so that desired information can be retrieved. Most relationships are one-to-many (1:*M*) or many-to-many (*M*:*N*). A customer can place (the Places relationship) more than one order with a company. However, each

FIGURE 1-3 Comparison of enterprise- and project-level data models



(a) Segment of an enterprise data model

(b) Segment of a project data model

order is usually associated with (the Is Placed By relationship) a particular customer. Figure 1-3a shows the 1:*M* relationship of customers who may place one or more orders; the 1:*M* nature of the relationship is marked by the crow's foot attached to the rectangle (entity) labeled ORDER. This relationship appears to be the same in Figures 1-3a and 1-3b. However, the relationship between orders and products is *M*:*N*. An order may be for one or more products, and a product may be included on more than one order. It is worthwhile noting that Figure 1-3a is an enterprise-level model, where it is necessary to include only the higher-level relationships of customers, orders, and products. The project-level diagram shown in Figure 1-3b includes additional levels of details, such as the further details of an order.

Relational database

A database that represents data as a collection of tables in which all data relationships are represented by common values in related tables.

Relational Databases

Relational databases establish the relationships between entities by means of common fields included in a file, called a *relation*. The relationship between a customer and the customer's order depicted in the data models in Figure 1-3 is established by including the customer's number with the customer's order. Thus, a customer's identification number

is included in the file (or relation) that holds customer information such as name, address, and so forth. Every time the customer places an order, the customer identification number is also included in the relation that holds order information. Relational databases use the identification number to establish the relationship between customer and order.

Database Management Systems

A **database management system (DBMS)** is a software system that enables the use of a database approach. The primary purpose of a DBMS is to provide a systematic method of creating, updating, storing, and retrieving the data stored in a database. It enables end users and application programmers to share data, and it enables data to be shared among multiple applications rather than propagated and stored in new files for every new application (Mullins, 2002). A DBMS also provides facilities for controlling data access, enforcing data integrity, managing concurrency control, and restoring a database. You will learn about these DBMS features in detail in Chapters 7 and 8.

Database management system (DBMS)
A software system that is used to create, maintain, and provide controlled access to user databases.

Now that you understand the basic elements of a database approach, you are in a good position to try to understand the differences between a database approach and a file-based approach. Let us begin by comparing Figures 1-2 and 1-4. Figure 1-4 depicts a representation (entities) of how the data can be considered to be stored in the database. Notice that unlike Figure 1-2, in Figure 1-4, there is only one place where the CUSTOMER information is stored rather than the two Customer Master Files. Both the Order Filling System and the Invoicing System will access the data contained in the single CUSTOMER entity. Further, what CUSTOMER information is stored, how it is stored, and how it is accessed are likely not closely tied to either of the two systems. All of this enables you to achieve the advantages listed in the next section. Of course, it is important to note that a real-life database will likely include thousands of entities and relationships among them.

Advantages of the Database Approach

The primary advantages of a database approach, enabled by DBMSs, are summarized in Table 1-3 and described next.

PROGRAM-DATA INDEPENDENCE The separation of data descriptions (metadata) from the application programs that use the data is called **data independence.** With the database approach, data descriptions are stored in a central location called the *repository*.

Data independence
The separation of data descriptions from the application programs that use the data.

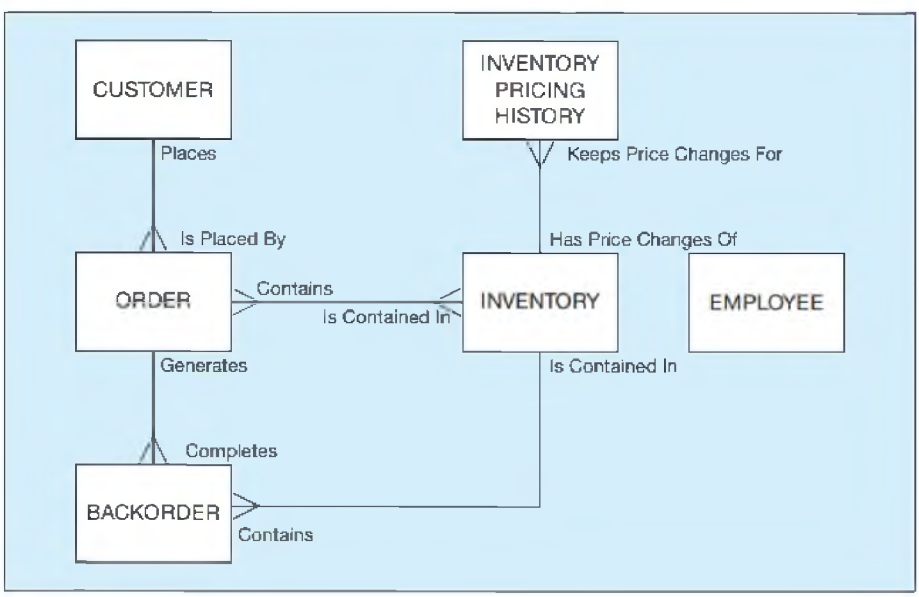


FIGURE 1-4 Enterprise model for Figure 1-3 segments

TABLE 1-3 Advantages of the Database Approach

Program-data independence
Planned data redundancy
Improved data consistency
Improved data sharing
Increased productivity of application development
Enforcement of standards
Improved data quality
Improved data accessibility and responsiveness
Reduced program maintenance
Improved decision support

This property of database systems allows an organization's data to change and evolve (within limits) without changing the application programs that process the data.

PLANNED DATA REDUNDANCY Good database design attempts to integrate previously separate (and redundant) data files into a single, logical structure. Ideally, each primary fact is recorded in only one place in the database. For example, facts about a product, such as the Pine Valley oak computer desk, its finish, price, and so forth, are recorded together in one place in the Product table, which contains data about each of Pine Valley's products. The database approach does not eliminate redundancy entirely, but it enables the designer to control the type and amount of redundancy. At other times, it may be desirable to include some limited redundancy to improve database performance, as you will see in later chapters.

IMPROVED DATA CONSISTENCY By eliminating or controlling data redundancy, you can greatly reduce the opportunities for inconsistency. For example, if a customer's address is stored only once, we cannot disagree about the customer's address. When the customer's address changes, recording the new address is greatly simplified because the address is stored in a single place. Finally, you avoid wasting storage space that results from redundant data storage.

IMPROVED DATA SHARING A database is designed as a shared corporate resource. Authorized internal and external users are granted permission to use the database, and each user (or group of users) is provided one or more user views into the database to facilitate this use. A **user view** is a logical description of some portion of the database that is required by a user to perform some task. A user view is often developed by identifying a form or report that the user needs on a regular basis. For example, an employee working in human resources will need access to confidential employee data; a customer needs access to the product catalog available on Pine Valley's Web site. The views for the human resources employee and the customer are drawn from completely different areas of one unified database.

User view

A logical description of some portion of the database that is required by a user to perform some task.

INCREASED PRODUCTIVITY OF APPLICATION DEVELOPMENT A major advantage of the database approach is that it greatly reduces the cost and time for developing new business applications. There are three important reasons that database applications can often be developed much more rapidly than conventional file applications:

1. Assuming that the database and the related data capture and maintenance applications have already been designed and implemented, the application developer can concentrate on the specific functions required for the new application without having to worry about file design or low-level implementation details.
2. The database management system provides a number of high-level productivity tools, such as forms and report generators, and high-level languages that automate

some of the activities of database design and implementation. You will learn about many of these tools in subsequent chapters.

3. Significant improvement in application developer productivity, estimated to be as high as 60 percent (Long, 2005), is currently being realized through the use of Web services based on the use of standard Internet protocols and a universally accepted data format (XML).

ENFORCEMENT OF STANDARDS When the database approach is implemented with full management support, the database administration function should be granted single-point authority and responsibility for establishing and enforcing data standards. These standards will include naming conventions, data quality standards, and uniform procedures for accessing, updating, and protecting data. The data repository provides database administrators with a powerful set of tools for developing and enforcing these standards. Unfortunately, the failure to implement a strong database administration function is perhaps the most common source of database failures in organizations. You will learn about the database administration (and related data administration) functions in Chapter 12.

IMPROVED DATA QUALITY Concern with poor quality data is a common theme in strategic planning and database administration today. In 2011 alone, poor data quality is estimated to have cost the U.S. economy almost $3 trillion, almost twice the size of the federal deficit (**http://hollistibbetts.sys-con.com/node/1975126**). The database approach provides a number of tools and processes to improve data quality. Two of the more important are the following:

1. Database designers can specify integrity constraints that are enforced by the DBMS. A **constraint** is a rule that cannot be violated by database users. We describe numerous types of constraints (also called "business rules") in Chapters 2 and 3. If a customer places an order, the constraint that ensures that the customer and the order remain associated is called a "relational integrity constraint," and it prevents an order from being entered without specifying who placed the order.
2. One of the objectives of a data warehouse environment is to clean up (or "scrub") operational data before they are placed in the data warehouse (Jordan, 1996). Do you ever receive multiple copies of a catalog? The company that sends you three copies of each of its mailings could recognize significant postage and printing savings if its data were scrubbed, and its understanding of its customers would also be enhanced if it could determine a more accurate count of existing customers. You will learn about data warehouses and data integration in Chapter 9 and the potential for improving data quality in Chapter 12.

Constraint
A rule that cannot be violated by database users.

IMPROVED DATA ACCESSIBILITY AND RESPONSIVENESS With a relational database, end users without programming experience can often retrieve and display data, even when they cross traditional departmental boundaries. For example, an employee can display information about computer desks at Pine Valley Furniture Company with the following query:

```
SELECT *
FROM Product_T
WHERE ProductDescription = "Computer Desk";
```

The language used in this query is called Structured Query Language, or SQL. (You will study this language in detail in Chapters 5 and 6.) Although the queries constructed can be *much* more complex, the basic structure of the query is easy for even novice, nonprogrammers to grasp. If they understand the structure and names of the data that fit within their view of the database, they soon gain the ability to retrieve answers to new questions without having to rely on a professional application developer. This can be dangerous; queries should be thoroughly tested to be sure they are returning accurate data before relying on their results, and novices may not understand that challenge.

REDUCED PROGRAM MAINTENANCE Stored data must be changed frequently for a variety of reasons: New data item types are added, data formats are changed, and so forth. A celebrated example of this problem was the well-known "year 2000" problem, in which common two-digit year fields were extended to four digits to accommodate the rollover from the year 1999 to the year 2000.

In a file processing environment, the data descriptions and the logic for accessing data are built into individual application programs (this is the program-data dependence issue described earlier). As a result, changes to data formats and access methods inevitably result in the need to modify application programs. In a database environment, data are more independent of the application programs that use them. Within limits, you can change either the data or the application programs that use the data without necessitating a change in the other factor. As a result, program maintenance can be significantly reduced in a modern database environment.

IMPROVED DECISION SUPPORT Some databases are designed expressly for decision support applications. For example, some databases are designed to support customer relationship management, whereas others are designed to support financial analysis or supply chain management. You will study how databases are tailored for different decision support applications and analytical styles in Chapters 9 through 11.

Cautions about Database Benefits

The previous section identified 10 major potential benefits of the database approach. However, we must caution you that many organizations have been frustrated in attempting to realize some of these benefits. For example, the goal of data independence (and, therefore, reduced program maintenance) has proven elusive due to the limitations of older data models and database management software. Fortunately, the relational model and the newer object-oriented model provide a significantly better environment for achieving these benefits. Another reason for failure to achieve the intended benefits is poor organizational planning and database implementation; even the best data management software cannot overcome such deficiencies. For this reason, you will learn about the importance of database planning and design throughout this text.

Costs and Risks of the Database Approach

A database is not a silver bullet, and it does not have the magic power of Harry Potter. As with any other business decision, the database approach entails some additional costs and risks that must be recognized and managed when it is implemented (see Table 1-4).

NEW, SPECIALIZED PERSONNEL Frequently, organizations that adopt the database approach need to hire or train individuals to design and implement databases, provide database administration services, and manage a staff of new people. Further, because of the rapid changes in technology, these new people will have to be retrained or upgraded on a regular basis. This personnel increase may be more than offset by other productivity gains, but an organization should recognize the need for these specialized skills, which are required to obtain the most from the potential benefits. You will learn about the staff requirements for database management in Chapter 12.

TABLE 1-4 Costs and Risks of the Database Approach

New, specialized personnel
Installation and management cost and complexity
Conversion costs
Need for explicit backup and recovery
Organizational conflict

INSTALLATION AND MANAGEMENT COST AND COMPLEXITY A multi-user database management system is a large and complex suite of software that has a high initial cost, requires a staff of trained personnel to install and operate, and has substantial annual maintenance and support costs. Installing such a system may also require upgrades to the hardware and data communications systems in the organization. Substantial training is normally required on an ongoing basis to keep up with new releases and upgrades. Additional or more sophisticated and costly database software may be needed to provide security and to ensure proper concurrent updating of shared data.

CONVERSION COSTS The term *legacy system* is widely used to refer to older applications in an organization that are based on file processing and/or older database technology. The cost of converting these older systems to modern database technology—measured in terms of dollars, time, and organizational commitment—may often seem prohibitive to an organization. The use of data warehouses is one strategy for continuing to use older systems while at the same time exploiting modern database technology and techniques (Ritter, 1999).

NEED FOR EXPLICIT BACKUP AND RECOVERY A shared corporate database must be accurate and available at all times. This requires that comprehensive procedures be developed and used for providing backup copies of data and for restoring a database when damage occurs. These considerations have acquired increased urgency in today's security-conscious environment. A modern database management system normally automates many more of the backup and recovery tasks than a file system. You will learn about procedures for security, backup, and recovery in Chapter 8.

ORGANIZATIONAL CONFLICT A shared database requires a consensus on data definitions and ownership as well as responsibilities for accurate data maintenance. Experience has shown that conflicts on data definitions, data formats and coding, rights to update shared data, and associated issues are frequent and often difficult to resolve. Handling these issues requires organizational commitment to the database approach, organizationally astute database administrators, and a sound evolutionary approach to database development.

If strong top management support of and commitment to the database approach are lacking, end-user development of stand-alone databases is likely to proliferate. These databases do not follow the general database approach that we have described, and they are unlikely to provide the benefits described earlier. In the extreme, they may lead to a pattern of inferior decision making that threatens the well-being or existence of an organization.

INTEGRATED DATA MANAGEMENT FRAMEWORK

The database approach described above is associated with relational database technologies and used primarily as a foundation for the design and implementation of transaction processing systems (*operational systems*). Data management technologies are also increasingly often used as *informational systems*, as a foundation for analytics, or the systematic analysis and interpretation of data to improve our understanding of a real-world domain. Transactional systems are still the core of this book with a focus on relational databases and the SQL language. These technologies continue, in practice, to be a fundamental source of data for all areas of data management, and no other technology is as widely used. They form the foundation on which business activities of modern organizations are built because they enable the way in which organizations interact and do business with their stakeholders.

This book does, however, also cover data management technologies intended primarily for enabling and supporting analytics. They can be divided into two major categories: data warehousing and big data. Data warehousing has existed as a concept since late 1980s, and, as you will learn in Chapter 9, both conceptual approaches and implementation technologies for data warehousing are already well developed and

FIGURE 1-5 Integrated data management framework

<table>
<tr><th></th><th>Operational</th><th colspan="2">Informational</th></tr>
<tr><td></td><td>Transactional</td><td>Analytical–
Data Warehousing</td><td>Analytical–
Big Data</td></tr>
<tr><td>Technology</td><td>Relational</td><td>Relational</td><td>Non-relational</td></tr>
<tr><td>Modeling</td><td>Conceptual data modeling with (E)ER (Chapters 2 and 3)</td><td rowspan="4">Data warehousing and data integration (Chapter 9)</td><td rowspan="4">Big data technologies, including Hadoop & NoSQL (Chapter 10)</td></tr>
<tr><td>Design</td><td>Logical data modeling with the relational model; Normalization (Chapter 4)</td></tr>
<tr><td>Infrastructure</td><td>Physical design of relational databases; Security; Cloud computing (Chapter 8)</td></tr>
<tr><td>Access</td><td>SQL (Chapters 5 and 6)

Applications with SQL (Chapter 7)</td></tr>
<tr><td>Data analysis</td><td colspan="3">Analytics and its implications (Chapter 11)</td></tr>
<tr><td>Governance and data management</td><td colspan="3">Lifecycle (Chapter 1)
Governance, data quality, and master data management (Chapter 12)</td></tr>
</table>

mature. Indeed, most traditional data warehouses are implemented using the same relational technologies as transactional systems. Big data technologies have emerged as another category of informational systems since the early 2010s. They are characterized by their ability to deal with large *volumes* of data with a *variety* of data types arriving to the organizational systems with high *velocity* (the so-called three Vs of big data). A major difference between big data systems compared to both data warehousing and operational, transaction-focused systems is that structures of the latter are typically expected to be carefully designed before data are stored in them ("schema on write"), whereas many of the big data analytics technologies are intended to be used in the "schema on read" mode. In the latter approach, the structure of the data and the relationships between the data elements will be determined later, either right before or at the time of the use of the data.

Figure 1-5 presents a framework that illustrates the structure and the contents of this book based three key categories: Transactional (an operational category), Analytical–Data Warehousing, and Analytical–Big Data (informational categories). As the framework demonstrates, the book explores data management in the operational context at a more detailed level than the informational one, dedicating Chapters 2 through 8 to transactional systems and separating the coverage of modeling, design, access, and infrastructure into different chapters. The framework also shows the following:

- This book recognizes the growing importance of informational uses of data management processes and technologies and presents them in the same broader context with operational uses. You will learn about the informational use of data management from two different analytical perspectives: data warehousing (Chapter 9) and big data (Chapter 10). Both of these chapters deal with questions regarding modeling, design, access, and infrastructure in an integrated way.
- In many areas, such as analytics (Chapter 11), and data management, governance, and quality (Chapter 12), the concerns and key questions are shared between the operational and informational perspectives.

COMPONENTS OF THE DATABASE ENVIRONMENT

Now that you have seen the advantages and risks of using the database approach to managing data, let us examine the major components of a typical database environment and their relationships (see Figure 1-6). You have already been introduced to some (but not all) of these components in previous sections. Following is a brief description of the nine components shown in Figure 1-6:

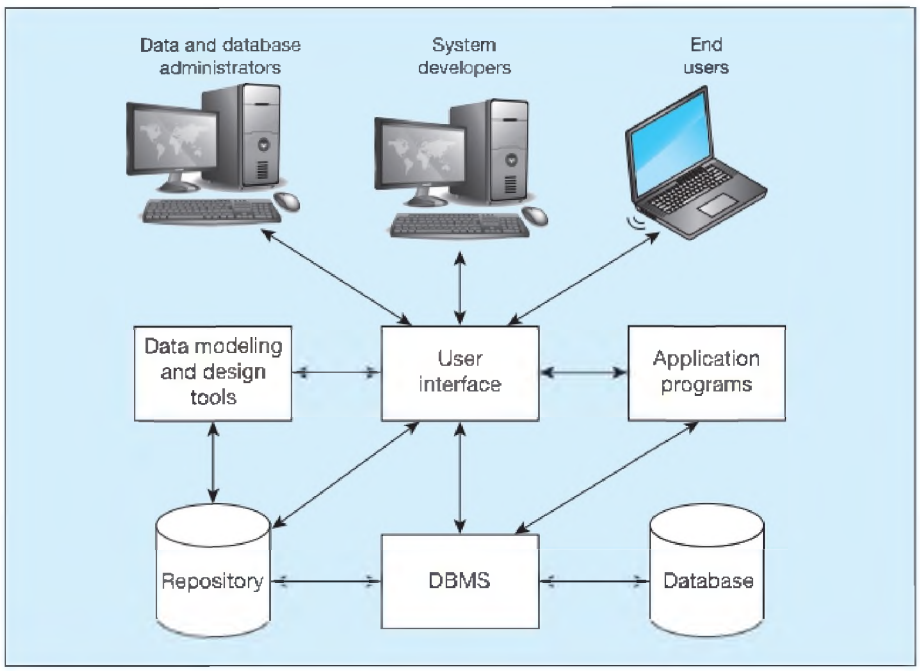


FIGURE 1-6 Components of the database environment

1. ***Data modeling and design tools*** **Data modeling and design tools** are automated tools used to design databases and application programs. These tools help with creation of data models and in some cases can also help automatically generate the "code" needed to create the database. You will learn more about the use of automated tools for database design and development throughout the text, particularly in Chapters 4 and 8.
2. ***Repository*** A **repository** is a centralized knowledge base for all data definitions, data relationships, screen and report formats, and other system components. A repository contains an extended set of metadata important for managing databases as well as other components of an information system. We describe the repository in Chapter 9.
3. ***DBMS*** A DBMS is a software system that is used to create, maintain, and provide controlled access to user databases. You will learn about many of the technical functions of a DBMS in Chapter 8.
4. ***Database*** A database is an organized collection of logically related data, usually designed to meet the information needs of multiple users in an organization. It is important to distinguish between the database and the repository. The repository contains definitions of data, whereas the database contains occurrences of data. You will explore the activities of database design and implementation in Chapters 4 through 8.
5. ***Application programs*** Computer-based application programs are used to create and maintain the database and provide information to users. Key database-related application programming skills are described in Chapters 5 through 10.
6. ***User interface*** The user interface includes languages, menus, and other facilities by which users interact with various system components, such as data modeling and design tools, application programs, the DBMS, and the repository. User interfaces are illustrated throughout this text, with a particular focus in Chapters 5, 6, 8, and 11.
7. ***Data and database administrators*** Data administrators are persons who are responsible for the overall management of data resources in an organization. Database administrators are responsible for physical database design and for managing technical issues in the database environment. You will learn about these functions in detail in Chapters 8 and 12.

Data modeling and design tools
Software tools that provide automated support for creating data models.

Repository
A centralized knowledge base of all data definitions, data relationships, screen and report formats, and other system components.

8. ***System developers*** System developers are persons, such as systems analysts and programmers, who design new application programs. The content of all chapters of this book is useful for system developers, but Chapters 2 through 8 are likely to be particularly valuable because of their focus on transactional systems.
9. ***End users*** End users are persons throughout the organization who add, delete, and modify data in the database and who request or receive information from it. All user interactions with the database must be routed through the DBMS. This text is targeted primarily to students striving to become data management and systems development professionals, but advanced end users can benefit from many of the skills covered in it (particularly conceptual data modeling in Chapters 2 to 3, SQL in Chapters 5 to 6, and Analytics in Chapter 11).

In summary, the database operational environment shown in Figure 1-6 is an integrated system of hardware, software, and people, designed to facilitate the storage, retrieval, and control of the information resource and to improve the productivity of the organization.

THE DATABASE DEVELOPMENT PROCESS

Enterprise data modeling
The first step in database development, in which the scope and general contents of organizational databases are specified.

How do organizations start developing a database? In many organizations, database development begins with **enterprise data modeling**, which establishes the range and general contents of organizational databases. Its purpose is to create an overall picture or explanation of organizational data, not the design for a particular database. A particular database provides the data for one or more information systems, whereas an enterprise data model, which may encompass many databases, describes the scope of data maintained by the organization. In enterprise data modeling, you review current systems, analyze the nature of the business areas to be supported, describe the data needed at a very high level of abstraction, and plan one or more database development projects.

Figure 1-3a showed a segment of an enterprise data model for Pine Valley Furniture Company, using a simplified version of the notation you will learn in Chapters 2 and 3. Besides such a graphical depiction of the entity types, a thorough enterprise data model would also include business-oriented descriptions of each entity type and a compendium of various statements about how the business operates, called *business rules*, which govern the validity of data. Relationships between business objects (business functions, units, applications, and so forth) and data are often captured using matrixes and complement the information captured in the enterprise data model. Figure 1-7 shows an example of such a matrix.

FIGURE 1-7 Example business function-to-data entity matrix

Business Functions \ Data Entity Types	Customer	Product	Raw Material	Order	Work Center	Work Order	Invoice	Equipment	Employee
Business Planning	X	X						X	X
Product Development		X	X		X			X	
Materials Management		X	X	X	X	X		X	
Order Fulfillment	X	X	X	X	X	X	X	X	X
Order Shipment	X	X		X	X		X		X
Sales Summarization	X	X		X			X		X
Production Operations		X	X	X	X	X		X	X
Finance and Accounting	X	X	X	X	X		X	X	X

X = data entity is used within business function

Enterprise data modeling as a component of a top-down approach to information systems planning and development represents one source of database projects. Such projects often develop new databases to meet strategic organizational goals, such as improved customer support, better production and inventory management, or more accurate sales forecasting. Many database projects arise, however, in a more bottom-up fashion. In this case, projects are requested by information systems users who need certain information to do their jobs or by other information systems professionals who see a need to improve data management in the organization.

A typical bottom-up database development project usually focuses on the creation of one database. Some database projects concentrate only on defining, designing, and implementing a database as a foundation for subsequent information systems development. In most cases, however, a database and the associated information processing functions are developed together as part of a comprehensive information systems development project.

Systems Development Life Cycle

As you may know from other information systems courses you've taken, a traditional process for conducting an information systems development project is called the **systems development life cycle (SDLC)**. The SDLC is a complete set of steps that a team of information systems professionals, including database designers and programmers, follow in an organization to specify, develop, maintain, and replace information systems. Textbooks and organizations use many variations on the life cycle and may identify anywhere from 3 to 20 different phases.

Systems development life cycle (SDLC)
The traditional methodology used to develop, maintain, and replace information systems.

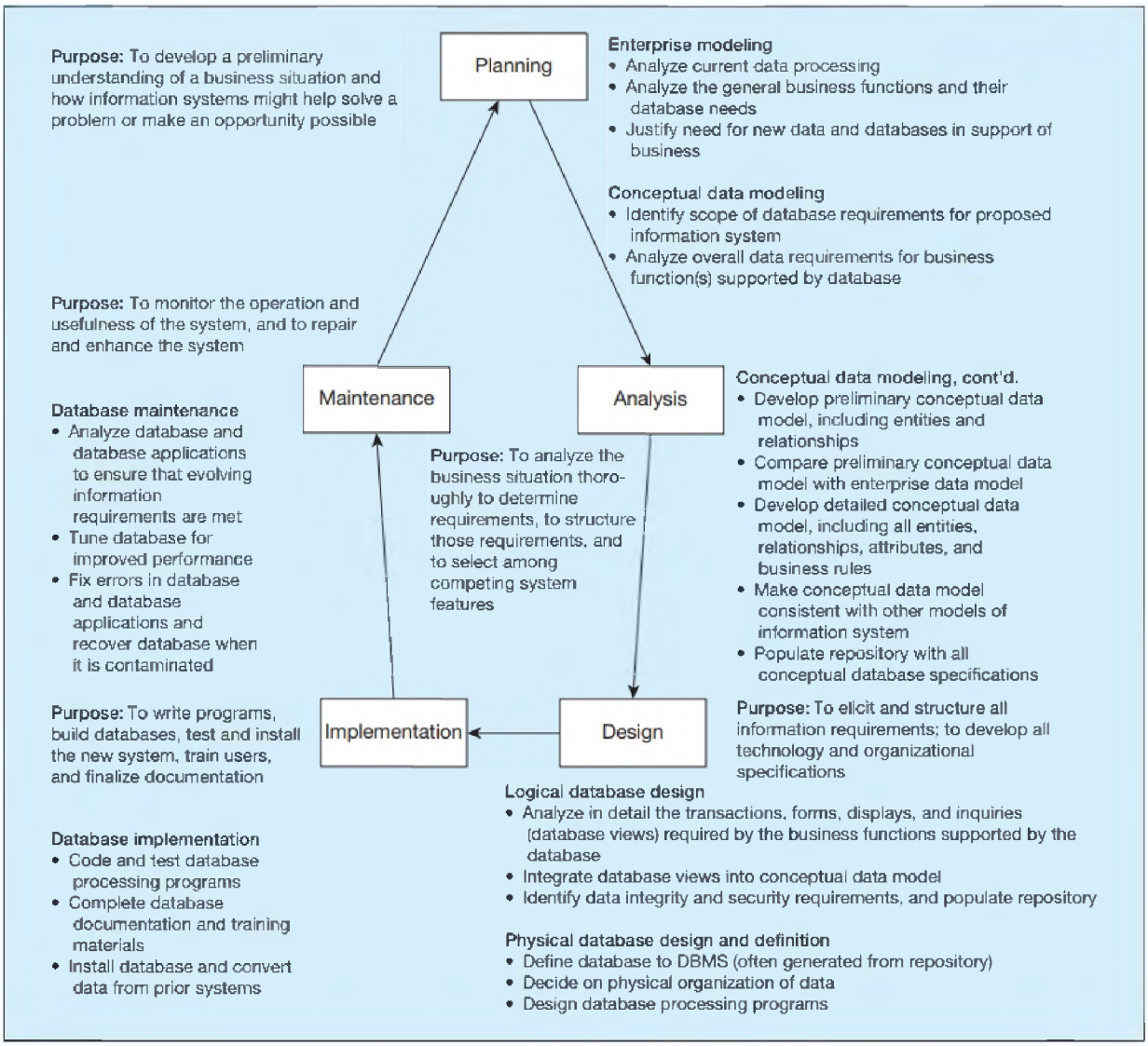
The various steps in the SDLC and their associated purpose are depicted in Figure 1-8 (Valacich and George, 2016). The process appears to be circular and is intended to convey the iterative nature of systems development projects. The steps may overlap in time, they may be conducted in parallel, and it is possible to backtrack to previous steps when prior decisions need to be reconsidered. Some believe that the most common path through the development process is to cycle through the steps depicted in Figure 1-8 but at more detailed levels on each pass as the requirements of the system become more concrete.

Figure 1-8 also provides an outline of the database development activities typically included in each phase of the SDLC. Note that there is not always a one-to-one correspondence between SDLC phases and database development steps. For example, conceptual data modeling occurs in both the Planning and the Analysis phase. We will briefly illustrate each of these database development steps for Pine Valley Furniture Company later in this chapter.

PLANNING—ENTERPRISE MODELING The database development process begins with a review of the enterprise modeling components that were developed during the information systems planning process. During this step, analysts review current databases and information systems, analyze the nature of the business area that is the subject of the development project, and describe, in general terms, the data needed for each information system under consideration for development. They determine what data are already available in existing databases and what new data will need to be added to support the proposed new project. Only selected projects move into the next phase based on the projected value of each project to the organization.

PLANNING—CONCEPTUAL DATA MODELING For an information systems project that is initiated, the overall data requirements of the proposed information system must be analyzed. This is done in two stages. First, during the Planning phase, the analyst develops a diagram similar to Figure 1-3a, as well as other documentation, to outline the scope of data involved in this particular development project without consideration of what databases already exist. Only high-level categories of data (entities) and major relationships are included at this point. This step in the SDLC is critical for improving the chances of a successful development process. The better the definition of the specific needs of the organization, the closer the conceptual model should come

FIGURE 1-8 Database development activities during the systems development life cycle (SDLC)



to meeting the needs of the organization and the less recycling back through the SDLC should be needed.

ANALYSIS—CONCEPTUAL DATA MODELING During the Analysis phase of the SDLC, the analyst produces a detailed data model that identifies all the organizational data that must be managed for this information system. Every data attribute is defined, all categories of data are listed, every business relationship between data entities is represented, and every rule that dictates the integrity of the data is specified. It is also during the Analysis phase that the conceptual data model is checked for consistency with other types of models developed to explain other dimensions of the target information system, such as processing steps, rules for handling data, and the timing of events. However, even this detailed conceptual data model is preliminary because subsequent SDLC activities may find missing elements or errors when designing specific transactions, reports, displays, and inquiries. With experience, the database developer gains mental models of common business functions, such as sales or financial record keeping,

but must always remain alert for the exceptions to common practices followed by an organization. The output of the conceptual modeling phase is a **conceptual schema**.

Conceptual schema
A detailed, technology-independent specification of the overall structure of organizational data.

DESIGN—LOGICAL DATABASE DESIGN Logical database design approaches database development from two perspectives. First, the conceptual schema must be transformed into a logical schema, which describes the data in terms of the data management technology that will be used to implement the database. For example, if relational technology will be used, the conceptual data model is transformed and represented using elements of the relational model, which include tables, columns, rows, primary keys, foreign keys, and constraints. (You will learn how to conduct this important process in Chapter 4.) This representation is referred to as the **logical schema**.

Logical schema
The representation of a database for a particular data management technology.

Then, as each application in the information system is designed, including the program's input and output formats, the analyst performs a detailed review of the transactions, reports, displays, and inquiries supported by the database. During this so-called bottom-up analysis, the analyst verifies exactly what data are to be maintained in the database and the nature of those data as needed for each transaction, report, and so forth. It may be necessary to refine the conceptual data model as each report, business transaction, and other user view is analyzed. In this case, one must combine, or integrate, the original conceptual data model along with these individual user views into a comprehensive design during logical database design. It is also possible that additional information processing requirements will be identified during logical information systems design, in which case these new requirements must be integrated into the previously identified logical database design.

The final step in logical database design is to transform the combined and reconciled data specifications into basic, or atomic, elements following well-established rules for well-structured data specifications. For most databases today, these rules come from relational database theory and a process called *normalization*, which you will learn about in detail in Chapter 4. The result is a complete picture of the database without any reference to a particular database management system for managing these data. With a final logical database design in place, the analyst begins to specify the logic of the particular computer programs and queries needed to maintain and report the database contents.

DESIGN—PHYSICAL DATABASE DESIGN AND DEFINITION A **physical schema** is a set of specifications that describe how data from a logical schema are stored in a computer's secondary memory by a specific database management system. There is one physical schema for each logical schema. Physical database design requires knowledge of the specific DBMS that will be used to implement the database. In physical database design and definition, an analyst decides on the organization of physical records, the choice of file organizations, the use of indexes, and so forth. To do this, a database designer needs to outline the programs to process transactions and to generate anticipated management information and decision support reports. The goal is to design a database that will efficiently and securely handle all data processing against it. Thus, physical database design is done in close coordination with the design of all other aspects of the physical information system: programs, computer hardware, operating systems, and data communications networks.

Physical schema
Specifications for how data from a logical schema are stored in a computer's secondary memory by a database management system.

IMPLEMENTATION—DATABASE IMPLEMENTATION In database implementation, a designer writes, tests, and installs the programs/scripts that access, create, or modify the database. The designer might do this using standard programming languages (e.g., Java, C#, or Visual Basic.NET) or in special database processing languages (e.g., SQL) or use special-purpose nonprocedural languages to produce stylized reports and displays, possibly including graphs. Also, during implementation, the designer will finalize all database documentation, train users, and put procedures into place for the ongoing support of the information system (and database) users. The last step is to load data from existing information sources (files and databases from legacy applications plus new data now needed). Loading is often done by first unloading data from existing files and databases into a neutral format (such as binary or text files) and then loading these data into the new database. Finally, the database and its associated applications

are put into production for data maintenance and retrieval by the actual users. During production, the database should be periodically backed up and recovered in case of contamination or destruction.

MAINTENANCE—DATABASE MAINTENANCE The database evolves during database maintenance. In this step, the designer adds, deletes, or changes characteristics of the structure of a database in order to meet changing business conditions, to correct errors in database design, or to improve the processing speed of database applications. The designer might also need to rebuild a database if it becomes contaminated or destroyed due to a program or computer system malfunction. This is typically the longest step of database development because it lasts throughout the life of the database and its associated applications. Each time the database evolves, view it as an abbreviated database development process in which conceptual data modeling, logical and physical database design, and database implementation occur to deal with proposed changes.

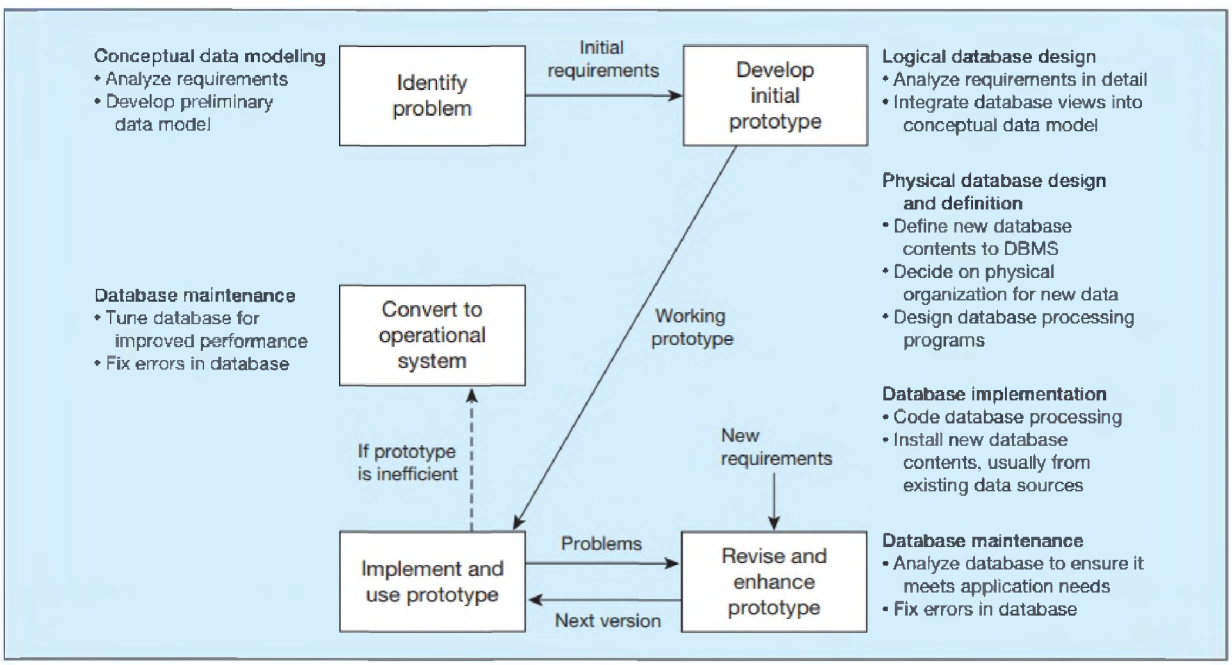
Alternative Information Systems Development Approaches

The systems development life cycle or slight variations on it are often used to guide the development of information systems and databases. The SDLC is a methodical, highly structured approach that includes many checks and balances to ensure that each step produces accurate results and that the new or replacement information system is consistent with existing systems with which it must communicate or for which there needs to be consistent data definitions. Whew! That's a lot of work! Consequently, the SDLC is often criticized for the length of time needed until a working system is produced, which occurs only at the end of the process. Instead, organizations increasingly use rapid application development (RAD) methods, which follow an iterative process of rapidly repeating analysis, design, and implementation steps until they converge on the system the user wants. These RAD methods work best when most of the necessary database structures already exist and hence for systems that primarily retrieve data rather than for those that populate and revise databases.

Prototyping
An iterative process of systems development in which requirements are converted to a working system that is continually revised through close work between analysts and users.

One of the most popular RAD methods is **prototyping**, which is an iterative process of systems development in which requirements are converted to a working system that is continually revised through close work between analysts and users. Figure 1-9 shows the

FIGURE 1-9 The prototyping methodology and database development process



prototyping process. This figure includes annotations to indicate roughly which database development activities occur in each prototyping phase. Typically, you make only a very cursory attempt at conceptual data modeling when the information system problem is identified. During the development of the initial prototype, you simultaneously design the displays and reports the user wants while understanding any new database requirements and defining a database to be used by the prototype. This is typically a new database, which is a copy of portions of existing databases, possibly with new content. If new content is required, it will usually come from external data sources, such as market research data, general economic indicators, or industry standards.

Database implementation and maintenance activities are repeated as new versions of the prototype are produced. Often, security and integrity controls are minimal because the emphasis is on getting working prototype versions ready as quickly as possible. Also, documentation tends to be delayed until the end of the project, and user training occurs from hands-on use. Finally, after an accepted prototype is created, the developer and the user decide whether the final prototype and its database can be put into production as is. If the system, including the database, is too inefficient, the system and database might need to be reprogrammed and reorganized to meet performance expectations. Inefficiencies, however, have to be weighed against violating the core principles behind sound database design.

With the increasing popularity of visual programming tools (such as Visual Basic, Java, or C#) that make it easy to modify the interface between user and system, prototyping is becoming the systems development methodology of choice to develop new applications internally. With prototyping, it is relatively easy to change the content and layout of user reports and displays.

The benefits from iterative approaches to systems development demonstrated by RAD and prototyping approaches have resulted in further efforts to create ever more responsive development approaches. In February 2001, a group of 17 individuals interested in supporting these approaches created "The Manifesto for Agile Software Development." For them, **agile software development** practices include valuing (www.agilemanifesto.org) the following:

Individuals and interactions over processes and tools

Working software over comprehensive documentation

Customer collaboration over contract negotiation

Responding to change over following a plan

Agile software development

An approach to database and software development that emphasizes "*individuals and interactions* over processes and tools, *working software* over comprehensive documentation, *customer collaboration* over contract negotiation, and *response to change* over following a plan."

Emphasis on the importance of people, both software developers and customers, is evident in their phrasing. This is in response to the turbulent environment within which software development occurs as compared to the more staid environment of most engineering development projects from which the earlier software development methodologies came. The importance of the practices established in the SDLC continues to be recognized and accepted by software developers, including the creators of "The Manifesto for Agile Software Development." However, it is impractical to allow these practices to stifle quick reactions to changes in the environment that change project requirements.

The use of agile or adaptive processes should be considered when a project involves unpredictable and/or changing requirements, responsible and collaborative developers, and involved customers who understand and can contribute to the process (Fowler, 2005). If you are interested in learning more about agile software development, investigate agile methodologies, such as eXtreme Programming, Scrum, the DSDM Consortium, and feature-driven development.

Three-Schema Architecture for Database Development

The explanation earlier in this chapter of the database development process referred to several different but related models of databases developed on a systems development project. These data models and the primary phase of the SDLC in which they are developed are summarized here:

- Enterprise data model (during the Information Systems Planning phase).
- External schema or user view (during the Analysis and Logical Design phases).

- Conceptual schema (during the Analysis phase).
- Logical schema (during the Logical Design phase).
- Physical schema (during the Physical Design phase).

In 1978, an industry committee commonly known as ANSI/SPARC published an important document that described three-schema architecture—external, conceptual, and internal schemas—for describing the structure of data. Figure 1-10 shows the relationship between the various schemas developed during the SDLC and the ANSI three-schema architecture. It is important to keep in mind that all these schemas are just different ways of visualizing the structure of the same database by different stakeholders.

The three schemas as defined by ANSI (depicted down the center of Figure 1-10) are as follows:

1. ***External schema*** This is the view (or views) of managers and other employees who are the database users. As shown in Figure 1-10, the external schema can be represented as a combination of the enterprise data model (a top-down view) and a collection of detailed (or bottom-up) user views.
2. ***Conceptual schema*** This schema combines the different external views into a single, coherent, and comprehensive definition of the enterprise's data. The conceptual schema represents the view of the data architect or data administrator.
3. ***Internal schema*** As shown in Figure 1-10, an internal schema today really consists of two separate schemas: a logical schema and a physical schema. The logical schema is the representation of data for a type of data management technology (e.g., relational). The physical schema describes how data are to be represented and stored in secondary storage using a particular DBMS (e.g., Oracle).

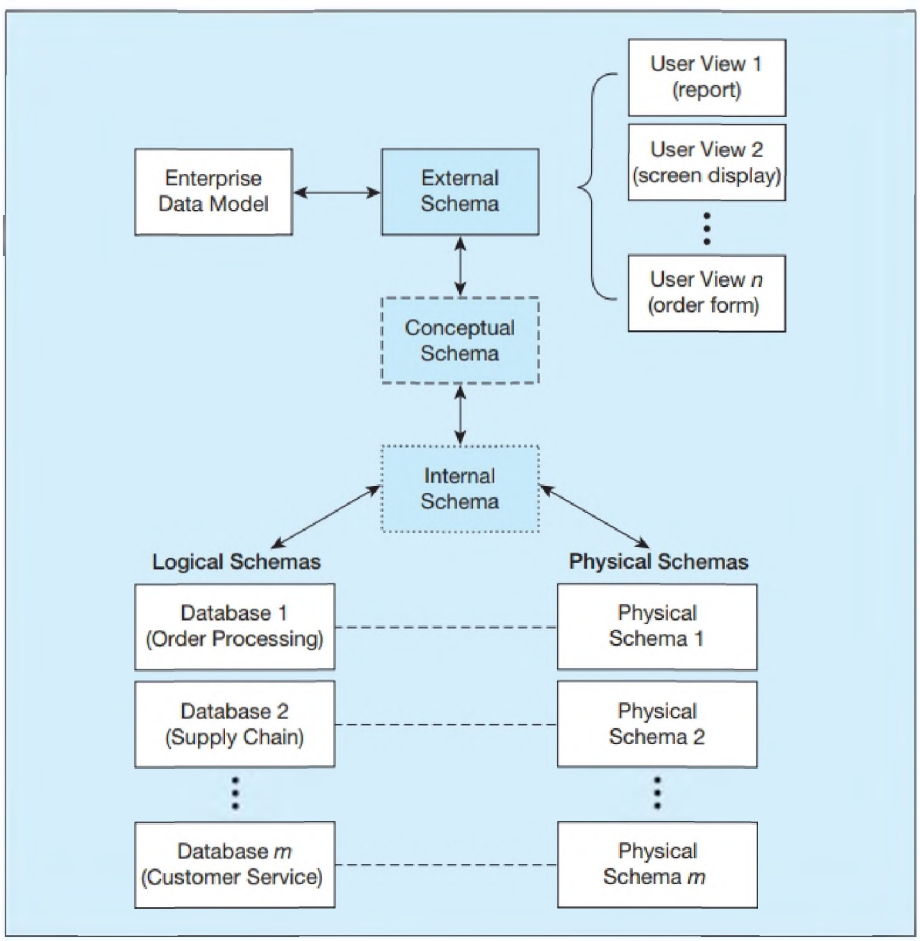


FIGURE 1-10 Three-schema architecture

Managing the People Involved in Database Development

Isn't it always ultimately about people working together? As implied in Figure 1-8, a database is developed as part of a project. A **project** is a planned undertaking of related activities to reach an objective that has a beginning and an end. A project begins with the first steps of the Project Initiation and Planning phase and ends with the last steps of the Implementation phase. A senior systems or database analyst will be assigned to be project leader. This person is responsible for creating detailed project plans as well as staffing and supervising the project team.

Project
A planned undertaking of related activities to reach an objective that has a beginning and an end.

A project is initiated and planned in the Planning phase; executed during Analysis, Logical Design, Physical Design, and Implementation phases; and closed down at the end of implementation. During initiation, the project team is formed. A systems or database development team can include one or more of the following:

- ***Business analysts*** These individuals work with both management and users to analyze the business situation and develop detailed system and program specifications for projects.
- ***Systems analysts*** These individuals may perform business analyst activities but also specify computer systems requirements and typically have a stronger systems development background than business analysts.
- ***Database analysts and data modelers*** These individuals concentrate on determining the requirements and design for the database component of the information system.
- ***Users*** Users provide assessments of their information needs and monitor that the developed system meets their needs.
- ***Programmers*** These individuals design and write computer programs that have commands to maintain and access data in the database embedded in them.
- ***Database architects*** These individuals establish standards for data in business units, striving to attain optimum data location, currency, and quality.
- ***Data administrators*** These individuals have responsibility for existing and future databases and ensure consistency and integrity across databases, and as experts on database technology, they provide consulting and training to other project team members.
- ***Project managers*** Project managers oversee assigned projects, including team composition, analysis, design, implementation, and support of projects.
- ***Other technical experts*** Other individuals are needed in areas such as networking, operating systems, testing, data warehousing, and documentation.

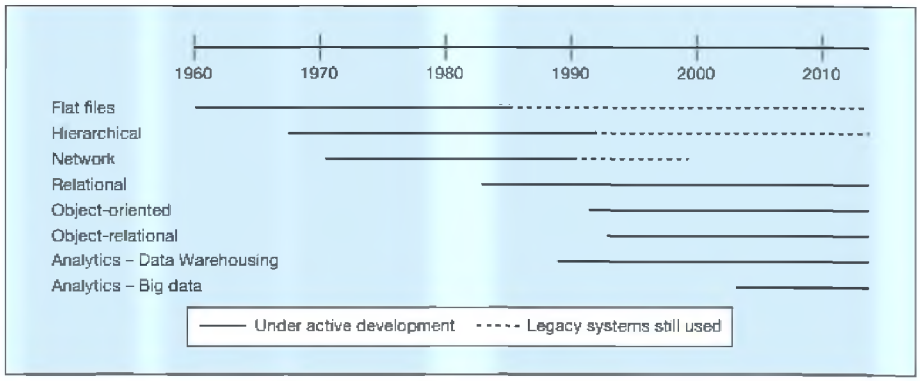
It is the responsibility of the project leader to select and manage all of these people as an effective team. See Valacich and George (2016) for details on how to manage a systems development project team and Henderson et al. (2005) for a more detailed description of career paths and roles in data management. The emphasis on people rather than roles when agile development processes are adopted means that team members will be less likely to be constrained to a particular role. They will be expected to contribute and collaborate across these roles, thus using their particular skills, interests, and capabilities more completely.

EVOLUTION OF DATABASE SYSTEMS

Database management systems were first introduced during the 1960s and have continued to evolve during subsequent decades. Figure 1-11a sketches this evolution by highlighting the database technology (or technologies) that was dominant during each decade. In most cases, the period of introduction was quite long, and the technology was first introduced during the decade preceding the one shown in the figure. For example, the relational model was first defined by E. F. Codd, an IBM research fellow, in a paper published in 1970 (Codd, 1970). However, the relational model did not realize widespread commercial success until the 1980s. For example, the challenge of the 1970s when programmers needed to write complex programs to access data was addressed by the introduction of SQL in the 1980s.

FIGURE 1-11 The range of database technologies: past and present

(a) Evolution of database technologies



(b) Database architectures

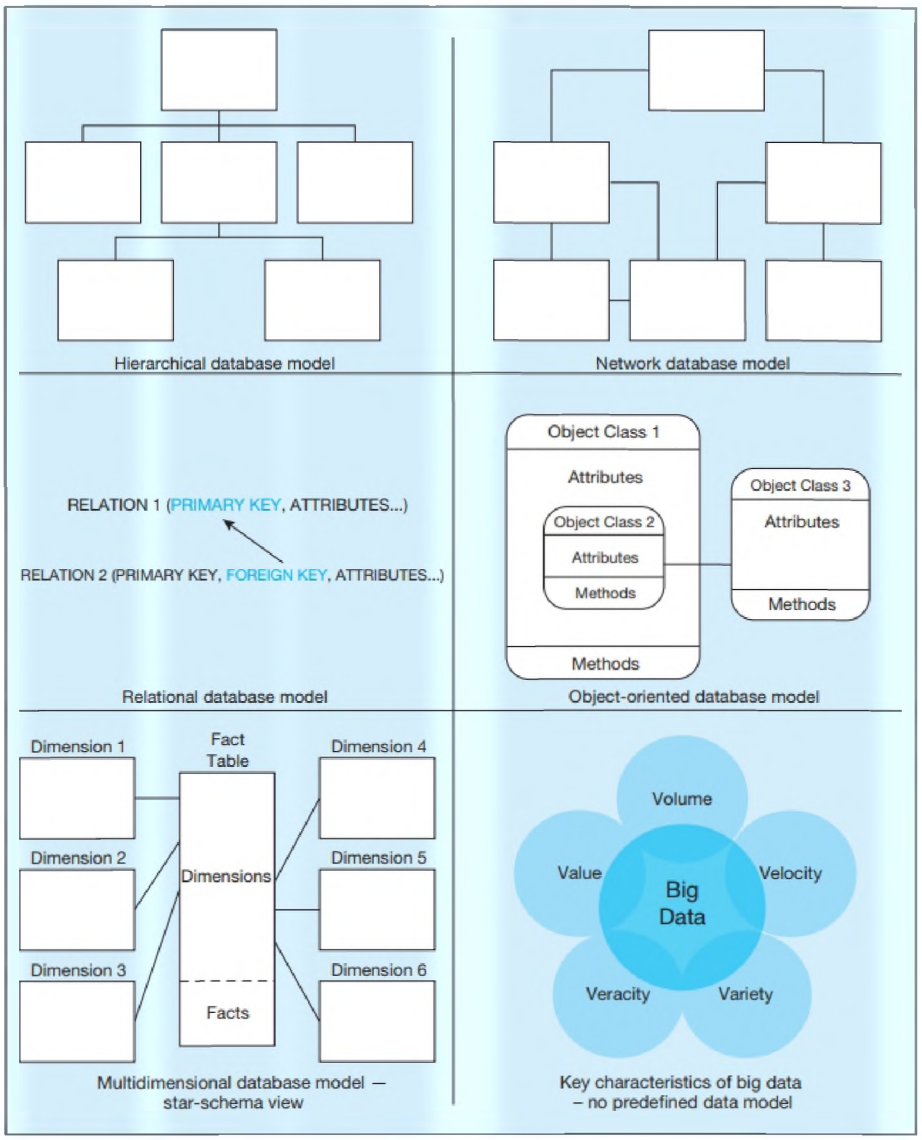


Figure 1-11b shows a visual depiction of the organizing principle underlying each of the major database technologies. For example, in the hierarchical model, files are organized in a top-down structure that resembles a tree or genealogy chart, whereas in the network model, each file can be associated with an arbitrary number of other files. The relational model (the primary focus of this book) organizes data in the form of tables and relationships among them. The object-oriented model (discussed in Chapter 14 on the book's Web site) is based on object classes and relationships among them. As shown in Figure 1-11b, an object class encapsulates attributes and methods. Object-relational databases are a hybrid between object-oriented and relational databases. Multidimensional databases, which form the basis for data warehouses, allow us to view data in the form of cubes or a star schema; you will learn more about this in Chapter 9. The final element of the diagram illustrates the main characteristics of the big data approach to data management (to be covered in Chapter 10), intentionally leaving a specific modeling approach (there is no single data model for big data).

Database management systems were developed to overcome the limitations of file processing systems, described in a previous section. To summarize, some of the following four objectives generally drove the development and evolution of database technology:

1. The need to provide greater independence between programs and data, thereby reducing maintenance costs.
2. The desire to manage increasingly complex data types and structures.
3. The desire to provide easier and faster access to data for users who have neither a background in programming languages nor a detailed understanding of how data are stored in databases.
4. The need to provide ever more powerful platforms for decision support applications.

1960s

File processing systems were still dominant during the 1960s. However, the first database management systems were introduced during this decade and were used primarily for large and complex ventures, such as the Apollo moon-landing project. We can regard this as an experimental "proof-of-concept" period in which the feasibility of managing vast amounts of data with a DBMS was demonstrated. Also, the first efforts at standardization were taken with the formation of the Data Base Task Group in the late 1960s.

1970s

During this decade, the use of database management systems became a commercial reality. The hierarchical and network database management systems were developed, largely to cope with increasingly complex data structures, such as manufacturing bills of materials that were extremely difficult to manage with conventional file processing methods. The hierarchical and network models are generally regarded as first-generation DBMS. Both approaches were widely used, and in fact many of these systems continue to be used today. However, they suffered from the same key disadvantages as file processing systems: limited data independence and lengthy development times for application development.

1980s

To overcome these limitations, E. F. Codd and others developed the relational data model during the 1970s. This model, considered second-generation DBMS, received widespread commercial acceptance and diffused throughout the business world during the 1980s. With the relational model, all data are represented in the form of tables. Typically, SQL is used for data retrieval. Thus, the relational model provides ease of access for nonprogrammers, overcoming one of the major objections to first-generation systems. The relational model has also proven well-suited to client/server computing, parallel processing, and graphical user interfaces (Gray, 1996).

1990s

The 1990s ushered in a new era of computing, first with client/server computing and then with data warehousing and Internet applications becoming increasingly important. Whereas the data managed by a DBMS during the 1980s were largely structured (such as accounting data), multimedia data (including graphics, sound, images, and video) became increasingly common during the 1990s. To cope with these increasingly complex data, object-oriented databases (considered third generation) were introduced during the late 1980s (Grimes, 1998); however, these never reached the popularity they were predicted to gain. Instead, new types of data management technologies were developed in 2000s and 2010s.

2000 and Beyond

Currently, relational databases are still the most widely used database technology. Because organizations must manage a vast amount of structured and unstructured data and because both the amounts and the variety of data are increasing rapidly, the need for new technologies has rapidly become clear. Since early 2000s, various nonrelational technologies have become increasingly popular, as you already learned earlier in this chapter in the context of the framework presented in Figure 1-5. One of the elements of this trend is the emergence of NoSQL (Not Only SQL) databases. NoSQL is an umbrella term that refers to a set of database technologies that is specifically designed to address large (structured and unstructured) data that are potentially stored across various locations. Popular examples of NoSQL databases are MongoDB and Apache Cassandra (**http://cassandra.apache.org**). In addition, Hadoop is an example of a nonrelational technology that is designed to handle the processing of large amounts of data. This search for nonrelational database technologies is fueled by the needs of social networking applications, such as blogs, wikis, and social networking sites (Facebook, Twitter, LinkedIn, and so forth); the ease of generating unstructured data, such as pictures and images, from devices, such as smartphones, tablets, and so forth; and the data needs and opportunities of the Internet of Things. Developing effective database practices to deal with these diverse types of data continues to be of prime importance. As larger computer memory chips become less expensive, new database technologies to manage in-memory databases are emerging. This trend opens up new possibilities for even faster database processing. You will find more about some of these new trends in Chapter 11.

Recent regulations such as Sarbanes-Oxley, the Health Insurance Portability and Accountability Act, and the Basel Convention have highlighted the importance of good data management practices, and the ability to reconstruct historical positions has gained prominence. This has led to developments in computer forensics with increased emphasis and expectations around discovery of electronic evidence. The importance of good database administration capabilities also continues to rise because effective disaster recovery and adequate security are mandated by these regulations.

An emerging trend that is making it more convenient to use database technologies (and to tackle some of the regulatory challenges identified here) is that of cloud computing. One popular technology available in the cloud is databases. Databases, relational and nonrelational, can now be created, deployed, and managed through the use of technologies owned and managed by a service provider. You will examine issues surrounding cloud databases in Chapters 7 and 8.

THE RANGE OF DATABASE APPLICATIONS

What can databases help us do? Recall that Figure 1-6 showed that there are several methods for people to interact with the data in the database. First, users can interact directly with the database using the user interface provided by the DBMS. In this manner, users can issue commands (called *queries*) against the database and examine the results or potentially even store them inside a Microsoft Excel spreadsheet or Word document. This method of interaction with the database is referred to as ad hoc querying and requires a level of understanding the query language on the part of the user.

Because most business users do not possess this level of knowledge, the second and more common mechanism for accessing the database is using application programs. An application program consists of two key components. A graphical user interface accepts the users' request (e.g., to input, delete, or modify data) and/or provides a mechanism for displaying the data retrieved from the database. The business logic contains the programming logic necessary to act on the users' commands. The machine that runs the user interface (and sometimes the business logic) is referred to as the *client*. The machine that runs the DBMS and contains the database is referred to as the *database server*.

It is important to understand that the applications and the database need not reside on the same computer (and, in most cases, they don't). In order to better understand the range of database applications, we divide them into three categories based on the location of the client (application) and the database software itself: personal, multi-tier, and enterprise databases. We introduce each category with a typical example, followed by some issues that generally arise within that category of use.

Personal Databases

Personal databases are designed to support one user. Personal databases have long resided on personal computers, including laptops, and now increasingly reside on smartphones, tablets, phablets, and so forth. The purpose of these databases is to provide the user with the ability to manage (store, update, delete, and retrieve) small amounts of data in an efficient manner. Simple database applications that store customer information and the details of contacts with each customer can be used from a personal computer and easily transferred from one device to the other for backup and work purposes. For example, consider a company that has a number of salespersons who call on actual or prospective customers. A database of customers and a pricing application can enable the salesperson to determine the best combination of quantity and type of items for the customer to order.

Personal databases are widely used because they can often improve personal productivity. However, they entail a risk: The data cannot easily be shared with other users. For example, suppose the sales manager wants a consolidated view of customer contacts. This cannot be quickly or easily provided from an individual salesperson's databases. This illustrates a common problem: If data are of interest to one person, they probably are or will soon become of interest to others as well. For this reason, personal databases should be limited to those rather special situations (e.g., in a very small organization) where the need to share the data among users of the personal database is unlikely to arise.

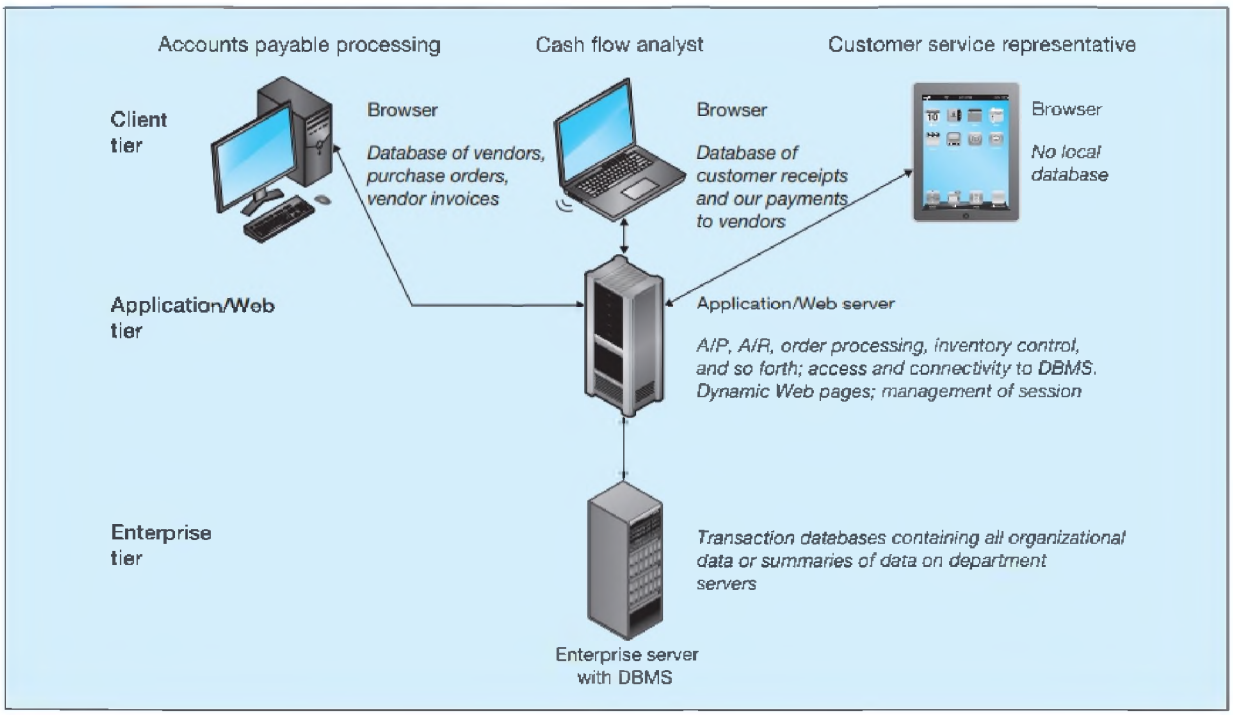
Departmental Multi-Tiered Client/Server Databases

As noted earlier, the utility of a personal (single-user) database is quite limited. Often, what starts off as a single-user database evolves into something that needs to be shared among several users.

To overcome these limitations, most modern applications that need to support a large number of users are built using the concept of multi-tiered architecture. In most organizations, these applications are intended to support a department (such as marketing or accounting) or a division (such as a line of business), which is generally larger than a work group (typically between 25 and 100 persons).

An example of a company that has several multi-tiered applications is shown in Figure 1-12. In a multi-tiered architecture, the user interface is accessible on the individual users' computers. This user interface may be either Web browser based or written using programming languages such as Visual Basic.NET, Visual C#, or Java. The application layer/Web server layer contains the business logic required to accomplish the business transactions requested by the users. This layer in turn talks to the database server. The most significant implication for database development from the use of multi-tiered client/server architectures is the ease of separating the development of the database and the modules that maintain the data from the information systems modules that focus on business logic and/or presentation logic. In addition, this architecture

FIGURE 1-12 Multi-tiered client/server database architecture



allows us to improve performance and maintainability of the application and database. We will consider both two and multi-tiered client/server architectures in more detail in Chapter 7.

Enterprise Applications

An enterprise (that's small "e," not capital "E," as in *Starship*) application/database is one whose scope is the entire organization or enterprise (or, at least, many different departments). Such databases are intended to support organization-wide operations and decision making in organizations for which departmental solutions are not sufficient. Note that an organization may have several enterprise databases, so such a database is not inclusive of all organizational data. A single operational enterprise database is impractical for many medium-size to large organizations due to difficulties in performance for very large databases, diverse needs of different users, and the complexity of achieving a single definition of data (metadata) for all database users. An enterprise database does, however, support information needs of many departments and divisions. It is possible that enterprise databases are architected using a multi-tiered approach described above.

The evolution of enterprise databases has resulted in three major developments:

1. Large-scale enterprise systems, such as enterprise resource planning and customer relationship management.
2. Data warehousing implementations providing a centralized perspective on enterprise data.
3. Most recently data lakes, which provide a less structured and less costly way to collect large amounts of heterogeneous data without a clear advance knowledge regarding their use.

ENTERPRISE SYSTEMS These make up the backbone for virtually every modern organization because they form the foundation for the processes that control and execute basic business tasks. They are the systems that keep an organization running, whether

it is a neighborhood grocery store with a few employees or a multinational corporation with dozens of divisions around the world and tens of thousands of employees. The focus of these applications is on capturing the data surrounding the *"transactions,"* that is, the hundreds or millions (depending on the size of the organization and the nature of the business) of events that take place in an organization every day and define how a business is conducted. For example, when you registered for your database course, you engaged in a transaction that captured data about your registration. Similarly, when you go to a store to buy a candy bar, a transaction takes place between you and the store, and data are captured about your purchase. When Wal-Mart pays its hundreds of thousands of hourly employees, data regarding these transactions are captured in Wal-Mart's systems.

It is very typical these days that organizations use packaged systems offered by outside vendors for their transaction processing needs. Examples of these types of systems include **enterprise resource planning (ERP)**, customer relationship management, supply chain management, human resource management, and payroll. All these systems are heavily dependent on databases for storing the data.

Enterprise resource planning (ERP)

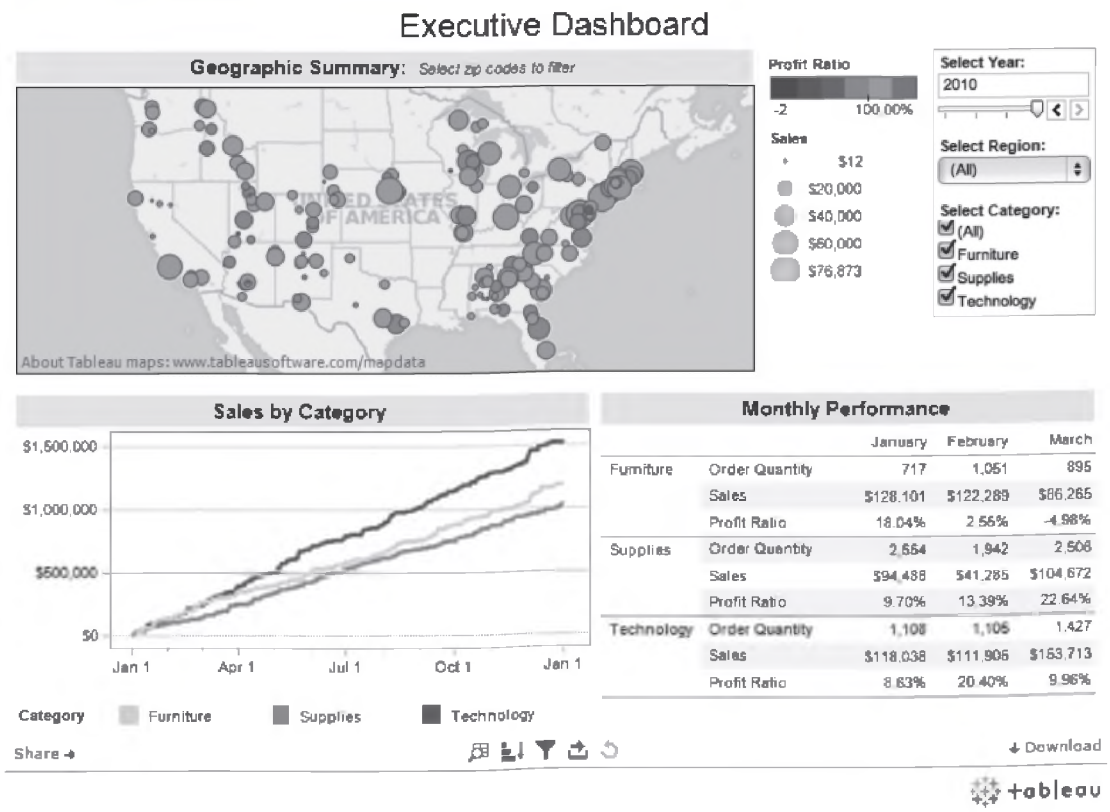
A business management system that integrates all functions of the enterprise, such as manufacturing, sales, finance, marketing, inventory, accounting, and human resources. ERP systems are software applications that provide the data necessary for the enterprise to examine and manage its activities.

DATA WAREHOUSES Whereas ERP systems work with the current operational data of the enterprise, **data warehouses** collect content from the various operational databases, including personal, work group, department, and ERP databases. Data warehouses provide users with the opportunity to work with historical data to identify patterns and trends and answers to strategic business questions. Figure 1-13 presents an example of what an output from a data warehouse might look like. You will learn about data warehouses at a detailed level in Chapter 9.

Data warehouse

An integrated decision support database whose content is derived from the various operational databases.

FIGURE 1-13 An example of an executive dashboard
(http://public.tableausoftware.com/profile/mirandali#!/vizhome/Executive-Dashboard_7/ExecutiveDashboard)
Courtesy Tableau Software



Data lake

A large integrated repository for internal and external data that does not follow a predefined schema.

DATA LAKE This is a relatively new enterprise-level concept introduced in the big data context. In the same way as a data warehouse, it is an integrated repository of data from a variety of sources, but data lakes have several characteristics that differentiate them from traditional data warehouses. First, they typically are not based on a predefined data model or schema (as we discussed above, they are often build following the "schema on read" principle). As you will find out in Chapter 10, organizations using data lakes often collect "everything," that is, store data from a rich variety of sources without knowing in advance when and for what purpose the stored data will be needed. Unless confidentiality requirements prevent it, data lakes are intended to be used for a variety of purposes and by multiple users. Because of the lack of the predefined structure, data lakes are particularly good for identifying new and creative connections between data items. Data lakes are by definition designed to be highly scalable, and they often are built on a technical foundation that uses low-cost commodity hardware (such as Hadoop).

Finally, one change that has dramatically affected the database environment is the ubiquity of the Internet and the subsequent development of applications that are used by the masses. Acceptance of the Internet by businesses has resulted in important changes in long-established business models. Even extremely successful companies have been shaken by competition from new businesses that have employed the Internet to provide improved customer information and service, to eliminate traditional marketing and distribution channels, and to implement employee relationship management. For example, customers configure and order their personal computers directly from the computer manufacturers. Bids are accepted for airline tickets and collectibles within seconds of submission, sometimes resulting in substantial savings for the end consumer. Information about open positions and company activities is readily available within many companies. Each of these Web-based applications use databases extensively.

In the previous examples, the Internet is used to facilitate interaction between the business and the customer (B2C) because the customers are necessarily external to the business. However, for other types of applications, the customers of the businesses are other businesses. Those interactions are commonly referred to as B2B relationships and are enabled by extranets. An *extranet* uses Internet technology, but access to the extranet is not universal, as is the case with an Internet application. Rather, access is restricted to business suppliers and customers with whom an agreement has been reached about legitimate access and use of one another's data and information. Finally, an *intranet* is used by employees of the firm to access applications and databases within the company.

Allowing such access to a business's database raises data security and integrity issues that are new to the management of information systems, whereby data have traditionally been closely guarded and secured within each company. These issues become even more complex as companies take advantage of the cloud. Now data are stored on servers that are not within the control of the company that is generating the data. You will learn about database security and cloud-based databases in more detail in Chapters 7 and 8.

Table 1-5 presents a brief summary of the types of databases outlined in this section.

TABLE 1-5 Summary of Database Applications

Type of Database/Application	Typical Number of Users	Typical Size of Database
Personal	1	Megabytes
Multi-tiered client/server	2–1,000	Gigabytes
Enterprise resource planning	>100	Gigabytes–terabytes
Data warehousing	>100	Terabytes–petabytes
Data lake	>100	Terabytes–petabytes

DEVELOPING A DATABASE APPLICATION FOR PINE VALLEY FURNITURE COMPANY

PINE VALLEY FURNITURE

Pine Valley Furniture Company was introduced earlier in this chapter. By the late 1990s, competition in furniture manufacturing had intensified, and competitors seemed to respond more rapidly than Pine Valley Furniture to new business opportunities. While there were many reasons for this trend, managers believed that the computer information systems they had been using (based on traditional file processing) had become outdated. After attending an executive development session led by Heikki Topi and Jeff Hoffer (we wish!), the company started a development effort that eventually led to adopting a database approach for the company. Data previously stored in separate files have been integrated into a single database structure. Also, the metadata that describe these data reside in the same structure. The DBMS provides the interface between the various database applications for organizational users and the database (or databases). The DBMS allows users to share the data and to query, access, and update the stored data.

To facilitate the sharing of data and information, Pine Valley Furniture Company uses a local area network that links employee workstations in the various departments to a database server, as shown in Figure 1-14. During the early 2000s, the company mounted a two-phase effort to introduce Internet technology. First, to improve intracompany communication and decision making, an intranet was installed that allows employees fast Web-based access to company information, including phone directories, furniture design specifications, e-mail, and so forth. In addition, Pine Valley Furniture Company also added a Web interface to some of its business applications, such as order entry, so that employees could conduct more internal business activities that require

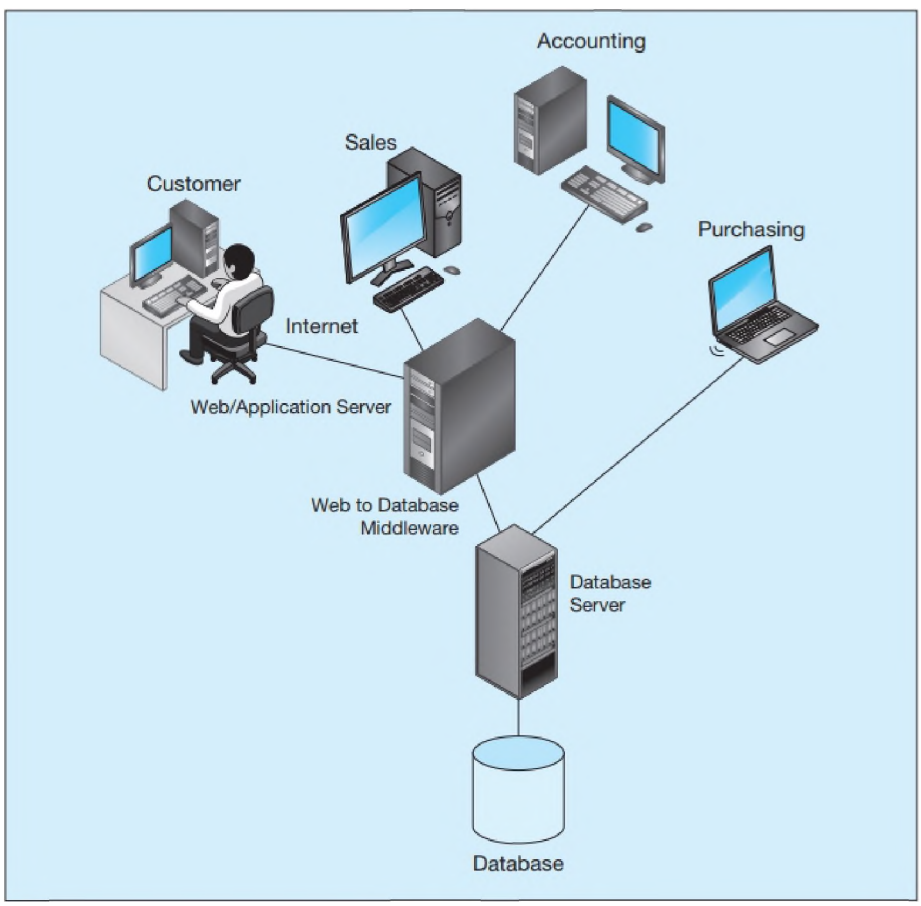


FIGURE 1-14 Computer System for Pine Valley Furniture Company

access to data in the database server through its intranet. However, most applications that use the database server still do not have a Web interface and require that the application itself be stored on employees' workstations.

Database Evolution at Pine Valley Furniture Company

A trait of a good database is that it does and can evolve! Helen Jarvis, product manager for home office furniture at Pine Valley Furniture Company, knows that competition has become fierce in this growing product line. Thus, it is increasingly important to Pine Valley Furniture that Helen be able to analyze sales of her products more thoroughly. Often these analyses are ad hoc, driven by rapidly changing and unanticipated business conditions, comments from furniture store managers, trade industry gossip, or personal experience. Helen has requested that she be given direct access to sales data with an easy-to-use interface so that she can search for answers to the various marketing questions she will generate.

Chris Martin is a systems analyst in Pine Valley Furniture's information systems development area. Chris has worked at Pine Valley Furniture for five years and has experience with information systems from several business areas within Pine Valley. With this experience, his information systems education at Western Florida University, and the extensive training Pine Valley has given him, he has become one of Pine Valley's best systems developers. Chris is skilled in data modeling and is familiar with several relational database management systems used within the firm. Because of his experience, expertise, and availability, the head of information systems has assigned Chris to work with Helen on her request for a marketing support system.

Because Pine Valley Furniture has been careful in the development of its systems, especially since adopting the database approach, the company already has databases that support its operational business functions. Thus, it is likely that Chris will be able to extract the data Helen needs from existing databases. Pine Valley's information systems architecture calls for systems such as the one Helen is requesting to be built as stand-alone databases so that the unstructured and unpredictable use of data will not interfere with the access to the operational databases needed to support efficient transaction processing systems.

Further, because Helen's needs are for data analysis, not creation and maintenance, and are personal, not institutional, Chris decides to follow a combination of prototyping and life cycle approaches in developing the system Helen has requested. This means that Chris will follow all the life cycle steps but focus his energy on the steps that are integral to prototyping. Thus, he will quickly address project planning and then use an iterative cycle of analysis, design, and implementation to work closely with Helen to develop a working prototype of the system she needs. Because the system will be personal and likely will require a database with limited scope, Chris hopes the prototype will end up being the actual system Helen will use. Chris has chosen to develop the system using Microsoft Access, Pine Valley's preferred technology for personal databases.

Project Planning

Chris begins the project by interviewing Helen. Chris asks Helen about her business area, taking notes about business area objectives, business functions, data entity types, and other business objects with which she deals. At this point, Chris listens more than he talks so that he can concentrate on understanding Helen's business area; he interjects questions and makes sure that Helen does not try to jump ahead to talk about what she thinks she needs with regard to computer screens and reports from the information system. Chris asks general questions, using business and marketing terminology as much as possible. For example, Chris asks Helen what issues she faces managing the home office products; what people, places, and things are of interest to her in her job; how far back in time she needs data to go to do her analyses; and what events occur in the business that are of interest to her. Chris pays particular attention to Helen's objectives as well as the data entities that she is interested in.

Chris does two quick analyses before talking with Helen again. First, he identifies all of the databases that contain data associated with the data entities Helen mentioned.

From these databases, Chris makes a list of all of the data attributes from these data entities that he thinks might be of interest to Helen in her analyses of the home office furniture market. Chris's previous involvement in projects that developed Pine Valley's standard sales tracking and forecasting system and cost accounting system helps him speculate on the kinds of data Helen might want. For example, the objective to exceed sales goals for each product finish category of office furniture suggests that Helen wants product annual sales goals in her system; also, the objective of achieving at least an eight percent annual sales growth means that the prior year's orders for each product need to be included. He also concludes that Helen's database must include all products, not just those in the office furniture line, because she wants to compare her line to others. However, he is able to eliminate many of the data attributes kept on each data entity. For example, Helen does not appear to need various customer data, such as address, phone number, contact person, store size, and salesperson. Chris does, though, include a few additional attributes, customer type and zip code, which he believes might be important in a sales forecasting system.

Second, from this list, Chris draws a conceptual data model (Figure 1-15) that represents the data entities with the associated data attributes as well as the major relationships among these data entities. The data model is represented using a notation called the entity-relationship model. You will learn more about this notation in Chapters 2 and 3. The data attributes of each entity Chris thinks Helen wants for the system are listed in Table 1-6. Chris lists in Table 1-6 only basic data attributes from existing databases because Helen will likely want to combine these data in various ways for the analyses she will want to do.

Analyzing Database Requirements

Prior to their next meeting, Chris sends Helen a rough project schedule outlining the steps he plans to follow and the estimated length of time each step will take. Because prototyping is a user-driven process in which the user says when to stop iterating on the new prototype versions, Chris can provide only rough estimates of the duration of certain project steps.

Chris does more of the talking at this second meeting, but he pays close attention to Helen's reactions to his initial ideas for the database application. He methodically walks through each data entity in Figure 1-15, explaining what it means and what business policies and procedures are represented by each line between entities.

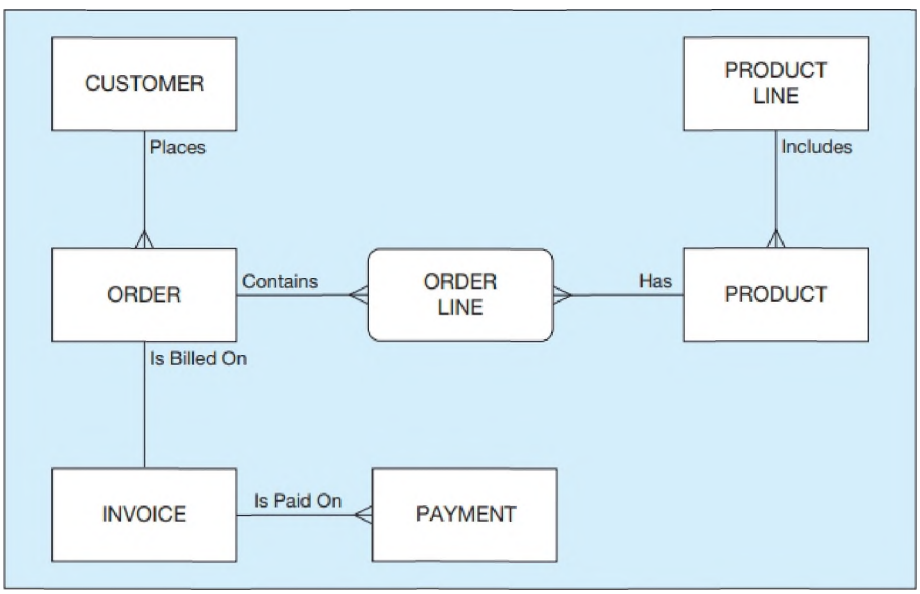


FIGURE 1-15 Preliminary data model for Home Office product line marketing support system

TABLE 1-6 Data Attributes for Entities in the Preliminary Data Model (Pine Valley Furniture Company)

Entity Type	Attribute
Customer	Customer Identifier
	Customer Name
	Customer Type
	Customer Zip Code
Product	Product Identifier
	Product Description
	Product Finish
	Product Price
	Product Cost
	Product Annual Sales Goal
	Product Line Name
Product Line	Product Line Name
	Product Line Annual Sales Goal
Order	Order Number
	Order Placement Date
	Order Fulfillment Date
	Customer Identifier
Ordered Product	Order Number
	Product Identifier
	Order Quantity
Invoice	Invoice Number
	Order Number
	Invoice Date
Payment	Invoice Number
	Payment Date
	Payment Amount

A few of the rules he summarizes are listed here:

1. Each CUSTOMER *Places* any number of ORDERs. Conversely, each ORDER *Is Placed By* exactly one CUSTOMER.
2. Each ORDER *Contains* any number of ORDER LINEs. Conversely, each ORDER LINE *Is Contained In* exactly one ORDER.
3. Each PRODUCT *Has* any number of ORDER LINEs. Conversely, each ORDER LINE *Is For* exactly one PRODUCT.
4. Each ORDER *Is Billed On* one INVOICE and each INVOICE *Is a Bill for* exactly one ORDER.

Places, *Contains*, and *Has* are called one-to-many relationships because, for example, one customer places potentially many orders and one order is placed by exactly one customer.

In addition to the relationships, Chris also presents Helen with some detail on the data attributes captured in Table 1-6. For example, Order Number uniquely identifies each order. Other data about an order that Chris thinks Helen might want to know include the date when the order was placed and the date when the order was filled. (This would be the latest shipment date for the products on the order.) Chris also explains that the Payment Date attribute represents the most recent date when the customer made any payments, in full or partial, for the order.

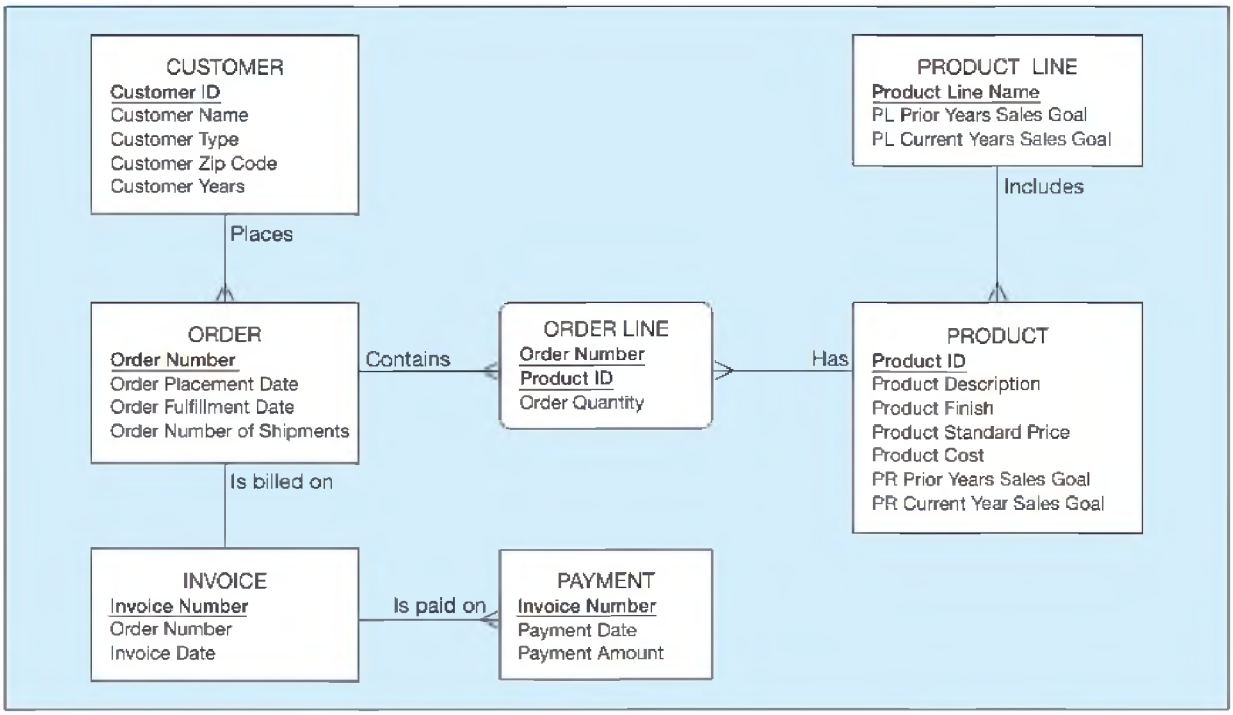
Maybe because Chris was so well prepared or so enthusiastic, Helen is excited about the possibilities, and this excitement leads her to tell Chris about some additional data she wants (the number of years a customer has purchased products from Pine Valley Furniture Company and the number of shipments necessary to fill each order). Helen also notes that Chris has only one year of sales goals indicated for a product line. She reminds him that she wants these data for both the past and current years. As she reacts to the data model, Chris asks her how she intends to use the data she wants. Chris does not try to be thorough at this point because he knows that Helen has not worked with an information set like the one being developed; thus, she may not yet be positive about what data she wants or what she wants to do with those data. Rather, Chris's objective is to understand a few ways in which Helen intends to use the data so that he can develop an initial prototype, including the database and several computer displays or reports. The final list of attributes that Helen agrees she needs appears in Table 1-7.

TABLE 1-7 Data Attributes for Entities in Final Data Model (Pine Valley Furniture Company)

Entity Type	Attribute
Customer	Customer Identifier
	Customer Name
	Customer Type
	Customer Zip Code
	Customer Years
Product	Product Identifier
	Product Description
	Product Finish
	Product Price
	Product Cost
	Product Prior Year Sales Goal
	Product Current Year Sales Goal
	Product Line Name
Product Line	Product Line Name
	Product Line Prior Year Sales Goal
	Product Line Current Year Sales Goal
Order	Order Number
	Order Placement Date
	Order Fulfillment Date
	Order Number of Shipments
	Customer Identifier
Ordered Product	Order Number
	Product Identifier
	Order Quantity
Invoice	Invoice Number
	Order Number
	Invoice Date
Payment	Invoice Number
	Payment Date
	Payment Amount

*Changes from preliminary list of attributes appear in italics.

FIGURE 1-16 Project data model for Home Office product line marketing support system



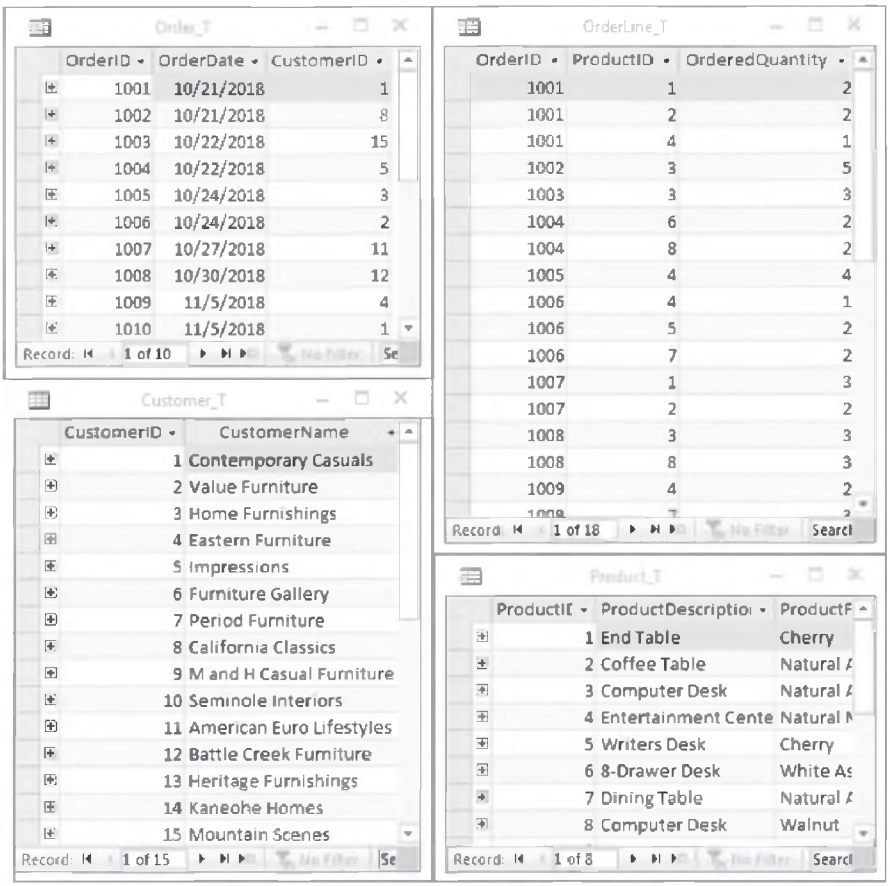
Designing the Database

Because Chris is following a prototyping methodology and the first two sessions with Helen quickly identified the data Helen might need, Chris is now ready to build a prototype. His first step is to create a project data model like the one shown in Figure 1-16. Notice the following characteristics of the project data model:

1. It is a model of the organization that provides valuable information about how the organization functions as well as important constraints.
2. The project data model focuses on entities, relationships, and business rules. It also includes attribute labels for each piece of data that will be stored in each entity.

Second, Chris translates the data model into a set of tables for which the columns are data attributes and the rows are different sets of values for those attributes. Tables are the basic building blocks of a relational database (you will learn about this in Chapter 4), which is the database style for Microsoft Access. Figure 1-17 shows four tables with sample data: Customer, Product, Order, and OrderLine. Notice that these tables represent the four entities shown in the project data model (Figure 1-16). Each column of a table represents an attribute (or characteristic) of an entity. For example, the attributes shown for Customer are CustomerID and CustomerName. Each row of a table represents an instance (or occurrence) of the entity. The design of the database also required Chris to specify the format, or properties, for each attribute (MS Access calls attributes *fields*). These design decisions were easy in this case because most of the attributes were already specified in the corporate data dictionary.

The tables shown in Figure 1-17 were created using SQL (you will learn about this in Chapters 5 and 6). Figures 1-18 and 1-19 show the SQL statements that Chris would have likely used to create the structure of the ProductLine and Product tables. It is customary to add the suffix _T to a table name. Also note that because Access does not allow for spaces between names, the individual words in the attributes from the data model have now been concatenated. Hence, Product Description in the data model has become ProductDescription in the table. Chris did this translation so that each table had



Order_T

OrderID	OrderDate	CustomerID
1001	10/21/2018	1
1002	10/21/2018	8
1003	10/22/2018	15
1004	10/22/2018	5
1005	10/24/2018	3
1006	10/24/2018	2
1007	10/27/2018	11
1008	10/30/2018	12
1009	11/5/2018	4
1010	11/5/2018	1

Record: 1 of 10

OrderLine_T

OrderID	ProductID	OrderedQuantity
1001	1	2
1001	2	2
1001	4	1
1002	3	5
1003	3	3
1004	6	2
1004	8	2
1005	4	4
1006	4	1
1006	5	2
1006	7	2
1007	1	3
1007	2	2
1008	3	3
1008	8	3
1009	4	2

Record: 1 of 18

Customer_T

CustomerID	CustomerName
1	Contemporary Casuals
2	Value Furniture
3	Home Furnishings
4	Eastern Furniture
5	Impressions
6	Furniture Gallery
7	Period Furniture
8	California Classics
9	M and H Casual Furniture
10	Seminole Interiors
11	American Euro Lifestyles
12	Battle Creek Furniture
13	Heritage Furnishings
14	Kaneohe Homes
15	Mountain Scenes

Record: 1 of 15

Product_T

ProductID	ProductDescription	ProductF
1	End Table	Cherry
2	Coffee Table	Natural A
3	Computer Desk	Natural A
4	Entertainment Cente	Natural M
5	Writers Desk	Cherry
6	8-Drawer Desk	White As
7	Dining Table	Natural A
8	Computer Desk	Walnut

Record: 1 of 8

FIGURE 1-17 Four relations (Pine Valley Furniture Company)

(a) Order and Order Line Tables

(b) Customer table

(c) Product table

```
CREATE TABLE ProductLine_T
(ProductLineID      VARCHAR (40) NOT NULL PRIMARY KEY,
PlPriorYearGoal     DECIMAL,
PlCurrentYearGoal   DECIMAL);
```

FIGURE 1-18 SQL definition of ProductLine_T table

```
CREATE TABLE Product_T
(ProductID                  NUMBER(11,0) NOT NULL PRIMARY KEY
ProductDescription          VARCHAR (50),
ProductFinish               VARCHAR (20),
ProductStandardPrice        DECIMAL(6,2),
ProductCost                 DECIMAL,
ProductPriorYearGoal        DECIMAL,
ProductCurrentYearGoal      DECIMAL,
ProductLineID               VARCHAR (40),
FOREIGN KEY                 (ProductLineID)  REFERENCES ProductLine_T (ProductLineID));
```

FIGURE 1-19 SQL definition of Product_T table

an attribute, called the table's "primary key," which will be distinct for each row in the table. The other major properties of each table are that there is only one value for each attribute in each row; if you know the value of the identifier, there can be only one value for each of the other attributes. For example, for any product line, there can be only one value for the current year's sales goal.

A final key characteristic of the relational model is that it represents relationships between entities by values stored in the columns of the corresponding tables. For example, notice that CustomerID is an attribute of both the Customer table and the Order table. As a result, you can easily link an order to its associated customer. For example, you can determine that OrderID 1003 is associated with CustomerID 1. Can you determine which ProductIDs are associated with OrderID 1004? In Chapters 5 and 6, you will also learn how to retrieve data from these tables by using SQL, which exploits these linkages. The other major decision Chris has to make about database design is how to physically organize the database to respond as fast as possible to the queries Helen will write. Because the database will be used for decision support, neither Chris nor Helen can anticipate all of the queries that will arise; thus, Chris must make the physical design choices from experience rather than precise knowledge of the way the database will be used. The key physical database design decision that SQL allows a database designer to make is on which attributes to create indexes. All primary key attributes (such as OrderNumber for the Order_T table)—those with unique values across the rows of the table—are indexed. In addition to this, Chris uses a general rule of thumb: Create an index for any attribute that has more than 10 different values and that Helen might use to segment the database. For example, Helen indicated that one of the ways she wants to use the database is to look at sales by product finish. Thus, it might make sense to create an index on the Product_T table using the Product Finish attribute.

However, Pine Valley uses only six product finishes, or types of wood, so this is not a useful index candidate. On the other hand, OrderPlacementDate (called a secondary key because there may be more than one row in the Order_T table with the same value of this attribute), which Helen also wants to use to analyze sales in different time periods, is a good index candidate.

Using the Database

Helen will use the database Chris has built mainly for ad hoc questions, so Chris will train her so that she can access the database and build queries to answer her ad hoc questions. Helen has indicated a few standard questions she expects to ask periodically. Chris will develop several types of prewritten routines (forms, reports, and queries) that can make it easier for Helen to answer these standard questions (so she does not have to program these questions from scratch).

During the prototyping development process, Chris may develop many examples of each of these routines as Helen communicates more clearly what she wants the system to be able to do. At this early stage of development, however, Chris wants to develop one routine to create the first prototype. One of the standard sets of information Helen says she wants is a list of each of the products in the Home Office product line showing each product's total sales to date compared with its current-year sales goal. Helen may want the results of this query to be displayed in a more stylized fashion—an opportunity to use a report—but for now Chris will present this feature to Helen only as a query.

The query to produce this list of products appears in Figure 1-20, with sample output in Figure 1-21. The query in Figure 1-20 uses SQL. You can see three of the six standard SQL clauses in this query: SELECT, FROM, and WHERE. SELECT indicates which attributes will be shown in the result. One calculation is also included and given the label "Sales to Date." FROM indicates which tables must be accessed to retrieve data. WHERE defines the links between the tables and indicates that results from only the Home Office product line are to be included. Only limited data are included for this example, so the Total Sales results in Figure 1-21 are fairly small, but the format is the result of the query in Figure 1-20.

```
SELECT Product.ProductID, Product.ProductDescription, Product.PRCurrentYearSalesGoal,
        (OrderQuantity * ProductPrice) AS SalesToDate
FROM Order.OrderLine, Product.ProductLine
WHERE Order.OrderNumber = OrderLine.OrderNumber
AND Product.ProductID = OrderedProduct.ProductID
AND Product.ProductID = ProductLine.ProductID
AND Product.ProductLineName = 'Home Office';
```

FIGURE 1-20 SQL query for Home Office sales-to-goal comparison

Home Office Sales to Date : Select Query

Product ID	Product Description	PR Current Year Sales Goal	Sales to Date
3	Computer Desk	$23,500.00	5625
10	96" Bookcase	$22,500.00	4400
5	Writer's Desk	$26,500.00	650
3	Computer Desk	$23,500.00	3750
7	48" Bookcase	$17,000.00	2250
5	Writer's Desk	$26,500.00	3900

FIGURE 1-21 Home Office product line sales comparison

Chris is now ready to meet with Helen again to see if the prototype is beginning to meet her needs. Chris shows Helen the system. As Helen makes suggestions, Chris is able to make a few changes online, but many of Helen's observations will have to wait for more careful work at his desk.

Space does not permit us to review the whole project to develop the Home Office marketing support system. Chris and Helen ended up meeting about a dozen times before Helen was satisfied that all the attributes she needed were in the database; that the standard queries, forms, and reports Chris wrote were of use to her; and that she knew how to write queries for unanticipated questions. Chris will be available to Helen at any time to provide consulting support when she has trouble with the system, including writing more complex queries, forms, or reports. One final decision that Chris and Helen made was that the performance of the final prototype was efficient enough that the prototype did not have to be rewritten or redesigned. Helen was now ready to use the system.

Administering the Database

The administration of the Home Office marketing support system is fairly simple. Helen decided that she could live with weekly downloads of new data from Pine Valley's operational databases into her MS Access database. Chris wrote a C# program with SQL commands embedded in it to perform the necessary extracts from the corporate databases and wrote an MS Access program in Visual Basic to rebuild the Access tables from these extracts; he scheduled these jobs to run every Sunday evening. Chris also updated the corporate information systems architecture model to include the Home Office marketing support system. This step was important so that when changes occurred to formats for data included in Helen's system, the corporate data modeling and design tools could alert Chris that changes might also have to be made in her system.

Future of Databases at Pine Valley

Although the databases currently in existence at Pine Valley adequately support the daily operations of the company, requests such as the one made by Helen have highlighted

that the current databases are often inadequate for decision support applications. For example, following are some types of questions that cannot be easily answered:

1. What is the pattern of furniture sales this year compared with the same period last year?
2. Who are our 10 largest customers, and what are their buying patterns?
3. Why can't we easily obtain a consolidated view of any customer who orders through different sales channels rather than viewing each contact as representing a separate customer?

To answer these and other questions, an organization often needs to build a separate database that contains historical and summarized information. Such a database is usually called a *data warehouse* or, in some cases, a *data mart*. Also, analysts need specialized decision support tools to query and analyze the database. One class of tools used for this purpose is called online analytical processing tools. You will learn more about data warehouses, data marts, data lakes, and related decision support tools in Chapters 9 through 11. There, you will learn of the interest in building a data warehouse that is now growing within Pine Valley Furniture Company. Who knows, maybe at some point in the near future the company will develop its own data lake for big data analysis.

Summary

Over the past two decades, there has been enormous growth in the number and importance of database applications. Databases are used to store, manipulate, and retrieve data in every type of organization. In the highly competitive environment of today, there is every indication that database technology will assume even greater importance. A course in modern database management is one of the most important courses in the information systems curriculum.

A database is an organized collection of logically related data. We define *data* as stored representations of objects and events that have meaning and importance in the user's environment. Information is data that have been processed in such a way that the knowledge of the person who uses the data increases. Both data and information may be stored in a database.

Metadata are data that describe the properties or characteristics of end-user data and the context of that data. A database management system (DBMS) is a software system that is used to create, maintain, and provide controlled access to user databases. A DBMS stores metadata in a repository, which is a central storehouse for all data definitions, data relationships, screen and report formats, and other system components. In addition to the databases enabling and supporting operational (transactional) systems, it is increasingly common that organizations maintain databases that are used for informational purposes (analytics) using either the data warehousing approach or the big data approach.

Computer file processing systems were developed early in the computer era so that computers could store, manipulate, and retrieve large files of data. These systems (still in use today) have a number of important limitations such as dependence between programs and data, data duplication, limited data sharing, and lengthy development times. The database approach was developed to overcome these limitations. This approach emphasizes the integration and sharing of data across the organization. Advantages of this approach include program-data independence, improved data sharing, minimal data redundancy, and improved productivity of application development.

Database development for transactional systems begins with enterprise data modeling, during which the range and general contents of organizational databases are established. In addition to the relationships among the data entities themselves, their relationship to other organizational planning objects, such as organizational units, locations, business functions, and information systems, also need to be established. Relationships between data entities and the other organizational planning objects can be represented at a high level by planning matrixes, which can be manipulated to understand patterns of relationships. Once the need for a database is identified, either from a planning exercise or from a specific request (such as the one from Helen Jarvis for a Home Office products marketing support system), a project team is formed to develop all elements. The project team follows a systems development process, such as the systems development life cycle or prototyping. The systems development life cycle can be represented by five methodical steps: (1) planning, (2) analysis, (3) design, (4) implementation, and (5) maintenance. Database development activities occur in each of these overlapping phases, and feedback may occur that causes a project to return to a prior phase. In prototyping, a database and its applications are iteratively refined through a close interaction of systems developers and users. Prototyping works best when the database application is small and stand-alone and a small number of users exist.

Those working on a database development project deal with three views, or schemas, for a database: (1) a conceptual schema, which provides a complete, technology-independent picture of the database; (2) an internal schema, which specifies the complete database as it will be stored in computer secondary memory in terms of a logical schema and a physical schema; and (3) an external schema or user view, which describes the database relevant to a specific set of users in terms of a set of user views combined with the enterprise data model.

Database applications can be arranged into the following categories: personal databases, multi-tiered databases, and enterprise databases. Enterprise databases include transactional databases supporting enterprise systems, data warehouses and integrated decision support databases whose content is derived from the various operational databases, and data lakes for storing large quantities of heterogeneous data for purposes that are often not predefined. Enterprise systems, such as enterprise resource planning (ERP) and customer relationship management rely heavily on enterprise databases. A modern database and the applications that use it may be located on multiple computers. Although any number of tiers may exist (from one to many), three tiers of computers relate to the client/server architecture for database processing: (1) the client tier, where database contents are presented to the user; (2) the application/Web server tier, where analyses on database contents are made and user sessions are managed; and (3) the enterprise server tier, where the data from across the organization are merged into an organizational asset.

We closed the chapter with the review of a hypothetical database development project at Pine Valley Furniture Company. This system to support marketing a Home Office furniture product line illustrated the use of a personal database management system and SQL coding for developing a retrieval-only database. The database in this application contained data extracted from the enterprise databases and then stored in a separate database on the client tier. Prototyping was used to develop this database application because the user, Helen Jarvis, had rather unstructured needs that could best be discovered through an iterative process of developing and refining the system. Also, her interest and ability to work closely with Chris was limited.

Chapter Review

Key Terms

Agile software development *25*
Conceptual schema *23*
Constraint *15*
Data *7*
Data independence *13*
Data lake *34*
Data model *11*
Data modeling and design tools *19*
Data warehouse *33*
Database *6*
Database application *10*
Database management system (DBMS) *13*
Enterprise data modeling *20*
Enterprise resource planning (ERP) *33*
Entity *11*
Information *7*
Logical schema *23*
Metadata *8*
Physical schema *23*
Project *27*
Prototyping *24*
Relational database *12*
Repository *19*
Systems development life cycle (SDLC) *21*
User view *14*

Review Questions

1-1. Define each of the following terms:

a. data
b. information
c. metadata
d. enterprise resource planning
e. data warehouse
f. constraint
g. database
h. entity
i. database management system
j. data lake
k. systems development life cycle
l. prototyping
m. enterprise data model
n. conceptual data model
o. logical data model
p. physical data model

1-2. Match the following terms and definitions:

_____ agile software development	a. data placed in context or summarized
_____ database application	b. application program(s)
_____ constraint	c. iterative, focused on working software and customer collaboration
_____ repository	d. a graphical model that shows the high-level entities for the organization and the relationships among those entities
_____ metadata	e. a real-world person or object about which the organization wishes to maintain data
_____ data warehouse	f. includes data definitions and constraints

_____ information	g. centralized storehouse for all data definitions
_____ user view	h. separation of data description from programs
_____ database management system	i. a business management system that integrates all functions of the enterprise
_____ data independence	j. logical description of portion of database
_____ entity	k. a software application that is used to create, maintain, and provide controlled access to user databases
_____ enterprise resource planning	l. a rule that cannot be violated by database users
_____ systems development life cycle	m. integrated decision support database
_____ prototyping	n. consist of the enterprise data model and multiple user views
_____ enterprise data model	o. a rapid approach to systems development
_____ conceptual schema	p. consists of two data models: a logical model and a physical model
_____ internal schema	q. a comprehensive description of business data
_____ external schema	r. a structured, step-by-step approach to systems development

1-3. Contrast the following terms:
- a. data dependence; data independence
- b. structured data; unstructured data
- c. metadata; data
- d. repository; database
- e. entity; enterprise data model
- f. data warehouse; data lake
- g. personal databases; multi-tiered databases
- h. systems development life cycle; prototyping
- i. enterprise data model; conceptual data model
- j. prototyping; agile software development

1-4. List five disadvantages of file processing systems.

1-5. Contrast transactional and analytical data management approaches.

1-6. What are the key differences between data warehousing and big data approaches to analytical data management?

1-7. List the nine major components in a database system environment.

1-8. How are relationships between tables expressed in a relational database?

1-9. What does the term *data independence* mean, and why is it an important goal?

1-10. List 10 potential benefits of the database approach over conventional file systems.

1-11. List five costs or risks associated with the database approach.

1-12. List nine key components that comprise a typical database environment.

1-13. Figure 1-5 specifies categories for Operational and Informational data management systems. Describe the main difference between these two categories.

1-14. Based on Figure 1-5, what are the four perspectives from which you will explore transactional systems in this book? What are the main competencies associated with each of these perspectives.

1-15. In the multi-tiered database architecture, is it possible for there to be no database on a particular tier? If not, why? If yes, give an example.

1-16. Specify the difference between database solutions supporting enterprise databases and departmental multi-tiered databases.

1-17. Describe the potential benefits of data lakes compared to other enterprise databases.

1-18. Name the five phases of the traditional systems development life cycle and explain the purpose and deliverables of each phase.

1-19. In which of the five phases of the SDLC do database development activities occur?

1-20. How does the use of an agile methodology affect decisions regarding data management?

1-21. Are there procedures and processes that are common to the use of SDLC, prototyping, and agile methodologies? Explain any that you can identify and then indicate why the methodologies are considered to be different even though fundamental procedures and processes are still included.

1-22. Explain the differences between user views, a conceptual schema, and an internal schema as different perspectives of the same database.

1-23. In the three-schema architecture:
- a. The view of a manager or other type of user is called the _____schema.
- b. The view of the data architect or data administrator is called the _____schema.
- c. The view of the database administrator is called the _____schema.

1-24. Revisit the section titled "Developing a Database Application for Pine Valley Furniture Company." What phase(s) of the database development process (Figure 1-9) do the activities that Chris performs in the following subsections correspond to:
- a. Project planning
- b. Analyzing database requirements
- c. Designing the database
- d. Using the database
- e. Administering the database

1-25. Why might Pine Valley Furniture Company need a data warehouse?

1-26. As the ability to handle large amounts of data improves, describe three business areas where these very large databases are being used effectively.

Problems and Exercises

1-27. For each of the following pairs of related entities, indicate whether (under typical circumstances) there is a one-to-many or a many-to-many relationship. Then, using the shorthand notation introduced in the text, draw a diagram for each of the relationships.
- a. STUDENT and COURSE (students register for courses)
- b. BOOK and BOOK COPY (books have copies)
- c. COURSE and SECTION (courses have sections)

d. SECTION and ROOM (sections are scheduled in rooms)
e. INSTRUCTOR and COURSE
f. COURSE and SEMESTER
g. MEAL and COURSE

1-28. Reread the definitions for *data* and *database* in this chapter. Database management systems only recently began to include the capability to store and retrieve more than numeric and textual data. What special data storage, retrieval, and maintenance capabilities do images, sound, video, and other advanced data types require that are not required or are simpler with numeric and textual data?

1-29. Table 1-1 shows example metadata for a set of data items. Identify three other columns for these data (i.e., three other metadata characteristics for the listed attributes) and complete the entries of the table in Table 1-1 for these three additional columns.

1-30. In the section "Disadvantages of File Processing Systems," the statement is made that the disadvantages of file processing systems can also be limitations of databases, depending on how an organization manages its databases. First, why do organizations create multiple databases, not just one all-inclusive database supporting all data processing needs? Second, what organizational and personal factors are at work that might lead an organization to have multiple, independently managed databases (and, hence, not completely follow the database approach)?

1-31. Consider the data needs of the student-run newspaper in your university or high school. What are the data entities of this enterprise? List and define each entity. Then develop an enterprise data model (such as Figure 1-3a) showing these entities and important relationships between them.

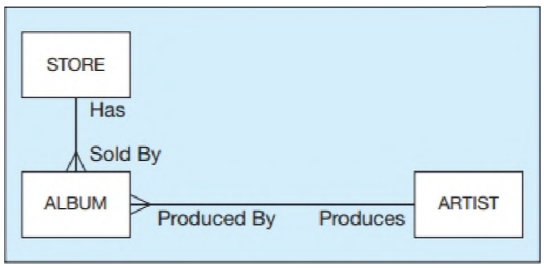
1-32. A driver's license bureau maintains a database of licensed drivers. State whether each of the following represents data or metadata. If it represents data, state whether it is structured or unstructured data. If it represents metadata, state whether it is a fact describing a property of data or a fact describing the context of data.

a. Driver's name, address, and birth date
b. The fact that the driver's name is a 30-character field
c. A photo image of the driver
d. The fact that birth date is stored using the column name BirthDate
e. An image of the driver's fingerprint
f. The make and serial number of the scanning device that was used to scan the fingerprint
g. The resolution (in megapixels) of the camera that was used to photograph the driver
h. The fact that the driver's birth date must precede today's date by at least 16 years
i. A 10-second video clip in which the driver states his or her name

1-33. Great Lakes Insurance would like to implement a relational database for both its in-house and its outside agents. The outside agents will use smartphones, tablets, and notebook computers to keep track of customers and policy information. Based on what you have learned in this chapter, what type (or types) of database(s) would you recommend for this application?

1-34. Figure 1-22 shows an enterprise data model for a music store.

FIGURE 1-22 Data model for Problem and Exercise 1-34



a. What is the relationship between Album and Store (one-to-one, many-to-many, or one-to-many)?
b. What is the relationship between Artist and Album?
c. Do you think there should be a relationship between Artist and Store? Describe at least one possible scenario that could justify such a relationship.

1-35. Consider Figure 1-12, which depicts a hypothetical multi-tiered database architecture. Identify potential duplications of data across all the databases listed on this figure. What problems might arise because of this duplication? Does this duplication violate the principles of the database approach outlined in this chapter? Why or why not?

1-36. What is your reaction to the representation of the systems development life cycle included in this chapter? Explain any problems you have with it.

1-37. List three additional entities that might appear in an enterprise data model for Pine Valley Furniture Company (Figure 1-3a).

1-38. Consider your business school or other academic unit as a business enterprise.

a. Define several major data entity types and draw a preliminary enterprise data model (similar in notation to Figure 1-3a).
b. Would your business school or academic unit benefit from a multi-tiered architecture for data management? Why or why not?

1-39. Contrast the top-down nature of database development during conceptual data modeling with the bottom-up nature of database development during logical database design. What major differences exist in the type of information considered in each of these two database development steps?

1-40. The objective of the prototyping systems development methodology is to rapidly build and rebuild an information system as the user and systems analyst learn from use of the prototype what features should be included in the evolving information system. Because the final prototype does not have to become the working system, where do you think would be an ideal location to develop a prototype: on a personal computer, departmental multi-tiered system, or enterprise database? Does your answer depend on any assumptions?

1-41. Explain the differences between an enterprise data model and a conceptual data model. How many databases does each represent? What scope of the organization does each address? What are other salient differences?

1-42. Is it possible that during the physical database design and creation step of database development you might want to return to the logical database design activity? Why or why not? If it is possible, give an example of what might arise during physical database design and creation that would

cause you to want to reconsider the conceptual and external database designs from prior steps.

1-43. Consider an organization with which you frequently interact, such as a bank, credit card company, university, or insurance company, from which you receive several computer-generated messages, such as monthly statements, transaction slips, and so forth. Depict the data included in each message you receive from the organization as its own user view; use the notation of Figure 1-3a to represent these views. Now combine all of these user views together into one conceptual data model, also using the notation of Figure 1-3a. What did you observe about the process of combining the different user views? Were there inconsistencies across the user views? Once you have created the conceptual data model, would you like to change anything about any of the user views?

1-44. Consider Figure 1-15. Explain the meaning of the line that connects CUSTOMER to ORDER and the line that connects ORDER to INVOICE. What does this say about how Pine Valley Furniture Company does business with its customers?

1-45. Consider the project data model shown in Figure 1-16.

a. Create a textual description of the diagrammatic representation shown in the figure. Ensure that the description captures the rules/constraints conveyed by the model.

b. In arriving at the requirements document, what aspect of the diagram did you find was the most difficult to describe? Which parts of the requirements do you still consider to be a little ambiguous? In your opinion, what is the underlying reason for this ambiguity?

1-46. Answer the following questions concerning Figures 1-18 and 1-19:

a. What will be the field size for the ProductLineName field in the Product table? Why?

b. In Figure 1-19, how is the ProductID field in the Product table specified to be required? Why is it a required attribute?

c. In Figure 1-19, explain the function of the FOREIGN KEY definition.

d. In Figure 1-19, explain the purpose of the NOT NULL specification associated with ProductID.

1-47. Consider the SQL query in Figure 1-20.

a. How is Sales to Date calculated?

b. How would the query have to change if Helen Jarvis wanted to see the results for all of the product lines, not just the Home Office product line?

c. The part of the query starting with WHERE (the so called WHERE clause) has two different types of conditions—the last one is clearly different from the first three. Explain how.

1-48. Helen Jarvis wants to determine the most important customers for Home Office products. She requests a listing of year-to-date total dollar sales for each customer who bought these products, as revealed by invoiced payments. The list is to be sorted in descending order so that the largest customer heads the list.

a. Look at Figure 1-16 and determine what entities are required to produce this list.

b. Which entities will be involved in the SQL query that will give Helen the information she needs?

1-49. In this chapter, we described four important data models and their properties: enterprise, conceptual, logical, and physical. In the following table, summarize the important properties of these data models by entering a Y (for yes) or an N (for no) in each cell of the table.

Table for Problem and Exercise 1-49

	All Entities?	All Attributes?	Technology Independent?	DBMS Independent?	Record Layouts?
Enterprise					
Conceptual					
Logical					
Physical					

Field Exercises

For Questions 1-50 through 1-58, choose an organization with a fairly extensive information systems department and set of information system applications. You should choose one with which you are familiar, possibly your employer, your university, or an organization where a friend works. Use the same organization for each question.

1-50. Investigate whether the organization follows more of a traditional file processing approach or the database approach to organizing data. How many different databases does the organization have? Try to draw a figure, similar to Figure 1-2, to depict some or all of the files and databases in this organization.

1-51. Talk with a database administrator or designer from the organization. What type of metadata does this organization maintain about its databases? Why did the organization choose to keep track of these and not other metadata? What tools are used to maintain these metadata?

1-52. Determine the company's use of intranet, extranet, or other Web-enabled business processes. For each type of process, determine its purpose and the database management system that is being used in conjunction with the networks. Ask what the company's plans are for the next year with regard to using intranets, extranets, or the Web in its business activities. Ask what new skills the company is looking for in order to implement these plans.

1-53. Consider a major database in this organization, such as one supporting customer interactions, accounting, or manufacturing. What is the architecture for this database? Is the organization using some form of client/server

architecture? Interview information systems managers in this organization to find out why they chose the architecture for this database.

1-54. Interview systems and database analysts at this organization. Ask them to describe their systems development process. Which does it resemble more: the systems development life cycle or prototyping? Do they use methodologies similar to both? When do they use their different methodologies? Explore the methodology used for developing applications to be used through the Web. How have they adapted their methodology to fit this new systems development process?

1-55. Interview a systems analyst or database analyst and ask questions about the typical composition of an information systems development team. Specifically, what role does a database analyst play in project teams? Is a database analyst used throughout the systems development process, or is the database analyst used only at selected points?

1-56. Interview a systems analyst or database analyst and ask questions about how that organization uses data modeling and design tools in the systems development process. Concentrate your questions on how data modeling and design tools are used to support data modeling and database design and how the data modeling and design tool's repository maintains the information collected about data, data characteristics, and data usage. If multiple data modeling and design tools are used on one or many projects, ask how the organization attempts to integrate data models and data definitions. Finally, inquire how satisfied the systems and database analysts are with data modeling and design tool support for data modeling and database design.

1-57. Interview one person from a key business function, such as finance, human resources, or marketing. Concentrate your questions on the following items: How does he or she retrieve data needed to make business decisions? From what kind of system (personal database, enterprise system, or data warehouse) are the data retrieved? How often are these data accessed? Is this person satisfied with the data available for decision making? If not, what are the main challenges in getting access to the right data?

1-58. You may want to keep a personal journal of ideas and observations about database management while you are studying this book. Use this journal to record comments you hear, summaries of news stories or professional articles you read, original ideas or hypotheses you create, uniform resource locators (URLs) for and comments about Web sites related to databases, and questions that require further analysis. Keep your eyes and ears open for anything related to database management. Your instructor may ask you to turn in a copy of your journal from time to time in order to provide feedback and reactions. The journal is an unstructured set of personal notes that will supplement your class notes and can stimulate you to think beyond the topics covered within the time limitations of most courses.

References

Anderson-Lehman, R., H. J. Watson, B. Wixom, and J. A. Hoffer. 2004. "Continental Airlines Flies High with Real-Time Business Intelligence." *MIS Quarterly Executive* 3,4 (December).

Codd, E. F. 1970. "A Relational Model of Data for Large Shared Data Banks." *Communications of the ACM* 13,6 (June): 377–87.

Fowler, M. 2005. "The New Methodology." Available at **www.martinfowler.com/articles/newMethodology.html**.

Gray, J. 1996. "Data Management: Past, Present, and Future." *IEEE Computer* 29,10: 38–46.

Grimes, S. 1998. "Object/Relational Reality Check." *Database Programming & Design* 11,7 (July): 26–33.

Groenfeldt, Tom. 2013. "Kroger Knows Your Shopping Patterns Better Than You Do." *Forbes Online*. Available at **www.forbes.com/sites/tomgroenfeldt/2013/10/28/kroger-knows-your-shopping-patterns-better-than-you-do**.

Henderson, D., B. Champlin, D. Coleman, P. Cupoli, J. Hoffer, L. Howarth, et al. 2005. "Model Curriculum Framework for Post Secondary Education Programs in Data Resource Management." Data Management Association International Foundation Committee on the Advancement of Data Management in Post Secondary Institutions Sub Committee on Curriculum Framework Development, DAMA International Foundation.

IBM. 2011. "The Essential CIO: Insights from the 2011 IBM Global CIO Study." Available at **https://www-935.ibm.com/services/c-suite/cio/study**.

Jordan, A. 1996. "Data Warehouse Integrity: How Long and Bumpy the Road?" *Data Management Review* 6,3 (March): 35–37.

Laskowski, N. (2014). "Ten Big Data Case Studies in a Nutshell." Available at **http://searchcio.techtarget.com/opinion/Ten-big-data-case-studies-in-a-nutshell**.

Long, D. 2005. ".Net Overview." Tampa Bay Technology Leadership Association, May 19.

Manyika, J., M. Chui, B. Brown, J. Bughin, R. Dobbs, C. Roxburgh, and A. H. Byers. 2011. "Big Data: The Next Frontier for Innovation, Competition and Productivity." McKinsey Global Institute, May.

Mullins, C. S. 2002. *Database Administration: The Complete Guide to Practices and Procedures*. New York: Addison-Wesley.

Ritter, D. 1999. "Don't Neglect Your Legacy." *Intelligent Enterprise* 2,5 (March 30): 70–72.

Valacich, J. S., and J. F. George. 2016. *Modern Systems Analysis and Design*. 8th ed. Upper Saddle River, NJ: Prentice Hall.

Further Reading

Ballou, D. P., and G. K. Tayi. 1999. "Enhancing Data Quality in Data Warehouse Environments." *Communications of the ACM* 42,1 (January): 73–78.

Date, C. J. 1998. "The Birth of the Relational Model, Part 3." *Intelligent Enterprise* 1,4 (December 10): 45–48.

Kimball, R., and M. Ross. 2002. The *Data Warehouse Toolkit: The Complete Guide to Dimensional Data Modeling*. 2nd ed. New York: Wiley.

Ritter, D. 1999. "The Long View." *Intelligent Enterprise* 2,12 (August 24): 58–67.

Silverston, L. 2001a. *The Data Model Resource Book, Vol. 1: A Library of Universal Data Models for All Enterprises*. New York: Wiley.

Silverston, L. 2001b. *The Data Model Resource Book, Vol. 2: A Library of Data Models for Specific Industries*. New York: Wiley.

Web Resources

www.webopedia.com An online dictionary and search engine for computer terms and Internet technology.

www.techrepublic.com A portal site for information technology professionals that users can customize to their own particular interests.

www.zdnet.com A portal site where users can review recent articles on information technology subjects.

www.information-management.com *DM Review* magazine Web site, with the tagline "Covering Business Intelligence, Integration and Analytics." Provides a comprehensive list of links to relevant resource portals in addition to providing many of the magazine articles online.

www.dbta.com *Data Base Trends and Applications* magazine Web site. Addresses enterprise-level information issues.

http://databases.about.com A comprehensive site with many feature articles, links, interactive forum, chat rooms, and so forth.

http://groups.google.com/group/comp.software-eng?lnk=gsch&hl=en The software engineering archives for a Google group that focuses on software engineering and related topics. This site contains many links that you may want to explore.

www.acinet.org/acinet America's Career InfoNet, which provides information about careers, outlook, requirements, and so forth.

www.collegegrad.com/salaries/index.shtml A site for finding recent salary information for a wide range of careers, including database-related careers.

www.essentialstrategies.com/publications/methodology/zachman.htm David Hay's Web site, which has considerable information on universal data models as well as how database development fits into the Zachman information systems architecture.

www.inmondatasystems.com Web site for one of the pioneers of data warehousing.

www.agilemanifesto.org Web site that explains the viewpoints of those who created "The Manifesto for Agile Software Development."

CASE

Forondo Artist Management Excellence Inc.

Case Description

FAME (Forondo Artist Management Excellence) Inc. is an artist management company that represents classical music artists (only soloists) both nationally and internationally. FAME has more than 500 artists under its management and wants to replace its spreadsheet-based system with a new state-of-the-art computerized information system.

Their core business idea is simple: FAME finds paid performance opportunities for the artists whom it represents and receives a 10 to 30 percent royalty for all the fees the artists earn (the royalties vary by artist and are based on a contract between FAME and each artist). To accomplish this objective, FAME needs technology support for several tasks. For example, it needs to keep track of prospective artists. FAME receives information regarding possible new artists both from promising young artists themselves and as recommendations from current artists and a network of music critics. FAME employees collect information regarding promising prospects and maintain that information in the system. When FAME management decides to propose a contract to a prospect, it first sends the artist a tentative contract, and if the response is positive, a final contract is mailed to the prospect. New contracts are issued annually to all artists.

FAME markets its artists to opera houses and concert halls (customers); in this process, a customer normally requests a specific artist for a specific date. FAME maintains the artists' calendars and responds back based on the requested artist's availability. After the performance, FAME sends an invoice to the customer, who sends a payment to FAME (note that FAME requires a security deposit, but you do not need to capture that aspect in your system). Finally, FAME pays the artist after deducting its own fee.

Currently, FAME has no IT staff. Its technology infrastructure consists of a variety of desktops, printers, laptops, tablets, and smartphones all connected with a simple wired and wireless network. A local company manages this infrastructure and provides the required support.

Martin Forondo, the owner of FAME, has commissioned your team to design and develop a database application. In his e-mail soliciting your help, he provides the following information:

E-mail from Martin Forondo, Owner

My name is Martin Forondo, and I am the owner and founder of FAME. I have built this business over the past 30 years together with my wonderful staff and I am very proud of my company. We are in the business of creating bridges between the finest classical musicians and the best concert venues and opera houses of the world and finding the best possible opportunities for the musicians we represent. It is very important for us to provide the best possible service to the artists we represent.

It used to be possible to run our business without any technology, particularly when the number of the artists we represented was much smaller than it currently is. The situation is, however, changing, and we seem to have a need to get some technical help for us. At this moment we have about 500 different artists and every one of them is very special for us. We have about 20 artist managers who are responsible for different numbers of artists; some of them have only 10, but some manage as many as 30 artists. The artist managers really keep this business going, and each of them has the ultimate responsibility for the artists for whom they work. Every manager has an administrative assistant to help him or her with daily routine work—the managers are focusing on relationship building and finding new talent for our company. The managers report to me but they are very independent in their work, and I am very pleased that I only very seldom have to deal with operational issues related to the managers' work. By the way, I also have my own artists (only a few but, of course, the very best within the company, if I may say so).

As I said, we find performance opportunities for the artists and, in practice, we organize their entire professional lives—of course, in agreement with them. Our main source of revenue consists of the royalties we get when we are successful in finding a performance opportunity for an artist: We get up to 30 percent of the fee paid to an artist (this is agreed separately with every artist and is a central part of our contract with the artist). Of course, we get the money only after the artist has successfully completed the performance; thus, if an artist has to cancel the performance, for example, because of illness, we will not get anything. Within the company the policy is very clear: A manager gets 50 percent of the royalties we earn based on the work of the artists he or she manages, and the remaining 50 percent will be used to cover administrative costs (including the administrative assistants' salaries), rent, electricity, computer systems, accounting services, and, of course, my modest profits. Each manager pays their own travel expenses from their 50 percent. Keeping track of the revenues by manager and by artist is one of the most important issues in running this business. Right now, we take care of it manually, which occasionally leads to unfortunate mistakes and a lot of extra work trying to figure out what the problem is. It is amazing how difficult simple things can sometimes become.

When thinking about the relationship between us and an artist whom we represent, it is important to remember that the artists are ultimately responsible for a lot of the direct expenses we pay when working for them, such as flyers, photos, prints of photos, advertisements, and publicity mailings. We don't, however, charge for phone calls made on behalf of a certain artist, but rather this is part of the general overhead. We would like to settle the accounts with each of the artists once per month so that either we pay them what we owe after our expenses are deducted from their portion of the fee or they pay us, if the expenses are higher than a particular month's fees. The artists take care of their own travel expenses, meals, etc.

From my perspective, the most important benefit of a new system would be an improved ability to know real-time how my managers are serving their artists. Are they finding opportunities for them and how good are the opportunities, what are the fees that their artists have earned and what are they projected to be, etc. Furthermore, the better the system

could predict the future revenues of the company, the better for me. Whatever we could do with the system to better cultivate new relationships between promising young artists, it would be great. I am not very computer savvy; thus, it is essential that the system will be easy to use.

Project Questions

1-59. Create a memo describing your initial analysis of the situation at FAME as it relates to the design of the database application. Write this as though you are writing a memo to Martin Forondo. Ensure that your memo addresses the following points:

a. Your approach to addressing the problem at hand (e.g., specify the systems development life cycle or whatever approach you plan on taking).

b. What will the new system accomplish? What functions will it perform? Which organizational goals will it support?

c. What will be the benefits of using the new system? Use concrete examples to illustrate this. Outline general categories of costs and resources needed for the project and implementation of the ultimate system.

d. A time line/road map for the project.

e. Questions, if any, you have for Mr. Forondo for which you need answers before you would be willing to begin the project.

1-60. Create an enterprise data model that captures the data needs of FAME. Use a notation similar to the one shown in Figure 1-4.

PART II

Database Analysis and Logical Design

AN OVERVIEW OF PART II

The first step in database development is database analysis, in which you determine user requirements for data and develop data models to represent those requirements. The first two chapters in Part II describe in depth the de facto standard for conceptual data modeling—entity-relationship (E-R) diagramming. A conceptual data model represents data from the viewpoint of the organization, independent of any technology that will be used to implement the model.

In Chapter 2 ("Modeling Data in the Organization") you will learn how to identify and document business rules, which are the policies and rules about the operation of a business that a data model represents. Characteristics of good business rules are described, and the process you will follow for gathering business rules is discussed. General guidelines for naming and defining elements of a data model are presented within the context of business rules.

Chapter 2 introduces the notations and main constructs of this modeling technique, including entities, relationships, and attributes; for each construct, you will see specific guidelines for naming and defining these elements of a data model. You will learn how to distinguish between strong and weak entity types and the use of identifying relationships. You will also learn about different types of attributes, including required versus optional attributes, simple versus composite attributes, single-valued versus multivalued attributes, derived attributes, and identifiers. You will contrast relationship types and instances and understand associative entities. You will study relationships of various degrees, including unary, binary, and ternary relationships. You will learn how to model the various relationship cardinalities that arise in modeling situations. You will study the common problem of how to model time-dependent data. Finally, you will see how multiple relationships can be defined between a given set of entities. The E-R modeling concepts are illustrated with an extended example for Pine Valley Furniture Company. This final example, as well as a few other examples throughout the chapter, is presented using Microsoft Visio, which shows how many data modeling tools represent data models.

Chapter 3 ("The Enhanced E-R Model") presents advanced concepts in E-R modeling; you will often need these additional modeling features to cope with the increasingly complex business environment encountered in organizations today.

The most important modeling construct incorporated in the enhanced entity-relationship (EER) diagram is supertype/subtype relationships. This facility allows you to model a general entity type (called a supertype) and then subdivide it into several specialized entity types called subtypes. For example, sports cars and

sedans are subtypes of automobiles. You will learn to use a simple notation for representing supertype/subtype relationships and several refinements. You will study generalization and specialization as two contrasting techniques for identifying supertype/subtype relationships. Supertype/subtype notation is necessary for the increasingly popular universal data model, which is motivated and explained in Chapter 3. The comprehensiveness of a well-documented relationship can be overwhelming, so we introduce a technique called entity clustering for simplifying the presentation of an E-R diagram to meet the needs of a given audience.

The concept of patterns has become a central element of many information systems development methodologies. The notion is that there are reusable component designs that you can combine and tailor to meet new information system requests. In the database world, these patterns are called universal data models, prepackaged data models, or logical data models. These patterns can be purchased or may be inherent in a commercial off-the-shelf package, such as an ERP or CRM application. Increasingly, it is from these patterns that new databases are designed. In Chapter 3, we describe the usefulness of such patterns and outline a modification of the database development process when such patterns are the starting point. Universal industry or business function data models extensively use the extended E-R diagramming notations introduced in this chapter.

There is another, alternative notation for data modeling: the Unified Modeling Language class diagrams for systems developed using object-oriented technologies. This technique is presented in a supplement found on this book's Web site. It is possible to read this supplement immediately after Chapter 3 if you want to compare these alternative, but conceptually similar, approaches.

Before you can implement a database, you must map the conceptual data model into a data model that is compatible with the database management system to be used. The activities of database design transform the requirements for data storage developed during database analysis into specifications to guide database implementation. There are two forms of specifications:

1. Logical specifications, which map the conceptual requirements into the data model associated with a specific database management system.
2. Physical specifications, which indicate all the parameters for data storage. Because database implementation often proceeds before these parameters can be finalized, we will delay discussing physical database design until later in this text.

In Chapter 4 ("Logical Database Design and the Relational Model"), you will study logical database design, with special emphasis on the relational data model. Logical database design is the process of transforming the conceptual data model (described in Chapters 2 and 3) into a logical data model. Most database management systems in use today are based on the relational data model, so this data model is the basis for our discussion of logical database design.

In Chapter 4, you will learn the important terms and concepts for this model, including *relation, primary key* and *surrogate primary key, foreign key, anomaly, normal form, normalization, functional dependency, partial functional dependency,* and *transitive dependency*. You next study the process of transforming an E-R model to the relational model. Many modeling tools support this transformation; however, it is important that you understand the underlying principles and procedures. You then will see in detail the important concepts of normalization (the process of designing well-structured relations). Appendix B, on the book's Web site, includes further discussion of normalization. Finally, you will learn how to merge relations from separate logical design activities (e.g., different groups within a large project team) while avoiding common pitfalls that may occur in this process. Finally, you will study enterprise keys, which make relational keys distinct across relations.

The conceptual and logical data modeling concepts presented in the three chapters in Part II provide the foundation for your career in database analysis and design. As a database analyst, you will be expected to apply the E-R notation and relational normalization in modeling user requirements for data and information.

Modeling Data in the Organization

LEARNING OBJECTIVES

After studying this chapter, you should be able to:

- Concisely define each of the following key terms: **business rule, term, fact, entity-relationship model (E-R model), entity-relationship diagram (E-R diagram), entity, entity type, entity instance, strong entity type, weak entity type, identifying owner, identifying relationship, attribute, required attribute, optional attribute, composite attribute, simple attribute, multivalued attribute, derived attribute, identifier, composite identifier, relationship type, relationship instance, associative entity, degree, unary relationship, binary relationship, ternary relationship, cardinality constraint, minimum cardinality, maximum cardinality,** and **time stamp.**
- State reasons why many system developers and business leaders believe that data modeling is the most important part of the systems development process with a high return on investment.
- Write good names and definitions for entities, relationships, and attributes.
- Distinguish unary, binary, and ternary relationships and give a common example of each.
- Model each of the following constructs in an E-R diagram: composite attribute, multivalued attribute, derived attribute, associative entity, identifying relationship, and minimum and maximum cardinality constraints.
- Draw an E-R diagram to represent common business situations.
- Convert a many-to-many relationship to an associative entity type.
- Model simple time-dependent data using time stamps and relationships in an E-R diagram.

Visit *www.pearsonhighered.com/hoffer* to view the accompanying video for this chapter.

INTRODUCTION

You have already been introduced to modeling data and the entity-relationship (E-R) data model through simplified examples in Chapter 1. (You may want to review, for example, the E-R models in Figures 1-3 and 1-4.) In this chapter, we formalize data modeling based on the powerful concept of business rules and describe the E-R data model in detail. This chapter begins your journey of learning how to design and use databases. It is exciting to create information systems that run organizations and help people do their jobs well.

Your excitement can, of course, lead to mistakes if you are not careful to follow best practices. Embarcadero Technologies, a leader in database design tools and processes, has identified "seven deadly sins" that are the culprits underlying the failure to follow best practices of database design (Embarcadero Technologies, 2014):

1. Poor or missing documentation for database(s) in production (this will be addressed in Chapters 2 and 3 via the topics of business rules and data modeling with entity relationship diagramming)
2. Little or no normalization (this will be a central topic of Chapter 4 on the relational data model)
3. Not treating the data model like a living, breathing organism (we encourage you through exercises and projects to develop database designs in phases and to realize that requirements evolve and emerge over time; in other words, design for change)
4. Improper storage of reference data (we will address this briefly in subsequent chapters in Parts II and III of this text)
5. Not using foreign keys or check constraints (this will be a significant topic in Chapters 4 and 8)
6. Not using domains and naming standards (we emphasize naming standards in Chapter 2 and provide guidelines on good standards to adopt in your practice)
7. Not choosing primary keys (we emphasize entity identifiers in Chapter 2 and address considerations in choosing primary keys in Chapters 4 and 8)

A specific quote from the referenced Embarcadero report that we believe sets the tone for the importance of what we present in this and subsequent chapters is "In the data management arena, you may constantly hear from data professionals that if you don't get the data right, nothing else matters. However, the business focus on applications often overshadows the priority for a well-organized database design. The database just comes along for the ride as the application grows in scope and functionality." That is, often in practice there is an emphasis on functionality over architecture and engineering. If the architecture and engineering are bad, you can never achieve the functionality the organization requires. So, let's begin at the beginning for the architecture and engineering of a database with business rules.

Business rules, the foundation of data models, are derived from policies, procedures, events, functions, and other business objects, and they state constraints on the organization. Business rules represent the language and fundamental structure of an organization (Hay, 2003). Business rules formalize the understanding of the organization by organization owners, managers, and leaders with that of information systems architects.

Business rules are important in data modeling because they govern how data are handled and stored. Examples of basic business rules are data names and definitions. This chapter explains guidelines you can follow for the clear naming and definition of data objects in a business. In terms of conceptual data modeling, you must provide names and definitions for the main data objects: entity types (e.g., Customer), attributes (e.g., Customer Name), and relationships (e.g., Customer Places Orders). Other business rules may state constraints on these data objects. These constraints can be captured in a data model, such as an E-R diagram, and associated documentation. Additional business rules govern the people, places, events, processes, networks, and objectives of the organization, which are all linked to the data requirements through other system documentation.

After decades of use, the E-R model remains the mainstream approach for conceptual data modeling. Its popularity stems from factors such as relative ease of use, widespread computer-aided software engineering (CASE) tool support, and the belief that entities and relationships are natural modeling concepts in the real world.

The E-R model is most used as a tool for communications between database designers (that is, you!) and end users during the analysis phase of database development (described in Chapter 1). The E-R model is used to construct a conceptual data model, which is a representation of the structure and constraints of a database that is independent of software (such as a database management system).

Some database professionals introduce terms and concepts peculiar to the relational data model when discussing E-R modeling; the relational data model is

the basis for most database management systems in use today. In particular, they recommend that the E-R model be completely normalized, with full resolution of primary and foreign keys. However, we believe that this forces a premature commitment to the relational data model. In today's database environment, the database may be implemented with a mixture of relational and nonrelational technology. Therefore, we defer discussion of normalization concepts to Chapter 4.

The E-R model was introduced in a key article by Chen (1976), in which he described the main constructs of the E-R model—entities and relationships—and their associated attributes. The model has subsequently been extended to include additional constructs by Chen and others; for example, see Teorey et al. (1986) and Storey (1991). The E-R model continues to evolve, but unfortunately there is not yet a standard notation for E-R modeling. Because data modeling software tools are now commonly used by professional data modelers, we adopt for use in this text a variation of the notation used in professional modeling tools. Appendix A, found on this book's Web site, will help you translate between our notation and other popular E-R diagramming notations.

As said in a once-popular travel service TV commercial, "we are doing important stuff here." Many systems developers believe that data modeling is the most important part of the systems development process for the following reasons (Valacich & George, 2016):

1. The characteristics of data captured during data modeling are crucial in the design of databases, programs, and other system components. The facts and rules captured during the process of data modeling are essential in assuring data integrity in an information system.
2. Data rather than processes are the most complex aspect of many modern information systems and hence require a central role in structuring system requirements. Often the goal is to provide a rich data resource that might support any type of information inquiry, analysis, and summary.
3. Data tend to be more stable than the business processes that use that data. Thus, an information system design that is based on a data orientation should have a longer useful life than one based on a process orientation.

Of course, we are all eager to build something new, so data modeling may still seem like a costly and unnecessary activity that simply delays getting to "the real work." If the above reasons for why data modeling is important are not enough to convince you, the following reasons are derived from what one industry leader demonstrates with examples are the benefits from and return on investment for data modeling (Haughey, 2010):

- Data modeling facilitates interaction/communication between designer, application programmer, and end user, thus reducing misunderstandings and improving the thoroughness of resultant systems; this is accomplished, in part, by providing a simplified (visual) understanding of data (data model) with agreed upon supporting documentation (metadata).
- Data modeling can foster understanding of the organization (rules) for which the data model is being developed; consistency and completeness of rules can be verified; otherwise, it is possible to create systems that are incorrect or inconsistent and unable to accommodate changes in user requirements (such as processing certain transactions or producing specific reports).
- The value of data modeling can be demonstrated as an overall savings in maintenance or development costs by determining the right requirements before the more costly steps of software development and hardware acquisition; further, data models can be reused in whole or in part on multiple projects, which can result in significant savings to any organization by reducing the costs for building redundant systems or complex interfaces between systems.
- Data modeling results in improved data quality because of consistent business data definitions (metadata) and hence greater accuracy of reporting and

consistency across systems and less organizational confusion; data modeling across the organization results in everyone having the same understanding of the same data.

- Data modeling reduces the significant costs of moving and translating data from one system to another; decisions can be made about the efficacy of sharing or having redundant data because data modeling creates a consistent enterprise-wide understanding of data; and when data must be transferred to achieve efficiencies, they do not have to be collected (possibly with inconsistencies) in multiple systems or from multiple sources.

To state it as simply as possible, the value of data modeling can be summarized by the phrase "measure twice, cut once."

In an actual work environment, you may not have to develop a data model from scratch. Because of the increased acceptance of packaged software (e.g., enterprise resource planning with a predefined data model) and purchased business area or industry data models (which we discuss in Chapter 3), your job of data modeling has a jump start. This is good because such components and patterns give you a starting point based on generally accepted practices. However, your job is not done for several reasons:

1. There are still many times when a new, custom-built application is being developed along with the associated database. The business rules for the business area supported by this application need to be modeled.
2. Purchased applications and data models need to be customized for your particular setting. Predefined data models tend to be very extensive and complex; hence, they require significant data modeling skill to tailor the models to be effective and efficient in a given organization. Although this effort can be much faster, thorough, and accurate than starting from scratch, the ability to understand a particular organization to match the data model to its business rules is an essential task.

In this chapter, you will learn the main features of E-R modeling, using common notation and conventions. We begin with a sample E-R diagram, including the basic constructs of the E-R model—entities, attributes, and relationships—and then we introduce the concept of business rules, which is the foundation for all the data modeling constructs. We define three types of entities that are common in E-R modeling: strong entities, weak entities, and associative entities; a few more entity types are defined in Chapter 3. We also define several important types of attributes, including required and optional attributes, single- and multivalued attributes, derived attributes, and composite attributes. We then introduce three important concepts associated with relationships: the degree of a relationship, the cardinality of a relationship, and participation constraints in a relationship. We conclude with an extended example of an E-R diagram for Pine Valley Furniture Company.

THE E-R MODEL: AN OVERVIEW

Entity-relationship model (E-R model)

A logical representation of the data for an organization or for a business area, using entities for categories of data and relationships for associations between entities.

Entity-relationship diagram (E-R diagram, or ERD)

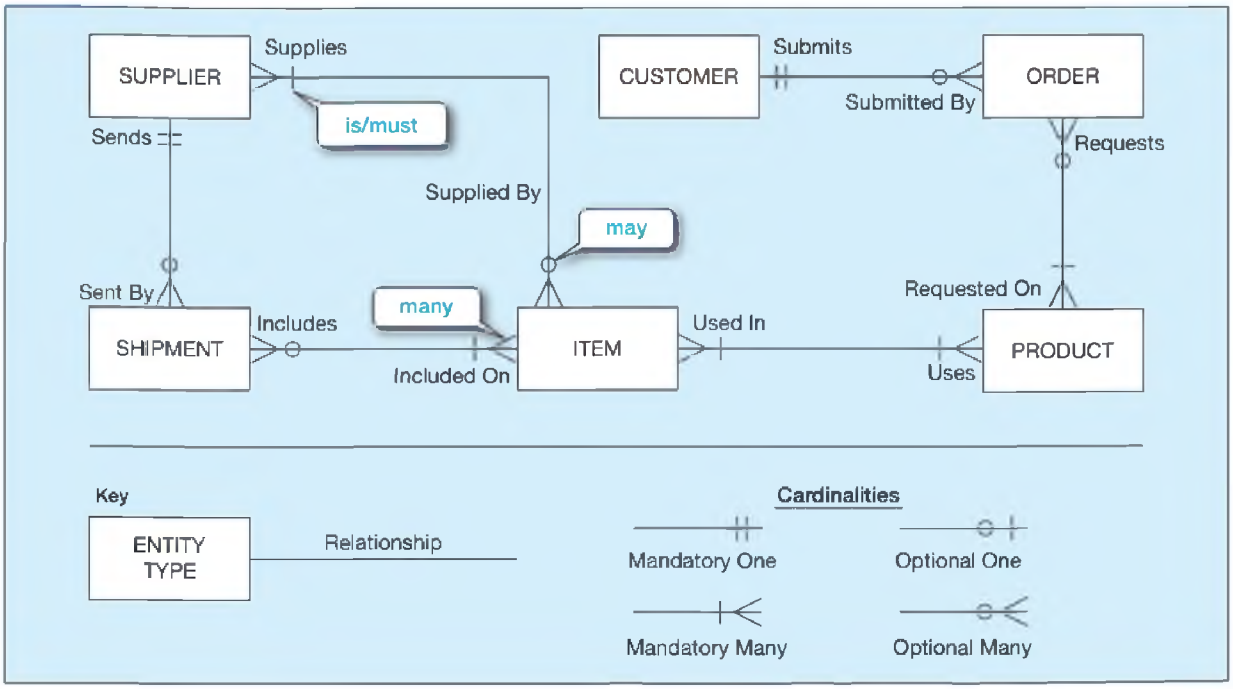
A graphical representation of an entity-relationship model.

An **entity-relationship model (E-R model)** is a detailed, logical representation of the data for an organization or for a business area. The E-R model is expressed in terms of entities in the business environment, the relationships (or associations) among those entities, and the attributes (or properties) of both the entities and their relationships. An E-R model is normally expressed as an **entity-relationship diagram (E-R diagram, or ERD)**, which is a graphical representation of an E-R model.

Sample E-R Diagram

To jump-start your understanding of E-R diagrams, Figure 2-1 presents a simplified E-R diagram for a small furniture manufacturing company, Pine Valley Furniture Company. (This figure, which does not include attributes, is often called an *enterprise data model*, which we introduced in Chapter 1; we use cartoon caption-like bubbles here and in subsequent figures to help you understand the meaning of symbols used in E-R diagrams.)

FIGURE 2-1 Sample E-R diagram



A number of suppliers supply and ship different items to Pine Valley Furniture. The items are assembled into products that are sold to customers who order the products. Each customer order may include one or more lines corresponding to the products appearing on that order.

The diagram in Figure 2-1 shows the entities and relationships for this company. (Attributes are omitted to simplify the diagram for now.) Entities (the objects of the organization) are represented by the rectangle symbol, whereas relationships between entities are represented by lines connecting the related entities. The entities in Figure 2-1 include the following:

CUSTOMER	A person or an organization that has ordered or might order products. *Example*: L. L. Fish Furniture.
PRODUCT	A type of furniture made by Pine Valley Furniture that may be ordered by customers. Note that a product is not a specific bookcase because individual bookcases do not need to be tracked. *Example*: A 6-foot, 5-shelf, oak bookcase called O600.
ORDER	The transaction associated with the sale of one or more products to a customer and identified by a transaction number from sales or accounting. *Example*: The event of L. L. Fish buying one product O600 and four products O623 on September 10, 2018.
ITEM	A type of component that goes into making one or more products and can be supplied by one or more suppliers. *Example*: A 4-inch ball-bearing caster called I-27–4375.
SUPPLIER	Another company that may provide items to Pine Valley Furniture. *Example*: Sure Fasteners, Inc.
SHIPMENT	The transaction associated with items received in the same package by Pine Valley Furniture from a supplier. All items in a shipment appear on one bill-of-lading document. *Example*: The receipt of 300 I-27-4375 and 200 I-27-4380 items from Sure Fasteners, Inc., on September 9, 2018.

Note that it is important to clearly define, as metadata, each entity. For example, it is important to know that the CUSTOMER entity includes persons or organizations that have not yet purchased products from Pine Valley Furniture. It is common for different departments in an organization to have different meanings for the same term (homonyms). For example, Accounting may designate as customers only those persons or organizations that have ever made a purchase, thus excluding potential customers, whereas Marketing designates as customers anyone they have contacted or who has purchased from Pine Valley Furniture or any known competitor. An accurate and thorough ERD without clear metadata may be interpreted in different ways by different people. We outline good naming and definition conventions as we formally introduce E-R modeling throughout this chapter.

The symbols at the end of each line on an ERD specify relationship cardinalities, which represent how many entities of one kind relate to how many entities of another kind. On examining Figure 2-1, we can see that these cardinality symbols express the following business rules:

1. A SUPPLIER may supply many ITEMs (by "may supply," we mean the supplier may not supply any items). Each ITEM is supplied by any number of SUPPLIERs (by "is supplied," we mean that the item must be supplied by at least one supplier). See annotations in Figure 2-1 that correspond to underlined words.
2. Each ITEM must be used in the assembly of at least one PRODUCT and may be used in many products. Conversely, each PRODUCT must use one or more ITEMs.
3. A SUPPLIER may send many SHIPMENTs. However, each shipment must be sent by exactly one SUPPLIER. Notice that sends and supplies are separate concepts. A SUPPLIER may be able to supply an item but may not yet have sent any shipments of that item.
4. A SHIPMENT must include one (or more) ITEMs. An ITEM may be included on several SHIPMENTs.
5. A CUSTOMER may submit any number of ORDERs. However, each ORDER must be submitted by exactly one CUSTOMER. Given that a CUSTOMER may not have submitted any ORDERs, some CUSTOMERs must be potential, inactive, or some other customer possibly without any related ORDERs.
6. An ORDER must request one (or more) PRODUCTs. A given PRODUCT may not be requested on any ORDER or may be requested on one or more orders.

There are actually two business rules for each relationship, one for each direction from one entity to the other. Note that each of these business rules roughly follows a certain grammar:

<entity> <minimum cardinality> <relationship> <maximum cardinality> <entity>

For example, rule 5 is

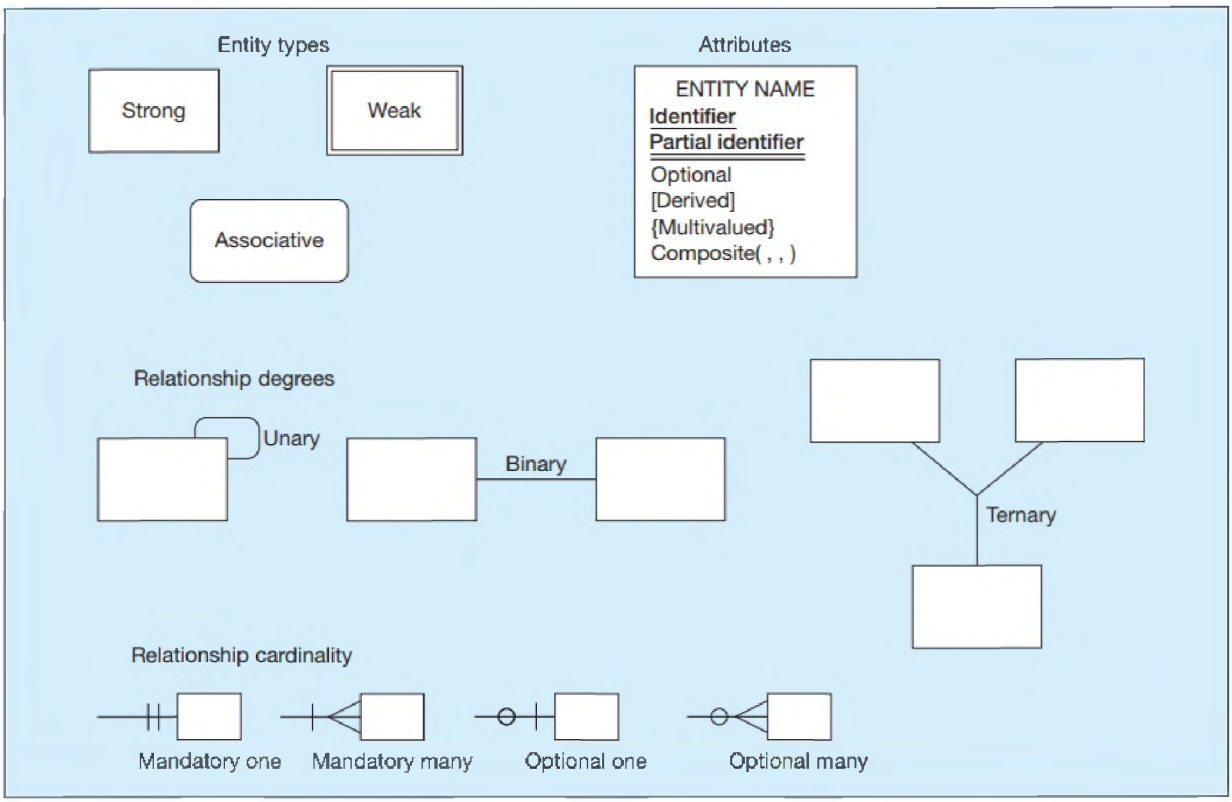
<CUSTOMER> <may> <Submit> <any number> <ORDER>

This grammar gives you a standard way to put each relationship into a natural English business rule statement.

E-R Model Notation

The notation we use for E-R diagrams is shown in Figure 2-2. As indicated in the previous section, there is no industry-standard notation (in fact, you saw a slightly simpler notation in Chapter 1). The notation in Figure 2-2 combines most of the desirable features of the different notations that are commonly used in E-R drawing tools today and also allows us to model accurately most situations that are encountered in practice. You will learn additional notation for enhanced E-R models (including class-subclass relationships) in Chapter 3.

FIGURE 2-2 Basic E-R notation



In many situations, however, a simpler E-R notation is sufficient. Most drawing tools, either stand-alone ones such as Microsoft Visio or SmartDraw (which we use in the video associated with this chapter) or those in CASE tools such as Oracle Designer, ERwin, or SAP PowerDesigner, do not show all the entity and attribute types we use. It is important to note that any notation requires special annotations, not always present in a diagramming tool, to show all the business rules of the organizational situation you are modeling. We will use the Visio notation for a few examples throughout the chapter and at the end of the chapter so that you can see the differences. Appendix A, found on this book's Web site, illustrates the E-R notation from several commonly used guidelines and diagramming tools. This appendix may help you translate between the notations in the text and the notations you use in classes.

MODELING THE RULES OF THE ORGANIZATION

Now that you have an example of a data model in mind, let's step back and consider more generally what a data model is representing. You will see in this and the next chapter how to use data models, in particular the E-R notation, to document rules and policies of an organization. *In fact, documenting rules and policies of an organization that govern data is exactly what data modeling is all about.* Business rules and policies govern creating, updating, and removing data in an information processing and storage system; thus, they must be described along with the data to which they are related. For example, the policy "every student in the university must have a faculty adviser" forces data (in a database) about each student to be associated with data about some student adviser. Also, the statement "a student is any person who has applied for admission or taken a course or training program from any credit or noncredit unit of the university" not only defines the concept of "student" for a particular university but also states a policy of that university (e.g., implicitly, alumni are students, and a high school student

who attended a college fair but has not applied is not a student, assuming the college fair is not a noncredit training program).

Business rules and policies are not universal; for example, different universities may have different policies for student advising and may include different types of people as students. Also, the rules and policies of an organization may change (usually slowly) over time; a university may decide that a student does not have to be assigned a faculty adviser until the student chooses a major.

Your job as a database analyst is to:

- Identify and understand those rules *that govern data*.
- Represent those rules so that they can be unambiguously understood by information systems developers and users.
- Implement those rules in database technology.

Data modeling is an important tool in this process. Because the purpose of data modeling is to document business rules about data, we introduce the discussion of data modeling and the E-R notation with an overview of business rules. Data models cannot represent all business rules (and do not need to, because not all business rules govern data); data models along with associated documentation and other types of information system models (e.g., models that document the processing of data) represent all business rules that must be enforced through information systems.

Overview of Business Rules

Business rule

A statement that defines or constrains some aspect of the business. It is intended to assert business structure or to control or influence the behavior of the business.

A **business rule** is "a statement that defines or constrains some aspect of the business. It is intended to assert business structure or to control or influence the behavior of the business ... rules prevent, cause, or suggest things to happen" (GUIDE Business Rules Project, 1997). For example, the following two statements are common expressions of business rules that affect data processing and storage:

- "A student may register for a section of a course only if he or she has successfully completed the prerequisites for that course."
- "A preferred customer qualifies for a 10 percent discount, unless he has an overdue account balance."

Most organizations (and their employees) today are guided by thousands of combinations of such rules. In the aggregate, these rules influence behavior and determine how the organization responds to its environment (Gottesdiener, 1997; von Halle, 1997). Capturing and documenting business rules is an important, complex task. Thoroughly capturing and structuring business rules, then enforcing them through database technologies, helps ensure that information systems work right and that users of the information understand what they enter and see.

THE BUSINESS RULES PARADIGM The concept of business rules has been used in information systems for some time. There are many software products that help organizations manage their business rules (e.g., IBM WebSphere ILOG JRules). In the database world, it has been more common to use the related term *integrity constraint* when referring to such rules. The intent of this term is somewhat more limited in scope, usually referring to maintaining valid data values and relationships in the database.

A business rules approach is based on the following premises:

- Business rules are a core concept in an enterprise because they are an expression of business policy and guide individual and aggregate behavior. Well-structured business rules can be stated in natural language for end users and in a data model for systems developers.
- Business rules can be expressed in terms that are familiar to end users. Thus, users can define and then maintain their own rules.
- Business rules are highly maintainable. They are stored in a central repository, and each rule is expressed only once, then shared throughout the organization. Each rule is discovered and documented only once, to be applied in all systems development projects.

- Enforcement of business rules can be automated through the use of software that can interpret the rules and enforce them using the integrity mechanisms of the database management system (Moriarty, 2000).

Although much progress has been made, the industry has not realized all of these objectives to date (Owen, 2004). Possibly the premise with greatest potential benefit is "Business rules are highly maintainable." The ability to specify and maintain the requirements for information systems as a set of rules has considerable power when coupled with an ability to generate automatically information systems from a repository of rules. Automatic generation and maintenance of systems will not only simplify the systems development process but also will improve the quality of systems.

Scope of Business Rules

In this chapter and the next, we are concerned with business rules that impact only an organization's databases. Most organizations have a host of rules and/or policies that fall outside this definition. For example, the rule "Friday is business casual dress day" may be an important policy statement, but it has no immediate impact on databases. In contrast, the rule "A student may register for a section of a course only if he or she has successfully completed the prerequisites for that course" is within our scope because it constrains the transactions that may be processed against the database. In particular, it causes any transaction that attempts to register a student who does not have the necessary prerequisites to be rejected. Some business rules cannot be represented in common data modeling notation; those rules that cannot be represented in a variation of an E-R diagram are stated in natural language, and some can be represented in the relational data model, which we describe in Chapter 4.

GOOD BUSINESS RULES Whether stated in natural language, a structured data model, or other information systems documentation, a business rule will have certain characteristics if it is to be consistent with the premises outlined previously. These characteristics are summarized in Table 2-1. These characteristics will have a better chance of being satisfied if a business rule is defined, approved, and owned by business, not technical, people. Businesspeople become stewards of the business rules. You, as the database analyst, facilitate the surfacing of the rules and the transformation of ill-stated rules into ones that satisfy the desired characteristics.

TABLE 2-1 Characteristics of a Good Business Rule

Characteristic	Explanation
Declarative	A business rule is a statement of policy, not how policy is enforced or conducted; the rule does not describe a process or implementation but rather describes what a process validates.
Precise	With the related organization, the rule must have only one interpretation among all interested people, and its meaning must be clear.
Atomic	A business rule marks one statement, not several; no part of the rule can stand on its own as a rule (i.e., the rule is indivisible, yet sufficient).
Consistent	A business rule must be internally consistent (i.e., not containing conflicting statements) and must be consistent with (and not contradict) other rules.
Expressible	A business rule must be able to be stated in natural language, but it will be stated in a structured natural language so that there is no misinterpretation.
Distinct	Business rules are not redundant, but a business rule may refer to other rules (especially to definitions).
Business-oriented	A business rule is stated in terms businesspeople can understand, and because it is a statement of business policy, only businesspeople can modify or invalidate a rule; thus, a business rule is owned by the business.

Source: Based on Gottesdiener (1999) and Plotkin (1999).

GATHERING BUSINESS RULES Business rules appear (possibly implicitly) in descriptions of business functions, events, policies, units, stakeholders, and other objects. You can find these descriptions in interview notes from individual and group information systems requirements collection sessions, organizational documents (e.g., personnel manuals, policies, contracts, marketing brochures, and technical instructions), and other sources. Rules are identified by asking questions about the who, what, when, where, why, and how of the organization. Usually, a data analyst has to be persistent in clarifying initial statements of rules because initial statements may be vague or imprecise (what some people have called "business ramblings"). Thus, precise rules are formulated from an iterative inquiry process. You should be prepared to ask such questions as "Is this always true?" "Are there special circumstances when an alternative occurs?" "Are there distinct kinds of that person?" "Is there only one of those or are there many?" and "Is there a need to keep a history of those, or is the current data all that is useful?" Such questions can be useful for surfacing rules for each type of data modeling construct we introduce in this chapter and the next.

Data Names and Definitions

Fundamental to understanding and modeling data are naming and defining data objects. Data objects must be named and defined before they can be used unambiguously in a model of organizational data. In the E-R notation you will learn in this chapter, you have to give entities, relationships, and attributes clear and distinct names and definitions.

DATA NAMES We will provide specific guidelines for naming entities, relationships, and attributes as we develop the entity-relationship data model, but there are some general guidelines about naming any data object. Data names should (Salin, 1990):

- ***Relate to business, not technical (hardware or software), characteristics;*** so, Customer is a good name, but File10, Bit7, and Payroll Report Sort Key are not good names.
- ***Be meaningful,*** almost to the point of being self-documenting (i.e., the definition will refine and explain the name without having to state the essence of the object's meaning); you should avoid using generic words such as *has*, *is*, *person*, or *it*.
- ***Be unique*** from the name used for every other distinct data object; words should be included in a data name if they distinguish the data object from other similar data objects (e.g., Home Address versus Campus Address).
- ***Be readable,*** so that the name is structured as the concept would most naturally be said (e.g., Grade Point Average is a good name, whereas Average Grade Relative To A, although possibly accurate, is an awkward name).
- ***Be composed of words taken from an approved list;*** each organization often chooses a vocabulary from which significant words in data names must be chosen (e.g., maximum is preferred, never upper limit, ceiling, or highest); alternative, or alias names, also can be used as can approved abbreviations (e.g., CUST for CUSTOMER), and you may be encouraged to use the abbreviations so that data names are short enough to meet maximum length limits of database technology.
- ***Be repeatable,*** meaning that different people or the same person at different times should develop exactly or almost the same name; this often means that there is a standard hierarchy or pattern for names (e.g., the birth date of a student would be Student Birth Date and the birth date of an employee would be Employee Birth Date).
- ***Follow a standard syntax,*** meaning that the parts of the name should follow a standard arrangement adopted by the organization.

Salin (1990) suggests that you develop data names by:

1. Preparing a definition of the data. (We talk about definitions next.)
2. Removing insignificant or illegal words (words not on the approved list for names); note that the presence of AND and OR in the definition may imply that

two or more data objects are combined, and you may want to separate the objects and assign different names.
3. Arranging the words in a meaningful, repeatable way.
4. Assigning a standard abbreviation for each word.
5. Determining whether the name already exists and, if so, adding other qualifiers that make the name unique.

You will see examples of good data names as we develop a data modeling notation in this chapter.

DATA DEFINITIONS A definition (sometimes called a *structural assertion*) is considered a type of business rule (GUIDE Business Rules Project, 1997). A definition is an explanation of a term or a fact. A **term** is a word or phrase that has a specific meaning for the business. Examples of terms are *course*, *section*, *rental car*, *flight*, *reservation*, and *passenger*. Terms are often the keywords used to form data names. Terms must be defined carefully and concisely. However, there is no need to define common terms such as *day*, *month*, *person*, or *television*, because these terms are understood without ambiguity by most persons.

Term
A word or phrase that has a specific meaning for the business.

A **fact** is an association between two or more terms. A fact is documented as a simple declarative statement that relates terms. Examples of facts that are definitions are the following (the defined terms are underlined):

Fact
An association between two or more terms.

- "A course is a module of instruction in a particular subject area." This definition associates two terms: *module of instruction* and *subject area*. We assume that these are common terms that do not need to be further defined.
- "A customer may request a model of car from a rental branch on a particular date." This fact, which is a definition of *model rental request*, associates the four underlined terms (GUIDE Business Rules Project, 1997). Three of these terms are business-specific terms that would need to be defined individually (date is a common term).

A fact statement places no constraints on instances of the fact. For example, it is inappropriate in the second fact statement to add that a customer may not request two different car models on the same date. Such constraints are separate business rules.

GOOD DATA DEFINITIONS We will illustrate good definitions for entities, relationships, and attributes as we develop the E-R notation in this and the next chapters. There are, however, some general guidelines to follow (Aranow, 1989):

- Definitions (and all other types of business rules) are gathered from the same sources as all requirements for information systems. Thus, systems and data analysts should be looking for data objects and their definitions as these sources of information systems requirements are studied.
- Definitions will usually be accompanied by diagrams, such as E-R diagrams. The definition does not need to repeat what is shown on the diagram but rather supplement the diagram.
- Definitions will be stated in the singular and explain what the data element is, not what it is not. A definition will use commonly understood terms and abbreviations and stand alone in its meaning and not embed other definitions within it. It should be concise and concentrate on the essential meaning of the data, but it may also state such characteristics of a data object as:
  - Subtleties.
  - Special or exceptional conditions.
  - Examples.
  - Where, when, and how the data are created or calculated in the organization.
  - Whether the data are static or change over time.
  - Whether the data are singular or plural in their atomic form.
  - Who determines the value for the data.
  - Who owns the data (i.e., who controls the definition and usage).

- Whether the data are optional or whether empty (what we will call null) values are allowed.
- Whether the data can be broken down into more atomic parts or are often combined with other data into some more composite or aggregate form.

If not included in a data definition, these characteristics need to be documented elsewhere, where other metadata are stored.

- A data object should not be added to a data model, such as an E-R diagram, until after it has been carefully defined (and named) and there is agreement on this definition. But expect the definition of the data to change once you place the object on the diagram because the process of developing a data model tests your understanding of the meaning of data. (In other words, *modeling data is an iterative process.*)

There is an unattributed phrase in data modeling that highlights the importance of good data definitions: "The person who controls the meaning of data controls the data." It might seem that obtaining concurrence in an organization on the definitions to be used for the various terms and facts should be relatively easy. However, this is usually far from the case. In fact, it is likely to be one of the most difficult challenges you will face in data modeling or, for that matter, in any other endeavor. It is not unusual for an organization to have multiple definitions (perhaps a dozen or more) for common terms such as *customer* or *order*.

To illustrate the problems inherent in developing definitions, consider a data object of Student found in a typical university. A sample definition for Student is "a person who has been admitted to the school and who has registered for at least one course during the past year." This definition is certain to be challenged because it is probably too narrow. A person who is a student typically proceeds through several stages in relationship with the school, such as the following:

1. Prospect—some formal contact, indicating an interest in the school.
2. Applicant—applies for admission.
3. Admitted applicant—admitted to the school and perhaps to a degree program.
4. Matriculated student—registers for at least one course.
5. Continuing student—registers for courses on an ongoing basis (no substantial gaps).
6. Former student—fails to register for courses during some stipulated period (now may reapply).
7. Graduate—satisfactorily completes some degree program (now may apply for another program).

Imagine the difficulty of obtaining consensus on a single definition in this situation! It would seem you might consider three alternatives:

1. ***Use multiple definitions to cover the various situations.*** This is likely to be highly confusing if there is only one entity type, so this approach is not recommended (multiple definitions are not good definitions). It might be possible to create multiple entity types, one for each student situation. However, because there is likely considerable similarity across the entity types, the fine distinctions between the entity types may be confusing, and the data model will show many constructs.
2. ***Use a very general definition that will cover most situations.*** This approach may necessitate adding additional data about students to record a given student's actual status. For example, data for a student's status, with values of prospect, applicant, and so forth, might be sufficient. On the other hand, if the same student could hold multiple statuses (e.g., prospect for one degree and matriculated for another degree), this might not work.
3. ***Consider using multiple, related data objects for Student.*** For example, we could create a general entity type for Student and then other specific entity types for kinds of students with unique characteristics. We describe the conditions that suggest this approach in Chapter 3.

MODELING ENTITIES AND ATTRIBUTES

The basic constructs of the E-R model are entities, relationships, and attributes. As shown in Figure 2-2, the model allows numerous variations for each of these constructs. The richness of the E-R model allows designers to model real-world situations accurately and expressively, which helps account for the popularity of the model.

Entities

An **entity** is a person, a place, an object, an event, or a concept in the user environment about which the organization wishes to maintain data. Thus, an entity has a singular noun name. Some examples of each of these *kinds* of entities follow:

Entity
A person, a place, an object, an event, or a concept in the user environment about which the organization wishes to maintain data.

Person:	EMPLOYEE, STUDENT, PATIENT
Place:	STORE, WAREHOUSE, STATE
Object:	MACHINE, BUILDING, AUTOMOBILE
Event:	SALE, REGISTRATION, RENEWAL
Concept:	ACCOUNT, COURSE, WORK CENTER

ENTITY TYPE VERSUS ENTITY INSTANCE There is an important distinction between entity types and entity instances. An **entity type** is a collection of entities that share common properties or characteristics. Each entity type in an E-R model is given a name. Because the name represents a collection (or set) of items, it is always singular. We use capital letters for names of entity type(s). In an E-R diagram, the entity name is placed inside the box representing the entity type (see Figure 2-1).

Entity type
A collection of entities that share common properties or characteristics.

An **entity instance** is a single occurrence of an entity type. Figure 2-3 illustrates the distinction between an entity type and two of its instances. An entity type is described just once (using metadata) in a database, whereas many instances of that entity type may be represented by data stored in the database. For example, there is one EMPLOYEE entity type in most organizations, but there may be hundreds (or even thousands) of instances of this entity type stored in the database. We often use the single term *entity* rather than *entity instance* when the meaning is clear from the context of our discussion.

Entity instance
A single occurrence of an entity type.

ENTITY TYPE VERSUS SYSTEM INPUT, OUTPUT, OR USER A common mistake people make when they are learning to draw E-R diagrams, especially if they are already familiar with data process modeling (such as data flow diagramming), is to confuse data entities with other elements of an overall information systems model. A simple rule to avoid such confusion is that *a true data entity will have many possible instances, each with a distinguishing characteristic, as well as one or more other descriptive pieces of data.*

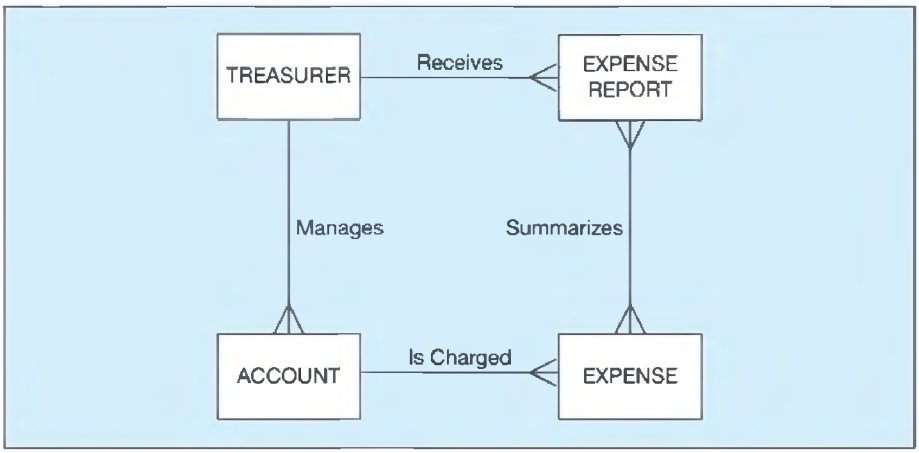
Entity type: EMPLOYEE

Attributes	Attribute Data Type	Example Instance	Example Instance
Employee Number	CHAR (10)	64217836	53410197
Name	CHAR (25)	Michelle Brady	David Johnson
Address	CHAR (30)	100 Pacific Avenue	450 Redwood Drive
City	CHAR (20)	San Francisco	Redwood City
State	CHAR (2)	CA	CA
Zip Code	CHAR (9)	98173	97142
Date Hired	DATE	03-21-1992	08-16-1994
Birth Date	DATE	06-19-1968	09-04-1975

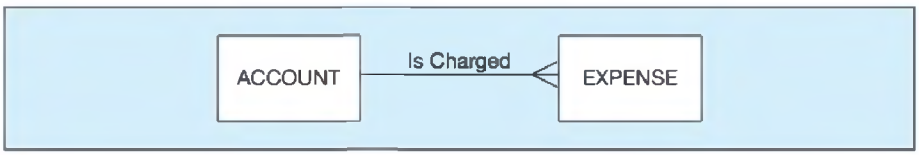
FIGURE 2-3 Entity type EMPLOYEE with two instances

FIGURE 2-4 Example of inappropriate entities

(a) System user (Treasurer) and output (Expense Report) shown as entities



(b) E-R diagram with only the necessary entities



Consider Figure 2-4a, which might be drawn to represent a database needed for a college sorority's expense system. (For simplicity in this and some other figures, we show only one name for a relationship.) In this situation, the sorority treasurer manages accounts, receives expense reports, and records expense transactions against each account. However, do we need to keep track of data about the Treasurer (the TREASURER entity type) and her supervision of accounts (the Manages relationship) and receipt of reports (the Receives relationship)? The Treasurer is the person entering data about accounts and expenses and receiving expense reports. That is, she is a user of the database. Because there is only one Treasurer, TREASURER data do not need to be kept. Further, is the EXPENSE REPORT entity necessary? Because an expense report is computed from expense transactions and account balances, it is the result of extracting data from the database and received by the Treasurer. Even though there will be multiple instances of expense reports given to the Treasurer over time, data needed to compute the report contents each time are already represented by the ACCOUNT and EXPENSE entity types.

Another key to understanding why the ERD in Figure 2-4a might be in error is the nature of the *relationship names*, Receives and Summarizes. These relationship names refer to business activities that transfer or translate data, not to simply the association of one kind of data with another kind of data. The simple E-R diagram in Figure 2-4b shows entities and a relationship that would be sufficient to handle the sorority expense system as described here. See Problem and Exercise 2-43 for a variation on this situation.

STRONG VERSUS WEAK ENTITY TYPES Most of the basic entity types to identify in an organization are classified as strong entity types. A **strong entity type** is one that exists independently of other entity types. (Some data modeling software, in fact, use the term *independent entity*.) Examples include STUDENT, EMPLOYEE, AUTOMOBILE, and COURSE. Instances of a strong entity type always have a unique characteristic (called an *identifier*)—that is, an attribute or a combination of attributes that uniquely distinguish each occurrence of that entity.

Strong entity type
An entity that exists independently of other entity types.

In contrast, a **weak entity type** is an entity type whose existence depends on some other entity type. (Some data modeling software, in fact, use the term *dependent entity*, and some data modeling tools make no distinction between strong and weak entities.) A weak entity type has no business meaning in an E-R diagram without the entity on which it depends. The entity type on which the weak entity type depends is called the **identifying owner** (or simply *owner* for short). A weak entity type does not typically

Weak entity type
An entity type whose existence depends on some other entity type.

Identifying owner
The entity type on which the weak entity type depends.

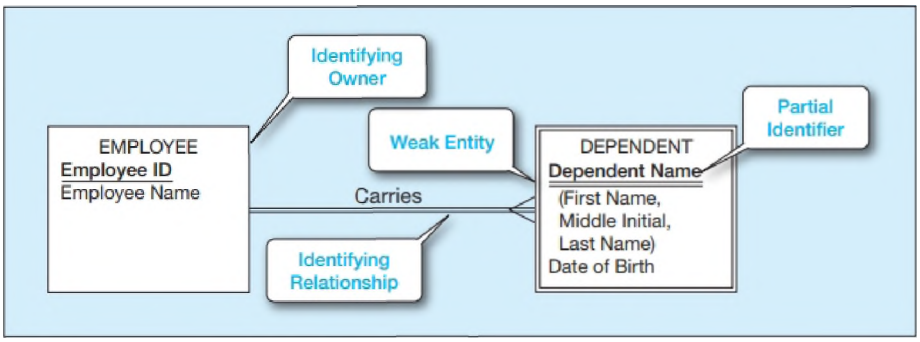


FIGURE 2-5 Example of a weak entity and its identifying relationship

have its own identifier. Generally, on an E-R diagram, a weak entity type has an attribute that serves as a *partial* identifier. During a later design stage (described in Chapter 4), a full identifier will be formed for the weak entity by combining the partial identifier with the identifier of its owner or by representing the weak entity as a strong entity with a surrogate, nonintelligent identifier attribute and the partial identifier as a regular attribute of this entity.

An example of a weak entity type with an identifying relationship is shown in Figure 2-5. EMPLOYEE is a strong entity type with identifier Employee ID (we note the identifier attribute by underlining it). DEPENDENT is a weak entity type, as indicated by the double-lined rectangle. The relationship between a weak entity type and its owner is called an **identifying relationship**. In Figure 2-5, Carries is the identifying relationship (indicated by the double line). The attribute Dependent Name serves as a *partial* identifier of DEPENDENT. (Dependent Name is a composite attribute that can be broken into component parts, as we describe later.) We use a double underline to indicate a partial identifier. During a later design stage, Dependent Name will be combined with Employee ID (the identifier of the owner) to form a full identifier for DEPENDENT. Some additional examples of strong and weak entity pairs are BOOK–BOOK COPY, PRODUCT–SERIAL PRODUCT, and COURSE–COURSE OFFERING.

Identifying relationship
The relationship between a weak entity type and its owner.

NAMING AND DEFINING ENTITY TYPES In addition to the general guidelines for naming and defining data objects, there are a few special guidelines for *naming* entity types, which follow:

- An entity type name is a *singular noun* (such as CUSTOMER, STUDENT, or AUTOMOBILE); an entity is a person, a place, an object, an event, or a concept, and the name is for the entity type, which represents a set of entity instances (i.e., STUDENT represents students Hank Finley, Jean Krebs, and so forth). It is common to also specify the plural form (possibly in a CASE tool repository accompanying the E-R diagram) because sometimes the E-R diagram is read best by using plurals. For example, in Figure 2-1, we would say that a SUPPLIER may supply ITEMs. Because plurals are not always formed by adding an *s* to the singular noun, it is best to document the exact plural form.
- An entity type name should be *specific to the organization*. Thus, one organization may use the entity type name CUSTOMER, and another organization may use the entity type name CLIENT (this is one task, e.g., done to customize a purchased data model). The name should be descriptive for everyone in the organization and distinct from all other entity type names within that organization. For example, a PURCHASE ORDER for orders placed with suppliers is distinct from a CUSTOMER ORDER for orders placed with a company by its customers. Both of these entity types cannot be named ORDER.
- An entity type name should be *concise*, using as few words as possible. For example, in a university database, an entity type REGISTRATION for the event of a student registering for a class is probably a sufficient name for this entity type; STUDENT REGISTRATION FOR CLASS, although precise, is probably too wordy because the reader will understand REGISTRATION from its use with other entity types.

- An *abbreviation*, or a *short name*, should be specified for each entity type name, and the abbreviation may be sufficient to use in the E-R diagram; abbreviations must follow all of the same rules as do the full entity names.
- *Event entity types* should be named for the result of the event, not the activity or process of the event. For example, the event of a project manager assigning an employee to work on a project results in an ASSIGNMENT, and the event of a student contacting his or her faculty adviser seeking some information is a CONTACT.
- The *name* used for the same entity type *should be the same* on all E-R diagrams on which the entity type appears. Thus, as well as being specific to the organization, the name used for an entity type should be a standard, adopted by the organization for all references to the same kind of data. However, some entity types will have aliases, or alternative names, which are synonyms used in different parts of the organization. For example, the entity type ITEM may have aliases of MATERIAL (for production) and DRAWING (for engineering). Aliases are specified in documentation about the database, such as the repository of a CASE tool.

There are also some specific guidelines for *defining* entity types, which follow:

- *An entity type definition usually starts with* "An X is ...." This is the most direct and clear way to state the meaning of an entity type.
- An entity type definition should *include a statement of what the unique characteristic is for each instance of the entity type*. In many cases, stating the identifier for an entity type helps convey the meaning of the entity. An example for Figure 2-4b is "An expense is a payment for the purchase of some good or service. An expense is identified by a journal entry number."
- An entity type definition should make it clear what *entity instances are included and not included* in the entity type; often, it is necessary to list the kinds of entities that are excluded. For example, "A customer is a person or organization that has placed an order for a product from us or one that we have contacted to advertise or promote our products. A customer does not include persons or organizations that buy our products only through our customers, distributors, or agents."
- An entity type definition often includes a description of *when an instance of the entity type is created and deleted*. For example, in the previous bullet point, a customer instance is implicitly created when the person or organization places its first order; because this definition does not specify otherwise, implicitly a customer instance is never deleted, or it is deleted based on general rules that are specified about the purging of data from the database. A statement about when to delete an entity instance is sometimes referred to as the retention of the entity type. A possible deletion statement for a customer entity type definition might be "A customer ceases to be a customer if it has not placed an order for more than three years."
- For some entity types, the definition must specify *when an instance might change into an instance of another entity type*. For example, consider the situation of a construction company for which bids accepted by potential customers become contracts. In this case, a bid might be defined by "A bid is a legal offer by our organization to do work for a customer. A bid is created when an officer of our company signs the bid document; a bid becomes an instance of contract when we receive a copy of the bid signed by an officer of the customer." This definition is also a good example to note how one definition can use other entity type names (in this case, the definition of bid uses the entity type name CUSTOMER).
- For some entity types, the definition must specify *what history is to be kept about instances of the entity type*. For example, the characteristics of an ITEM in Figure 2-1 may change over time, and we may need to keep a complete history of the individual values and when they were in effect. As you will see in some examples later, such statements about keeping history may have ramifications about how we represent the entity type on an E-R diagram and eventually how we store data for the entity instances.

Attributes

Each entity type has a set of attributes associated with it. An **attribute** is a property or characteristic of an entity type that is of interest to the organization. (Later, you will see that some types of relationships may also have attributes.) Thus, an attribute has a noun name. Following are some typical entity types and their associated attributes:

Attribute
A property or characteristic of an entity or relationship type that is of interest to the organization.

STUDENT	Student ID, Student Name, Home Address, Phone Number, Major
AUTOMOBILE	Vehicle ID, Color, Weight, Horsepower
EMPLOYEE	Employee ID, Employee Name, Payroll Address, Skill

In naming attributes, we use an initial capital letter followed by lowercase letters. If an attribute name consists of more than one word, we use a space between the words and we start each word with a capital letter, for example, Employee Name or Student Home Address. In E-R diagrams, we represent an attribute by placing its name in the entity it describes. Attributes may also be associated with relationships, as described later. Note that an attribute is associated with exactly one entity or relationship.

Notice in Figure 2-5 that all of the attributes of DEPENDENT are characteristics only of an employee's dependent, not characteristics of an employee. In traditional E-R notation, an entity type (not just weak entities but any entity) does not include attributes of entities to which it is related (what might be called foreign attributes). For example, DEPENDENT does not include any attribute that indicates to which employee this dependent is associated. This nonredundant feature of the E-R data model is consistent with the shared data property of databases. Because of relationships, which we discuss shortly, someone accessing data from a database will be able to associate attributes from related entities (e.g., show on a display screen a Dependent Name and the associated Employee Name).

REQUIRED VERSUS OPTIONAL ATTRIBUTES Each entity (or instance of an entity type) potentially has a value associated with each of the attributes of that entity type. An attribute that must be present for each entity instance is called a **required attribute**, whereas an attribute that may not have a value is called an **optional attribute**. For example, Figure 2-6 shows two STUDENT entities (instances) with their respective attribute values. The only optional attribute for STUDENT is Major. (Some students, specifically Melissa Kraft in this example, have not chosen a major yet; MIS would, of course, be a great career choice!) However, every student must, by the rules of the organization, have values for all the other attributes; *that is, we cannot store any data about a student in a STUDENT entity instance unless there are values for all the required attributes.* In various E-R diagramming notations, a symbol might appear in front of each attribute to indicate whether it is required (e.g., *) or optional (e.g., o), or required attributes will be in **boldface**, whereas optional attributes will be in normal font (the format we use in this text); in many cases, required or optional is indicated within supplemental documentation.

Required attribute
An attribute that must have a value for every entity (or relationship) instance with which it is associated.

Optional attribute
An attribute that may not have a value for every entity (or relationship) instance with which it is associated.

Entity type: STUDENT

Attributes	Attribute Data Type	Required or Optional	Example Instance	Example Instance
Student ID	CHAR (10)	Required	28-618411	26-844576
Student Name	CHAR (40)	Required	Michael Grant	Melissa Kraft
Home Address	CHAR (30)	Required	314 Baker St.	1422 Heft Ave
Home City	CHAR (20)	Required	Centerville	Miami
Home State	CHAR (2)	Required	OH	FL
Home Zip Code	CHAR (9)	Required	45459	33321
Major	CHAR (3)	Optional	MIS	

FIGURE 2-6 Entity type STUDENT with required and optional attributes

In Chapter 3, when you study entity supertypes and subtypes, you will see how sometimes optional attributes imply that there are different types of entities. (For example, we may want to consider students who have not declared a major as a subtype of the STUDENT entity type.) An attribute without a value is said to be null. Thus, each entity has an identifying attribute, which we discuss in a subsequent section, plus one or more other attributes. If you try to create an entity that has only an identifier, that entity is likely not legitimate. Such a data structure may simply hold a list of legal values for some attribute, which is better kept outside the database.

SIMPLE VERSUS COMPOSITE ATTRIBUTES Some attributes can be broken down into meaningful component parts (detailed attributes). A common example is Name, which you saw in Figure 2-5; another is Address, which can usually be broken down into the following component attributes: Street Address, City, State, and Postal Code. A **composite attribute** is an attribute, such as Address, that has meaningful component parts, which are more detailed attributes. Figure 2-7 shows the notation that we use for composite attributes applied to this example. Most drawing tools do not have a notation for composite attributes, so you simply list all the component parts.

Composite attribute
An attribute that has meaningful component parts (attributes).

Composite attributes provide considerable flexibility to users, who can either refer to the composite attribute as a single unit or else refer to individual components of that attribute. Thus, for example, a user can either refer to Address or refer to one of its components, such as Street Address. The decision about whether to subdivide an attribute into its component parts depends on whether users will need to refer to those individual components, and hence, they have organizational meaning. Of course, you must always attempt to anticipate possible future usage patterns for the database.

A **simple (or atomic) attribute** is an attribute that cannot be broken down into smaller components that are meaningful for the organization. For example, all the attributes associated with AUTOMOBILE are simple: Vehicle ID, Color, Weight, and Horsepower.

Simple (or atomic) attribute
An attribute that cannot be broken down into smaller components that are meaningful to the organization.

SINGLE-VALUED VERSUS MULTIVALUED ATTRIBUTES Figure 2-6 shows two entity instances with their respective attribute values. For each entity instance, each of the attributes in the figure has one value. It frequently happens that there is an attribute that may have more than one value for a given instance. For example, the EMPLOYEE entity type in Figure 2-8 has an attribute named Skill, whose values record the skill

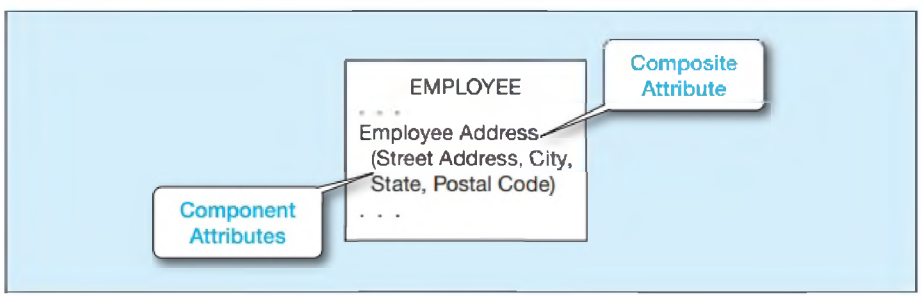


FIGURE 2-7 A composite attribute

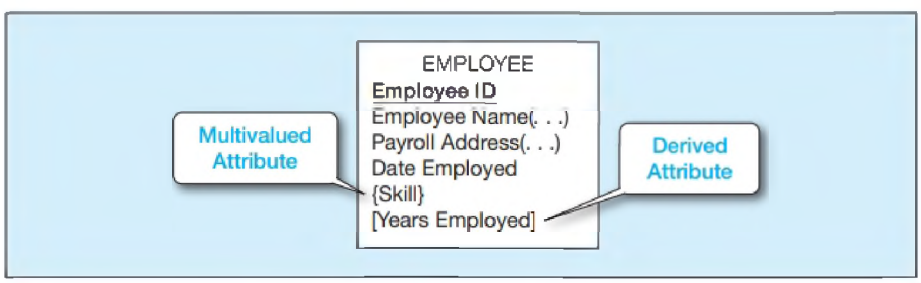


FIGURE 2-8 Entity with multivalued attribute (Skill) and derived attribute (Years Employed)

(or skills) for that employee. Of course, some employees may have more than one skill, such as PHP Programmer and C++ Programmer. A **multivalued attribute** is an attribute that may take on more than one value for a given entity (or relationship) instance. In this text, we indicate a multivalued attribute with curly brackets around the attribute name, as shown for the Skill attribute in the EMPLOYEE example in Figure 2-8. In Microsoft Visio, once an attribute is placed in an entity, you can edit that attribute (column), select the Collection tab, and choose one of the options. (Typically, MultiSet will be your choice, but one of the other options may be more appropriate for a given situation.) Other E-R diagramming tools may use an asterisk (*) after the attribute name, or you may have to use supplemental documentation to specify a multivalued attribute.

Multivalued attribute

An attribute that may take on more than one value for a given entity (or relationship) instance.

Multivalued and composite are different concepts, although beginner data modelers often confuse these terms. Skill, a multivalued attribute, may occur multiple times for each employee; Employee Name and Payroll Address are both likely composite attributes, each of which occurs once for each employee but which have component, more atomic attributes that are not shown in Figure 2-8 for simplicity. It is possible to have a multivalued composite attribute. For example, a Customer Address composite attribute may have a variable number of values for a given customer (home, office, seasonal, etc.). See Problem and Exercise 2-38 to review the concepts of composite and multivalued attributes.

STORED VERSUS DERIVED ATTRIBUTES Some attribute values that are of interest to users can be calculated or derived from other related attribute values that are stored in the database. When users want to use such derived values in calculations and displays, it is likely helpful for them to show these derived attributes in an E-R diagram. For example, suppose for an organization, the EMPLOYEE entity type has a Date Employed attribute. If users need to know how many years a person has been employed, that value can be calculated using Date Employed and today's date. A **derived attribute** is an attribute whose values can be calculated from related attribute values (plus possibly data not in the database, such as today's date, the current time, or a security code provided by a system user). We indicate a derived attribute in an E-R diagram by using square brackets around the attribute name, as shown in Figure 2-8 for the Years Employed attribute. Some E-R diagramming tools use a notation of a forward slash (/) in front of the attribute name to indicate that it is derived. (This notation is borrowed from UML for a virtual attribute.) The method for calculating the derived attribute may be stored with the description of the associated entity (or relationship), and this method may be either coded into the physical database definition or programmed into application programs based on the one method description associated with the E-R diagram and database documentation.

Derived attribute

An attribute whose values can be calculated from related attribute values.

In some situations, the value of an attribute can be derived from attributes in related entities. For example, consider an invoice created for each customer at Pine Valley Furniture Company. Order Total would be an attribute of the INVOICE entity, which indicates the total dollar amount that is billed to the customer. The value of Order Total can be computed by summing the Extended Price values (unit price times quantity sold) for the various line items that are billed on the invoice. Formulas for computing values such as this are one type of business rule.

IDENTIFIER ATTRIBUTE An **identifier** is an attribute (or combination of attributes) whose value distinguishes individual instances of an entity type. That is, no two instances of the entity type may have the same value for the identifier attribute. The identifier for the STUDENT entity type introduced earlier is Student ID, whereas the identifier for AUTOMOBILE is Vehicle ID. Notice that an attribute such as Student Name is not a candidate identifier because many students may potentially have the same name and students, like all people, can change their names. To be a candidate identifier, each entity instance must have a single value for the attribute, and the attribute must be associated with the entity. We underline identifier names on the E-R diagram, as shown in the STUDENT entity type example in Figure 2-9a. To be an identifier, the attribute is also required (so the distinguishing value must exist), so an identifier

Identifier

An attribute (or combination of attributes) whose value distinguishes instances of an entity type.

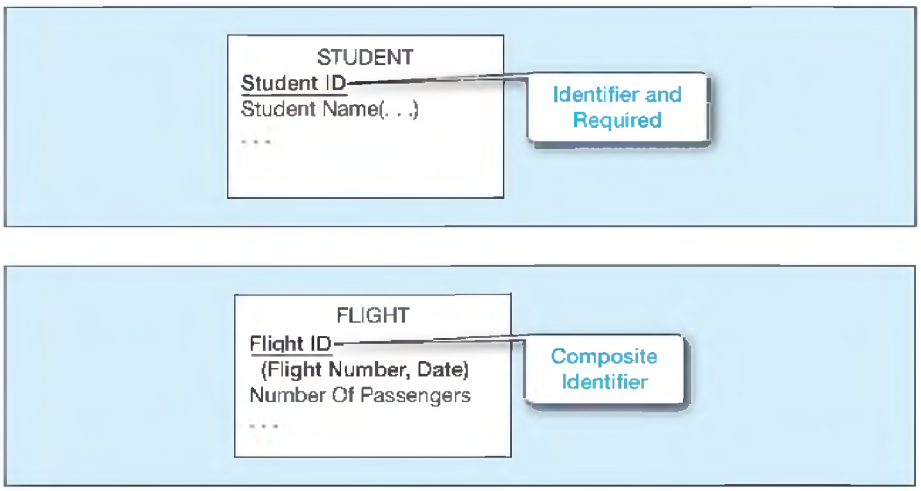


FIGURE 2-9 Simple and composite identifier attributes

is also in bold. Some E-R drawing software will place a symbol, called a stereotype, in front of the identifier (e.g., <<ID>> or <<PK>>).

Composite identifier
An identifier that consists of a composite attribute.

For some entity types, there is no single (or atomic) attribute that can serve as the identifier (i.e., that will ensure uniqueness). However, two (or more) attributes used in combination may serve as the identifier. A **composite identifier** is an identifier that consists of a composite attribute. Figure 2-9b shows the entity FLIGHT with the composite identifier Flight ID. Flight ID in turn has component attributes Flight Number and Date. This combination is required to identify uniquely individual occurrences of FLIGHT.

We use the convention that the composite attribute (Flight ID) is underlined to indicate it is the identifier, whereas the component attributes are not underlined. Some data modelers think of a composite identifier as "breaking a tie" created by a simple identifier. Even with Flight ID, a data modeler would ask a question, such as "Can two flights with the same number occur on the same date?" If so, yet another attribute is needed to form the composite identifier and to break the tie.

Some entities may have more than one candidate identifier. If there is more than one candidate identifier, the designer must choose one of them as the identifier. Bruce (1992) suggests the following criteria for selecting identifiers:

1. Choose an identifier that will not change its value over the life of each instance of the entity type. For example, the combination of Employee Name and Payroll Address (even if unique) would be a poor choice as an identifier for EMPLOYEE because the values of Employee Name and Payroll Address could easily change during an employee's term of employment.
2. Choose an identifier such that for each instance of the entity, the attribute is guaranteed to have valid values and not be null (or unknown). If the identifier is a composite attribute, such as Flight ID in Figure 2-9b, make sure that all parts of the identifier will have valid values.
3. Avoid the use of so-called intelligent identifiers (or keys), whose structure indicates classifications, locations, and so on. For example, the first two digits of an identifier value may indicate the warehouse location. Such codes are often changed as conditions change, which renders the identifier values invalid.
4. Consider substituting single-attribute surrogate identifiers for large composite identifiers. For example, an attribute called Game Number could be used for the entity type GAME instead of the combination of Home Team and Visiting Team.

NAMING AND DEFINING ATTRIBUTES In addition to the general guidelines for naming data objects, there are a few special guidelines for naming attributes, which follow:

- An attribute name is a *singular noun or noun phrase* (such as Customer ID, Age, Product Minimum Price, or Major). Attributes, which materialize as data values,

are concepts or physical characteristics of entities. Concepts and physical characteristics are described by nouns.

- An attribute name should be *unique*. No two attributes of the same entity type may have the same name, and it is desirable, for clarity purposes, that no two attributes across all entity types have the same name.
- To make an attribute name unique and for clarity purposes, *each attribute name should follow a standard format*. For example, your university may establish Student GPA, as opposed to GPA of Student, as an example of the standard format for attribute naming. The format to be used will be established by each organization. A common format is [Entity type name { [Qualifier] }] Class, where [. . .] is an optional clause, and { . . . } indicates that the clause may repeat. *Entity type name* is the name of the entity with which the attribute is associated. The entity type name may be used to make the attribute name explicit. It is almost always used for the identifier attribute (e.g., Customer ID) of each entity type. *Class* is a phrase from a list of phrases defined by the organization that are the permissible characteristics or properties of entities (or abbreviations of these characteristics). For example, permissible values (and associated approved abbreviations) for Class might be Name (Nm), Identifier (ID), Date (Dt), or Amount (Amt). Class is, obviously, required. *Qualifier* is a phrase from a list of phrases defined by the organization that are used to place constraints on classes. One or more qualifiers may be needed to make each attribute of an entity type unique. For example, a qualifier might be Maximum (Max), Hourly (Hrly), or State (St). A qualifier may not be necessary: Employee Age and Student Major are both fully explicit attribute names. Sometimes a qualifier is necessary. For example, Employee Birth Date and Employee Hire Date are two attributes of Employee that require one qualifier. More than one qualifier may be necessary. For example, Employee Residence City Name (or Emp Res Cty Nm) is the name of an employee's city of residence, and Employee Tax City Name (or Emp Tax Cty Nm) is the name of the city in which an employee pays city taxes.
- *Similar attributes* of different entity types *should use the same qualifiers and classes*, as long as those are the names used in the organization. For example, the city of residence for faculty and students should be, respectively, Faculty Residence City Name and Student Residence City Name. Using similar names makes it easier for users to understand that values for these attributes come from the same possible set of values, what we will call *domains*. Users may want to take advantage of common domains in queries (e.g., find students who live in the same city as their adviser), and it will be easier for users to recognize that such a matching may be possible if the same qualifier and class phrases are used.

There are also some specific guidelines for defining attributes, which follow:

- An attribute definition states *what the attribute is and possibly why it is important*. The definition will often parallel the attribute's name; for example, Student Residence City Name could be defined as "The name of the city in which a student maintains his or her permanent residence."
- An attribute definition should make it clear *what is included and not included* in the attribute's value; for example, "Employee Monthly Salary Amount is the amount of money paid each month in the currency of the country of residence of the employee, exclusive of any benefits, bonuses, reimbursements, or special payments."
- Any *aliases*, or alternative names, for the attribute can be specified in the definition or may be included elsewhere in documentation about the attribute, possibly stored in the repository of a CASE tool used to maintain data definitions.
- It may also be desirable to state in the definition *the source of values for the attribute*. Stating the source may make the meaning of the data clearer. For example, "Customer Standard Industrial Code is an indication of the type of business for the customer. Values for this code come from a standard set of values provided by the Federal Trade Commission and are found on a CD we purchase named SIC provided annually by the FTC."

- An attribute definition (or other specification in a CASE tool repository) also should indicate *if a value for the attribute is required or optional*. This business rule about an attribute is important for maintaining data integrity. The identifier attribute of an entity type is, by definition, required. If an attribute value is required, then to create an instance of the entity type, a value of this attribute must be provided. Required means that an entity instance must always have a value for this attribute, not just when an instance is created. Optional means that a value may not exist for an instance of an entity instance to be stored. Optional can be further qualified by stating whether once a value is entered, a value must always exist. For example, "Employee Department ID is the identifier of the department to which the employee is assigned. An employee may not be assigned to a department when hired (so this attribute is initially optional), but once an employee is assigned to a department, the employee must always be assigned to some department."
- An attribute definition (or other specification in a CASE tool repository) may also indicate *whether a value for the attribute may change* once a value is provided and before the entity instance is deleted. This business rule also controls data integrity. Nonintelligent identifiers may not change values over time. To assign a new nonintelligent identifier to an entity instance, that instance must first be deleted and then re-created.
- For a multivalued attribute, the attribute definition should indicate *the maximum and minimum number of occurrences of an attribute value for an entity instance*. For example, "Employee Skill Name is the name of a skill an employee possesses. Each employee must possess at least one skill, and an employee can choose to list at most 10 skills." The reason for a multivalued attribute may be that a history of the attribute needs to be kept. For example, "Employee Yearly Absent Days Number is the number of days in a calendar year the employee has been absent from work. An employee is considered absent if he or she works less than 50 percent of the scheduled hours in the day. A value for this attribute should be kept for each year in which the employee works for our company."
- An attribute definition may also indicate *any relationships that attribute has with other attributes*. For example, "Employee Vacation Days Number is the number of days of paid vacation for the employee. If the employee has a value of 'Exempt' for Employee Type, then the maximum value for Employee Vacation Days Number is determined by a formula involving the number of years of service for the employee."

MODELING RELATIONSHIPS

Relationships are the glue that holds together the various components of an E-R model. Intuitively, a *relationship* is an association representing an interaction among the instances of one or more entity types that is of interest to the organization. Thus, a relationship has a verb phrase name. Relationships and their characteristics (degree and cardinality) represent business rules, and usually relationships represent the most complex business rules shown in an ERD. In other words, this is where data modeling gets really interesting and fun, as well as crucial for controlling the integrity of a database. Relationships are essential for almost every meaningful use of a database; for example, relationships allow iTunes to find the music you've purchased, your cell phone company to find all the text messages in one of your SMS threads, or the campus nurse to see how different students have reacted to different treatments to the latest influenza on campus. So, fun and essential—modeling relationships will be a rewarding skill for you.

To understand relationships more clearly, we must distinguish between relationship types and relationship instances. To illustrate, consider the entity types EMPLOYEE and COURSE, where COURSE represents training courses that may be taken by employees. To track courses that have been completed by particular employees, you would define a relationship called Completes between the two entity types (see Figure 2-10a). This is a many-to-many relationship because each employee may complete any number of courses (zero, one, or many courses), whereas a given course may be completed by any number of employees (nobody, one employee, or many employees). For example, in Figure 2-10b, the employee Melton has completed three courses (C++, COBOL, and

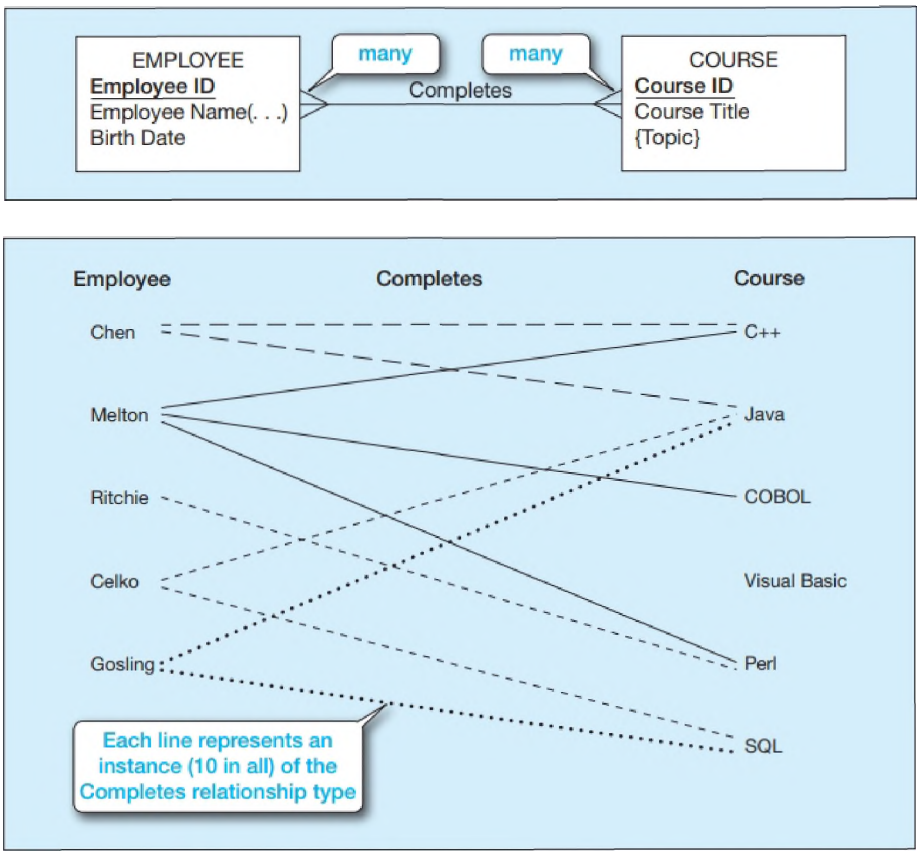


FIGURE 2-10 Relationship type and instances

(a) Relationship type (Completes)

(b) Relationship instances

Perl). The SQL course has been completed by two employees (Celko and Gosling), and the Visual Basic course has not been completed by anyone.

In this example, there are two entity types (EMPLOYEE and COURSE) that participate in the relationship named Completes. In general, any number of entity types (from one to many) may participate in a relationship.

We frequently use in this and subsequent chapters the convention of a single verb phrase label to represent a relationship. Because relationships often occur due to an organizational event, entity instances are related because an action was taken; thus, a verb phrase is appropriate for the label. This verb phrase should be in the present tense and descriptive. There are, however, many ways to represent a relationship. Some data modelers prefer the format with two relationship names, one to name the relationship in each direction. One or two verb phrases have the same structural meaning, so you may use either format as long as the meaning of the relationship in each direction is clear.

Basic Concepts and Definitions in Relationships

A **relationship type** is a meaningful association between (or among) entity types. The phrase *meaningful association* implies that the relationship allows us to answer questions that could not be answered given only the entity types. A relationship type is denoted by a line labeled with the name of the relationship, as in the example shown in Figure 2-10a, or with two names, as in Figure 2-1. We suggest you use a short, descriptive verb phrase that is meaningful to the user in naming the relationship. (We say more about naming and defining relationships later in this section.)

Relationship type
A meaningful association between (or among) entity types.

A **relationship instance** is an association between (or among) entity instances, where each relationship instance associates exactly one entity instance from each participating entity type (Elmasri and Navathe, 1994). For example, in Figure 2-10b, each of the 10 lines in the figure represents a relationship instance between one employee and

Relationship instance
An association between (or among) entity instances where each relationship instance associates exactly one entity instance from each participating entity type.

TABLE 2-2 Instances Showing Date Completed

Employee Name	Course Title	Date Completed
Chen	C++	06/2017
Chen	Java	09/2017
Melton	C++	06/2017
Melton	COBOL	02/2018
Melton	SQL	03/2017
Ritchie	Perl	11/2017
Celko	Java	03/2017
Celko	SQL	03/2018
Gosling	Java	09/2017
Gosling	Perl	06/2017

one course, indicating that the employee has completed that course. For example, the line between Employee Ritchie and Course Perl is one relationship instance.

ATTRIBUTES ON RELATIONSHIPS It is probably obvious to you that entities have attributes, but attributes may be associated with a many-to-many (or one-to-one) relationship, too. For example, suppose the organization wishes to record the date (month and year) when an employee completes each course. This attribute is named Date Completed. For some sample data, see Table 2-2.

Where should the attribute Date Completed be placed on the E-R diagram? Referring to Figure 2-10a, you will notice that Date Completed has not been associated with either the EMPLOYEE or the COURSE entity. That is because Date Completed is a property of the relationship Completes rather than a property of either entity. In other words, for each instance of the relationship Completes, there is a value for Date Completed. One such instance, for example, shows that the employee named Melton completed the course titled C++ in 06/2017.

A revised version of the ERD for this example is shown in Figure 2-11a. In this diagram, the attribute Date Completed is in a rectangle connected to the Completes relationship line. Other attributes might be added to this relationship if appropriate, such as Course Grade, Instructor, and Room Location. We will explain the A and B annotations below.

It is interesting to note that an attribute cannot be associated with a one-to-many relationship, such as Carries in Figure 2-5. For example, consider Dependent Date, similar to Date Completed above, for when the DEPENDENT begins to be carried by the EMPLOYEE. Because each DEPENDENT is associated with only one EMPLOYEE, such a date is unambiguously a characteristic of the DEPENDENT (i.e., for a given DEPENDENT, Dependent Date cannot vary by EMPLOYEE). So, if you ever have the urge to associate an attribute with a one-to-many relationship, "step away from the relationship!"

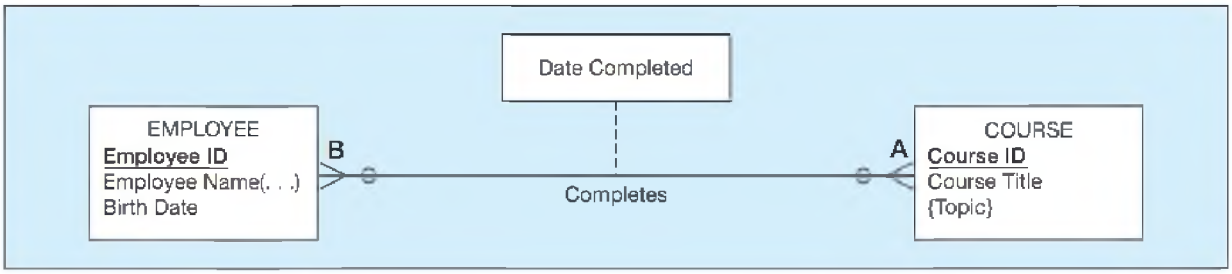
ASSOCIATIVE ENTITIES The presence of one or more attributes on a relationship suggests to the designer that the relationship should perhaps instead be represented as an entity type. To emphasize this point, most E-R drawing tools require that such attributes be placed in an entity type. An **associative entity** is an entity type that associates the instances of one or more entity types and contains attributes that are peculiar to the relationship between those entity instances. The associative entity CERTIFICATE is represented with the rectangle with rounded corners, as shown in Figure 2-11b. Most E-R drawing tools do not have a special symbol for an associative entity. Associative entities are sometimes referred to as gerunds because the relationship name (a verb) is usually converted to an entity name that is a noun. Note in Figure 2-11b that there are no relationship names on the lines between an associative entity and a strong entity. This is because the associative entity represents the relationship. Figure 2-11c shows how associative entities are drawn using Microsoft Visio, which is representative of how you

Associative entity

An entity type that associates the instances of one or more entity types and contains attributes that are peculiar to the relationship between those entity instances.

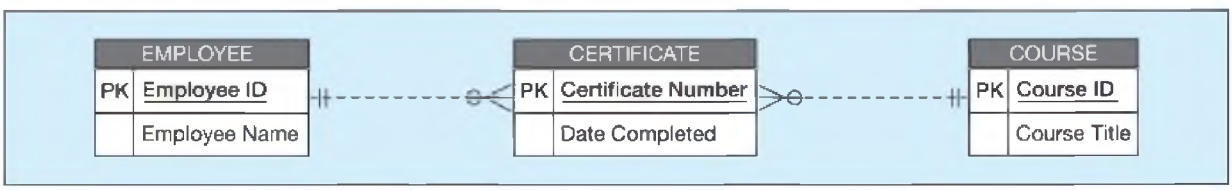
FIGURE 2-11 An associative entity

(a) Attribute on a relationship



(b) An associative entity (CERTIFICATE)

(c) An associative entity using Microsoft VISIO



would draw an associative entity with most E-R diagramming tools. In Visio, the relationship lines are dashed because CERTIFICATE does not include the identifiers of the related entities in its identifier. (Certificate Number is sufficient.)

How do you know whether to convert a relationship to an associative entity type? Following are four conditions that should exist:

1. All the relationships for the participating entity types are "many" relationships.
2. The resulting associative entity type has independent meaning to end users and, preferably, can be identified with a single-attribute identifier.
3. The associative entity has one or more attributes in addition to the identifier.
4. The associative entity participates in one or more relationships independent of the entities related in the associated relationship.

Figure 2-11b shows the relationship Completes converted to an associative entity type. In this case, the training department for the company has decided to award a certificate to each employee who completes a course. Thus, the entity is named CERTIFICATE, which certainly has independent meaning to end users. Also, each certificate has a number (Certificate Number) that serves as the identifier.

The attribute Date Completed is also included. Note also in Figure 2-11b and the Visio version of Figure 2-11c that both EMPLOYEE and COURSE are mandatory participants in the two relationships with CERTIFICATE. This is exactly what occurs when you have to represent a many-to-many relationship (Completes in Figure 2-11a) as two one-to-many relationships (the ones associated with CERTIFICATE in Figures 2-11b and 2-11c).

Notice that converting a relationship to an associative entity has caused the relationship notation to move. That is, the "many" cardinality now terminates at the associative entity rather than at each participating entity type. In Figure 2-11, this shows that an employee, who may complete one or more courses (notation A in Figure 2-11a), may be awarded more than one certificate (notation A in Figure 2-11b) and that a course, which may have one or more employees complete it (notation B in Figure 2-11a), may have

many certificates awarded (notation B in Figure 2-11b). See Problem and Exercise 2-42 for an interesting variation on Figure 2-11a, which emphasizes the rules for when to convert a many-to-many relationship, such as Completes, into an associative entity.

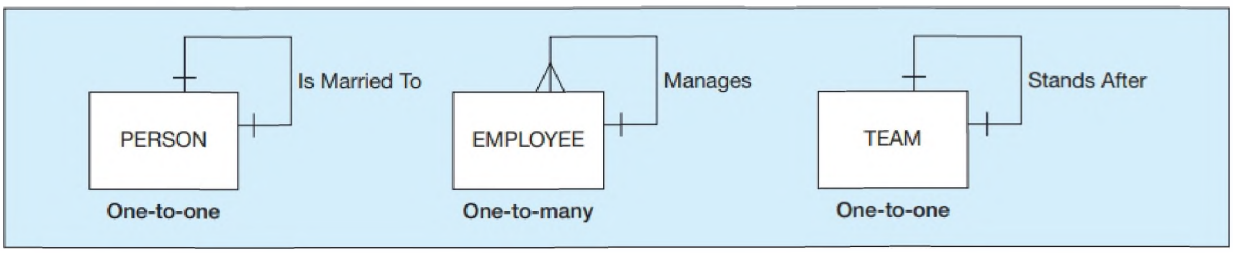
Degree of a Relationship

Degree
The number of entity types that participate in a relationship.

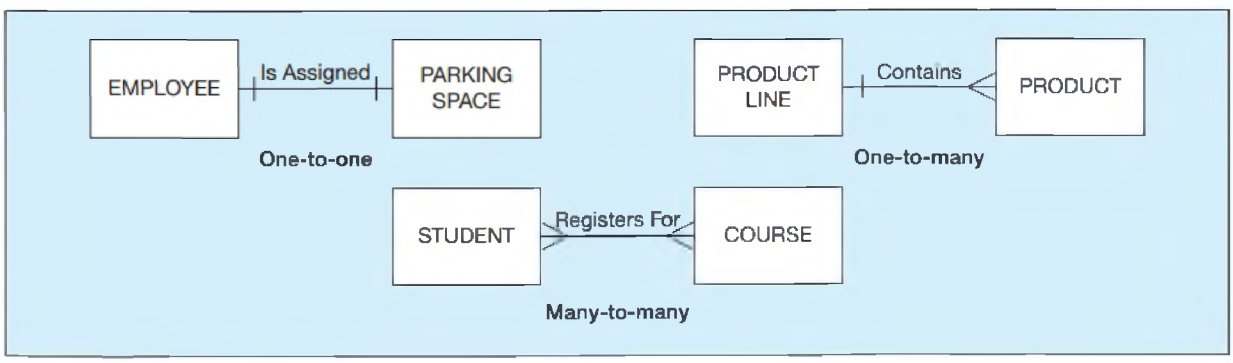
The **degree** of a relationship is the number of entity types that participate in that relationship. Thus, the relationship Completes in Figure 2-11 is of degree 2 because there are two entity types: EMPLOYEE and COURSE. The three most common relationship degrees in E-R models are unary (degree 1), binary (degree 2), and ternary (degree 3). Higher-degree relationships are possible, but they are rarely encountered in practice, so we restrict our discussion to these three cases. Examples of unary, binary, and ternary relationships appear in Figure 2-12. (Attributes are not shown in some figures for simplicity.)

FIGURE 2-12 Examples of relationships of different degrees

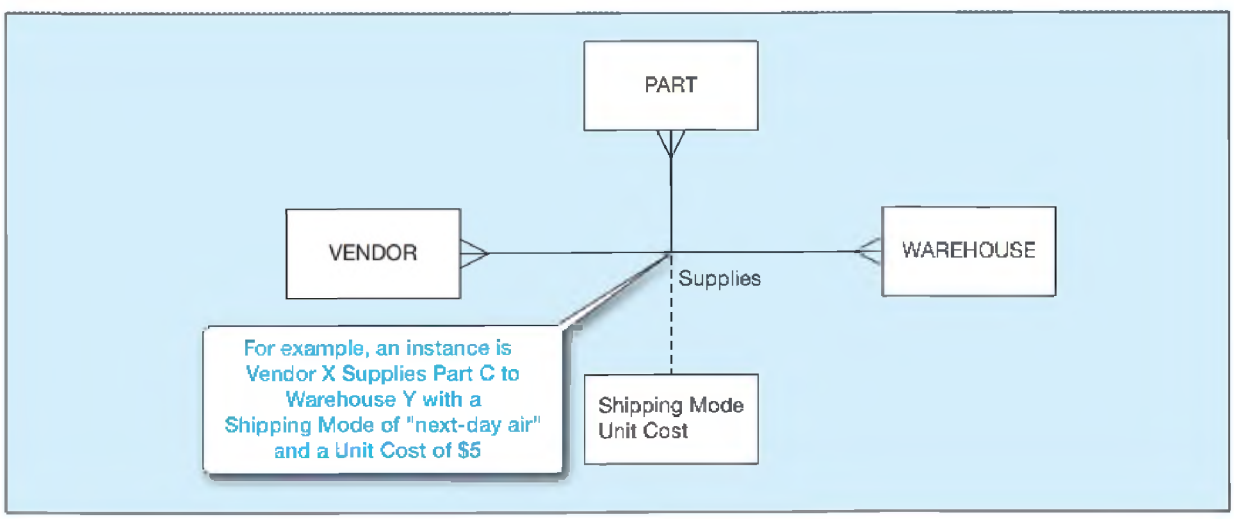
(a) Unary relationships



(b) Binary relationships



(c) Ternary relationship



As you look at Figure 2-12, understand that any particular data model represents a specific situation, not a generalization. For example, consider the Manages relationship in Figure 2-12a. In some organizations, it may be possible for one employee to be managed by many other employees (e.g., in a matrix organization). It is important when you develop an E-R model that you understand the business rules of the particular organization you are modeling.

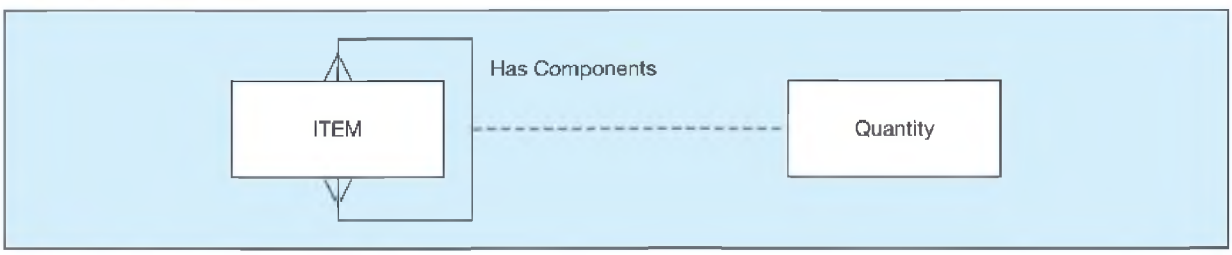
UNARY RELATIONSHIP A **unary relationship** is a relationship between the instances of a *single* entity type. (Unary relationships are also called *recursive relationships*.) Three examples are shown in Figure 2-12a. In the first example, Is Married To is shown as a one-to-one relationship between instances of the PERSON entity type. Because this is a one-to-one relationship, this notation indicates that only the current marriage, if one exists, needs to be kept about a person. What would change if we needed to retain the history of marriages for each person? See Review Question 2-20 and Problem and Exercise 2-34 for other business rules and their effect on the Is Married To relationship representation. In the second example, Manages is shown as a one-to-many relationship between instances of the EMPLOYEE entity type. Using this relationship, you could identify, for example, the employees who report to a particular manager. The third example is one case of using a unary relationship to represent a sequence, cycle, or priority list. In this example, sports teams are related by their standing in their league (the Stands After relationship). (Note: In these examples, we ignore whether these are mandatory- or optional-cardinality relationships or whether the same entity instance can repeat in the same relationship instance; we will introduce mandatory and optional cardinality in a later section of this chapter.)

Unary relationship
A relationship between instances of a single entity type.

Figure 2-13 shows an example of another unary relationship, called a *bill-of-materials structure*. Many manufactured products are made of assemblies, which in turn are composed of subassemblies and parts and so on. As shown in Figure 2-13a,

FIGURE 2-13 Representing a bill-of-materials structure

(a) Many-to-many relationship



(b) Two ITEM bill-of-materials structure instances

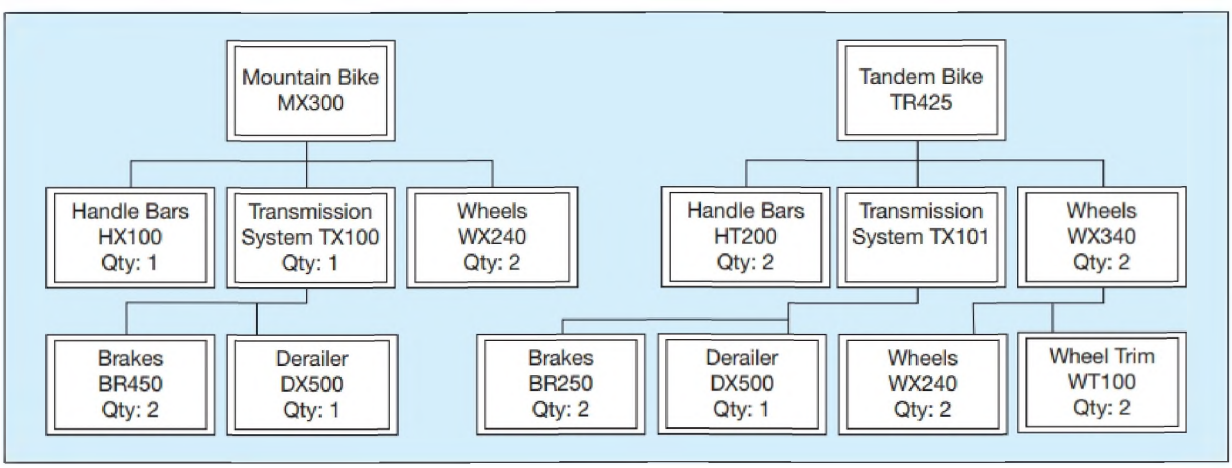
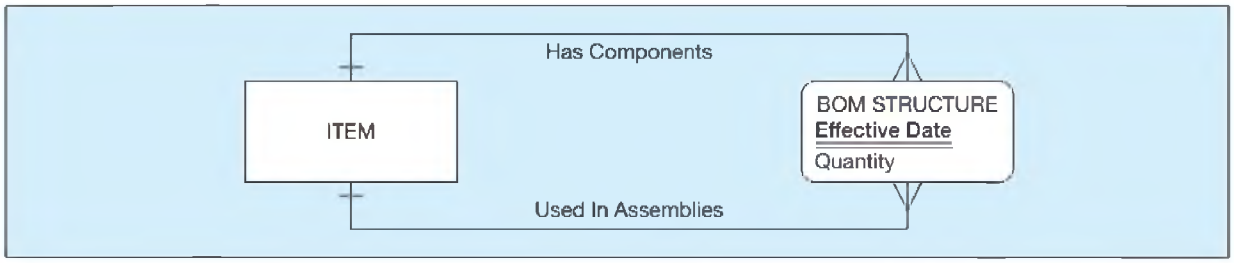


FIGURE 2-13 (*continued*)

(c) Associative entity



we can represent this structure as a many-to-many unary relationship. In this figure, the entity type ITEM is used to represent all types of components, and we use Has Components for the name of the relationship type that associates lower-level items with higher-level items.

Two occurrences of this bill-of-materials structure are shown in Figure 2-13b. Each of these diagrams shows the immediate components of each item as well as the quantities of that component. For example, item TX100 consists of item BR450 (quantity 2) and item DX500 (quantity 1). You can easily verify that the associations are in fact many-to-many. Several of the items have more than one component type (e.g., item MX300 has three immediate component types: HX100, TX100, and WX240). Also, some of the components are used in several higher-level assemblies. For example, item WX240 is used in both item MX300 and item WX340, even at different levels of the bill-of-materials. The many-to-many relationship guarantees that, for example, the same subassembly structure of WX240 (not shown) is used each time item WX240 goes into making some other item.

The presence of the attribute Quantity on the relationship suggests that the analyst consider converting the relationship Has Components to an associative entity. Figure 2-13c shows the entity type BOM STRUCTURE, which forms an association between instances of the ITEM entity type. A second (partial identifier) attribute (named Effective Date) has been added to BOM STRUCTURE to record the date when the specified quantity of this component was first used in the related assembly. Effective dates are often needed when a history of values is required. In practice, the identifiers of the component and assembly items along with Effective Date often become nonidentifier attributes, and a surrogate, nonintelligent identifier is used. Other data model structures can be used for unary relationships involving such hierarchies; we show some of these other structures in Chapter 9.

Binary relationship

A relationship between the instances of two entity types.

BINARY RELATIONSHIP A **binary relationship** is a relationship between the instances of two entity types and is the most common type of relationship encountered in data modeling. Figure 2-12b shows three examples. The first (one-to-one) indicates that an employee is assigned one parking place and that each parking place is assigned to one employee (at a given time). The second (one-to-many) indicates that a product line may contain several products and that each product belongs to only one product line. The third (many-to-many) shows that a student may register for more than one course and that each course may have many student registrants.

Ternary relationship

A simultaneous relationship among the instances of three entity types.

TERNARY RELATIONSHIP A **ternary relationship** is a *simultaneous* relationship among the instances of three entity types. A typical business situation that leads to a ternary relationship is shown in Figure 2-12c. In this example, vendors can supply various parts to warehouses. The relationship Supplies is used to record the specific parts that are supplied by a given vendor to a particular warehouse. Thus, there are three entity types involved: VENDOR, PART, and WAREHOUSE. There are two attributes on the relationship Supplies: Shipping Mode and Unit Cost. For example, one instance of Supplies might record the fact that vendor X can ship part C to warehouse Y, that the shipping mode is next-day air, and that the cost is $5 per unit.

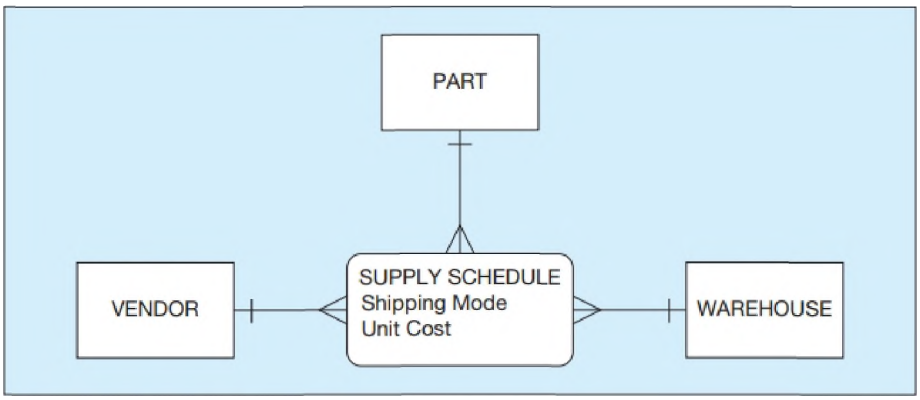


FIGURE 2-14 Ternary relationship as an associative entity

Don't be confused: A ternary relationship is not the same as three binary relationships. For example, Unit Cost is an attribute of the Supplies relationship in Figure 2-12c. Unit Cost cannot be properly associated with any one of the three possible binary relationships among the three entity types, such as that between PART and WAREHOUSE. Thus, for example, if we were told that vendor X can ship part C for a unit cost of $8, those data would be incomplete because they would not indicate to which warehouse the parts would be shipped.

As usual, the presence of an attribute on the relationship Supplies in Figure 2-12c suggests converting the relationship to an associative entity type. Figure 2-14 shows an alternative (and preferable) representation of the ternary relationship shown in Figure 2-12c. In Figure 2-14, the (associative) entity type SUPPLY SCHEDULE is used to replace the Supplies relationship from Figure 2-12c. Clearly, the entity type SUPPLY SCHEDULE is of independent interest to users. However, notice that an identifier has not yet been assigned to SUPPLY SCHEDULE. This is acceptable. If no identifier is assigned to an associative entity during E-R modeling, an identifier (or key) will be assigned during logical modeling (discussed in Chapter 4). This will be a composite identifier whose components will consist of the identifier for each of the participating entity types (in this example, PART, VENDOR, and WAREHOUSE) or a surrogate, non-intelligent identifier. Can you think of other attributes that might be associated with SUPPLY SCHEDULE?

As noted earlier, we do not label the lines from SUPPLY SCHEDULE to the three entities. This is because these lines do not represent binary relationships. To keep the same meaning as the ternary relationship of Figure 2-12c, we cannot break the Supplies relationship into three binary relationships, as we have already mentioned.

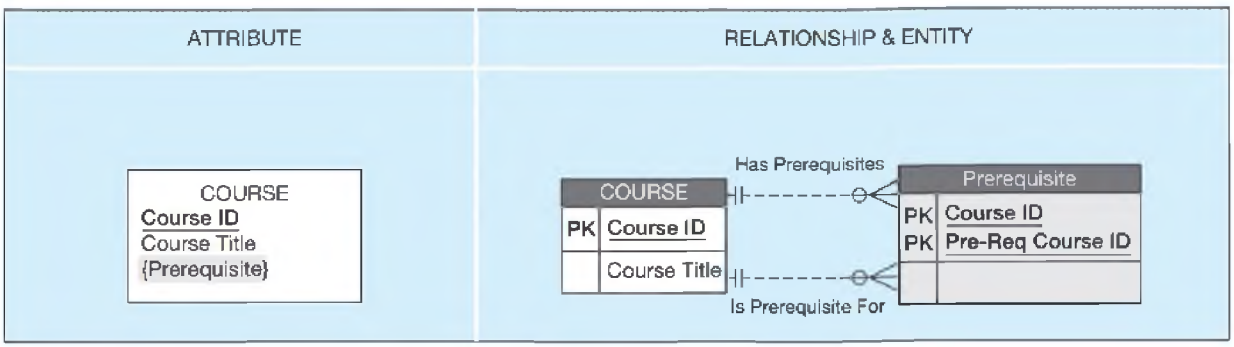
So, here is a guideline to follow: Convert all ternary (or higher) relationships to associative entities, as in this example. Song et al. (1995) show that participation constraints (described in a following section on cardinality constraints) cannot be accurately represented for a ternary relationship, given the notation with attributes on the relationship line. However, by converting to an associative entity, the constraints can be accurately represented. Also, many E-R diagram drawing tools, including most CASE tools, cannot represent ternary relationships. So, although not semantically accurate, you must use these tools to represent the ternary or higher-order relationship with an associative entity and three binary relationships, which have a mandatory association with each of the three related entity types.

Attributes or Entity?

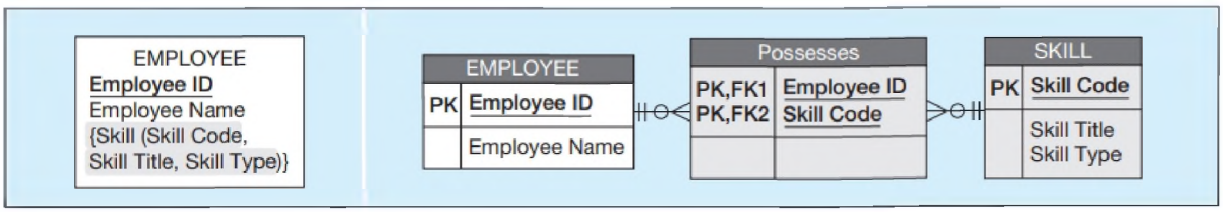
Sometimes you will wonder if you should represent data as an attribute or an entity; this is a common dilemma. Figure 2-15 includes three examples of situations when an attribute could be represented via an entity type. We use this text's E-R notation in the left column and the notation from Microsoft Visio in the right column; it is

FIGURE 2-15 Using relationships and entities to link related attributes

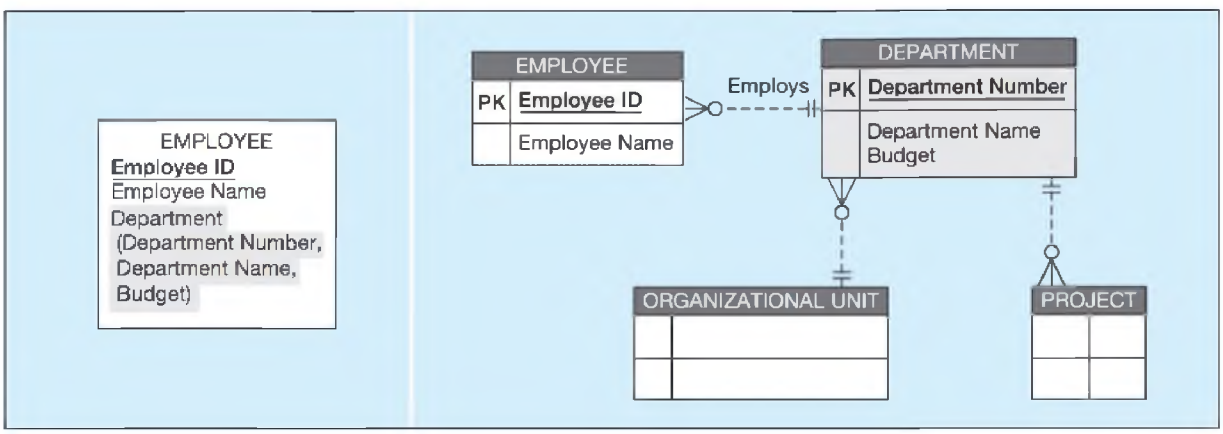
(a) Multivalued attribute versus relationships via bill-of-materials structure



(b) Composite, multivalued attribute versus relationship



(c) Composite attribute of data shared with other entity types



important that you learn how to read ERDs in several notations because you will encounter various styles in different publications and organizations. In Figure 2-15a, the potentially multiple prerequisites of a course (shown as a multivalued attribute in the Attribute cell) are also courses (and a course may be a prerequisite for many other courses). Thus, prerequisite could be viewed as a bill-of-materials structure (shown in the Relationship & Entity cell) between courses, not a multivalued attribute of COURSE. Representing prerequisites via a bill-of-materials structure also means that finding the prerequisites of a course and finding the courses for which a course is prerequisite both deal with relationships between entity types. When a prerequisite is a multivalued attribute of COURSE, finding the courses for which a course is a prerequisite means looking for a specific value for a prerequisite across all COURSE instances. As was shown in Figure 2-13a, such a situation could also be modeled as a

unary relationship among instances of the COURSE entity type. In Visio, this specific situation requires creating the equivalent of an associative entity (see the Relationship & Entity cell in Figure 2-15a; Visio does not use the rectangle with rounded corners symbol). By creating the associative entity, it is now easy to add characteristics to the relationship, such as a minimum grade required. Also note that Visio shows the identifier (in this case composite) with a PK stereotype symbol and boldface on the composite attribute names, signifying these are required attributes.

In Figure 2-15b, employees potentially have multiple skills (shown in the Attribute cell), but skill could be viewed instead as an entity type (shown in the Relationship & Entity cell as the equivalent of an associative entity) about which the organization wants to maintain data (the unique code to identify each skill, a descriptive title, and the type of skill, e.g., technical or managerial). An employee has skills, which are not viewed as attributes but rather as instances of a related entity type. In the cases of Figures 2-15a and 2-15b, representing the data as a multivalued attribute rather than via a relationship with another entity type may, in the view of some people, simplify the diagram. On the other hand, the right-hand drawings in these figures are closer to the way the database would be represented in a standard relational database management system, the most popular type of DBMS in use today. Although we are not concerned with implementation during conceptual data modeling, there is some logic for keeping the conceptual and logical data models similar. Further, as you will see in the next example, there are times when an attribute, whether simple, composite, or multivalued, should be in a separate entity.

So, when *should* an attribute be linked to an entity type via a relationship? The answer is when the attribute is the identifier or some other characteristic of an entity type in the data model and multiple entity instances need to share these same attributes. Figure 2-15c represents an example of this rule. In this example, EMPLOYEE has a composite attribute of Department. Because Department is a concept of the business and multiple employees will share the same department data, department data could be represented (nonredundantly) in a DEPARTMENT entity type, with attributes for the data about departments that all other related entity instances need to know. With this approach, not only can different employees share the storage of the same department data, but projects (which are assigned to a department) and organizational units (which are composed of departments) also can share the storage of this same department data.

Cardinality Constraints

There is one more important data modeling notation for representing common and important business rules. Suppose there are two entity types, A and B, that are connected by a relationship. A **cardinality constraint** specifies the number of instances of entity B that can (or must) be associated with each instance of entity A. For example, consider a video store that rents DVDs of movies. Because the store may stock more than one DVD for each movie, this is intuitively a one-to-many relationship, as shown in Figure 2-16a (technically, movie is a strong entity and DVD is a weak entity; thus, we represent these entities this way with a partial identifier for DVD). Yet it is also true that the store may not have any DVDs of a given movie in stock at a particular time (e.g., all copies may be checked out). We need a more precise notation to indicate the range of cardinalities for a relationship. This notation was introduced in Figure 2-2, which you may want to review at this time.

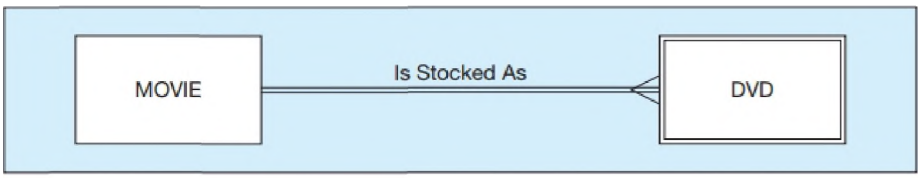
Cardinality constraint
A rule that specifies the number of instances of one entity that can (or must) be associated with each instance of another entity.

MINIMUM CARDINALITY The **minimum cardinality** of a relationship is the minimum number of instances of entity B that may be associated with each instance of entity A. In our DVD example, the minimum number of DVDs for a movie is zero. When the minimum number of participants is zero, we say that entity type B is an optional participant in the relationship. In this example, DVD (a weak entity type) is an optional participant in the Is Stocked As relationship. This fact is indicated by the symbol zero through the line near the DVD entity in Figure 2-16b.

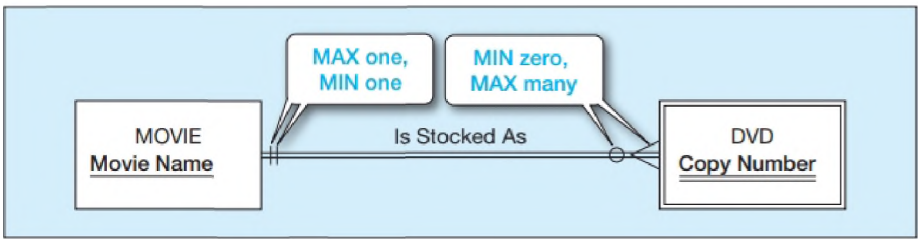
Minimum cardinality
The minimum number of instances of one entity that may be associated with each instance of another entity.

FIGURE 2-16 Introducing cardinality constraints

(a) Basic relationship



(b) Relationship with cardinality constraints



Maximum cardinality

The maximum number of instances of one entity that may be associated with each instance of another entity.

MAXIMUM CARDINALITY The **maximum cardinality** of a relationship is the maximum number of instances of entity B that may be associated with each instance of entity A. In the video example, the maximum cardinality for the DVD entity type is "many"—that is, an unspecified number greater than one. This is indicated by the "crow's foot" symbol on the line next to the DVD entity symbol in Figure 2-16b. (You might find interesting the explanation of the origin of the crow's foot notation found in the Wikipedia entry about the E-R model; this entry also shows the wide variety of notation used to represent cardinality; see **http://en.wikipedia.org/wiki/Entity-relationship_model.**)

A relationship is, of course, bidirectional, so there is also cardinality notation next to the MOVIE entity. Notice that the minimum and maximum are both one (see Figure 2-16b). This is called a *mandatory one* cardinality. In other words, each DVD of a movie must be a copy of exactly one movie. In general, participation in a relationship may be optional or mandatory for the entities involved. If the minimum cardinality is zero, participation is optional; if the minimum cardinality is one, participation is mandatory.

In Figure 2-16b, some attributes have been added to each of the entity types. Notice that DVD is represented as a weak entity. This is because a DVD cannot exist unless the owner movie also exists. The identifier of MOVIE is Movie Name. DVD does not have a unique identifier. However, Copy Number is a *partial* identifier, which together with Movie Name would uniquely identify an instance of DVD.

Some Examples of Relationships and Their Cardinalities

Examples of three relationships that show all possible combinations of minimum and maximum cardinalities appear in Figure 2-17. Each example states the business rule for each cardinality constraint and shows the associated E-R notation. Each example also shows some relationship instances to clarify the nature of the relationship. You should study each of these examples carefully. Following are the business rules for each of the examples in Figure 2-17:

1. ***PATIENT Has Recorded PATIENT HISTORY (Figure 2-17a)*** Each patient has one or more patient histories. (A PATIENT cannot exist unless there is an initial instance of PATIENT HISTORY.) Each instance of PATIENT HISTORY "belongs to" exactly one PATIENT.
2. ***EMPLOYEE Is Assigned To PROJECT (Figure 2-17b)*** Each PROJECT has at least one EMPLOYEE assigned to it. (Some projects have more than one.) Each EMPLOYEE may or (optionally) may not be assigned to any existing PROJECT (e.g., employee Pete) or may be assigned to one or more PROJECTs.
3. ***PERSON Is Married To PERSON (Figure 2-17c)*** This is an optional zero or one cardinality in both directions because a person may or may not be married at a given point in time.

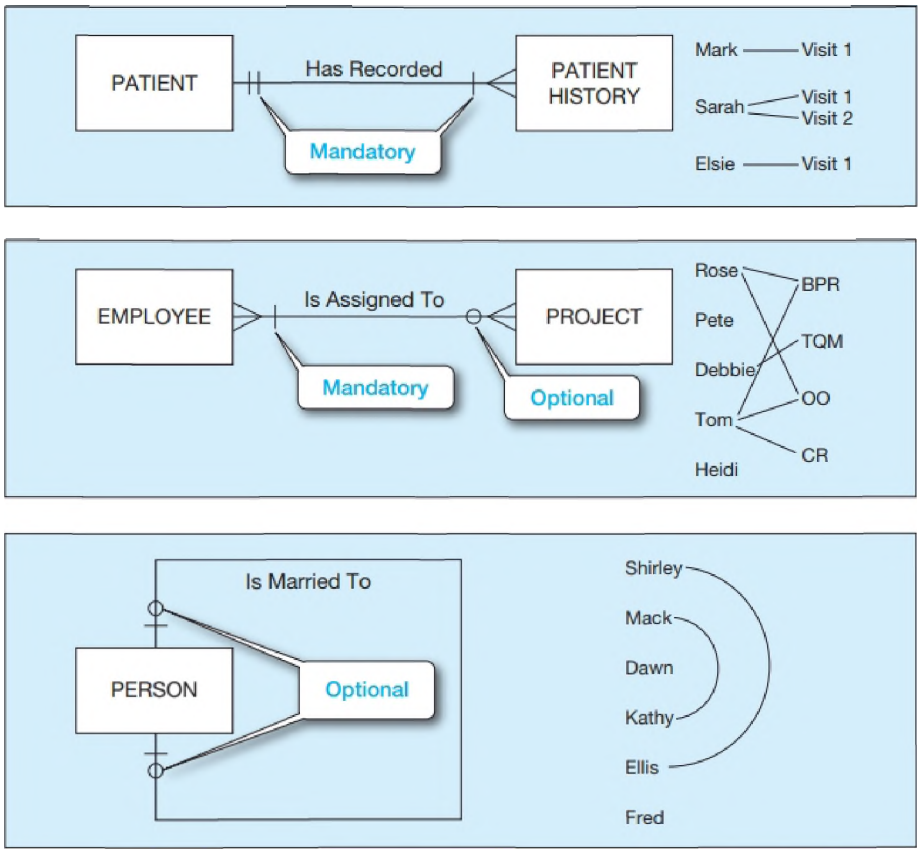


FIGURE 2-17 Examples of cardinality constraints

(a) Mandatory cardinalities

(b) One optional, one mandatory cardinality

(c) Optional cardinalities

It is possible for the maximum cardinality to be a fixed number, not an arbitrary "many" value. For example, suppose corporate policy states that an employee may work on at most five projects at the same time. We could show this business rule by placing a 5 above or below the crow's foot next to the PROJECT entity in Figure 2-17b.

A TERNARY RELATIONSHIP We showed the ternary relationship with the associative entity type SUPPLY SCHEDULE in Figure 2-14. Now let's add cardinality constraints to this diagram, based on the business rules for this situation. The E-R diagram, with the relevant business rules, is shown in Figure 2-18. Notice that PART and WAREHOUSE must relate to some SUPPLY SCHEDULE instance, and a VENDOR optionally may

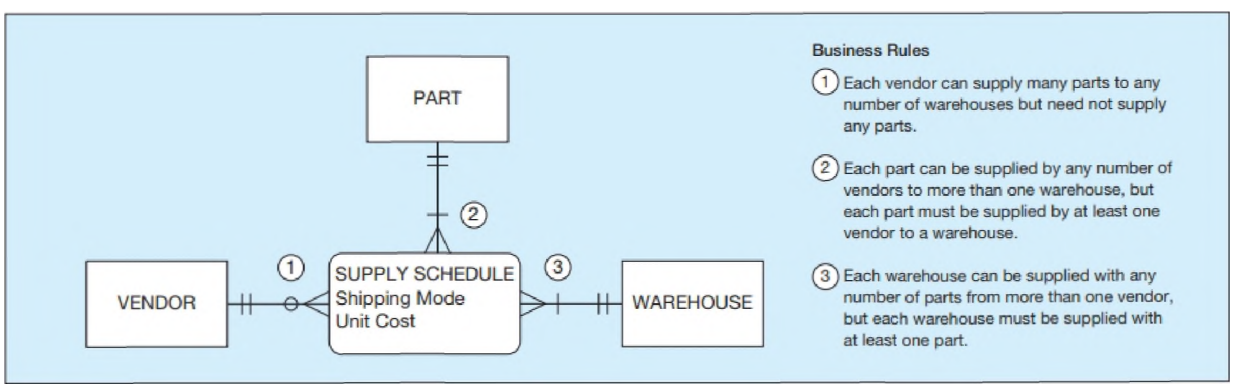


FIGURE 2-18 Cardinality constraints in a ternary relationship

not participate. The cardinality at each of the participating entities is a mandatory one because each SUPPLY SCHEDULE instance must be related to exactly one instance of each of these participating entity types. (Remember, SUPPLY SCHEDULE is an associative entity.)

As noted earlier, a ternary relationship is not equivalent to three binary relationships. Unfortunately, you are not able to draw ternary relationships with many CASE tools; instead, you are forced to represent ternary relationships as three binaries (i.e., an associative entity with three binary relationships). If you are forced to draw three binary relationships, then do not draw the binary relationships with names and be sure that the cardinality next to the three strong entities is a mandatory one.

Modeling Time-Dependent Data

Database contents vary over time. With renewed interest today in traceability and reconstruction of a historical picture of the organization for various regulatory requirements, such as HIPAA and Sarbanes-Oxley, and for business intelligence and other analytical purposes, the need to include a time series of data has become essential. For example, in a database that contains product information, the unit price for each product may be changed as material and labor costs and market conditions change. If only the current price is required, Price can be modeled as a single-valued attribute. However, for accounting, billing, financial reporting, and other purposes, we are likely to need to preserve a history of the prices and the time period during which each was in effect. As Figure 2-19 shows, we can conceptualize this requirement as a series of prices and the effective date for each price. This results in the (composite) multivalued attribute named Price History, with components Price and Effective Date. An important characteristic of such a composite, multivalued attribute is that the component attributes go together. Thus, in Figure 2-19, each Price is paired with the corresponding Effective Date.

Time stamp
A time value that is associated with a data value, often indicating when some event occurred that affected the data value.

In Figure 2-19, each value of the attribute Price is time stamped with its effective date. A **time stamp** is simply a time value, such as date and time, that is associated with a data value. A time stamp may be associated with any data value that changes over time when we need to maintain a history of those data values. Time stamps may be recorded to indicate the time the value was entered (transaction time), the time the value becomes valid or stops being valid, or the time when critical actions were performed, such as updates, corrections, or audits. This situation is similar to the employee skill diagrams in Figure 2-15b; thus, an alternative, not shown in Figure 2-19, is to make Price History a separate entity type, as was done with Skill using Microsoft Visio.

The use of simple time stamping (as in the preceding example) is often adequate for modeling time-dependent data. However, time can introduce subtler complexities to data modeling. For example, consider again Figure 2-17c. This figure is drawn for a given point in time, not to show history. If, on the other hand, we needed to record the full history of marriages for individuals, the Is Married To relationship would be an optional many-to-many relationship. Further, we might want to know the beginning and ending date (optional) of each marriage; these dates would be, similar to the bill-of-materials structure in Figure 2-13c, attributes of the relationship or associative entity.

Financial and other compliance regulations, such as Sarbanes-Oxley and Basel II, require that a database maintain history rather than just current status of critical data. In addition, some data modelers will argue that a data model should always be able to

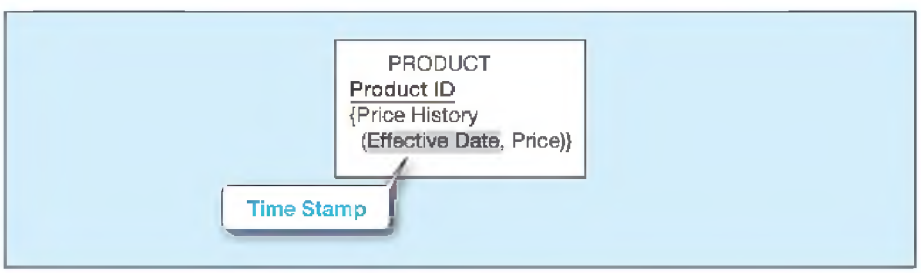


FIGURE 2-19 Simple example of time stamping

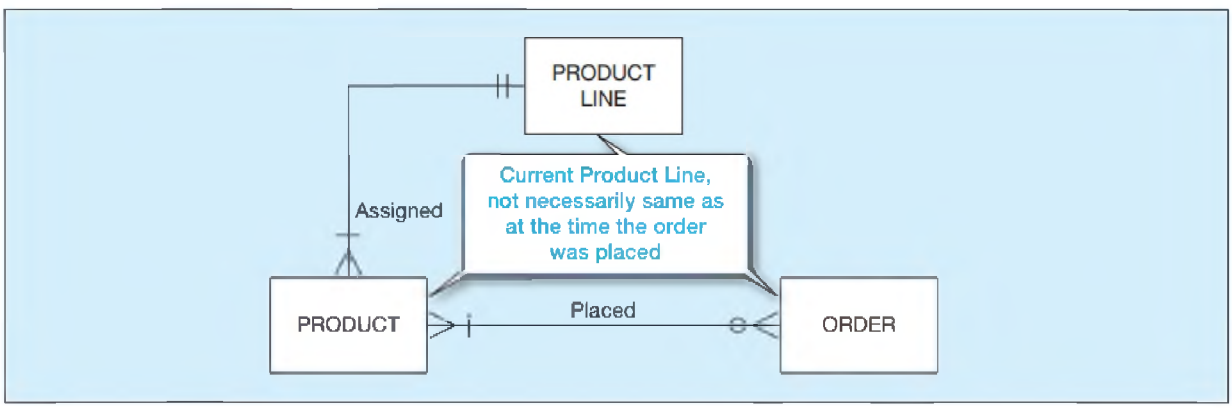
represent history, even if today's users say they need only current values. These factors suggest that all relationships should be modeled as many-to-many (which is often done in a purchased data model). Thus, for most databases, this will necessitate forming an associative entity along every relationship. There are two obvious negatives to this approach. First, many additional (associative) entities are created, thus cluttering ERDs. Second, a many-to-many (*M:N*) relationship is less restrictive than a one-to-many (1:*M*). So, if initially you want to enforce only one associated entity instance for some entity (i.e., the "one" side of the relationships), this cannot be enforced by the data model with an *M:N* relationship. It would seem likely that some relationships would never be *M:N*; for example, would a 1:*M* relationship between customer and order ever become *M:N* (but, of course, maybe someday our organization would sell items that would allow and often have joint purchasing, like vehicles or houses)? The conclusion is that if history or a time series of values might ever be desired or required by regulation, you should consider using an *M:N* relationship.

An even more subtle situation of the effect of time on data modeling is illustrated in Figure 2-20a, which represents a portion of an ERD for Pine Valley Furniture Company. Each product is assigned (i.e., current assignment) to a product line (or related group of products). Customer orders are processed throughout the year, and monthly summaries are reported by product line and by product within product line.

Suppose that in the middle of the year, due to a reorganization of the sales function, some products are reassigned to different product lines. The model shown in Figure 2-20a is not designed to track the reassignment of a product to a new product line. Thus, all sales reports will show cumulative sales for a product based on its current product line rather than the one at the time of the sale. For example, a product may have total year-to-date sales of $50,000 and be associated with product line B, yet $40,000 of

FIGURE 2-20 Example of time in Pine Valley Furniture product database

(a) E-R diagram not recognizing product reassignment



(b) E-R diagram recognizing product reassignment

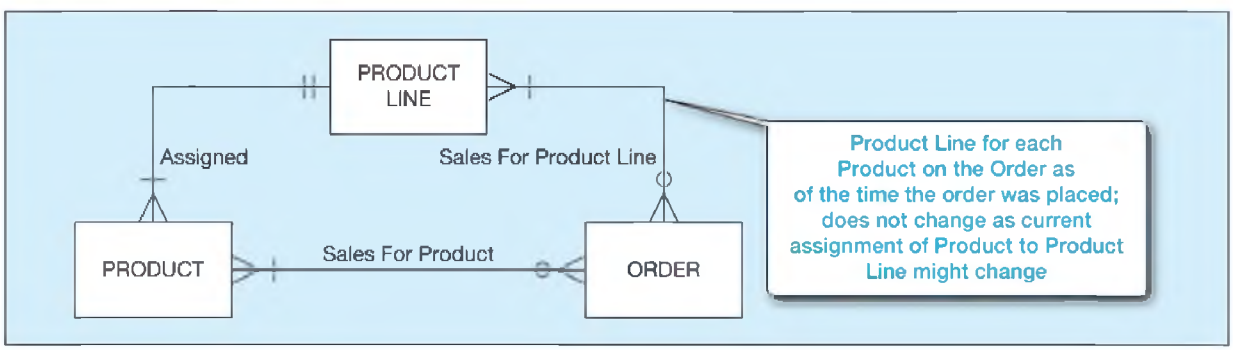
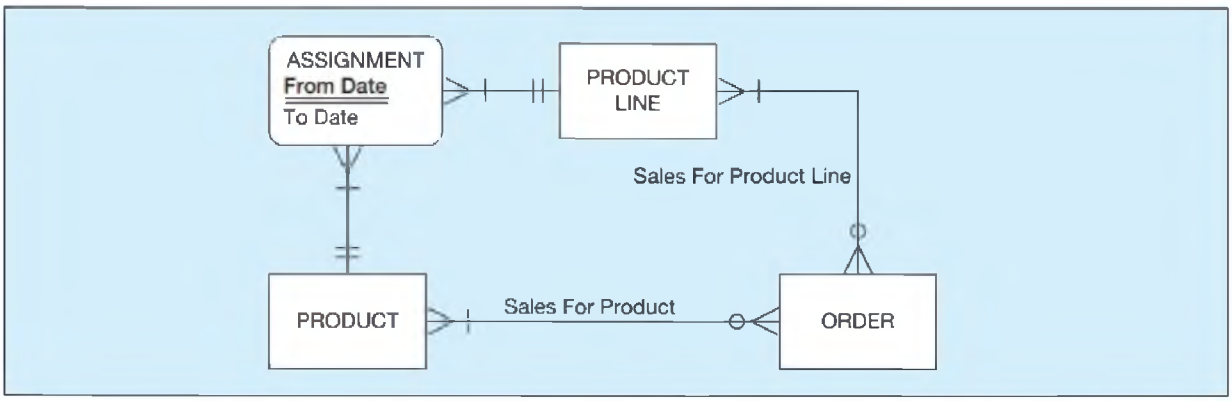


FIGURE 2-20 *(continued)*

(c) E-R diagram with associative entity for product assignment to product line over time



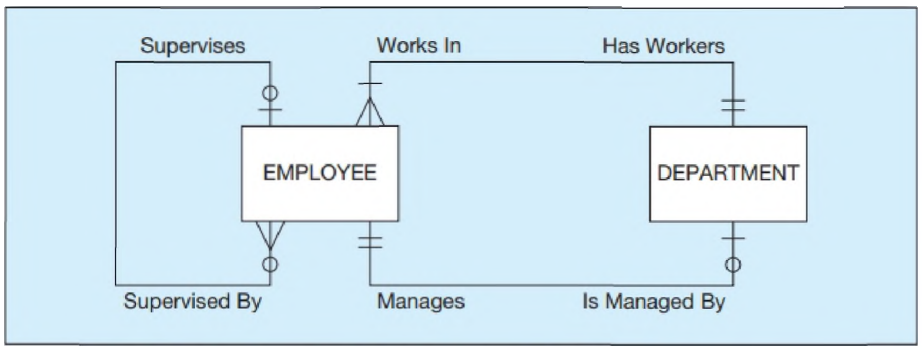
those sales may have occurred while the product was assigned to product line A. This fact will be lost using the model in Figure 2-20a. The simple design change shown in Figure 2-20b will correctly recognize product reassignments. A new relationship, called Sales For Product Line, has been added between ORDER and PRODUCT LINE. As customer orders are processed, they are credited to both the correct product (via Sales For Product) and the correct product line (via Sales For Product Line) as of the time of the sale. The approach of Figure 2-20b is similar to what is done in a data warehouse to retain historical records of the precise situation at any point in time. (We will return to dealing with the time dimension in Chapter 9.)

Another aspect of modeling time is recognizing that although the requirements of the organization today may be to record only the current situation, the design of the database may need to change if the organization ever decides to keep history. In Figure 2-20b, we know the current product line for a product and the product line for the product each time it is ordered. But what if the product were ever reassigned to a product line during a period of zero sales for the product? Based on this data model in Figure 2-20b, we would not know of these other product line assignments. A common solution to this need for greater flexibility in the data model is to consider whether a one-to-many relationship, such as Assigned, should become a many-to-many relationship. Further, to allow for attributes on this new relationship, this relationship should actually be an associative entity. Figure 2-20c shows this alternative data model with the ASSIGNMENT associative entity for the Assigned relationship. The advantage of the alternative is that we now will not miss recording any product line assignment, and we can record information about the assignment (such as the from and to effective dates of the assignment); the disadvantage is that the data model no longer has the restriction that a product may be assigned to only one product line at a time.

We have discussed the problem of time-dependent data with managers in several organizations who are considered leaders in the use of data modeling and database management. Before the recent wave of financial reporting disclosure regulations, these discussions revealed that data models for operational databases were generally inadequate for handing time-dependent data and that organizations often ignored this problem and hoped that the resulting inaccuracies balanced out. However, with these new regulations, you need to be alert to the complexities posed by time-dependent data as you develop data models in your organization. For a thorough explanation of time as a dimension of data modeling, see a series of articles by T. Johnson and R. Weis beginning in May 2007 in *DM Review* (now *Information Management*; see References at the end of this chapter).

Modeling Multiple Relationships Between Entity Types

There may be more than one relationship between the same entity types in a given organization. Two examples are shown in Figure 2-21. Figure 2-21a shows two relationships



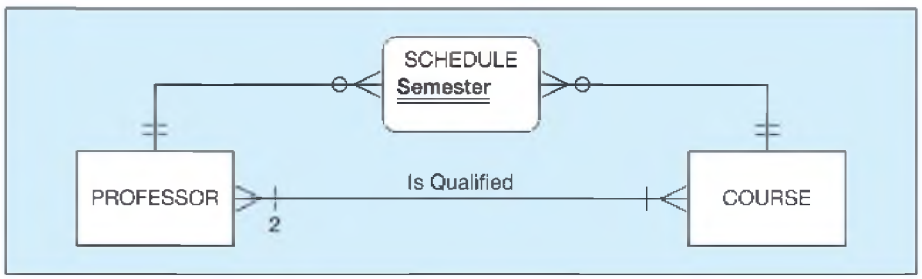


FIGURE 2-21 Examples of multiple relationships

(a) Employees and departments

(b) Professors and courses (fixed lower limit constraint)

between the entity types EMPLOYEE and DEPARTMENT. In this figure, we use the notation with names for the relationship in each direction; this notation makes explicit what the cardinality is for each direction of the relationship (which becomes important for clarifying the meaning of the unary relationship on EMPLOYEE). One relationship associates employees with the department in which they work. This relationship is one-to-many in the Has Workers direction and is mandatory in both directions. That is, a department must have at least one employee who works there (perhaps the department manager), and each employee must be assigned to exactly one department. (Note: These are specific business rules we assume for this illustration. It is crucial when you develop an E-R diagram for a particular situation that you understand the business rules that apply for that setting. For example, if EMPLOYEE were to include retirees, then each employee may not be currently assigned to exactly one department; further, the E-R model in Figure 2-21a assumes that the organization needs to remember in which DEPARTMENT each EMPLOYEE currently works rather than remembering the history of department assignments. Again, the structure of the data model reflects the information the organization needs to remember.)

The second relationship between EMPLOYEE and DEPARTMENT associates each department with the employee who manages that department. The relationship from DEPARTMENT to EMPLOYEE (called Is Managed By in that direction) is a mandatory one, indicating that a department must have exactly one manager. From EMPLOYEE to DEPARTMENT, the relationship (Manages) is optional because a given employee either is or is not a department manager.

Figure 2-21a also shows the unary relationship that associates each employee with his or her supervisor and vice versa. This relationship records the business rule that each employee may have exactly one supervisor (Supervised By). Conversely, each employee may supervise any number of employees or may not be a supervisor.

The example in Figure 2-21b shows two relationships between the entity types PROFESSOR and COURSE. The relationship Is Qualified associates professors with the courses they are qualified to teach. A given course must have at a minimum two qualified instructors (an example of how to use a fixed value for a minimum or maximum cardinality). This might happen, for example, so that a course is never the "property" of one instructor. Conversely, each instructor must be qualified to teach at least one course (a reasonable expectation).

The second relationship in this figure associates professors with the courses they are actually scheduled to teach during a given semester. Because Semester is a characteristic of the relationship, we place an associative entity, SCHEDULE, between PROFESSOR and COURSE.

One final point about Figure 2-21b: Have you figured out what the identifier is for the SCHEDULE associative entity? Notice that Semester is a partial identifier; thus, the full identifier will be the identifier of PROFESSOR along with the identifier of COURSE as well as Semester. Because such full identifiers for associative entities can become long and complex, it is often recommended that surrogate identifiers be created for each associative entity; so, Schedule ID would be created as the identifier of SCHEDULE, and Semester would be an attribute. What is lost in this case is the explicit business rule that the combination of the PROFESSOR identifier, COURSE identifier, and Semester must be unique for each SCHEDULE instance (because this combination is the identifier of SCHEDULE). Of course, this can be added as another business rule.

Naming and Defining Relationships

In addition to the general guidelines for naming data objects, there are a few special guidelines for naming relationships, which follow:

- A relationship name is a *verb phrase* (such as Assigned To, Supplies, or Teaches). Relationships represent actions being taken, usually in the present tense, so transitive verbs (an action on something) are the most appropriate. A relationship name states the action taken, not the result of the action (e.g., use Assigned To, not Assignment). The name states the essence of the interaction between the participating entity types, not the process involved (e.g., use an Employee is *Assigned To* a project, not an Employee is *Assigning* a project).
- You should *avoid vague names*, such as Has or Is Related To. Use descriptive, powerful verb phrases, often taken from the action verbs found in the definition of the relationship.

There are also some specific guidelines for defining relationships, which follow:

- A relationship definition *explains what action is being taken and possibly why it is important*. It may be important to state who or what does the action, but it is not important to explain how the action is taken. Stating the business objects involved in the relationship is natural, but because the E-R diagram shows what entity types are involved in the relationship and other definitions explain the entity types, you do not have to describe the business objects.
- It may also be important to *give examples to clarify the action*. For example, for a relationship of Registered For between student and course, it may be useful to explain that this covers both on-site and online registration, and includes registrations made during the drop/add period.
- The definition should explain any *optional participation*. You should explain what conditions lead to zero associated instances, whether this can happen only when an entity instance is first created, or whether this can happen at any time. For example, "Registered For links a course with the students who have signed up to take the course, and the courses a student has signed up to take. A course will have no students registered for it before the registration period begins and may never have any registered students. A student will not be registered for any courses before the registration period begins and may not register for any classes (or may register for classes and then drop any or all classes)."
- A relationship definition should also *explain the reason for any explicit maximum cardinality* other than many. For example, "Assigned To links an employee with the projects to which that employee is assigned and the employees assigned to a project. Due to our labor union agreement, an employee may not be assigned to more than four projects at a given time." This example, typical of many upper-bound business rules, suggests that maximum cardinalities tend not to be permanent. In this example, the next labor union agreement could increase or decrease this limit. Thus, the implementation of maximum cardinalities must be done to allow changes.

- A relationship definition should *explain any mutually exclusive relationships.* Mutually exclusive relationships are ones for which an entity instance can participate in only one of several alternative relationships. We will show examples of this situation in Chapter 3. For now, consider the following example: "Plays On links an intercollegiate sports team with its student players and indicates on which teams a student plays. Students who play on intercollegiate sports teams cannot also work in a campus job (i.e., a student cannot be linked to both an intercollegiate sports team via Plays On and a campus job via the Works On relationship)." Another example of a mutually exclusive restriction is when an employee cannot both be Supervised By and be Married To the same employee.
- A relationship definition should *explain any restrictions on participation in the relationship.* Mutual exclusivity is one restriction, but there can be others. For example, "Supervised By links an employee with the other employees he or she supervises and links an employee with the other employee who supervises him or her. An employee cannot supervise him- or herself, and an employee cannot supervise other employees if his or her job classification level is below 4."
- A relationship definition should *explain the extent of history that is kept in the relationship.* For example, "Assigned To links a hospital bed with a patient. Only the current bed assignment is stored. When a patient is not admitted, that patient is not assigned to a bed, and a bed may be vacant at any given point in time." Another example of describing history for a relationship is "Places links a customer with the orders he or she has placed with our company and links an order with the associated customer. Only two years of orders are maintained in the database, so not all orders can participate in this relationship."
- A relationship definition should *explain whether an entity instance involved in a relationship instance can transfer participation to another relationship instance.* For example, "Places links a customer with the orders he or she has placed with our company and links an order with the associated customer. An order is not transferable to another customer." Another example is "Categorized As links a product line with the products sold under that heading and links a product to its associated product line. Due to changes in organization structure and product design features, products may be recategorized to a different product line. Categorized As keeps track of only the current product line to which a product is linked."

E-R MODELING EXAMPLE: PINE VALLEY FURNITURE COMPANY

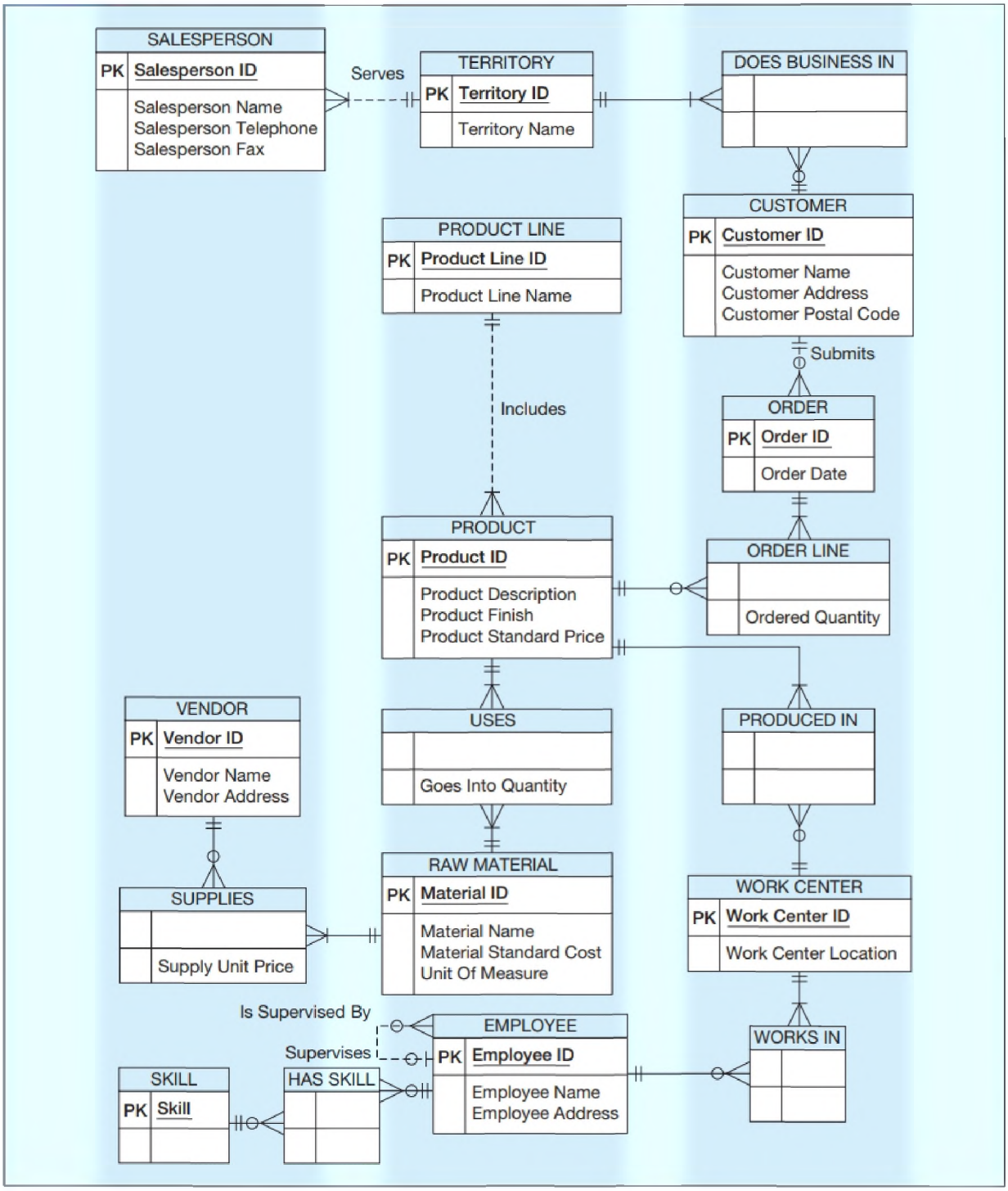
You can develop an E-R diagram from one (or both) of two perspectives. With a top-down perspective, you proceed from basic descriptions of the business, including its policies, processes, and environment. This approach is most appropriate for developing a high-level E-R diagram with only the major entities and relationships and with a limited set of attributes (such as just the entity identifiers). With a bottom-up approach, you proceed from detailed discussions with users, and from a detailed study of documents, screens, and other data sources. This approach is necessary for developing a detailed, "fully attributed" E-R diagram.

In this section, we illustrate a high-level ERD for Pine Valley Furniture Company, based largely on the first of these approaches (see Figure 2-22 for a Microsoft Visio version). For simplicity, we do not show any composite or multivalued attributes (e.g., skill is shown as a separate entity type associated with EMPLOYEE via an associative entity, which allows an employee to have many skills and a skill to be held by many employees).

Figure 2-22 provides many examples of common E-R modeling notations, and hence, it can be used as an excellent review of what you have learned in this chapter. In a moment, we will explain the business rules that are represented in this figure. However, before you read that explanation, one way to use Figure 2-22 is to search for typical E-R model constructs in it, such as one-to-many, binary, or unary relationships. Then ask yourself why the business data was modeled this way. For example, ask yourself:

- Where is a unary relationship, what does it mean, and for what reasons might the cardinalities on it be different in other organizations?

FIGURE 2-22 Data model for Pine Valley Furniture Company in Microsoft Visio notation



- Why is Includes a one-to many relationship, and why might this ever be different in some other organization?
- Does Includes allow for a product to be represented in the database before it is assigned to a product line (e.g., while the product is in research and development)?

- If there were a different customer contact person for each sales territory in which a customer did business, where in the data model would we place this person's name?
- What is the meaning of the Does Business In associative entity, and why does each Does Business In instance have to be associated with exactly one SALES TERRITORY and one CUSTOMER?
- In what way might Pine Valley change the way it does business that would cause the Supplies associative entity to be eliminated and the relationships around it to change?

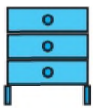
Each of these questions is included in Problem and Exercise 2-25 at the end of the chapter, but we suggest you use these now as a way to review your understanding of E-R diagramming.

From a study of the business processes at Pine Valley Furniture Company, we have identified the following entity types. An identifier is also suggested for each entity, together with selected important attributes:

- The company sells a number of different furniture products. These products are grouped into several product lines. The identifier for a product is Product ID, whereas the identifier for a product line is Product Line ID. We identify the following additional attributes for product: Product Description, Product Finish, and Product Standard Price. Another attribute for product line is Product Line Name. A product line may group any number of products but must group at least one product. Each product must belong to exactly one product line.
- Customers submit orders for products. The identifier for an order is Order ID, and another attribute is Order Date. A customer may submit any number of orders but need not submit any orders. Each order is submitted by exactly one customer. The identifier for a customer is Customer ID. Other attributes include Customer Name, Customer Address, and Customer Postal Code.
- A given customer order must request at least one product and only one product per order line item. Any product sold by Pine Valley Furniture may not appear on any order line item or may appear on one or more order line items. An attribute associated with each order line item is Ordered Quantity.
- Pine Valley Furniture has established sales territories for its customers. Each customer may do business in any number of these sales territories or may not do business in any territory. A sales territory has one to many customers. The identifier for a sales territory is Territory ID and an attribute is Territory Name.
- Pine Valley Furniture Company has several salespersons. The identifier for a salesperson is Salesperson ID. Other attributes include Salesperson Name, Salesperson Telephone, and Salesperson Fax. A salesperson serves exactly one sales territory. Each sales territory is served by one or more salespersons.
- Each product is assembled from a specified quantity of one or more raw materials. The identifier for the raw material entity is Material ID. Other attributes include Unit Of Measure, Material Name, and Material Standard Cost. Each raw material is assembled into one or more products, using a specified quantity of the raw material for each product.
- Raw materials are supplied by vendors. The identifier for a vendor is Vendor ID. Other attributes include Vendor Name and Vendor Address. Each raw material can be supplied by one or more vendors. A vendor may supply any number of raw materials or may not supply any raw materials to Pine Valley Furniture. Supply Unit Price is the unit price at which a particular vendor supplies a particular raw material.
- Pine Valley Furniture has established a number of work centers. The identifier for a work center is Work Center ID. Another attribute is Work Center Location. Each product is produced in one or more work centers. A work center may be used to produce any number of products or may not be used to produce any products.
- The company has more than 100 employees. The identifier for employee is Employee ID. Other attributes include Employee Name, Employee Address, and Skill. An employee may have more than one skill. Each employee may work in one or more work centers. A work center must have at least one employee working in

that center but may have any number of employees. A skill may be possessed by more than one employee or possibly no employees.

- Each employee has exactly one supervisor; however, a manager has no supervisor. An employee who is a supervisor may supervise any number of employees, but not all employees are supervisors.



DATABASE PROCESSING AT PINE VALLEY FURNITURE

The purpose of the data model diagram in Figure 2-22 is to provide a conceptual design for the Pine Valley Furniture Company database. It is important to check the quality of such a design through frequent interaction with the persons who will use the database after it is implemented. An important and often performed type of quality check is to determine whether the E-R model can easily satisfy user requests for data and/or information. Employees at Pine Valley Furniture have many data retrieval and reporting requirements. In this section, we show how a few of these information requirements can be satisfied by database processing against the database shown in Figure 2-22.

We use the SQL database processing language (explained in Chapters 5 and 6) to state these queries. To fully understand these queries, you will need to understand concepts introduced in Chapter 4. However, a few simple queries in this chapter should help you understand the capabilities of a database to answer important organizational questions and give you a jump-start toward understanding SQL queries in Chapter 5 as well as in later chapters.

Showing Product Information

Many different users have a need to see data about the products Pine Valley Furniture produces (e.g., salespersons, inventory managers, and product managers). One specific need is for a salesperson who wants to respond to a request from a customer for a list of products of a certain type. An example of this query is

> List all details for the various computer desks that are stocked by the company.

The data for this query are maintained in the PRODUCT entity (see Figure 2-22). The query scans this entity and displays all the attributes for products that contain the description Computer Desk.

The SQL code for this query is

```
SELECT *
FROM Product
WHERE ProductDescription LIKE "Computer Desk%";
```

Typical output for this query is

PRODUCTID	PRODUCTDESCRIPTION	PRODUCTFINISH	PRODUCTSTANDARDPRICE
3	Computer Desk 48"	Oak	375.00
8	Computer Desk 64"	Pine	450.00

SELECT * FROM Product says display all attributes of PRODUCT entities. The WHERE clause says to limit the display to only products whose description begins with the phrase Computer Desk.

Showing Product Line Information

Another common information need is to show data about Pine Valley Furniture product lines. One specific type of person who needs this information is a product manager. The following is a typical query from a territory sales manager:

List the details of products in product line 4.

The data for this query are maintained in the PRODUCT entity. As we explain in Chapter 4, the attribute Product Line ID will be added to the PRODUCT entity when a data model in Figure 2-22 is translated into a database that can be accessed via SQL. The query scans the PRODUCT entity and displays all attributes for products that are in the selected product line.

The SQL code for this query is

```
SELECT *
FROM Product
WHERE ProductLineID = 4;
```

Typical output for this query is

PRODUCTID	PRODUCTDESCRIPTION	PRODUCTFINISH	PRODUCTSTANDARDPRICE	PRODUCTONHAND	PRODUCTLINEID
18	Grandfather Clock	Oak	890.0000	0	4
19	Grandfather Clock	Oak	1100.0000	0	4

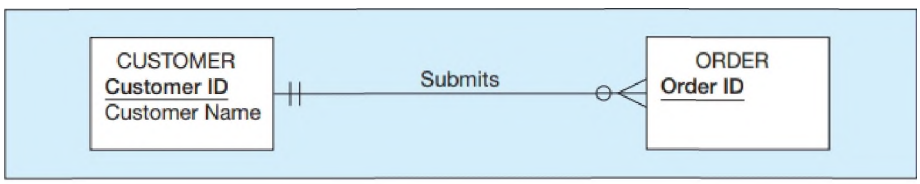
The explanation of this SQL query is similar to the explanation of the previous one.

Showing Customer Order Status

The previous two queries are relatively simple, involving data from only one table in each case. Often, data from multiple tables are needed in one information request. Although the previous query is simple, we did have to look through the whole database to find the entity and attributes needed to satisfy the request.

To simplify query writing and for other reasons, many database management systems support creating restricted views of a database suitable for the information needs of a particular user. For queries related to customer order status, Pine Valley utilizes such a user view called "Orders for customers," which is created from the segment of an E-R diagram for PVFC shown in Figure 2-23a. This user view allows users to see only CUSTOMER and ORDER entities in the database, and only the attributes of these entities shown in the figure. For the user, there is only one (virtual) table, ORDERS FOR CUSTOMERS, with the listed attributes. As we explain in Chapter 4, the attribute Customer ID will be added to the ORDER entity (as shown in Figure 2-23a). A typical order status query is:

How many orders have we received from Value Furniture?



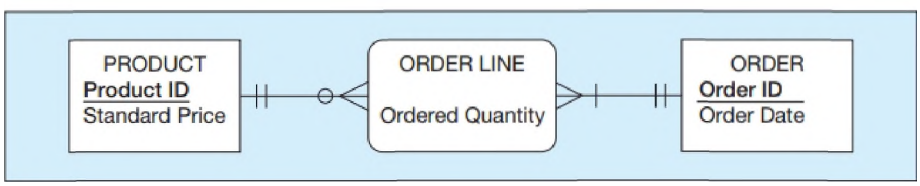


FIGURE 2-23 Two user views for Pine Valley Furniture

(a) User View 1: Orders for customers

(b) User View 2: Orders for products

Assuming that all the data we need are pulled together into this one user view, or virtual entity, called OrdersForCustomers, we can simply write the query as follows:

```
SELECT COUNT(Order ID)
FROM OrdersForCustomers
WHERE CustomerName = "Value Furniture";
```

Without the user view, we can write the SQL code for this query in several ways. The way we have chosen is to compose a query within a query, called a *subquery*. (We will explain subqueries in Chapter 6, with some diagramming techniques to assist you in composing the query.) The query is performed in two steps. First, the subquery (or inner query) scans the CUSTOMER entity to determine the Customer ID for the customer named Value Furniture. (The ID for this customer is 5, as shown in the output for the previous query.) Then the query (or outer query) scans the ORDER entity and counts the order instances for this customer.

The SQL code for this query without the "Orders for customer" user view is as follows:

```
SELECT COUNT (OrderID)
FROM Order
WHERE CustomerID =
  (SELECT CustomerID
  FROM Customer
  WHERE CustomerName = "Value Furniture");
```

For this example query, using a subquery rather than a view did not make writing the query much more complex.

Typical output for this query using either of the query approaches above is:

```
COUNT(ORDERID)
             4
```

Showing Product Sales

Salespersons, territory managers, product managers, production managers, and others have a need to know the status of product sales. One kind of sales question is what products are having an exceptionally strong sales month. Typical of this question is the following query:

> What products have had total sales exceeding $25,000 during the past month (June, 2018)?

This query can be written using the user view "Orders for products," which is created from the segment of an E-R diagram for PVFC shown in Figure 2-23b. Data to respond to the query are obtained from the following sources:

- Order Date from the ORDER entity (to find only orders in the desired month).
- Ordered Quantity for each product on each order from the associative entity ORDER LINE for an ORDER entity in the desired month.
- Standard Price for the product ordered from the PRODUCT entity associated with the ORDER LINE entity.

For each item ordered during the month of June 2018, the query needs to multiply Ordered Quantity by Product Standard Price to get the dollar value of a sale. For the

user, there is only one (virtual) table, ORDERS FOR PRODUCTS, with the listed attributes. The total amount is then obtained for that item by summing all orders. Data are displayed only if the total exceeds $25,000.

The SQL code for this query is beyond the scope of this chapter because it requires techniques introduced in Chapter 6. We introduce this query now only to suggest the power that a database such as the one shown in Figure 2-22 has to find information for management from detailed data. In many organizations today, users can use a Web browser to obtain the information described here. The programming code associated with a Web page then invokes the required SQL commands to obtain the requested information.

Summary

This chapter has described the fundamentals of modeling data in the organization. Business rules, derived from policies, procedures, events, functions, and other business objects, state constraints that govern the organization and, hence, how data are handled and stored. Using business rules is a powerful way to describe the requirements for an information system, especially a database. The power of business rules results from business rules being core concepts of the business, being able to be expressed in terms familiar to end users, being highly maintainable, and being able to be enforced through automated means, mainly through a database. Good business rules are ones that are declarative, precise, atomic, consistent, expressible, distinct, and business oriented.

Examples of basic business rules are data names and definitions. This chapter explained guidelines for the clear naming and definition of data objects in a business. In terms of conceptual data modeling, names and definitions must be provided for entity types, attributes, and relationships. Other business rules may state constraints on these data objects. These constraints can be captured in a data model and associated documentation.

The data modeling notation most frequently used today is the E-R data model. An E-R model is a detailed, logical representation of the data for an organization. An E-R model is usually expressed in the form of an E-R diagram, which is a graphical representation of an E-R model. The E-R model was introduced by Chen in 1976. However, at the present time, there is no standard notation for E-R modeling. Notations such as those found in Microsoft Visio are used in many CASE tools.

The basic constructs of an E-R model are entity types, relationships, and related attributes. An entity is a person, a place, an object, an event, or a concept in the user environment about which the organization wishes to maintain data. An entity type is a collection of entities that share common properties, whereas an entity instance is a single occurrence of an entity type. A strong entity type is an entity that has its own identifier and can exist without other entities. A weak entity type is an entity whose existence depends on the existence of a strong entity type. Weak entities do not have their own identifier, although they normally have a partial identifier. Weak entities are identified through an identifying relationship with their owner entity type.

An attribute is a property or characteristic of an entity or relationship that is of interest to the organization. There are several types of attributes. A required attribute must have a value for an entity instance, whereas an optional attribute value may be null. A simple attribute is one that has no component parts. A composite attribute is an attribute that can be broken down into component parts. For example, Person Name can be broken down into the parts First Name, Middle Initial, and Last Name.

A multivalued attribute is one that can have multiple values for a single instance of an entity. For example, the attribute College Degree might have multiple values for an individual. A derived attribute is one whose values can be calculated from other attribute values. For example, Average Salary can be calculated from values of Salary for all employees.

An identifier is an attribute that uniquely identifies individual instances of an entity type. Identifiers should be chosen carefully to ensure stability and ease of use. Identifiers may be simple attributes, or they may be composite attributes with component parts.

A relationship type is a meaningful association between (or among) entity types. A relationship instance is an association between (or among) entity instances. The degree of a relationship is the number of entity types that participate in the relationship. The most common relationship types are unary (degree 1), binary (degree 2), and ternary (degree 3).

In developing E-R diagrams, you sometimes encounter many-to-many (and one-to-one) relationships that have one or more attributes associated with the relationship, rather than with one of the participating entity types. In such cases, you might consider converting the relationship to an associative entity. This type of entity associates the instances of one or more entity types and contains attributes that are peculiar to the relationship. Associative entity types may have their own simple identifier, or they may be assigned a composite identifier during logical design.

A cardinality constraint is a constraint that specifies the number of instances of entity B that may (or must) be associated with each instance of entity A. Cardinality constraints normally specify the minimum and maximum number of instances. The possible constraints are mandatory one, mandatory many, optional one, optional

many, and a specific number. The minimum cardinality constraint is also referred to as the participation constraint. A minimum cardinality of zero specifies optional participation, whereas a minimum cardinality of one specifies mandatory participation.

Because many databases need to store the value of data over time, modeling time-dependent data is an important part of data modeling. Data that repeat over time may be modeled as multivalued attributes or as separate entity instances; in each case, a time stamp is necessary to identify the relevant date and time for the data value. Sometimes separate relationships need to be included in the data model to represent associations at different points in time. The recent wave of financial reporting disclosure regulations have made it more important to include time-sensitive and historical data in databases.

Chapter Review

Key Terms

Associative entity *78*
Attribute *71*
Binary relationship *82*
Business rule *62*
Cardinality constraint *85*
Composite attribute *72*
Composite identifier *74*
Degree *80*
Derived attribute *73*
Entity *67*
Entity instance *67*
Entity-relationship diagram (E-R diagram) *58*
Entity-relationship model (E-R model) *58*
Entity type *67*
Fact *65*
Identifier *73*
Identifying owner *68*
Identifying relationship *69*
Maximum cardinality *85*
Minimum cardinality *86*
Multivalued attribute *73*
Optional attribute *71*
Relationship instance *77*
Relationship type *77*
Required attribute *71*
Simple (or atomic) attribute *72*
Strong entity type *68*
Term *65*
Ternary relationship *82*
Time stamp *88*
Unary relationship *81*
Weak entity type *68*

Review Questions

2-1. Define each of the following terms:
a. entity type
b. entity-relationship model
c. entity instance
d. attribute
e. relationship type
f. strong entity type
g. multivalued attribute
h. associative entity
i. cardinality constraint
j. weak entity
k. binary relationship
l. derived attribute
m. business rule

2-2. Match the following terms and definitions.

_____ composite attribute
_____ associative entity
_____ unary relationship
_____ weak entity
_____ attribute
_____ entity
_____ relationship type
_____ cardinality constraint
_____ degree
_____ identifier
_____ entity type
_____ ternary
_____ optional attribute

a. uniquely identifies entity instances
b. relates instances of a single entity type
c. specifies maximum and minimum number of instances
d. relationship modeled as an entity type
e. association between entity types
f. collection of similar entities
g. number of participating entity types in relationship
h. property of an entity
i. can be broken into component parts
j. depends on the existence of another entity type
k. relationship of degree 3
l. may not have a value
m. person, place, object, concept, or event

2-3. Contrast the following terms:
a. stored attribute; derived attribute
b. minimum cardinality; maximum cardinality
c. entity type; relationship type
d. strong entity type; weak entity type
e. degree; cardinality
f. required attribute; optional attribute
g. composite attribute; multivalued attribute
h. ternary relationship; three binary relationships

2-4. Give four reasons why many system designers believe that data modeling is important and arguably the most important part of the systems development process.

2-5. Give four reasons why a business rules approach is advocated as a new paradigm for specifying information systems requirements.

2-6. Explain where you can find business rules in an organization.

2-7. State six general guidelines for naming data objects in a data model.

2-8. State four criteria for selecting identifiers for entities.

2-9. Why must some identifiers be composite rather than simple?

2-10. State three conditions that suggest the designer should model a relationship as an associative entity type.

2-11. List the four types of cardinality constraints, and draw an example of each.

2-12. Give an example, other than those described in this chapter, of a weak entity type. What is the identifier attribute of a weak entity?

2-13. What is the degree of a relationship? List the three types of relationship degrees described in the chapter and give an example of each.

2-14. Give an example (other than those described in this chapter) for each of the following, and justify your answer:
a. derived attribute
b. multivalued attribute
c. atomic attribute
d. composite attribute
e. composite identifier attribute
f. optional attribute

2-15. Give an example of each of the following, other than those described in this chapter, and clearly explain why your example is this type of relationship and not of some other degree.
a. ternary relationship
b. unary relationship

2-16. Give an example of the use of effective (or effectivity) dates as attributes of an entity.

2-17. State a rule that says when to extract an attribute from one entity type and place it in a linked entity type.

2-18. What are the special guidelines for naming relationships?

2-19. In addition to explaining what action is being taken, what else should a relationship definition explain?

2-20. For the Manages relationship in Figure 2-12a, describe one or more situations that would result in different cardinalities on the two ends of this unary relationship. Based on your description for this example, do you think it is always clear simply from an E-R diagram what the business rule is that results in certain cardinalities? Justify your answer.

2-21. Explain the distinction between entity type and entity instance.

2-22. Why is it recommended that every ternary relationship be converted into an associative entity?

Problems and Exercises

2-23. A cellular operator needs a database to keep track of its customers, their subscription plans, and the handsets (mobile phones) that they are using. The E-R diagram in Figure 2-24 illustrates the key entities of interest to the operator and the relationships between them. Based on the figure, answer the following questions and explain the rationale for your response. For each question, identify the element(s) in the E-R diagram that you used to determine your answer.
a. Can a customer have an unlimited number of plans?
b. Can a customer exist without a plan?
c. Is it possible to create a plan without knowing who the customer is?
d. Does the operator want to limit the types of handsets that can be linked to a specific plan type?
e. Is it possible to maintain data regarding a handset without connecting it to a plan?
f. Can a handset be associated with multiple plans?
g. Assume a handset type exists that can utilize multiple operating systems. Could this situation be accommodated within the model included in Figure 2-24?
h. Is the company able to track a manufacturer without maintaining information about its handsets?
i. Can the same operating system be used on multiple handset types?
j. There are two relationships between Customer and Plan. Explain how they differ.
k. Characterize the degree and the cardinalities of the relationship that connects Customer to itself. Explain its meaning.
l. Is it possible to link a handset to a specific customer in a plan with multiple customers?
m. Can the company track a handset without identifying its operating system?

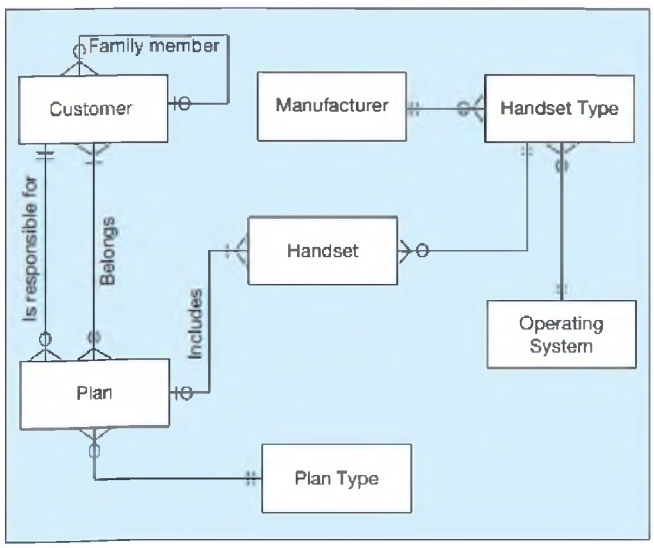
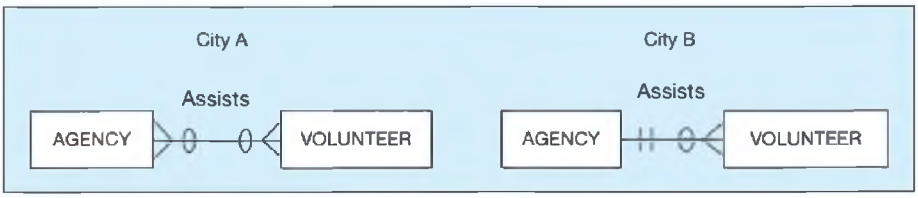


FIGURE 2-24 Diagram for Problem and Exercise 2-23

2-24. For each of the descriptions below, perform the following tasks:
i. Identify the degree and cardinalities of each relationship.
ii. Express the relationships in each description graphically with an E-R diagram.

a. A book is identified by its ISBN number, and it has a title, a price, and a date of publication. It is published by a publisher, which has its own ID number and a name. Each book has exactly one publisher, but one publisher typically publishes multiple books over time.
b. A book (see 2a) is written by one or multiple authors. Each author is identified by an author number and has a name and date of birth. Each author has either one or multiple books; in addition, occasionally data are needed regarding prospective authors who have not yet published any books.
c. In the context specified in 2a and 2b, better information is needed regarding the relationship between a book and its authors. Specifically, it is important to record the percentage of the royalties that belongs to a specific author, whether or not a specific author is a lead author of the book, and each author's position in the sequence of the book's authors.

FIGURE 2-25 Diagram for Problem and Exercise 2-28



d. A book (see 2a) can be part of a series, which is also identified as a book and has its own ISBN number. One book can belong to several sets, and a set consists of at least one but potentially many books.

e. Ebony and Ivory, a piano manufacturer, wants to keep track of all the pianos it makes individually. Each piano has an identifying serial number and a manufacturing completion date. Each instrument represents exactly one piano model, all of which have an identification number and a name. In addition, Ebony and Ivory wants to maintain information about the designer of the model. Over time, the company often manufactures thousands of pianos of a certain model, and the model design is specified before any single piano exists.

f. Ebony and Ivory (see 2e) employs piano technicians who are responsible for inspecting the instruments before they are shipped to the customers. Each piano is inspected by at least two technicians (identified by their employee number). For each separate inspection, the company needs to record its date and a quality evaluation grade.

g. The piano technicians (see 2f) have a hierarchy of reporting relationships: Some of them have supervisory responsibilities in addition to their inspection role and have multiple other technicians report to them. The supervisors themselves report to the chief technician of the company.

h. Chiclets Electronics (CE) builds multiple types of tablet computers. Each has a type identification number and a name. The key specifications for each type include amount of storage space and display type. The company uses multiple processor types, exactly one of which is used for a specific tablet computer type; obviously, the same processor can be used in multiple types of tablets. Each processor has a manufacturer and a manufacturer's unique code that identifies it.

i. Each individual tablet computer manufactured by CE (see 2h) is identified by the type identification number and a serial number that is unique within the type identification. CE wants to maintain information about when each tablet is shipped to a customer.

j. Each of the tablet computer types (see 2h) has a specific operating system. Each technician the company employs is certified to assemble a specific tablet type–operating system combination. The validity of a certification starts on the day the employee passes a certification examination for the combination, and the certification is valid for a specific period of time that varies depending on tablet type–operating system combination.

2-25. Answer the following questions concerning Figure 2-22:

a. Where is a unary relationship, what does it mean, and for what reasons might the cardinalities on it be different in other organizations?

b. Why is Includes a one-to-many relationship, and why might this ever be different in some other organization?

c. Does Includes allow for a product to be represented in the database before it is assigned to a product line (e.g., while the product is in research and development)?

d. If there is a rating of the competency for each skill an employee possesses, where in the data model would we place this rating?

e. What is the meaning of the DOES BUSINESS IN associative entity, and why does each DOES BUSINESS IN instance have to be associated with exactly one TERRITORY and one CUSTOMER?

f. In what way might Pine Valley change the way it does business that would cause the Supplies associative entity to be eliminated and the relationships around it to change?

2-26. There is a bulleted list associated with Figure 2-22 that describes the entities and their relationships in Pine Valley Furniture. For each of the 10 points in the list, identify the subset of Figure 2-22 described by that point.

2-27. You may have been assigned a CASE or a drawing tool to develop conceptual data models. Using this tool, attempt to redraw all the E-R diagrams in this chapter. What difficulties did you encounter? What E-R notations did not translate well to your tool? How did you incorporate the E-R notation that did not directly translate into the tool's notation?

2-28. Consider the two E-R diagrams in Figure 2-25, which represent a database of community service agencies and volunteers in two different cities (A and B). For each of the following three questions, place a check mark under City A, City B, or Can't Tell for the choice that is the best answer.

	City A	City B	Can't Tell
a. Which city maintains data about only those volunteers who currently assist agencies?			
b. In which city would it be possible for a volunteer to assist more than one agency?			
c. In which city would it be possible for a volunteer to change which agency or agencies he or she assists?			

2-29. Draw an E-R diagram for the following situation: ShinyShoesForAll (SSFA) is a small shoe repair shop located in a suburban town in the Boston area. SSFA repairs shoes, bags, wallets, luggage, and other similar items. Its customers are individuals and small businesses. The store wants to track the categories to which a customer belongs. SSFA also needs each customer's name and phone number. A job at SSFA is initiated when a customer brings an item or a set of items to be repaired to the shop. At that time, an SSFA employee evaluates the condition

MILLENNIUM COLLEGE
GRADE REPORT
FALL SEMESTER 2018

NAME: Emily Williams ID: 268300458
CAMPUS ADDRESS: 208 Brooks Hall
MAJOR: Information Systems

COURSE ID	TITLE	INSTRUCTOR NAME	INSTRUCTOR LOCATION	GRADE
IS 350	Database Mgt.	Codd	B104	A
IS 465	System Analysis	Parsons	B317	B

FIGURE 2-26 Grade report

of the items to be repaired and gives a separate estimate of the repair cost for each item. The employee also estimates the completion date for the entire job. Each of the items to be repaired will be classified into one of many item types (such as shoes, luggage, etc.); it should be possible and easy to create new item types even before any item is assigned to a type and to remember previous item types when no item in the database is currently of that type. At the time when a repair job is completed, the system should allow the completion date to be recorded as well as the date when the order is picked up. If a customer has comments regarding the job, it should be possible to capture them in the system.

2-30. Are associative entities also weak entities? Why or why not? If yes, is there anything special about their "weakness"?

2-31. Because Visio does not explicitly show associative entities, it is not clear in Figure 2-22 which entity types are associative. List the associative entities in this figure. Why are there so many associative entities in Figure 2-22?

2-32. Figure 2-26 shows a grade report that is mailed to students at the end of each semester. Prepare an ERD reflecting the data contained in the grade report. Assume that each course is taught by one instructor. Also, draw this data model using the tool you have been told to use in the course. Explain what you chose for the identifier of each entity type on your ERD.

2-33. Add minimum and maximum cardinality notation to each of the following figures, as appropriate:

a. Figure 2-5
b. Figure 2-10a
c. Figure 2-12 (all parts)
d. Figure 2-13c
e. Figure 2-14

2-34. The Is Married To relationship in Figure 2-12a would seem to have an obvious answer in Problem and Exercise 2-33d—that is, until time plays a role in modeling data. Draw a data model for the PERSON entity type and the Is Married To relationship for each of the following variations by showing the appropriate cardinalities and including, if necessary, any attributes:

a. All we need to know is who a person is currently married to, if anyone. (This is likely what you represented in your answer to Problem and Exercise 2-33d.)
b. We need to know who a person has ever been married to, if anyone.
c. We need to know who a person has ever been married to, if anyone, as well as the date of their marriage and the date, if any, of the dissolution of their marriage.
d. The same situation as in c, but now assume (which you likely did not do in c) that the same two people can remarry each other after a dissolution of a prior marriage to each other.
e. In history, and even in some cultures today, there may be no legal restriction on the number of people to whom one can be currently married. Does your answer to part c of this Problem and Exercise handle this situation or must you make some changes (if so, draw a new ERD).

2-35. Figure 2-27 represents a situation of students who attend and work in schools and who also belong to certain clubs that are located in different schools. Study this diagram carefully to try to discern what business rules are represented.

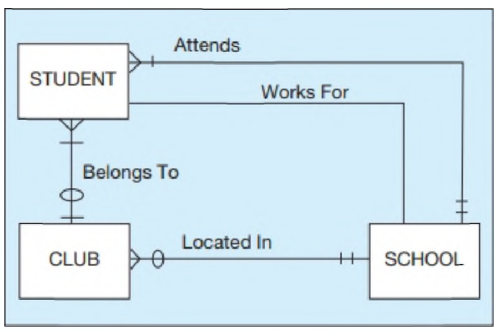
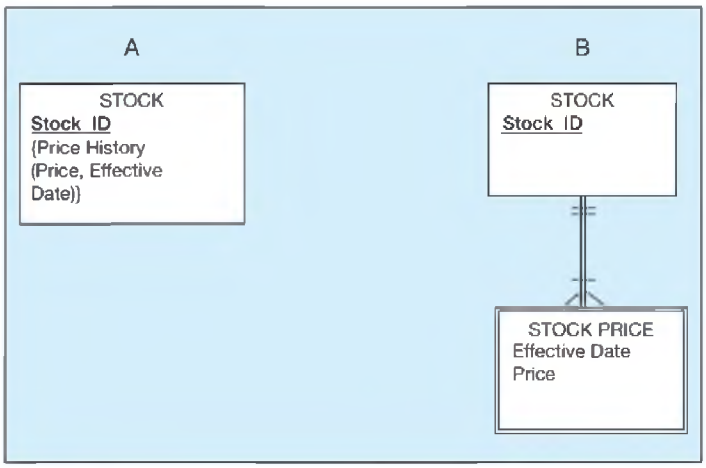


FIGURE 2-27 E-R diagram for Problem and Exercise 2-35

a. You will notice that cardinalities are not included on the Works For relationship. State a business rule for this relationship, and then represent this rule with the cardinalities that match your rule.
b. State a business rule that would make the Located In relationship redundant (i.e., where the school in which a club is located can be surmised or derived in some way from other relationships).
c. Suppose a student could work for only a school the student attends but might not work. Would the Works For relationship still be necessary, or could you represent whether a student works for the school she attends in some other way (if so, how)?
d. Based on the rules of the ERD in Figure 2-27, is it possible for a student to belong to a club located in a school that the student does not attend? Explain.

FIGURE 2-28 E-R diagram for Problem and Exercise 2-36



e. Is it possible to determine which schools do not have any of their students belonging to any club in that school? Explain.

2-36. Figure 2-28 shows two diagrams (A and B), both of which are legitimate ways to represent that a stock has a history of many prices. Which of the two diagrams do you consider a better way to model this situation and why?

2-37. Modify Figure 2-11b to model the following additional information requirements: The training director decides for each employee who completes each class, what course, if any, that employee should take next. The training director needs to keep track of a suggested date by when the employee should take this follow-on course. This date is the only attribute recorded about this suggestion.

2-38. Review Figure 2-8 and Figure 2-22.

a. Identify any attributes in Figure 2-22 that might be composite attributes but are not shown that way. Justify your suggestions. Redraw the ERD to reflect any changes you suggest.

b. Identify any attributes in Figure 2-22 that might be multivalued attributes but are not shown that way. Justify your suggestions. Redraw the ERD to reflect any changes you suggest.

c. Is it possible for the same attribute to be both composite and multivalued? If no, justify your answer; if yes, give an example. (Hint: Consider the CUSTOMER attributes in Figure 2-22.)

2-39. Draw an ERD for each of the following situations. (If you believe that you need to make additional assumptions, clearly state them for each situation.) Draw the same situation using the tool you have been told to use in the course.

a. A company has a number of employees. The attributes of EMPLOYEE include Employee ID (identifier), Name, Address, and Birthdate. The company also has several projects. Attributes of PROJECT include Project ID (identifier), Project Name, and Start Date. Each employee may be assigned to one or more projects or may not be assigned to a project. A project must have at least one employee assigned and may have any number of employees assigned. An employee's billing rate may vary by project, and the company wishes to record the applicable billing rate (Billing Rate) for each employee when assigned to a particular project. Do the attribute names in this description follow the guidelines for naming attributes? If not, suggest better names. Do you have any associative entities on your ERD? If so, what are the identifiers for those associative entities? Does your ERD allow a project to be created before it has any employees assigned to it? Explain. How would you change your ERD if the Billing Rate could change in the middle of a project?

b. A laboratory has several chemists who work on one or more projects. Chemists also may use certain chemicals on each project. Attributes of CHEMIST include Employee ID (identifier), Name, and Phone No. Attributes of PROJECT include Project ID (identifier) and Start Date. Attributes of CHEMICAL include Compound No and Cost. The organization wishes to record Volume Used—that is, the amount of a given chemical used by a particular chemist working on a specified project. A chemist must be assigned to at least one project and one chemical on each project to which he or she is assigned. A given chemical need not be assigned to any project, and a given project need not be assigned to either a chemist or a chemical. Provide good definitions for all of the relationships in this situation.

c. A college course may have one or more scheduled sections or may not have a scheduled section. Attributes of COURSE include Course ID, Course Name, and Units. Attributes of SECTION include Section Number and Semester ID. Semester ID is composed of two parts: Semester and Year. Section Number is an integer (such as 1 or 2) that distinguishes one section from another for the same course but does not uniquely identify a section. How did you model SECTION? Why did you choose this way versus alternative ways to model SECTION?

d. A hospital has a large number of registered physicians. Attributes of PHYSICIAN include Physician ID (the identifier) and Specialty. Patients are admitted to the hospital by physicians. Attributes of PATIENT include Patient ID (the identifier) and Patient Name. Any patient who is admitted must have exactly one admitting physician. A physician may optionally admit any number of patients. Once admitted, a given patient must be treated by at least one physician. A particular physician may treat any number of patients or may not treat any patients. Whenever a patient is treated by a physician, the hospital wishes to record the details of the treatment (Treatment Detail). Components of Treatment Detail include Date, Time, and Results. Did you

draw more than one relationship between physician and patient? Why or why not? Did you include hospital as an entity type? Why or why not? Does your ERD allow for the same patient to be admitted by different physicians over time? How would you include on the ERD the need to represent the date on which a patient is admitted for each time he or she is admitted?

e. The loan office in a bank receives from various parties requests to investigate the credit status of a customer. Each credit request is identified by a Request ID and is described by a Request Date and Requesting Party Name. The loan office also received results of credit checks. A credit check is identified by a Credit Check ID and is described by the Credit Check Date and the Credit Rating. The loan office matches credit requests with credit check results. A credit request may be recorded before its result arrives; a particular credit result may be used in support of several credit requests. Draw an ERD for this situation. Now, assume that credit results may not be reused for multiple credit requests. Redraw the ERD for this new situation using two entity types, and then redraw it again using one entity type. Which of these two versions do you prefer, and why?

f. Companies, identified by Company ID and described by Company Name and Industry Type, hire consultants, identified by Consultant ID and described by Consultant Name and Consultant Specialty, which is multivalued. Assume that a consultant can work for only one company at a time, and we need to track only current consulting engagements. Draw an ERD for this situation. Now, consider a new attribute, Hourly Rate, which is the rate a consultant charges a company for each hour of his or her services. Redraw the ERD to include this new attribute. Now, consider that each time a consultant works for a company, a contract is written describing the terms for this consulting engagement. Contract is identified by a composite identifier of Company ID, Consultant ID, and Contract Date. Assuming that a consultant can still work for only one company at a time, redraw the ERD for this new situation. Did you move any attributes to different entity types in this latest situation? As a final situation, now consider that although a consultant can work for only one company at a time, we now need to keep the complete history of all consulting engagements for each consultant and company. Draw an ERD for this final situation. Explain why these different changes to the situation led to different data models, if they did.

g. A parking garage in downtown Baltimore offers its services to both monthly customers, who pay a fixed fee every month, and to visitors, who pay an hourly fee (assume that the hourly fee is the same regardless of the day or time of day). Each monthly customer gets an ID assigned by the garage, and the garage wants to maintain the customer's basic contact information. The monthly fee paid is negotiated separately for each customer, and it changes periodically; it is important for the garage to maintain a history of the fee details for each customer. The garage has more than 700 parking spots, each of which is equipped with a sensor that recognizes whether there is a car in the spot; in addition, the sensor can read each monthly customer's customer ID card with an RFID reader and, thus, knows which monthly visitor has parked at which spot and when. Each spot is also equipped with a camera that is able to take a picture of each visitor's license plate number; this information is stored to help locate any vehicle that an owner has misplaced. The system should keep track of each time a parking spot is used, including an image of the license plate, start and end times, and a link to the monthly customer, if appropriate.

h. Each case handled by the law firm of Dewey, Cheetim, and Howe has a unique case number; a date opened, date closed, and judgment description are also kept on each case. A case is brought by one or more plaintiffs, and the same plaintiff may be involved in many cases. Each plaintiff in each case has a requested judgment characteristic, such as a requested dollar award, possession of some asset, or other outcome. A case is against one or more defendants, and the same defendant may be involved in many cases. A plaintiff or defendant may be a person or an organization. Over time, the same person or organization may be a defendant or a plaintiff in cases. In either situation, such legal entities are identified by an entity number, and other attributes are name and net worth. As you develop the ERD for this problem, follow good data naming guidelines.

i. A professional society, CAM, organizes hundreds of meetings for its members every year, and it would like to develop an application to support the organizational processes needed to ensure the success of these meetings. Each of the meetings will have five to 500 participants (sometimes even more), and each meeting takes place in a specific hotel in a specific city in a specific state. The organizers need to know each participant's full name, cell phone number, and e-mail address. CAM wants to maintain information regarding dietary restrictions for each participant at the general level, but the restrictions also need to be confirmed and recorded separately for each meeting. For each member's participation for a specific meeting, the organizers need to know how he or she will travel to the meeting and when he or she is planning to arrive and leave. It is important that CAM can produce an accurate report on how many meetings have been organized in a specific city or a specific state.

2-40. Star Hoist is owned by Darth and his wife Ella Vader. The company has had its ups and downs since Darth and Ella built it from the ground up several years ago. The company had some initial difficulties when Darth's brother, Tacksi, was their accountant and got in trouble with the IRS. Finally, the company is doing well, and the owners are ready to expand the business to new heights. Star Hoist sells and installs replacement parts for lifts and similar equipment from a variety of manufacturers. Business can be very competitive, especially from the original manufacturers, which directly sell replacement parts and service to end customers. Darth and Ella need every aspect of their business to work smoothly so that they don't get the shaft in deals with customers. Darth and Ella try to encourage their employees to do the best they can for each customer, which is symbolized by the company motto: "Oh, be the one." There are many rogue competitors, so accurate service is also key for Star Hoist.

You are to draw an ERD for Star Hoist. The fundamental need for Star Hoist is a computer database to keep track of their in-house inventory and of installed parts. Because the business offers negotiated warranties with customers, all parts installed at customer sites need to be tracked. Each part instance is identified by a number assigned by Ella, but because a particular part might come from the original manufacturer or an alternative supplier, the database must record the source of each part and its supplier's part number. In general, a part has a description and standard prices that Star Hoist charges a customer for the part and its installation. Each instance of the part has a cost to Star, based on what the supplier actually charged when Star Hoist acquired that part (many parts, due to their materials, have frequent price changes). Each particular part instance must be tracked, whether it is in inventory or sold to a particular customer. When a part is sold to a customer, there is a negotiated warranty end date, until which time Star assumes all replacement costs for the part, and an actual selling and installation price. Customers have a name, account number, contact person name, and a code that specifies special terms that have been negotiated with each customer. Each supplier has a name, Star's account number with that supplier, and the phone number for the supplier. Each supplier can supply only certain parts. Because many parts can be sourced from multiple suppliers, each part in inventory or installed at a customer must be associated with its source supplier; in addition, Star also needs to know which suppliers can supply which parts. Because many of the parts are very expensive, Darth has placed a limit on how many part instances of a given part can be held in the company's inventory. The limit is three part instances to be held in inventory. As Darth tells the customers, "May the fourth be with you."

2-41. Emerging Electric wishes to create a database with the following entities and attributes:

- Customer, with attributes Customer ID, Name, Address (Street, City, State, Zip Code), and Telephone.
- Location, with attributes Location ID, Address (Street, City, State, Zip Code), and Type (values of Business or Residential).
- Rate, with attributes Rate Class and RatePerKWH.

After interviews with the owners, you have come up with the following business rules:

- Customers can have one or more locations.
- Each location can have one or more rates, depending on the time of day.

Draw an ERD for this situation and place minimum and maximum cardinalities on the diagram. Also, draw a data model for this situation using the tool you have been told to use in your course. State any assumptions that you have made.

2-42. Each semester, each student must be assigned an adviser who counsels students about degree requirements and helps students register for classes. Each student must register for classes with the help of an adviser, but if the student's assigned adviser is not available, the student may register with any adviser. We must keep track of students, the assigned adviser for each, and the name of the adviser with whom the student registered for the current term. Represent this situation of students and advisers with an E-R diagram. Also, draw a data model for this situation using the tool you have been told to use in your course.

2-43. In the chapter, when describing Figure 2-4a, it was argued that the Received and Summarizes relationships and TREASURER entity were not necessary. Within the context of this explanation, this is true. Now, consider a slightly different situation. Suppose it is necessary, for compliance purposes (e.g., Sarbanes-Oxley compliance), to know when each expense report was produced and which officers (not just the treasurer) received each expense report and when each signed off on that report. Redraw Figure 2-4a, now including any attributes and relationships required for this revised situation.

2-44. Virtual Campus (VC) is a social media firm that specializes in creating virtual meeting places for students, faculty, staff, and others associated with different college campuses. VC was started as a student project in a database class at Cyber University, an online polytechnic college, with headquarters in a research park in Dayton, Ohio. The following parts of this exercise relate to different phases in the development of the database VC now provides to client institutions to support a threaded discussion application. Your assignment is to draw an ERD to represent each phase of the development of the VC database and to answer questions that clients raised about the capabilities (business rules) of the database in each phase. The description of each phase will state specific requirements as seen by clients, but other requirements may be implied or possibly should be implemented in the design slightly differently than the clients might see them, so be careful to not limit yourself to only the specifics provided.

a. The first phase was fairly simplistic. Draw an ERD to represent this initial phase, described by the following:
   - A client may maintain several social media sites (e.g., for intercollegiate sports, academics, local food and beverage outlets, or a specific student organization). Each site has attributes of Site Identifier, Site Name, Site Purpose, Site Administrator, and Site Creation Date.
   - Any person may become a participant in any public site. Persons need to register with the client's social media presence to participate in any site, and when they do the person is assigned a Person Identifier; the person provides his or her Nickname and Status (e.g., student, faculty, staff, or friend, or possibly several such values); the Date Joined the site is automatically generated. A person may also include other information, which is available to other persons on the site; this information includes Name, Twitter Handle, Facebook Page link, and SMS Contact Number. Anyone may register (no official association with the client is necessary).
   - An account is created each time a person registers to use a particular site. An account is described by an Account ID, User Name, Password, Date Created, Date Terminated, and Date/Time the person most recently used that account.
   - Using an account, a person creates a posting, or message, for others to read. A posting has a Posting Date/Time and Content. The person posting the message may also add a Date when the posting should be made invisible to other users.

- A person is permitted to have multiple accounts, each of which is for only one site.
- A person, over time, may create multiple postings from an account.

b. After the first phase, a representative from one of the initial clients asked if it were possible for a person to have multiple accounts on the same site. Answer this question based on your ERD from part a of this exercise. If your answer is yes, could you enforce via the ERD a business rule of only one account per site per person, or would other than a data modeling requirement be necessary? If your answer is no, justify how your ERD enforces this rule.

c. The database for the first phase certainly provided only the basics. VC quickly determined that two additional features needed to be added to the database design, as follows (draw a revised ERD to represent the expanded second phase database):
   - From their accounts, persons might respond to postings with an additional posting. Thus, postings may form threads, or networks of response postings, which then may have other response postings and so forth.
   - It also became important to track not only postings but also when persons from their accounts read a posting. This requirement is needed to produce site usage reports concerning when postings are made, when they are read and by whom, frequency of reading, etc.

d. Clients liked the improvements to the social media application supported by the database from the second phase. How useful the social media application is depends, in part, on questions administrators at a client organization might be able to answer from inquiries against the database using reports or online queries. For each of the example client inquiries that follow, justify for your answer to part c whether your database could provide answers to that inquiry (if you already know SQL, you could provide justification by showing the appropriate SQL query; otherwise, explain the entities, attributes, and relationships from your ERD in part c that would be necessary to produce the desired result):
   - How many postings has each person created for each site?
   - Which postings appear under multiple sites?
   - Has any person created a posting and then responded to his or her own posting before any other person has read the original posting?
   - Which sites, if any, have no associated postings?

e. The third phase of database development by VC dealt with one of the hazards of social media sites—irresponsible, objectionable, or harmful postings (e.g., bullying or inappropriate language). So for the third phase, draw a revised ERD to the ERD you drew for the second phase to represent the following:
   - Any person from one of their accounts may file a complaint about any posting. Most postings, of course, are legitimate and not offensive, but some postings generate lots of complaints. Each complaint has a Complaint ID, Date/Time the complaint is posted, the Content of the complaint, and a Resolution Code. Complaints and the status of resolution are visible to only the person making the complaint and to the site administrator.
   - The administrator for the site about which a complaint has been submitted (not necessarily a person in the database, and each site may have a different administrator) reviews complaints. If a complaint is worthy, the associated offensive posting is marked as removed from the site; however, the posting stays in the database so that special reports can be produced to summarize complaints in various ways, such as by person, so that persons who make repeated objectionable postings can be dealt with. In any case, the site administrator after his or her review fills in the date of resolution and the Resolution Code value for the complaint. As stated, only the site administrator and the complaining person, not other persons with accounts on the site, see complaints for postings on the associated site. Postings marked as removed as well as responses to these postings are then no longer seen by the other persons.

f. You may see various additional capabilities for the VC database. However, in the final phase you will consider in this exercise, you are to create an expansion of the ERD you drew for phase three to handle the following:
   - Not all sites are public, that is, open for anyone to create an account. A person may create one or more sites as well as groups and then invite other persons in a group to be part of a site he or she has created. A group has a Group ID, Group Name, Date Created, Date Terminated, Purpose, and Number of Members.
   - The person creating a "private" site is then, by default, the site administrator for that site.
   - Only the members of a group associated with a private site may then create accounts for that site, post to that site, and perform any other activities for that site.

2-45. After completing a course in database management, you are asked to develop a preliminary ERD for a symphony orchestra. You discover the entity types that should be included as shown in Table 2-3.

During further discussions you discover the following:
- A concert season schedules one or more concerts. A particular concert is scheduled for only one concert season.
- A concert includes the performance of one or more compositions. A composition may be performed at one or more concerts or may not be performed.
- For each concert there is one conductor. A conductor may conduct any number of concerts or may not conduct any concerts.
- Each composition may require one or more soloists or may not require a soloist. A soloist may perform one or more compositions at a given concert or may not perform any composition. The symphony orchestra wishes to record the date when a soloist last performed a given composition (Date Last Performed).

Draw an ERD to represent what you have discovered. Identify a business rule in this description and explain how this business rule is modeled on the E-R diagram. Also draw a data model for this situation using the tool you have been told to use in your course.

TABLE 2-3 Entity Types for Problem and Exercise 2-45

CONCERT SEASON	The season during which a series of concerts will be performed. Identifier is Opening Date, which includes Month, Day, and Year.
CONCERT	A given performance of one or more compositions. Identifier is Concert Number. Another important attribute is Concert Date, which consists of the following: Month, Day, Year, and Time. Each concert typically has more than one concert date.
COMPOSITION	Compositions to be performed at each concert. Identifier is Composition ID, which consists of the following: Composer Name and Composition Name. Another attribute is Movement ID, which consists of two parts: Movement Number and Movement Name. Many, but not all, compositions have multiple movements.
CONDUCTOR	Person who will conduct the concert. Identifier is Conductor ID. Another attribute is Conductor Name.
SOLOIST	Solo artist who performs a given composition on a particular concert. Identifier is Soloist ID. Another attribute is Soloist Name.

2-46. Draw an ERD for the following situation, which is based on Lapowsky (2016): The Miami-Dade County, Florida, court system believes that jail populations can be reduced, reincarceration rates lowered, and court system costs lessened and, most important, that better outcomes can occur for people in and potentially in the court system if there is a database that coordinates activities for county jails, metal health facilities, shelters, and hospitals. Based on the contents of this database, algorithms can be used to predict what kind of help a person might need to reduce his or her involvement in the justice system. Eventually, such a database could be extensive (involving many agencies and lots of personal history and demographic data) once privacy issues are resolved. However, for now, the desire is to create a prototype database with the following data. Data about persons will be stored in the database, including professionals who work for the various participating agencies as well as those who have contact with an agency (e.g., someone who is a client of a mental health facility, who is incarcerated, or both). Data about people include name, birth date, education level, job title (if the person is an employee of one of the participating agencies), and (permanent) address. Some people in the system will have been prescribed certain medicines while in the care of county hospitals and mental health facilities. A medicine has a name and a manufacturer. Each prescription is for a particular medicine and has a dosage. A prescription is due to some diagnosis, which was identified on a certain date, to treat some illness, was diagnosed by some facility professional, and has notes explaining family history at the time of the diagnosis. Each illness has a name and some medicines or other treatments commonly prescribed (e.g., certain type of counseling). Each participating agency is of a certain type (e.g., criminal justice, mental health) and has a name and a contact person. People visit or contact an agency (e.g., they are arrested by the justice system or stay at a shelter). For each contact a person has with an agency, the database needs to record the contact date, employment status at time of contact, address at time of contact, reason for visit/contact, and the name of the responsible agency employee.

2-47. Draw an ERD diagram for the following situation: The Sensing Building Company (SBC) installs wireless microsensors throughout buildings and building campuses to give building managers, maintenance personnel, and others real-time data about the status of almost any part of a building. Sensors can be placed on doors, trash cans, plumbing fixtures, windows, lighting fixtures, and heating systems—almost any building element. Sensor data are used to create dashboards to indicate when, for example, a plumber needs to be dispatched to fix a leaking pipe in a particular wall of an identified building. In addition, data are analyzed over time to determine, for example, where and when electricity and heating/cooling are used so that measures can be taken to reduce energy consumption costs. All the collected data must be kept in a database, although some data are used to trigger alerts when immediate action must be taken in response to a security or safety issue. Each sensor has various features, depending on its purpose and location. For example, a sensor on a trash can is designed to periodically transmit how full the container is and to immediately send a message when the can is within 5 percent of being full or when the can is no longer upright. In general, each sensor sends periodic as well as critical event messages, the latter of which may cause an alert and immediate action to be taken. The following is a somewhat simplified description of the database requirements. Data must be kept on each sensor, sensor transmission, building personnel, building, location within or outside a building, alert, and action taken. A sensor has a unique 12 character ID, a title, type, date installed, frequency of transmission, and location. Each sensor transmission includes the sensor ID, time stamp, and one or more readings. Personnel have an ID, name, job title, a set of skills, and a set of locations for which he or she is responsible. Each building has a number, description, and a senior person responsible for the building. Each location has an ID, type of location, and coordinates of where within a building or the campus it is. An alert has an ID, the ID of the sensor transmission(s) that generated the alert, and a time stamp for when the alert occurred. Finally, an action has an ID, the ID of the alert that caused the action, the individual or several personnel taking the action, and the result of the action.

2-48. Draw an ERD for the following situation. (State any assumptions you believe you have to make in order to develop a complete diagram.) Also, draw a data model for this situation using the tool you have been told to use in your course: Stillwater Antiques buys and sells one-of-a-kind antiques of all kinds (e.g., furniture, jewelry, china, and clothing). Each item is uniquely identified by an item number and is also characterized by a description, asking price, condition, and open-ended comments. Stillwater works with many different individuals, called clients, who sell items to and buy items from the store. Some clients only sell items to Stillwater, some only buy items, and some others both sell and buy. A client is identified by a client number and is also described by a client name and client address. When Stillwater sells an item in stock to a client, the owners want to record the commission paid, the actual selling price, sales tax (tax of zero indicates a tax exempt sale), and date sold. When Stillwater buys an item from a client, the owners want to record the purchase cost, date purchased, and condition at time of purchase.

2-49. Draw an ERD for the following situation. (State any assumptions you believe you have to make in order to develop a complete diagram.) Also, draw a data model for this situation using the tool you have been told to use in your course: The A. M. Honka School of Business operates international business programs in 10 locations throughout Europe. The school had its first class of 9,000 graduates in 1965. The school keeps track of each graduate's student number, name when a student, country of birth, current country of citizenship, current name, and current home address and current business address, as well as the name of each major the student completed. (Each student has one or two majors.) To maintain strong ties to its alumni, the school distributes various communications, including publications and messages. Each communication has a title, date, and medium (e.g., printed magazine, electronic newsletter, invitation). The school needs to keep track of which graduates have received which communications. For each communication with a graduate, a comment may be recorded with any feedback the school received from the graduate. When a school official knows that he or she will be meeting or talking to a graduate, a report is produced showing the latest information about that graduate and the information learned during the past two years of comments from that graduate.

2-50. Wally Los Gatos, owner of Wally's Wonderful World of Wallcoverings, Etc., has hired you as a consultant to design a database management system for his new online marketplace for wallpaper, draperies, and home decorating accessories. He would like to track sales, prospective sales, and customers. Ultimately, he'd like to become the leading online retailer for all things related to home decorating. During an initial meeting with Wally, you and Wally developed a list of business requirements to begin the design of an E-R model for the database to support his business needs.

a. Wally was called away unexpectedly after only a short discussion with you, due to a sticky situation with the pre-pasted line of wall coverings he sells. He gave you only a brief description of his needs and asked that you fill in details for what you expect he might need for these requirements. Wally expected to be away for only a short time, so he asked that you go ahead with some first suggestions for the database; but he said, "Keep it basic for now, we'll do the faux finishes later." Before Wally left, he requested the following features for his system:
   - At a basic level, Wally needs to track his customers (both those who have bought and those Wally has identified as prospective buyers based on his prior brick-and-mortar business outlets), the products he sells, and the products they have bought.
   - Wally wants a variety of demographic data about his customers so he can better understand who is buying his products. He'd like a few suggestions from you on appropriate demographic data, but he definitely wants to know customer interests, hobbies, and activities that might help him proactively suggest products customers might like.

b. True to his word, Wally soon returned, but said he could only step into the room for a short time because the new Tesla he had ordered had been delivered, and he wanted to take it for a test drive. But before he and his friend Elon left, he had a few questions that he wanted the database to allow him to answer, including the following (you can answer Wally with an SQL query that would produce the result because Wally is proficient in SQL or by explaining the entities, attributes, and relationships that would allow the questions to be answered):
   - Would the database be able to tell him which other customers had bought the same product a given customer or prospective customer had bought or was considering buying?
   - Would the database be able to tell him even something deeper, that is, what other products other customers bought who also bought the product the customer just bought (i.e., an opportunity for cross-selling)?
   - Would he be able to find other customers with at least three interests that overlap with those of a given customer so that he can suggest to these other customers other products they might want to purchase?

   Prepare queries or explanations to demonstrate for Wally why your database design in part a of this exercise can support these needs or draw a revised design to support these specific questions.

c. Wally is thrilled with his new Tesla and returns from the test drive eager to expand his business to now pay for this new car. The test drive was so invigorating that it helped him to generate more ideas for the new online shopping site, including the following requirements:
   - Wally wants to be able to suggest products for customers to buy. Wally knows that most of the products he sells have similar alternatives that a customer might want to consider. Similarity is fairly subtle, so he or his staff would have to specify for each product what other products, if any, are similar.
   - Wally also thinks that he can improve sales by reminding customers about products they have previously considered or viewed when on his online marketplace.

   Unfortunately, Wally's administrative assistant, Helen, in her hunt for Wally, knocked on the door and told Wally that his first born child, Julia, had just come in asking to see her father so that she could show him

her new tattoo, introduce him to her new "goth" boyfriend, and let him know about her new life plans. This declaration, obviously, got Wally's attention. Wally left abruptly but asked that you fill in the blanks for these new database requirements.

d. Fortunately, Wally's assistant was just kidding, and the staff had actually thrown a surprise birthday party for Wally. They needed Wally in the staff dining room quickly before the ice cream melted and the festive draperies adorning the table of presents had to be returned to the warehouse for shipment to Tokyo for display at the summer Olympics. Now overjoyed by the warm reception from his trusted associates, Wally was even more enthusiastic about making his company successful. Wally came in with two additional requirements for this database:
   - Wally had learned the hard way that in today's world, some of his customers have multiple homes or properties for which they order his products. Thus, different orders for the same customer may go to different addresses, but several orders for the same customer often go to the same address.
   - Customers also like to see what other people think about products they are considering to buy. So, the database needs to be able to allow customers to rate and review products they buy and for other customers considering purchasing a product to see the reviews and ratings from those who have already purchased the product being considered. Wally also wants to know what reviews customers have viewed so that he can tell which reviews might be influencing purchases.

   Yet again, Wally has to leave the meeting, this time because it is time for his weekly pickle ball game, and he doesn't want to brush off his partner in the ladder tournament, which he and partner now lead. He asks that you go ahead and work on adding these requirements to the database, and he'll be back after he and his partner hang their new trophy in Wally's den.

e. Although still a little sweaty and not in his normal dapper business attire, Wally triumphantly hobbled back to the meeting room. Wally's thigh was wrapped in what seemed to be a whole reel of painter's tape (because he didn't have any sports tape), nursing his agony of victory. Before limping off to his doctor, Wally, ever engaged in his business, wanted to make sure your database design could handle the following needs:
   - One of the affinities people have for buying is what other people in their same geographical area are buying (a kind of "keep up with the Jones" phenomenon). Justify to Wally why your database design can support this requirement or suggest how the design can be changed to meet this need.
   - Customers want to search for possible products based on categories and characteristics, such as paint brushes, lamps, bronze color, etc.
   - Customers want to have choices for the sequence in which products are shown to them, such as by rating, popularity, and price.

   Justify to Wally why your database design can support these needs or redesign your database to support these additional requirements.

2-51. Doctors Information Technology (DocIT) is an IT services company supporting medical practices with a variety of computer technologies to make medical offices more efficient and less costly to run. Medical offices are rapidly becoming automated with electronic medical records, automated insurance claims processing and prescription submissions, patient billing, and other typical aspects of medical practices. In this assignment, you will address only insurance claims processing; however, what you develop must be able to be generalized and expanded to these other areas of a medical practice. Your assignment is to draw an ERD to represent each phase of the development of an insurance claims processing database and to answer questions that clients might raise about the capabilities of the application the database supports in each phase.

a. The first phase deals with a few core elements. Draw an ERD to represent this initial phase, described by the following:
   - A patient is assigned a patient ID, and you need to keep track of a patient's gender, date of birth, name, current address, and list of allergies.
   - A staff member (doctor, nurse, physician's assistant, etc.) has a staff ID, job title, gender, name, address, and list of degrees or qualifications.
   - A patient may be included in the database even if no staff member has ever seen the patient (e.g., family member of another patient or a transfer from another medical practice). Similarly, some staff members never have a patient contact that requires a claim to be processed (e.g., a receptionist greeting a patient does not generate a claim).
   - A patient sees a staff member via an appointment. An appointment has an appointment ID, a date and time of when the appointment is scheduled or when it occurred as well as a date and time when the appointment was made, and a list of reasons for the appointment.

b. As was noted in part a of this exercise the first phase, information about multiple members of the same family may need to be stored in the database because they are all patients. Actually, there is a broader need. A medical practice may need to recognize various people related to a particular patient (e.g., spouse, child, caregiver, power of attorney, an administrator at a nursing home, etc.) who can see patient information and make emergency medical decisions on behalf of the patient. Augment your answer to part a of this exercise to represent the relationships between people in the database and the nature of any relationships.

c. In the next phase, you will extend the database design to begin to handle insurance claims. Draw a revised ERD to your answer to part b of this exercise to represent the expanded second phase database:
   - Each appointment may generate several insurance claims (some patients are self-pay, with no insurance coverage). Each claim is for a specific action taken in the medical practice, such as seeing a staff member, performing a test, administering a specific treatment, etc. Each claim has an ID, a claim code (taken from a list of standard codes that all insurance companies recognize), date the action was

done, date the claim was filed, amount claimed, amount paid on the claim, optionally a reason code for not paying full amount, and the date the claim was (partially) paid.
- Each patient may be insured under policies with many insurance companies. Each patient policy has a policy number; possibly a group code; a designation of whether the policy is primary, secondary, tertiary, or whatever in the sequence of processing claims for a given patient; and the type of coverage (e.g., medicines, office visit, outpatient procedure).
- A medical practice deals with many insurance companies because of the policies for their patients. Each company has an ID, name, mailing address, IP address, and company contact person.
- Each claim is filed under exactly one policy with one insurance company. If for some reason a particular action with a patient necessitates more than one insurance company to be involved, then a separate claim is filed with each insurance company (e.g., a patient might reach some reimbursement limit under her primary policy, so a second claim must be filed for the same action with the company associated with the secondary policy).

d. How useful and sufficient a database is depends, in part, on questions it can be used to answer using reports or online queries. For each of the example inquiries that follow, justify for your answer to part c of this exercise whether your database could provide answers to that inquiry (if you already know SQL, you could provide justification by showing the appropriate SQL query; otherwise, explain the entities, attributes, and relationships from your ERD in part c that would be necessary to produce the desired result):
- How many claims are currently fully unreimbursed?
- Which insurance company has the most fully or partially unreimbursed claims?
- What is the total claims amount per staff member?
- Is there a potential conflict of interest in which a staff member is related to a patient for which that staff member has generated a claim?

e. As was stated in previous parts of this exercise, some claims may be only partially paid or even denied by the insurance company. When this occurs, the medical practice may take follow-up steps to resolve the disputed claim, and this can cycle through various negotiation stages. Draw a revised ERD to replace the ERD you drew for part c to represent the following:
- Each disputed claim may be processed through several stages. In each stage, the medical practice needs to know the date processed, the dispute code causing the processing step, the staff person handling the dispute in this stage, the date when this stage ends, and a description of the dispute status at the end of the stage.
- There is no limit to the number of stages a dispute may go through.
- One possible result of a disputed claim processing stage is the submission of a new claim, but usually it is the same original claim that is processed in subsequent stages.

2-52. Review your answer to Problem and Exercise 2-49; if necessary, change the names of the entities, attributes, and relationships to conform to the naming guidelines presented in this chapter. Then, using the definition guidelines, write a definition for each entity, attribute, and relationship. If necessary, state assumptions so that each definition is as complete as possible.

Field Exercises

2-53. Interview a database analyst or systems analyst and document how he or she decides on names for data objects in data models. Does the organization in which this person works have naming guidelines? If so, describe the pattern used. If there are no guidelines, ask whether your contact has ever had any problems because guidelines did not exist. Does the organization use any tool to help manage metadata, including data names?

2-54. Visit two local small businesses, one in the service sector (e.g., dry cleaner, auto repair shop, veterinarian, or bookstore) and one that manufactures tangible goods. Interview employees from these organizations to elicit from them the entities, attributes, and relationships that are commonly encountered in these organizations. Use this information to construct E-R diagrams. What differences and similarities are there between the diagrams for the service- and the product-oriented companies? Does the E-R diagramming technique handle both situations equally well? Why or why not?

2-55. Ask a database or systems analyst to give you examples of unary, binary, and ternary relationships that the analyst has dealt with personally at his or her company. Ask which is most common and why. Ask them if they ever model weak or dependent entities and, if so, what they use for the identifier of such entities.

2-56. Ask a database or systems analyst in a local company to show you an E-R diagram for one of the organization's primary databases. Ask questions to be sure you understand what each entity, attribute, and relationship means. Does this organization use the same E-R notation used, in this text? If not, what other or alternative symbols are used, and what do these symbols mean? Does this organization model associative entities on the E-R diagram? If not, how are associative entities modeled? What metadata are kept about the objects on the E-R diagram?

2-57. For the same E-R diagram used in Field Exercise 2-56 or for a different database in the same or a different organization, identify any uses of time stamping or other means to model time-dependent data. Why are time-dependent data necessary for those who use this database? Would the E-R diagram be much simpler if it were not necessary to represent the history of attribute values?

2-58. Research various graphics and drawing packages (e.g., Microsoft Office, SmartDraw) and compare the E-R diagramming capabilities of each. Is each package capable of using the notation found in this text? Is it possible to draw a ternary or higher-order relationship with each package?

References

Aranow, E. B. 1989. "Developing Good Data Definitions." *Database Programming & Design* 2,8 (August): 36–39.

Bruce, T. A. 1992. *Designing Quality Databases with IDEF1X Information Models*. New York: Dorset House.

Chen, P. P.-S. 1976. "The Entity-Relationship Model—Toward a Unified View of Data." *ACM Transactions on Database Systems* 1,1 (March): 9–36.

Elmasri, R., and S. B. Navathe. 1994. *Fundamentals of Database Systems*. 2d ed. Menlo Park, CA: Benjamin/Cummings.

Embarcadero Technologies. 2014. "Seven Deadly Sins of Database Design: How to Avoid the Worst Problems in Database Design." April. Available at **www.embarcadero.com**.

Gottesdiener, E. 1997. "Business Rules Show Power, Promise." *Application Development Trends* 4,3 (March): 36–54.

Gottesdiener, E. 1999. "Turning Rules into Requirements." *Application Development Trends* 6,7 (July): 37–50.

GUIDE. 1997 (October)."GUIDE Business Rules Project." Final Report, revision 1.2.

Haughey, T. 2010 (March) "The Return on Investment (ROI) of Data Modeling." White paper published by Computer Associates–Erwin Division.

Hay, D. C. 2003. "What Exactly IS a Data Model?" Parts 1, 2, and 3. *DM Review* 13,2 (February: 24–26), 3 (March: 48–50), and 4 (April: 20–22, 46).

Johnson, T. and R. Weis. 2007. "Time and Time Again: Managing Time in Relational Databases, Part 1." May. *DM Review*. This and other related articles by Johnson and Weis on "Time and Time Again" can be found by doing a search on Weis at **www.information-management.com**.

Lapowsky, I. 2016. "The Justice Machine: Law Enforcement and Mental Health Workers Are Getting Help from Algorithms." *Wired* 24,11 (November): 68–70.

Moriarty, T. 2000. "The Right Tool for the Job." *Intelligent Enterprise* 3,9 (June 5): 68, 70–71.

Owen, J. 2004. "Putting Rules Engines to Work." *InfoWorld* (June 28): 35–41.

Plotkin, D. 1999. "Business Rules Everywhere." *Intelligent Enterprise* 2,4 (March 30): 37–44.

Salin, T. 1990. "What's in a Name?" *Database Programming & Design* 3,3 (March): 55–58.

Song, I.-Y., M. Evans, and E. K. Park. 1995. "A Comparative Analysis of Entity-Relationship Diagrams." *Journal of Computer & Software Engineering* 3,4: 427–59.

Storey, V. C. 1991. "Relational Database Design Based on the Entity-Relationship Model." *Data and Knowledge Engineering* 7: 47–83.

Teorey, T. J., D. Yang, and J. P. Fry. 1986. "A Logical Design Methodology for Relational Databases Using the Extended Entity-Relationship Model." *Computing Surveys* 18, 2 (June): 197–221.

Valacich, J. S., and J. F. George. 2016. *Modern Systems Analysis and Design*. 8th ed. Upper Saddle River, NJ: Prentice Hall.

von Halle, B. 1997. "Digging for Business Rules." *Database Programming & Design* 8,11: 11–13.

Further Reading

Batini, C., S. Ceri, and S. B. Navathe. 1992. *Conceptual Database Design: An Entity-Relationship Approach*. Menlo Park, CA: Benjamin/Cummings.

Bodart, F., A. Patel, M. Sim, and R. Weber. 2001. "Should Optional Properties Be Used in Conceptual Modelling? A Theory and Three Empirical Tests." *Information Systems Research* 12,4 (December): 384–405.

Carlis, J., and J. Maguire. 2001. *Mastering Data Modeling: A User-Driven Approach*. Upper Saddle River, NJ: Prentice Hall.

Keuffel, W. 1996. "Battle of the Modeling Techniques." *DBMS* 9,8 (August): 83, 84, 86, 97.

Moody, D. 1996. "The Seven Habits of Highly Effective Data Modelers." *Database Programming & Design* 9,10 (October): 57, 58, 60–62, 64.

Teorey, T. 1999. *Database Modeling & Design*. 3d ed. San Francisco, CA: Morgan Kaufman.

Tillman, G. 1994. "Should You Model Derived Data?" *DBMS* 7,11 (November): 88, 90.

Tillman, G. 1995. "Data Modeling Rules of Thumb." *DBMS* 8,8 (August): 70, 72, 74, 76, 80–82, 87.

Web Resources

www.adtmag.com Web site of *Application Development Trends*, a leading publication on the practice of information systems development.

www.axisboulder.com Web site for one vendor of business rules software.

www.businessrulesgroup.org Web site of the Business Rules Group, formerly part of GUIDE International, which formulates and supports standards about business rules.

http://en.wikipedia.org/wiki/Entity-relationship_model The Wikipedia entry for entity-relationship model, with an explanation of the origins of the crow's foot notation, which is used in this book.

http://ss64.com/ora/syntax-naming.html Web site that suggests naming conventions for entities, attributes, and relationships within an Oracle database environment.

www.tdan.com Web site of *The Data Administration Newsletter*, an online journal that includes articles on a wide variety of data management topics. This Web site is considered a "must follow" Web site for data management professionals.

CASE

Forondo Artist Management Excellence Inc.

Case Description

Martin was very impressed with your project plan and has given you the go-ahead for the project. He also indicates to you that he has e-mails from several key staff members that should help with the design of the system. The first is from Alex Martin (administrative assistant to Pat Smith, an artist manager). Pat is on vacation, and Martin has promised that Pat's perspective will be provided at a later date. The other two are from Dale Dylan, an artist who Pat manages, and Sandy Wallis, an event organizer. The text of these e-mails is provided below.

E-mail from Alex Martin, Administrative Assistant

My name is Alex Martin, and I am the administrative assistant to Pat Smith. While Pat's role is to create and maintain relationships with our clients and the event organizers, I am responsible for running the show at the operational level. I take care of Pat's phone calls while Pat is on the road, respond to inquiries and relay the urgent ones to Pat, write letters to organizers and artists, collect information on prospective artists, send bills to the event organizers and make sure that they pay their bills, take care of the artist accounts, and arrange Pat's travel (and keep track of travel costs). Most of my work I manage with Word and simple Excel spreadsheets, but it would be very useful to be able to have a system that would help me to keep track of the event fees that have been agreed upon, the events that have been successfully completed, cancellations (in the current system, I sometimes don't get information about a cancellation and I end up sending an invoice for a cancelled concert—pretty embarrassing), payments that need to be made to the artists, etc. Pat and other managers seem to think that it would be a good idea if they could better track their travel costs and the impact these costs have on their income.

We don't have a very good system for managing our artist accounts because we have separate spreadsheets for keeping track of a particular artist's fees earned and the expenses incurred, and then at the end of each month we manually create a simple statement for each of the artists. This is a lot of work, and it would make much more sense to have a computer system that would allow us to be able to keep the books constantly up to date.

A big thing for me is to keep track of the artists whom Pat manages. We need to keep in our databases plenty of information on them—their name, gender, address (including country, as they live all over the world), phone number(s), instrument(s), e-mail, etc. We also try to keep track of how they are doing in terms of the reviews they get, and thus we are subscribing to a clipping service that provides us articles on the artists whom we manage. For some of the artists, the amount of material we get is huge, and we would like to reduce it somehow. At any rate, we would at least like to be able to have a better idea of what we have in our archives on a particular artist, and thus we should probably start to maintain some kind of a list of the news items we have for a particular artist. I don't know if this is worth it but it would be very useful if we could get it done.

Scheduling is, of course, a major headache for me. Although Pat and the artists negotiate the final schedules, I do, in practice, at this point maintain a big schedule book for each artist whom we manage. You know, somebody has to have the central copy. This means that Pat, the artists, and the event organizers are calling me all the time to verify the current situation and make changes to the schedule. Sometimes things get mixed up and we don't get the latest changes to the central calendar (for example, an artist schedules a vacation and forgets to tell us—as you can understand, this can lead to a pretty difficult situation). It would be so wonderful to get a centralized calendar which both Pat and the artists could access; it is probably, however, better if Pat (and the other managers for the other artists, of course) was the only person in addition to me who had the right to change the calendar. Hmmm . . . I guess it would be good if the artists could block time out if they decide that they need if for personal purposes (they are not, however, allowed to book any performances without discussing it first with us).

One more thing: I would need to have something that would remind me of the upcoming changes in artist contracts. Every artist's contract has to be renewed annually, and sometimes I forget to remind Pat to do this with the artist. Normally this is not a big deal, but occasionally we have had a situation where the lack of a valid contract led to unfortunate and unnecessary problems. It seems that we would need to maintain some type of list of the contracts with their start dates, end dates, royalty percentages, and simple notes related to each of the contracts.

This is a pretty hectic job, and I have not had time to get as good computer training as I would have wanted. I think I am still doing pretty well. It is very important that whatever you develop for us, it has to be easy to use because we are in such a hurry all the time and we cannot spend much time learning complex commands.

E-mail from Dale Dylan, Established Artist

Hi! I am Dale Dylan, a pianist from Austin, TX. I have achieved reasonable success during my career and I am very thankful that I have been able to work with Pat Smith and Mr. Forondo during the past five years. They have been very good at finding suitable performance opportunities for me, particularly after I won an international piano competition in Amsterdam a few years ago. Compared to some other people with whom I have worked, Pat is very conscientious and works hard for me.

During the recent months, FAME and its managers' client base has grown quite a lot, and unfortunately I have seen this in the service they have been able to provide to me. I know that Pat and Alex don't mean any harm but it seems that they simply have too much to do, particularly in scheduling and getting my fees to me. Sometimes things seem to get lost pretty easily these days, and occasionally I have been waiting for my money for 2–3 months. This was never the case earlier but it has been pretty typical during the last year or so. Please don't say anything to Pat or Alex about this; I don't want to hurt their feelings, but it just simply seems that they have too much to do. Do you think your new system could help them?

What I would like to see in a new system—if you will develop one for them—are just simple facilities that would help them do even better what they have always done pretty well (except very recently): collecting money from the concert organizers and getting it to me fast (they are, after all, taking 20 percent of my money—at least they should get the rest of it to me quickly) and maintaining my schedule. I have either a laptop or at least my smartphone/iPad with me all the time while I am on the road, thus I certainly should be able to check my schedule on the Web. Now I always need to call Alex to get any last-minute changes. It seems pretty silly that Pat has to be in touch with Alex before any changes can be made to the calendar; I feel that I should be allowed to make my own changes. Naturally, I would always notify Pat about anything that changes (or maybe the system could do that for me). The calendar system should be able to give me at least a simple list of the coming events in the chronological order for any time period I want. Furthermore, I would like to be able to search for events using specific criteria (location, type, etc.).

In addition, we do, of course, get annual summaries from FAME regarding the fees we have earned, but it would be nice to have this information a bit more often. I don't need it on paper but if I could access that information on the Web, it would be very, very good. It seems to me that Alex is doing a lot of work with these reports by hand; if you could help her with any of the routine work she is doing, I am sure she would be quite happy. Maybe then she and Pat would have more time for getting everything done as they always did earlier.

E-mail from Sandy Wallis, Event Organizer

I am Sandy Wallis, the executive director of the Greater Tri-State Area Concert Halls, and it has been a pleasure to have a good working relationship with Pat Smith at FAME for many years. Pat has provided me and my annual concert series several excellent artists per year, and I believe that our cooperation has a potential to continue into the foreseeable future. This does, however, require that Pat is able to continue to give me the best service in the industry during the years to come.

Our business is largely based on personal trust, and the most important aspect of our cooperation is that I can know that I can rely on the artists managed by Pat. I am not interested in the technology Pat is using, but it is important for us that practical matters such as billing and scheduling work smoothly and that technology does not prevent us from making decisions fast, if necessary. We don't want to be billed for events that were cancelled and never rescheduled, and we are quite unhappy if we need to spend our time on these types of technicalities.

At times, we need a replacement artist to substitute for a musician who becomes ill or cancels for some other reason, and the faster we can get information about the availability of world-class performers in these situations, the better it is for us. Yes, we work in these situations directly with Pat, but we have seen that occasionally all the information required for fast decision making is not readily available, and this is something that is difficult for us to understand. We would like to be able to assume that Pat's able assistant Alex should be able to give us information regarding the availability of a certain artist on a certain date on the phone without any problems. Couldn't this information be available on the Web, too? Of course, we don't want anybody to know in advance whom we have booked before we announce our annual program; therefore, security is very important for us.

I hope you understand that we run multiple venues but we definitely still want to be treated as one customer. With some agencies we have seen silly problems that have forced them to send us invoices with several different names and customer numbers, which does not make any sense from our perspective and causes practical problems with our systems.

Project Questions

2-59. Redo the enterprise data model you created in Chapter 1 to accommodate the information gleaned from Alex Martin's, Dale Dylan's, and Sandy Wallis's e-mails.

2-60. Create an E-R diagram for FAME based on the enterprise data model you developed in 1-60. Clearly state any assumptions you made in developing the diagram.

2-61. Use the narratives in Chapter 1 and above to identify the typical outputs (reports and displays) the various stakeholders might want to retrieve from your database. Now, revisit the E-R diagram you created in 2-60 to ensure that your model has captured the information necessary to generate the outputs desired. Update your E-R diagram as necessary.

2-62. Prepare a list of questions that you have as a result of your E-R modeling efforts and that need to be answered to clarify your understanding of FAME's business rules and data requirements.