

# DATABASE MANAGEMENT SYSTEMS

## LEARNING OBJECTIVES

AFTER READING THIS CHAPTER, YOU SHOULD BE ABLE TO:

- DESCRIBE THE LIMITATIONS OF TRADITIONAL APPLICATION APPROACHES TO MANAGING DATA.
- ANALYZE THE ADVANTAGES GAINED BY USING THE DATABASE APPROACH TO MANAGING DATA.
- CREATE NORMALIZED TABLES IN A RELATIONAL DATABASE.
- USE ENTITY-RELATIONSHIP DIAGRAMS IN DATABASE DESIGN AND IMPLEMENTATION.
- EXPLAIN THE IMPORTANCE OF ADVANCED DATABASE APPLICATIONS IN DECISION SUPPORT AND KNOWLEDGE MANAGEMENT.

“To bring all the world’s information to users seeking answers”—a huge if not an impossible task, but still it’s the goal that Google researchers are seeking.<sup>1</sup> How would you even begin such a task? Google began by storing and organizing pointers to Internet sites. That information had to be stored in such a way that it was easily retrievable by the user—the person that wanted to find specific information. Today, billions of documents are indexed and quickly available to any user. However, there is much information in traditional print format that is not available online. To make these documents available online, Google’s Book Search is in the process of working with several major libraries to scan, index, and provide the contents of millions of books to users, which is another massive data management project.

Most readers of this book face challenges in data management that are better defined and easier to manage than “all the world’s information.” Your tasks will generally be centered on capturing business events, as they happen, and using that information for making excellent decisions for your organization. You will need tools to accomplish this task—tools such as an understanding of database management systems—as well as understanding the information tools (such as intelligent systems) that interact with database systems to help provide those excellent decisions.

---

<sup>1</sup> From Google.com at <http://www.google.com/corporate>, August 2006.

## Synopsis

---

In this chapter, you will learn about the approaches that organizations use to process business event data. These events, as you learned in Chapter 2, include but are not limited to, sales, purchases, cash receipts, and cash disbursements. In this chapter, you will learn how data from business events are recorded and processed using differing accounting systems designs. As business events occur, data are processed and recorded. In a traditional manual accounting system, or in an automated system designed in the format of a manual accounting system, the accounting data are recorded in journals and classified in ledgers. Increasingly, however, accounting systems are built on underlying databases of business event data. In these databases, accounting information (along with other business information) is stored in database tables. Accounting reports, such as financial statements, and traditional accounting records, such as journals and ledgers, are generated from the information stored in these database tables. In a database accounting system, data management is broken into two functions, the creation and maintenance of master data (see Figure 2.4 on pg. 47) and the recording of business event data (see Figure 2.5 on pg. 48). You will learn the basic elements of database design and implementation that organizations use when they create databases for their accounting information. In some larger organizations, the business event data is copied periodically into a separate database (a *data warehouse*) where it is stored. Managers can gain important insights by analyzing this collected historical data with multidimensional analytic tools and exploratory techniques (called *data mining*). In some companies, these data warehouses are combined with business event databases to create sophisticated reporting systems that help managers make better decisions. These systems include decision support systems, executive information systems, group support systems, and expert systems. Many of these advanced systems use software tools called *intelligent agents*. Finally, you will learn about knowledge management systems that combine event processing databases, data warehouses, and decision support systems, and make the combined knowledge contained in these systems available across the organization.

## Introduction

---

Organizations engage in various business processes, such as hiring employees, purchasing inventory, making sales, and collecting cash from customers. As you learned in Chapters 1 and 2, the activities that occur during the execution of these business processes are called *events*. Among the most important elements in any organization's information systems, whether those systems deal with accounting information or other information that managers use to make decisions, are the data describing these events that are stored in those systems. This is the *business event data* first described in Chapter 1.

As an aspiring decision maker, you should know that no matter what career path you take, data and databases will become an integral part of your day-to-day work. In this chapter, the major approaches used to manage data are described and compared. You will learn about the benefits and costs of alternative methods for collecting, storing, and using business data.

## Two Approaches to Business Event Processing

---

As we begin, let's describe the processing of business event data. First, we know that as organizations engage in business processes, such as purchasing inventory, several business events, such as preparing a purchase order and receiving the goods, will occur.

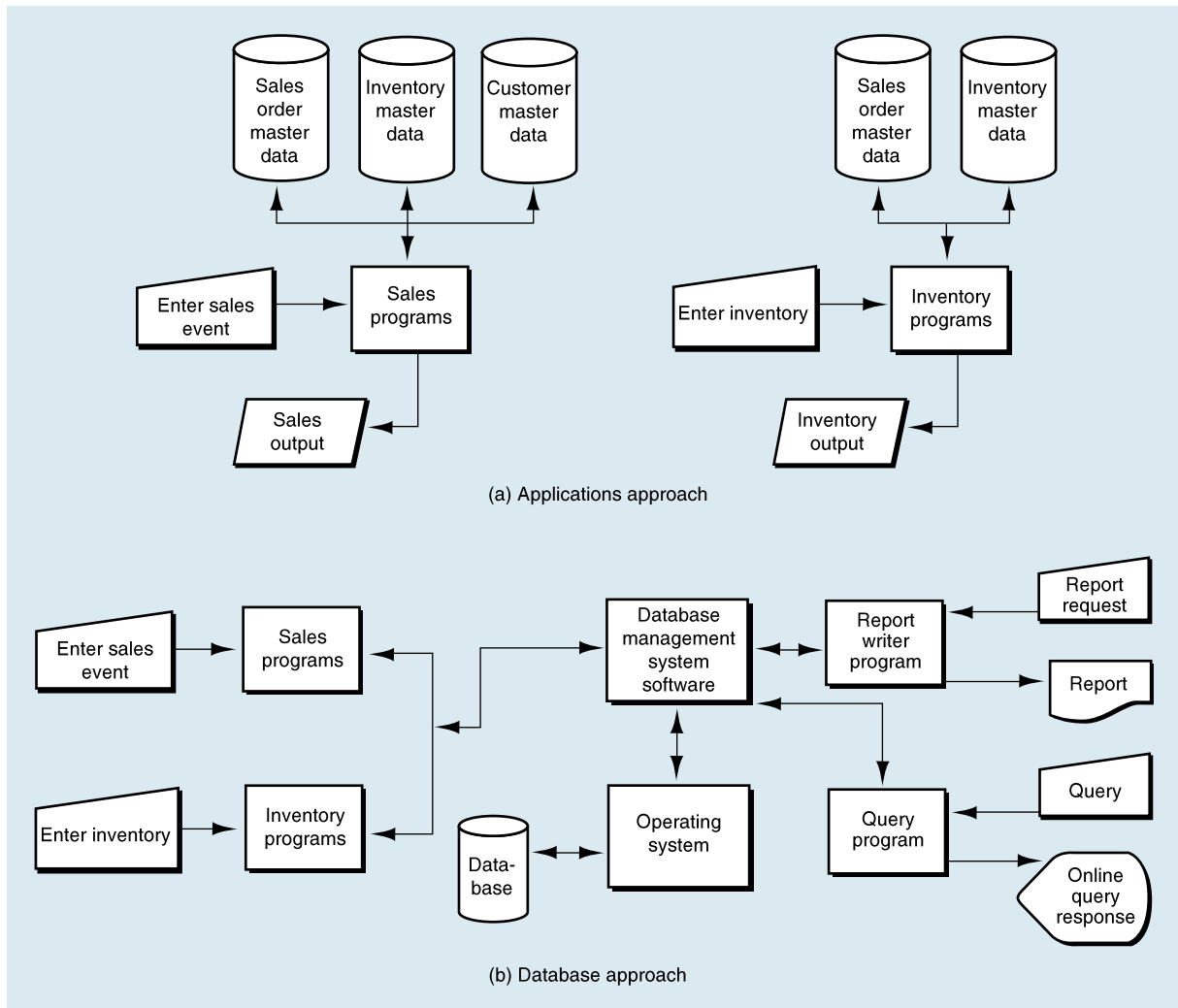
Second, as these business events occur, business event data is captured to describe the *who*, *what*, *where*, and *when* about that event. In the following sections, you will learn about the *applications* and *database* approaches to capturing and storing that business event data.

### The Applications Approach to Business Event Processing

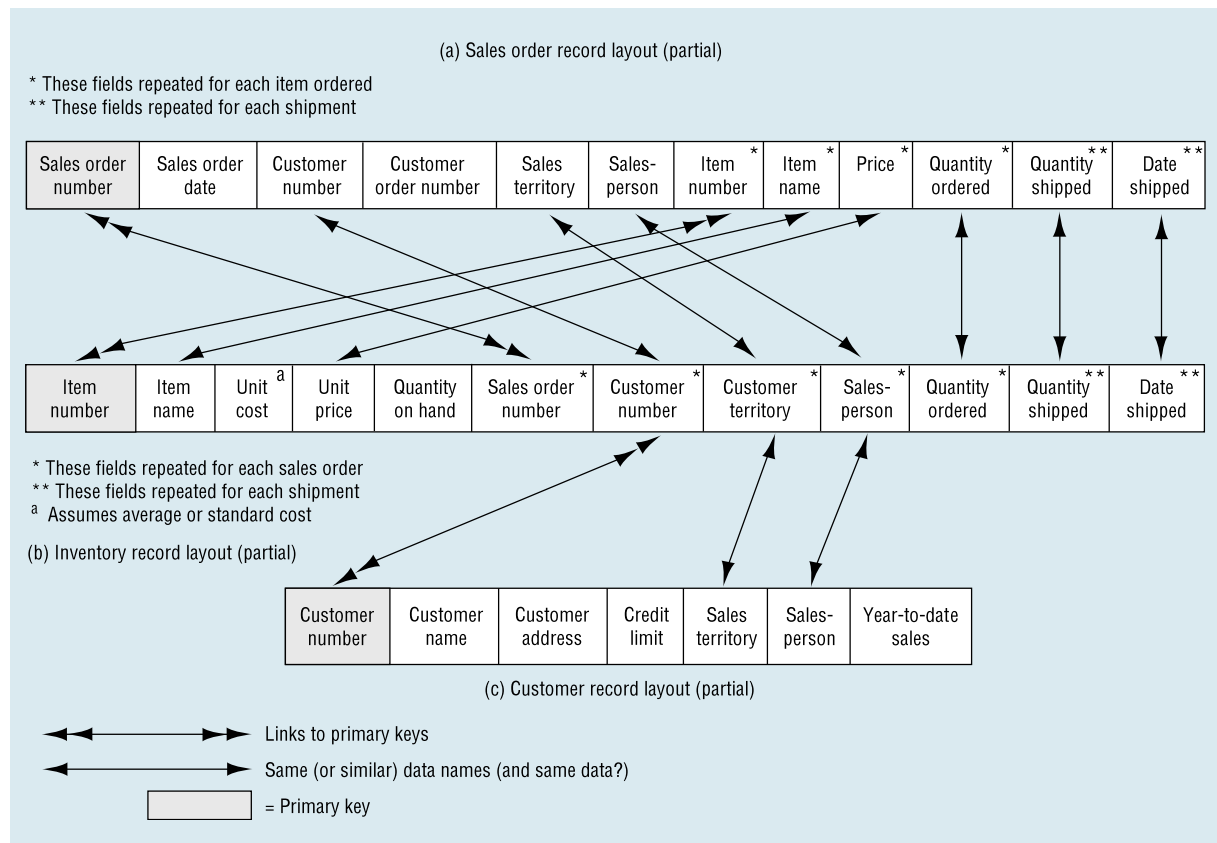
Figure 5.1 compares the applications approach (discussed in this section) with the database approach to business event processing (discussed in the next section). Figure 5.2 (pg. 138) contains the record layouts for the files in Figure 5.1, part (a).

Before databases became widely used in business information systems, organizations tended to view their data as a subordinate element to the program that used the data. As you can see in part (a) of Figure 5.1, the traditional **applications approach to business event processing** view concentrates on the process being performed. In this case, data play a secondary or supportive role to the programs that run in each application system.

**FIGURE 5.1** Two Approaches to Business Event Processing



**FIGURE 5.2** Record Layouts Under an Applications Approach to Business Event Processing



Under this approach, each application collects and manages its own data, generally in dedicated, separate, physically distinguishable files for each application.

**Data Redundancy**

An important consequence of the applications approach is that **data redundancy**, data stored in multiple locations, occurs among the various files. For example, notice the redundancies (indicated by double-ended arrows) depicted in the record layouts in Figure 5.2. This redundancy increases storage costs because the system must store and maintain multiple versions of the same data in different files. In addition, data residing in separate files may not be shareable among applications. The worst consequence of using an applications approach is that the redundant data stored in multiple files can become inconsistent when information is updated in one file and not in other files where it also resides.

In Chapter 1, you learned about horizontal (business operations) and vertical (managerial decision support) information flows. The data in Figure 5.2 have two purposes. The data (1) mirror and monitor the business operations (the *horizontal information flows*) and (2) provide the basis for managerial decisions (the *vertical information flows*). In addition to data derived from the horizontal flows, managers use information unrelated to event data processing. These data would be collected and stored with the business event related data.

Consider the following example: A manager asked that a sales application be designed to perform sales analysis and generate reports such as product sales by territory, by customer, or by salesperson. To do this, the sales application would store data for sales territory and salesperson in the sales order record shown in part (a) of Figure 5.2. Assume that a different manager asked to have the inventory application conduct similar analyses. To do this, the inventory application would store similar (and redundant) data about territory and salesperson such as that depicted in part (b) of Figure 5.2. As implied by Figure 5.1, part (a) (pg. 137) the sales data in the inventory file—including customer territory and salesperson—could be updated by the sales application or updated separately by the inventory application. This would create a condition in which different information about the same fact is stored in different files. This condition violates the *integrity* of the data.

As a second example, consider a feature in the sales application that allowed managers to find the current amount of sales for a particular customer. The summary data in the customer master file (in the year-to-date sales field) could be stored as shown in part (c) of Figure 5.2. Alternatively, the information could be obtained as needed by summarizing data on the sales order or in the inventory data. As a final sales example, assume that a sales manager would like to know all the products that a particular customer has purchased (perhaps so the company can promote those products that the customer is no longer buying). Given the record layouts depicted in Figure 5.2, the information could be obtained by sorting the inventory or sales order data by customer number. Alternatively, the company could have collected these data in the customer master data. In all of these examples, the data are difficult and expensive to obtain. Also, if the applications were not originally designed to yield these data, the applications approach to business event processing makes it difficult to add this access after the fact.

## The Database Approach to Business Event Processing

The preceding examples have all involved business event data related to selling merchandise. The applications approach leaves us with similar problems for *standing data*. In Figure 5.2, you can see the redundancies among the three files with respect to master file standing data such as customer number, territory, and salesperson. These fields could easily take on different values for the same facts over time as changes are made in one file but not all files in which the data are stored. The **database approach to business event processing**, in which facts about events are stored in relational database tables instead of separate files, solves many of the problems caused by data redundancy. You will learn about the database approach in the next section, and then you can return to Figure 5.2 and see how these data are handled in a database approach, rather than an application approach.

## Databases and Business Events

Traditional file management approaches that focus on an applications approach to data management are often sufficient to support a conventional applications approach to processing business event data. The use of databases has improved the efficiency of processing business event data by eliminating data redundancies and improving data integrity. However, the big change that databases have helped make possible is the creation of integrated business information systems that include data about all of a company's operations in one massive collection of relational tables (in one database or in a set of databases that are linked to each other). Multiple users from throughout the organization can view and aggregate *event* data in a manner most conducive to their needs.

ENTERPRISE  
SYSTEMS

At the heart of this movement is a fundamental shift in the view of information processing in business organizations. Traditionally, organizational information systems have been focused on capturing data for the purpose of generating reports and using the reporting function to support decision making. Increasingly, management's view is shifting to one of viewing information systems processing as a decision-support activity first and a reporting function second. This perspective leads to a focus on aggregating and maintaining data in an original form from which reports can be derived, but users also can access and manipulate data using their own models and their own data aggregations. In Chapter 2, you learned about strategic shifts in information design, including the trend toward the use of enterprise systems that might include one or more ERP systems, CRM systems, and other special-purpose systems. Some of these systems include their own databases, but most are built on existing relational databases.

## Database Management Systems

---

A **database management system (DBMS)** is a set of integrated programs designed to simplify the tasks of creating, accessing, and managing data. *Database management systems* integrate a collection of files that are independent of application programs and are available to satisfy a number of different processing needs. Organizations use *DBMSs* to coordinate the activities of their many functional areas. The DBMS, containing data related to all an organization's applications, supports normal data processing needs and enhances the organization's management activities by providing data useful to managers. We will use the term *enterprise database* synonymously with *database management system* or DBMS. This use is consistent with the meaning intended by most computer users and developers.

### Logical versus Physical Database Models

The concept underlying the database approach to business event processing is to decouple the data from the system applications (that is, to make the data independent of the application or other users). This decoupling, called **data independence**, is a major difference between the database approach and the applications approach. Recall that in the applications approach, the data is subordinate to, or dependent upon, the application program that uses the data. Therefore, as reflected in part (b) of Figure 5.1 (pg. 137), the data become the focus of attention. Several other aspects of the database approach are noteworthy:

- The database is now shared by multiple system applications that support related business processes, as shown at the left of Figure 5.1, part (b).
- In addition to being used by application programs, the data can be accessed through two other user interfaces that are built into most database management software: (1) *report generation*, the creation of onscreen or printed summaries of specific data as shown in the upper-right portion of part (b), and (2) ad hoc user inquiries, also called *queries*, which allow users to ask questions about the data in the database handled through *query language* software, depicted in the lower-right portion of part (b).
- Two layers of software are needed to translate user views into instructions for retrieving the data from the physical location in which it is stored (for example, a computer's disk drive). The distinction between the way a user thinks of the data in a database, called a user's *logical view*, and the way the data is actually stored on the computer hardware, called the *physical view* of the data, is important for computer scientists who develop database software, but it is not very important for accountants and managers who use the software because the DBMS includes the software that

## TECHNOLOGY SUMMARY 5.1

**DATABASE MANAGEMENT SYSTEMS (DBMSs)**

A DBMS is a set of integrated programs designed to simplify the tasks of creating, accessing, and managing a database. The DBMS performs several functions:

- Defining the data
- Defining the relations among data
- Interfacing with the *operating system* for storage of the data on the *physical media*
- Mapping each user's view of the data

In the language of DBMS, a **schema** is a complete description of the configuration of record types, data items, and the relationships among them. The

schema defines the *logical* structure of the database. The schema, therefore, defines the organizational view of the data. A **subschema** is a description of a *portion* of a schema. The DBMS maps each user's view of the data from subschemas to the schema. In this way, the DBMS provides flexibility in identifying and selecting records. Each of the many database users may want to access records in his or her own way. For example, the accounts receivable manager might want to access customer records by invoice number, whereas a marketing manager may want to access the customer records by geographic location. The following figure portrays the schema-subschema relationship.

Customer number	Customer name	Customer address	Credit limit	Sales-person	Sales territory	Year-to-date sales
-----------------	---------------	------------------	--------------	--------------	-----------------	--------------------

(a) Schema

Customer number	Customer name	Credit limit
-----------------	---------------	--------------

(b) Credit department subschema

Customer number	Customer name	Sales-person	Sales territory	Year-to-date sales
-----------------	---------------	--------------	-----------------	--------------------

(c) Sales manager subschema

A chief advantage of a DBMS is that it contains a **query language** (also called a **data manipulation language**, or **DML**), which is a language much like ordinary language. A query language is used to access a database and to produce inquiry reports. These languages allow nontechnical users to bypass the programmer and to access the database directly. Deriving data from the database using a query does not replace applications programs, which are still required to perform routine data processing tasks. However, when information is needed quickly or when a manager wants to browse through the database combining data in a variety of ways, the query facility of a DBMS is a vast improvement over the traditional method of

requesting that a program be written to generate a report. Later in this chapter and in Chapter 6, you will see examples of *SQL (Structured Query Language)*, the de facto standard for DBMS query languages.

A DBMS normally contains a number of security controls to protect the data from access by unauthorized users as well as from accidental or deliberate alteration or destruction. A DBMS also includes software that allows the data to be simultaneously shared by multiple users. This software often allows managers to manage access rights for specific users. For example, all employees might be able to view employee names and addresses, but only specific authorized users would be able to view employee's pay rates.

handles this translation automatically. Some of the more technical design issues of DBMSs are described in Technology Summary 5.1.

Figure 5.3 depicts how a database might look if the data from Figure 5.2 (pg. 138) were stored in a database that used a relational structure, which is the most common type of database structure used in businesses today. The data from our three files are now stored in four tables: CUSTOMERS (instead of the customer master data file), INVENTORY\_ITEMS (instead of the inventory master data file), and SALES\_ORDERS and

**FIGURE 5.3** Record Layouts as Tables

Shaded\_Attribute(s) = Primary Key

CUSTOMERS				
Cust_Code	Cust_Name	Cust_City	Credit_Limit	Sales_YTD
ETC	Bikes Et Cetera	Elgin	10000.00	9561.55
IBS	Inter. Bicycle Sales	New York	5000.00	4191.18
RODEBYKE	Rodebyke Bic. & Mopeds	San Jose	2000.00	1142.50
STANS	Stan's Cyclery	Hawthorne	10000.00	8330.00
WHEEL	Wheelaway Cycle Center	Campbell	10000.00	6854.00

INVENTORY_ITEMS				
Item_Number	Item_Name	Qty_On_Hand	Unit_Cost	Unit_Price
1000-1	20 in. Bicycle	247	55.00	137.50
1001-1	26 in. Bicycle	103	60.00	150.00
1002-1	24 in. Bicycle	484	60.00	150.00
1003-1	20 in. Bicycle	4	24.37	60.93
1280-054	Kickstand	72	6.50	16.25
2010-0050	Formed Handlebar	90	4.47	11.25
3050-2197	Pedal	23	0.75	1.88
3961-1010	Tire, 26 in.	42	1.45	3.13
3961-1041	Tire Tube, 26 in.	19	1.25	3.13
3965-1050	Spoke Reflector	232	0.29	0.63
3970-1011	Wheel, 26 in.	211	10.50	25.00

SALES_ORDERS			
SO_Number	Cust_Code	Cust_Order_Number	SO_Date
1010	WHEEL	453	061205
1011	ETC	347	061205
1012	WHEEL	56-6	061205
1013	IBS	3422	061205
1014	ETC	778	061205
1015	WHEEL	5673	061206
1016	ETC	3345	061206

SALES_LINES				
SO_Number	Item_Number	Qty_Ordered	Sales_Price	Qty_Shipped
1010	1000-1	5	137.50	0
1010	2010-0050	2	11.25	0
1011	1001-1	10	127.50	8
1011	1002-1	5	150.00	4
1012	1003-1	5	60.93	0
1012	1001-1	10	127.50	5
1013	1001-1	50	78.30	0
1014	1003-1	25	37.42	0
1015	1003-1	25	37.42	0
1016	1003-1	5	60.93	0
1016	3965-1050	50	33.00	0
1016	3961-1041	5	3.13	0
1016	1000-1	4	137.50	0

SALES\_LINES (these two tables replace the sales order master data file). These tables are *logical views* of data that are *physically* stored in a database. Users can access the data in the tables by:

- Formulating a query.
- Preparing a report using a report writer.
- Including a request for data within an application program.



These three methods are depicted in the flowchart in Figure 5.1, part (b) (pg. 137). The following two examples can help you see how easily data can be obtained from the tables in Figure 5.3. These examples use the database query language *SQL (Structured Query Language)*. SQL is widely used because it works with a number of DBMSs (although some variations exist in syntax and allowed commands in different DBMSs), and it resembles English. You should be able to follow these two simple query examples here and you will learn more about SQL in the next chapter.

1. A query that uses the SQL SELECT command can return the customers assigned to salesperson Garcia.

```
SELECT    Cust_Code Cust_Name Cust_City
FROM      CUSTOMERS
WHERE     Salesperson = 'Garcia'
```

Cust_Code	Cust_Name	Cust_City
STANS	Stan's Cyclery	Hawthorne
WHEEL	Wheelaway Cycle Center	Campbell

You can see that there are two customers, STANS and WHEEL, who are assigned to salesperson Garcia.

2. You also can create more complex queries using the SELECT command. Consider the query shown here with its results:

```
SELECT    SO_Number INVENTORY_ITEMS.Item_Number Sales_Price
          Unit_Price
FROM      SALES_LINES INVENTORY_ITEMS
WHERE     Sales_Price <> Unit_Price AND INVENTORY_ITEMS
          .Item_Number = SALES_LINES.Item_Number
```

SO_Number	Item_Number	Sales_Price	Unit_Price
1011	1001-1	\$127.50	\$150.00
1012	1001-1	\$127.50	\$150.00
1013	1001-1	\$78.30	\$150.00
1014	1003-1	\$37.42	\$60.93
1015	1003-1	\$37.42	\$60.93
1016	3965-1050	\$33.00	\$0.63

In this query, the SELECT command examines fields in both the SALES\_LINES and INVENTORY\_ITEMS tables and finds those items in the combined table that were sold at a price (Sales\_Price) other than the price contained on the INVENTORY\_ITEMS table (Unit\_Price). The query result shows that there are six instances where this occurred.

In some cases, a user formulates a query and enters it to receive the query results immediately on the computer screen. In other cases, queries are placed into onscreen forms or reports that are generated by software within the DBMS. A third alternative is to include queries in the program code that use the data in the DBMS. In the second and third cases, users might not be aware that a query was operating to help them obtain the information. They simply open the form, print the report, or run the program.

## Overcoming the Limitations of the Applications Approach

Earlier in this chapter, you learned about some of the limitations of the applications approach to business event processing. In this section, you will learn how the database approach can overcome these limitations. You also will learn about other advantages of the database approach. Of course, the database approach has its own limitations. You will learn about those in this section also.

### CONTROLS

- **Eliminating data redundancy:** With the database approach to business event processing, an item of data is stored only once. Applications that need data can access the data from the central database. For example, in Figure 5.1, part (a) (pg. 137), multiple versions of the inventory master data are present, whereas in part (b) of that figure, only one exists. Further, Figure 5.2 (pg. 138) depicts the same data elements on more than one file, whereas Figure 5.3 (pg. 142) shows each data element only once. An organization using the applications approach to business event processing must incur the costs and risks of storing and maintaining these duplicate files and data elements.
  - **Ease of maintenance:** Because each data element is stored only once, additions, deletions, or changes to the database are accomplished easily. Contrast this to the illustration in Figure 5.2 (pg. 138), where changes in a salesperson, territory, or customer combination would require changes in three different files.
  - **Reduced storage costs:** By eliminating redundant data, storage space is reduced, which results in associated cost savings. However, in most database installations, this savings is *more than offset* by the additional costs of DBMS software.
- ### CONTROLS
- **Data integrity:** This advantage, like several others, results from eliminating data redundancy. As mentioned earlier, storing multiple versions of the same data element is bound to produce inconsistencies among the versions. For instance, the salesperson and sales territory data might differ among their many versions, not only because of clerical errors but because of timing differences in making *data maintenance* changes. Inconsistent data could also result from the timing differences that can occur during *business event processing* of the inventory master data by the sales and inventory applications. With only one version of each data element stored in the database, such inconsistencies are no longer a threat.
  - **Data independence:** The database approach allows multiple application programs to use the data concurrently. The data can be accessed in several ways (for example, through applications processing, online query, and report writing programs), and the access can be quickly changed by modifying the definition of the tables or views. With the traditional applications approach to business event processing, the programs would need revisions to provide access to more or less data.
- ### CONTROLS
- **Privacy:** The security modules available in most DBMS software include powerful features to protect the database against unauthorized disclosure, alteration, or destruction. Control over data access can typically be exercised down to the data element level. Users can be granted access to data for reading or updating (add, revise, delete) data. Other ways to implement security include *data classification* (i.e., data objects are given classification levels, and users are assigned clearance levels) and *data encryption* (discussed in Chapter 9).

Despite the many advantages of using a DBMS instead of an application approach, some organizations do not use a DBMS. A DBMS can be expensive to implement. In general, a DBMS requires more powerful, and thus more expensive, hardware. The DBMS itself costs money. Hiring people to maintain and operate the database can be more expensive than hiring application maintenance programmers. Also, drawbacks

exist that are related to *operating* the DBMS. Operational issues can include the following:

- Although database sharing is an advantage, it carries with it a downside risk. If the DBMS fails, all of the organization's information processing halts.
- Because all applications depend on the DBMS, *continuous data protection (CDP)* and *contingency planning* (discussed in Chapter 8) are more important than in the applications approach to data management.
- When more than one user attempts to access data at the *same* time, the database can face “contention” or “concurrency” problems. Procedures such as *record locking* or *field locking* can mitigate these problems, but these solutions are not foolproof. Further discussion of this topic is beyond the scope of this book.
- Territorial disputes can arise over who “owns” the data. For instance, disputes can arise regarding who is responsible for data maintenance (additions/deletions/changes) to customer data. The sales department might think it should own those data, but the credit department and the accountants managing accounts receivable might argue with that contention.

To cope with these and other problems, most companies that have adopted the database approach have found it necessary to create a database administrator function. In most organizations, the database administrator is responsible for administrative and technical issues related to the DBMS.

## Database Essentials

---

The design and implementation of a DBMS can be a more complex process than creating specific applications with subordinate data. To understand how the database approach works, you need some background information in database essentials. You will learn about logical database structures and gain an understanding of some key database elements in this section. You also will be introduced to the process of designing and implementing a database. Chapter 6 includes a more detailed treatment of database design, implementation, and use.

### Logical Database Models

Four types of logical DBMS models exist: hierarchical, network, relational, and object-oriented. As a *designer* or *user* of an AIS, you will participate in the selection of the DBMS for your organization, and the choice from among these logical models will affect the speed and flexibility of the DBMS. In addition, as a user or auditor of business information systems, your effective use of a DBMS often depends on your understanding of these logical models.

The first DBMSs used a **hierarchical database model**. In this model, records are organized in a pyramid structure. The records at or near the top of the structure contain records below them. This structure works well for simple situations. For example, a bank that wants to record information about its customers and their accounts could use a hierarchical DBMS. The top-level records may hold information about customers. The next level down could include records with information about accounts. A customer might have a savings account, a checking account, and a loan account. All of a customer's accounts would be below that customer record in the hierarchy. The next level down may include records that stored information about transactions in each account.

In a hierarchical DBMS, records that are included in a record one level above them are called **child records** of that upper-level record. **Parent records** include the

lower-level child records. A child of a parent record can be the parent of another child record. Each parent record can have many child records, but each child record can have only one parent record. In the bank example, the customer records would be the parent records of the child account records, which in turn would be the parent records of the child records that stored information about account increases and decreases.

Hierarchical DBMSs work well for simple data structures; however, they fall apart quickly when the data becomes more complex. For example, the hierarchical bank example DBMS described previously could not handle joint accounts unless the joint owners were considered a single entity. To handle more complex data structures, database researchers created a different database model. In the **network database model**, a child record can have more than one parent record. This was a significant improvement over the early hierarchical designs. Network DBMSs were adopted by a number of organizations that had been frustrated by the limitations of hierarchical DBMSs. The wholesale move to network DBMSs was interrupted, however, by the development of a vastly more flexible model, the relational database.

### Relational Database Model

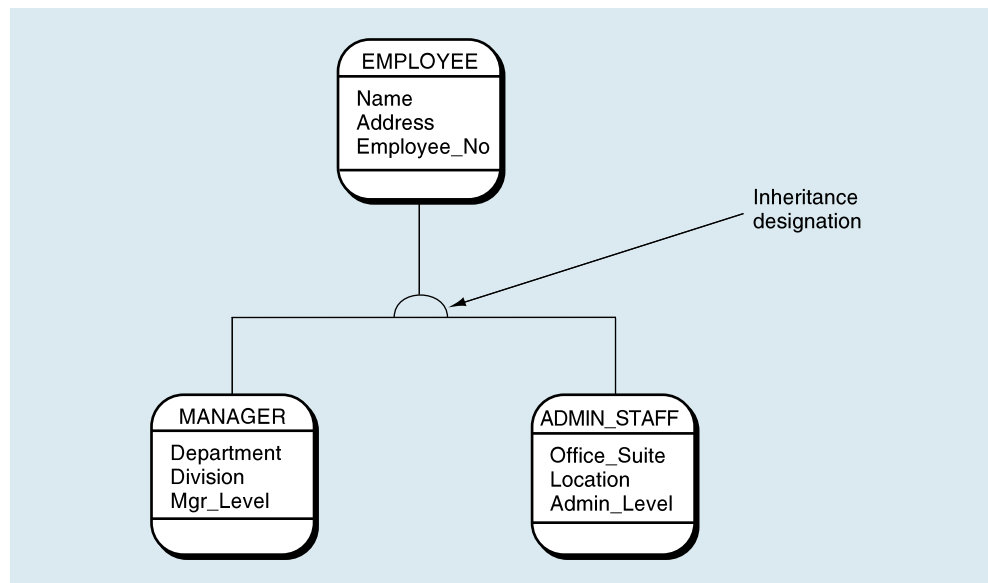
In a **relational database model**, data are logically organized in two-dimensional tables. Each individual fact or type of information is stored in its own table. The relational model was developed using a branch of mathematics called *set theory*. In set theory, a two-dimensional collection of information is called a *relation* (thus the name “relational”). However, today most people call these collections of information “tables.” A relational DBMS allows users to query the tables to obtain information from one or more tables in a very flexible way. The tables in Figure 5.3 (pg. 142) are relational tables.

The relational structure is attractive from a user’s standpoint because end users often think of the data they need as a table. This mental picture translates well to the logical data model of a relational DBMS. The capability of a relational DBMS to handle complex queries also is important. Although the relational model is considered to be a dramatic improvement over the network and relational models, it does have two disadvantages. First, a relational DBMS requires much more computer memory and processing time than the earlier models. Increases in computer processing as well as a steady decrease in hardware costs have reduced the impact of this first disadvantage. The second disadvantage is that the relational model, as originally conceived, allows only text and numerical information to be stored in the database. It did not allow the inclusion of complex object types in the database such as graphics, audio, video, or geographic information. The desire to include these complex objects in databases led to the development of object-oriented databases.

### Object-Oriented Database Model

Both simple and complex objects can be stored in an **object-oriented database model**. Earlier data models were designed to store text-based data (which includes numbers). In *object-oriented* data models, other types of data can be stored. For example, video clips or pictures can be stored in an object-oriented database. Object-oriented databases include abstract data types that allow users to define characteristics of the data to be stored when developing an application. This overcomes the limitations of relational databases. Relational databases limit the types of data that can be stored in table columns. Instead of tables, an object-oriented DBMS stores data in objects.

An object can store attributes (similar to the attributes stored in table columns in a relational database), and instructions for actions that can be performed on the object or its attributes. These instructions are called *encapsulated methods* (“encapsulated” because they are included as part of the object). Objects can be placed in a hierarchy so that other

**FIGURE 5.4** Object-Oriented Database Model

objects lower in the hierarchy (subclass objects) can obtain (inherit) attributes from objects higher in the hierarchy (superclass objects). Figure 5.4 shows three objects in an object-oriented DBMS. The superclass object EMPLOYEE provides the same set of attributes to both subclasses—MANAGER and ADMIN\_STAFF. In other words, every MANAGER would have a Name, Address, and Employee\_No (as would every ADMIN\_STAFF). Objects are drawn using a rectangle with rounded corners, which is divided into three parts: the object name, the attributes, and any encapsulated methods.

Although many researchers have argued that object-oriented DBMSs are superior to relational DBMSs, most organizations still use relational DBMSs. The main advantage of an object-oriented model is to store complex data types. In recent years, **object-relational databases** have been developed. This combined structure includes a relational DBMS framework with the capability to store complex data types. Although developers continue to refine object-oriented DBMSs, most companies seem to be satisfied with relational *or* object-relational DBMSs at this time. The next section describes some of the important elements of relational databases.

## Elements of Relational Databases

Although many vendors offer DBMS software products, all DBMSs have the same essential elements. In this section, you will learn about these elements and some important concepts that underlie database design and implementation.

The elements that make up all DBMSs include **tables**, a place to store data; **queries**, tools that allow users and programmers to access the data stored in various tables; **forms**, onscreen presentations of data in tables and collected by queries from one or more tables; and **reports**, which provide printed lists and summaries of data stored in tables or collected by queries from one or more tables.

The most important step in creating a useful database is designing the tables properly. Each database table only stores data about one specific thing. It might be a person, such as a customer; an object, such as inventory; or an event, such as a sale. You are probably familiar with spreadsheet software. In spreadsheet software, the main element of the user interface is a worksheet. The worksheet has rows and columns.

A database table is like a worksheet in that it has rows and columns, but it is unlike a worksheet in that a database table has very strict rules about what can be put in those rows and columns. In a database table, a specific row contains all information about a particular instance of the type of thing stored in the table. For example, the first row in the CUSTOMERS table (see Figure 5.3 on pg. 142) stores all of the information about the customer Bikes Et Cetera. No other row can contain information about that customer. Database table columns each store one specific attribute of the type of things stored in the table. For example, the column labeled Cust\_Name in the CUSTOMERS table in Figure 5.3 stores all of the customer names. No customer name can ever appear in any other column.

A spreadsheet cell (the intersection of a row and a column) can hold text, numbers, a formula, a graphic, a command button, a chart, or any of a number of different types of data. A cell in a database table can only hold the type of data allowed in its column and must contain the value of that specific attribute for the instance of the item recorded in its row. For example, the first row of the SALES\_LINES table (Figure 5.3) contains a number, 137.50, in the Sales\_Price column. This column contains the sales price of each item on each sales order. The values in this column must be numbers, and the numbers must each have two decimal places. The number in the first row is the sales price of the five Item Number 1000-1 products that were sold on sales order number 1010. This fact is stored in this row and column and does not appear anywhere else in the database.

Each row in a database must be unique; that is, no other row can exist in the table that stores identical information. Each row must include a unique identifier that serves as an address for the row. This unique identifier is called the table's **primary key**, and its value is often stored in the first column of the table. In Figure 5.3, you can see that the primary key of the CUSTOMERS table is the first column of the table, labeled Cust\_Code. Some tables use two or more columns in combination to provide a primary key for each row. This type of key is called a **composite primary key**. The SALES\_LINES table in Figure 5.3 has a composite primary key formed by combining the first two columns in that table. Note that each of the first two columns individually contains nonunique values in some rows, but when the two columns are combined, the resulting values across both columns are unique for all rows in the table.

The primary key fields in tables must be unique identifiers, which can present problems for database designers who are not very careful in creating the values to be used in these fields. For example, a designer who wants to create a primary key for an employee table might be tempted to use employees' last names as the value. As soon as the business hires more than one person with the same last name, the primary key becomes unworkable. Because the requirement that primary key fields contain unique values is so important, most database designers create an artificial value for each row in each table. They do this by following procedures for classifying and coding as described in Technology Summary 5.2.

DBMSs have many built-in tools for creating tables and enforcing the strict rules on the columns and rows. After the tables have been properly designed, the DBMS helps to enforce the design by using these tools. First, however, the tables need to be designed in accordance with the relational model. This design task is the subject of the next two sections.

## Normalization in Relational Databases

Two approaches are used in designing a relational database: bottom-up and top-down. You will learn about top-down design in the next section. In the bottom-up design

## TECHNOLOGY SUMMARY 5.2

## CLASSIFYING AND CODING

Classifying and coding data are important elements of any database design. **Classifying** is the process of grouping or categorizing data according to common attributes. Your college or university probably has numerous occasions to classify you (and your peers who have the same characteristics) according to class (freshman, sophomore, etc.), housing (resident versus commuter), financial aid status, major, class enrollment (e.g., all students enrolled in section 001 of AC 322), and so forth.

To classify students into meaningful categories, schools do not create computer records that include values such as “junior,” “resident,” “work study assigned to department X,” or “20-meal plan.” To be efficient, they use a shorthand substitute for these long labels. The creation of these substitute values, or codes, is called **coding**. Many different coding schemes exist; the following list outlines the most common and useful examples. The accompanying figure illustrates five common coding types.

- **Sequential Coding:** Sequential coding (also known as **serial coding**) assigns numbers to objects in chronological sequence. This coding scheme provides limited flexibility. Additions can be made only at the end of the sequence; deletions result in unused numbers unless the numbers are recycled; and the codes tell nothing about the objects’ attributes.
- **Block Coding:** In block coding, groups of numbers are dedicated to particular characteristics of the objects being identified. This provides some improvement over sequential coding. For example, in the universal product codes (UPCs) that supermarkets and other retailers use, the first block of five digits represents the manufacturer, and the second block of five digits designates the product. However, within each block, no significance is given to any of the digits. For example, Kellogg’s has a manufacturer’s code of 38000, whereas Ralston Purina’s code is 17800. Cracklin’ Oat Bran™ (Kellogg’s) has a 04510 product identifier, which appears to have no relationship to Ralston’s Wheat Chex™ code, 15011. Within each block of digits, numbers are usually assigned sequentially (see the employee example in the accompanying figure), which means that block coding may have the same


limitations related to additions and deletions as sequential coding.

- **Significant Digit Coding:** Significant digit coding assigns meanings to specific digits. The accompanying figure shows how this method can be used for inventory items. Parts of the inventory item number describe the product group, the product type (part, subassembly, or end-item), the warehouse in which the part is stored, and a unique number that identifies the specific item.
- **Hierarchical Coding:** Like significant digit codes, hierarchical codes also attach specific meaning to particular character positions. Hierarchical coding orders items in descending order, where each successive rank order is a subset of the rank above it. Reading from left to right in a hierarchical code, each digit is a subcategory of the digit to its immediate left. The five-digit postal ZIP code illustrated in the accompanying figure shows the hierarchical elements in this type of coding.
- **Mnemonic Coding:** Computers are good at handling numeric data, and typically the previous coding schemes use numbers. Most humans, however, have trouble learning and remembering strings of numbers. In mnemonic coding, some or all of the code is made of letters. The word *mnemonic* comes from the Greek *mnemonikos*, to *remember*, and means “assisting or related to memory.” The accompanying figure shows a mnemonic code used for college courses.
- **Other Coding Schemes:** Other coding schemes can be useful in creating primary key fields in databases. A **self-checking digit code** includes an extra digit that can be used to check the accuracy of the code. The extra digit is computed by applying a mathematical formula to the primary code. For example, a bank account number of 1234-0784 might have an extra digit of 9 (it might appear on the account as 1234-0784-9). The number 9 is calculated from the other numbers using a formula. In this case, the formula is the sum of the last four digits ( $0 + 7 + 8 + 4 = 19$ ) minus the sum of the first four digits (e.g.,  $1 + 2 + 3 + 4 = 10$ ). If a data entry clerk enters the first eight digits incorrectly, the check digit will likely be different. The formulae used in practice are much more complex, but they work much the same as in this example.

Coding Type	Everyday Example(s)	Example Based on Employee ID Codes
A. Sequential (serial)	<ul style="list-style-type: none"> <li>• Student ID numbers</li> <li>• Ticket taken to identify your turn to be waited on at the supermarket deli counter</li> </ul>	001 = first employee hired 002 = second employee hired ⋮ etc.
B. Block	Universal product code (UPC): <sup>Ⓐ</sup> <div style="display: flex; justify-content: space-around; align-items: center;"> <div style="text-align: center;">                         73805                          {                          Manufacturer code                     </div> <div style="text-align: center;">                         80248                          {                          Product identifier                     </div> </div>	001-100 fabricating department employees 101-200 assembly department employees ⋮ etc. <div style="float: right; font-size: 2em;">}</div> Within department blocks, codes are usually assigned to individual employees on a sequential basis.
C. Significant digit	Inventory item: <div style="display: flex; justify-content: space-around; align-items: center;"> <div style="text-align: center;">                         16                          {                          Product group                     </div> <div style="text-align: center;">                         2   Part, subassembly or end-item                     </div> <div style="text-align: center;">                         17                          {                          Warehouse                     </div> <div style="text-align: center;">                         4389                          {                          Unique item identifier                     </div> </div>	<div style="display: flex; justify-content: space-around; align-items: center;"> <div style="text-align: center;">                         2                          ⋮                          Work center                     </div> <div style="text-align: center;">                         0                          ⋮                          Exempt or nonexempt                     </div> <div style="text-align: center;">                         4                          ⋮                          Pay rate code                     </div> <div style="text-align: center;">                         623                          {                          Unique employee identifier                     </div> </div>
D. Hierarchical	Postal ZIP Codes: <div style="display: flex; justify-content: space-around; align-items: center;"> <div style="text-align: center;">                         0                          ⋮                          Section of country                     </div> <div style="text-align: center;">                         18                          {                          Region within section                     </div> <div style="text-align: center;">                         90                          {                          Locality (e.g., town) within region                     </div> </div>	<div style="display: flex; justify-content: space-around; align-items: center;"> <div style="text-align: center;">                         01                          {                          Company division                     </div> <div style="text-align: center;">                         3                          ⋮                          Plant within division                     </div> <div style="text-align: center;">                         9                          ⋮                          Department within plant                     </div> <div style="text-align: center;">                         623                          {                          Unique employee identifier                     </div> </div>
E. Mnemonic	College course numbering: AC 340 = Accounting Information Systems EN 101 = English Composition	<div style="display: flex; justify-content: space-around; align-items: center;"> <div style="text-align: center;">                         F                          ⋮                          Female                     </div> <div style="text-align: center;">                         M                          ⋮                          Married                     </div> <div style="text-align: center;">                         C                          ⋮                          Caucasian                     </div> <div style="text-align: center;">                         623                          {                          Unique employee identifier                     </div> </div>

NOTE:

Ⓐ The universal product code (UPC) is physically implemented through bar codes attached to the product or its container. Therefore →



approach, the designer identifies the attributes that are of interest and organizes those attributes into tables. Referring to Figure 5.3 (pg. 142), you can see that the attributes of sales events might include such things as customer name, customer city, salesperson, sales territory, credit limit, inventory item name, unit cost, unit price, sales order number, and so on.

The structure of the tables must comply with several rules that are based on set theory, the branch of mathematics on which relational database models are based. These rules are called **normal forms** and include specifications that must be met by relational database tables. Following the normal forms yields tables that prevent errors (also called **anomalies**) that otherwise might occur when adding, changing, or deleting data stored in the database.

Applying the normal forms to collections of data transforms tables that are not in normal form into tables that comply with the rules. The resulting tables are said to be “in normal form.” There are six levels of normal form, but business systems usually work well if they comply with the first three. A table that is in first normal form (1NF) is



usually preferable to a table that is not in 1NF; a table in second normal form (2NF) is preferred to a table in 1NF; and a table in third normal form (3NF) is preferred to a table in 2NF. The goal of normalization is to produce a database model that contains relations that are in 3NF. The normal forms are inclusive, which means that each higher normal form includes all lower normal forms. That is, a table in 3NF is in 1NF and in 2NF. Two concepts are essential to an understanding of normal forms: functional dependence and primary keys.

### Functional Dependence and Primary Keys

An attribute (a column in a table) is **functionally dependent** on a second attribute (or a collection of other attributes) if a value for the first attribute determines a single value for the second attribute at any time. When functional dependence exists, the first attribute determines the second attribute.

Consider a table that contains information about purchases in two columns, one for purchase order number (PO\_Num) and another for purchase order date (PO\_Date). A value in PO\_Date does not determine the value in PO\_Num because a particular date could have several purchase orders (and thus, several distinct values for PO\_Num) associated with it. In this case, PO\_Num is not functionally dependent on PO\_Date. However, PO\_Date is functionally dependent on PO\_Num because the value in PO\_Num will always be associated with a single value for PO\_Date; the value will always be the date on which that purchase order was issued.

The second concept that is essential to understanding normalization is that of the primary key. Although you know from its definition earlier in this chapter that a primary key contains a value that uniquely identifies a specific row in a table, the use of this concept in normalization requires that you learn a more formal specific definition. A candidate attribute (a column or collection of columns) in a table is that table's primary key if:

- All attributes in the table are functionally dependent on the candidate attribute.
- No collection of other columns in the table, taken together, has the first property.

### First Normal Form (1NF)

An **unnormalized table** contains repeating *attributes* (or *fields*) within each *row* (or *record*). We call these repeated attributes “repeating groups.” Figure 5.5 (pg. 152) is an unnormalized table because it contains repeating groups. Each sales order occupies one row, but then Item\_Number, Item\_Name, Qty\_Ordered, Cust\_Code, and Cust\_Name are repeated as many times as necessary.

A table is in **first normal form (1NF)** if it does not contain repeating groups. Transforming this table into 1NF requires the removal of the repeating groups. Figure 5.6 (pg. 152) shows the table SALES\_ORDERS in 1NF. Instead of one row with repeating groups, each sales order is represented in the number of rows required. For example, sales order 1010 now has two rows, rather than the one row it had in Figure 5.5. The *primary key* for the new table is a combination of SO\_Number and Item\_Number. Recall that a primary key that is formed by the combination of two or more columns is called a *composite primary key*. The simpler table structure shown in Figure 5.6 solves a big problem. If a table has repeating groups, the designer must decide in advance how many repeats to allow. The risk is always that the designer will not allocate enough columns. With tables in 1NF, the designer does not need to speculate because the table expands vertically to accommodate any number of items.

**FIGURE 5.5** Unnormalized Relation

SALES_ORDERS					
SO_Number	Item_Number	Item_Name	Qty_Ordered	Cust_Code	Cust_Name
1010	2010-0050	Formed Handlebar	2	WHEEL	Wheelaway Cycle Center
	1000-1	20 in. Bicycle	5	WHEEL	Wheelaway Cycle Center
1011	1002-1	24 in. Bicycle	5	ETC	Bikes Et Cetera
	1001-1	26 in. Bicycle	10	ETC	Bikes Et Cetera
1012	1003-1	20 in. Bicycle	5	WHEEL	Wheelaway Cycle Center
	1001-1	26 in. Bicycle	10	WHEEL	Wheelaway Cycle Center
1013	1001-1	26 in. Bicycle	50	IBS	Inter. Bicycle Sales
1014	1003-1	20 in. Bicycle	25	ETC	Bikes Et Cetera
1015	1003-1	20 in. Bicycle	25	WHEEL	Wheelaway Cycle Center
1016	3961-1041	Tire Tube, 26 in.	5	ETC	Bikes Et Cetera
	3965-1050	Spoke Reflector	50	ETC	Bikes Et Cetera
	1003-1	20 in. Bicycle	5	ETC	Bikes Et Cetera
	1000-1	20 in. Bicycle	4	ETC	Bikes Et Cetera

**FIGURE 5.6** Relation in First Normal Form (1NF)

Shaded\_Attribute(s) = Primary Key

SALES_ORDERS					
SO_Number	Item_Number	Item_Name	Qty_Ordered	Cust_Code	Cust_Name
1010	2010-0050	Formed Handlebar	2	WHEEL	Wheelaway Cycle Center
1010	1000-1	20 in. Bicycle	5	WHEEL	Wheelaway Cycle Center
1011	1002-1	24 in. Bicycle	5	ETC	Bikes Et Cetera
1011	1001-1	26 in. Bicycle	10	ETC	Bikes Et Cetera
1012	1003-1	20 in. Bicycle	5	WHEEL	Wheelaway Cycle Center
1012	1001-1	26 in. Bicycle	10	WHEEL	Wheelaway Cycle Center
1013	1001-1	26 in. Bicycle	50	IBS	Inter. Bicycle Sales
1014	1003-1	20 in. Bicycle	25	ETC	Bikes Et Cetera
1015	1003-1	20 in. Bicycle	25	WHEEL	Wheelaway Cycle Center
1016	3961-1041	Tire Tube, 26 in.	5	ETC	Bikes Et Cetera
1016	3965-1050	Spoke Reflector	50	ETC	Bikes Et Cetera
1016	1003-1	20 in. Bicycle	5	ETC	Bikes Et Cetera
1016	1000-1	20 in. Bicycle	4	ETC	Bikes Et Cetera

**Second Normal Form (2NF)**

Although the table shown in Figure 5.6 is in 1NF, it still has problems. The table includes the following *functional dependencies*:

- Item\_Number functionally determines Item\_Name. Therefore, item names, such as “26 in. Bicycle,” are repeated several times. This data redundancy should be eliminated.
- Cust\_Code functionally determines Cust\_Name.
- The combination of SO\_Number and Item\_Number together functionally determine Item\_Name, Qty\_Ordered, Cust\_Code, and Cust\_Name.

These dependencies cause several problems, called **update anomalies**:

- **Update:** A change to the name of any item requires not one change but several. Each row in which any item, such as the 26 in. Bicycle, appears must be changed if the description is updated.
- **Inconsistent data:** Nothing is preventing an item from having several different names in different rows of the table.
- **Additions:** If a user tries to add a new inventory item to the database, a problem arises. Because the primary key to the table is the item number *and* the sales order number, a user cannot add a new inventory item to the database unless it has a sales order. This is an impossible requirement for a business, which would want to have information about inventory items stored in its database before accepting orders to sell those inventory items.
- **Deletions:** Deleting an inventory item from the database (by deleting its row) could cause the table to lose the information it has stored about all sales orders that contained that item.

These problems arise because we have an attribute, Item\_Name that is dependent on a portion of the primary key, Item\_Number, *not* on the entire key. Database designers call this problem a **partial dependency**.

A table is in **second normal form (2NF)** if it is in first normal form and has no partial dependencies; that is, no non-key attribute is dependent on only a portion of the primary key. An attribute is a **non-key attribute** if it is not part of the primary key. For instance, in Figure 5.5 (pg. 152), Item\_Name, Quantity\_Ordered, Cust\_Code, and Cust\_Name are non-key attributes.

A designer would perform two steps to get this 1NF table into 2NF. First, create a new table for each subset of the table that is partially dependent on a part of the composite primary key (that is, SO\_Number and Item\_Number). In this case, that procedure would yield two new tables, one with SO\_Number as its primary key (a SALES\_ORDERS table) and another with Item\_Number as its primary key (an INVENTORY\_ITEMS table). Second, place each of the non-key attributes that are dependent on a part of the composite primary key into the table that now has a primary key that is the field on which the non-key attribute is partially dependent. For example, the Item\_Name field is partially dependent on the Item\_Number field portion of the composite primary key, so it would be moved into the new INVENTORY\_ITEMS table. This transformation yields the three tables shown in Figure 5.7 (pg. 154).

With this set of three tables, the *update anomaly* problems mentioned earlier are resolved. Users can add inventory items without having a sales order by adding them to the table INVENTORY\_ITEMS. Because item names are stored only once, the potential for inconsistencies no longer exists, and updates to names will require only one change. Finally, users can delete inventory items from the database and not lose any sales order information.

### Third Normal Form (3NF)

Before proceeding to third normal form, we need one more definition. A **transitive dependency** exists in a table when a non-key attribute is functionally dependent on another non-key attribute (of course, the second non-key attribute will be dependent on the primary key). If you examine the table SALES\_ORDERS in Figure 5.7, you will notice that the values in Cust\_Name are functionally dependent on Cust\_Code, which is a non-key attribute. Thus, a transitive dependency exists in this table.

**FIGURE 5.7** Relations in Second Normal Form (2NF)

Shaded\_Attribute(s) = Primary Key

SALES_ORDERS		
SO_Number	Cust_Code	Cust_Name
1010	WHEEL	Wheelaway Cycle Center
1011	ETC	Bikes Et Cetera
1012	WHEEL	Wheelaway Cycle Center
1013	IBS	Inter. Bicycle Sales
1014	ETC	Bikes Et Cetera
1015	WHEEL	Wheelaway Cycle Center
1016	ETC	Bikes Et Cetera

INVENTORY_ITEMS	
Item_Number	Item_Name
1000-1	20 in. Bicycle
1001-1	26 in. Bicycle
1002-1	24 in. Bicycle
1003-1	20 in. Bicycle
1280-054	Kickstand
2010-0050	Formed Handlebar
3050-2197	Pedal
3961-1010	Tire, 26 in.
3961-1041	Tire Tube, 26 in.
3965-1050	Spoke Reflector
3970-1011	Wheel, 26 in.

SALES_ORDER	line item	INVENTORY
SO_Number	Item_Number	Qty_Ordered
1010	2010-0050	2
1010	1000-1	5
1011	1002-1	5
1011	1001-1	10
1012	1003-1	5
1012	1001-1	10
1013	1001-1	50
1014	1003-1	25
1015	1003-1	25
1016	3961-1041	5
1016	3965-1050	50
1016	1003-1	5
1016	1000-1	4

Some of the customer names—Wheelaway Cycle Center, for example—are repeated several times. This transitive dependency causes these *update anomalies*:

- **Update:** A change to the name of any customer could require not one change but several. The user would have to change each row in which any customer appears. For example, changing Wheelaway's name would require changing three rows in the SALES\_ORDERS table.
- **Inconsistent data:** Nothing in this design prevents users from entering several different names for a single customer.
- **Additions:** A new customer cannot be added to the database unless the customer already has a sales order. Good internal control dictates that an authorized customer should exist before a sales order can be created for that customer.
- **Deletions:** If a user deletes a sales order from the database, the name of a customer might be erased from the database.

These problems arise because the transitive dependency exists in the table SALES\_ORDERS. To summarize, a table is in **third normal form (3NF)** if it is in second normal form and has no transitive dependencies.

Figure 5.8 contains the 3NF tables that are the final result of the normalization process. The tables in Figure 5.8 are free of the anomalies outlined earlier. Users can add customers without sales orders. Because customer names are stored only once, each customer will have only one name, and updates to names will require only one change. Finally, users can delete customers from the database without regard to the sales order information.

**FIGURE 5.8** Relations in Third Normal Form (3NF)

Shaded\_Attribute(s) = Primary Key

SALES_ORDERS	
SO_Number	Cust_Code
1010	WHEEL
1011	ETC
1012	WHEEL
1013	IBS
1014	ETC
1015	WHEEL
1016	ETC

SALES_ORDER	line item	INVENTORY
SO_Number	Item_Number	Qty_Ordered
1010	2010-0050	2
1010	1000-1	5
1011	1002-1	5
1011	1001-1	10
1012	1003-1	5
1012	1001-1	10
1013	1001-1	50
1014	1003-1	25
1015	1003-1	25
1016	3961-1041	5
1016	3965-1050	50
1016	1003-1	5
1016	1000-1	4

CUSTOMERS	
Cust_Code	Cust_Name
ETC	Bikes Et Cetera
IBS	Inter. Bicycle Sales
RODEBYKE	Rodebyke Bic. & Mopeds
STANS	Stan's Cyclery
WHEEL	Wheelaway Cycle Center

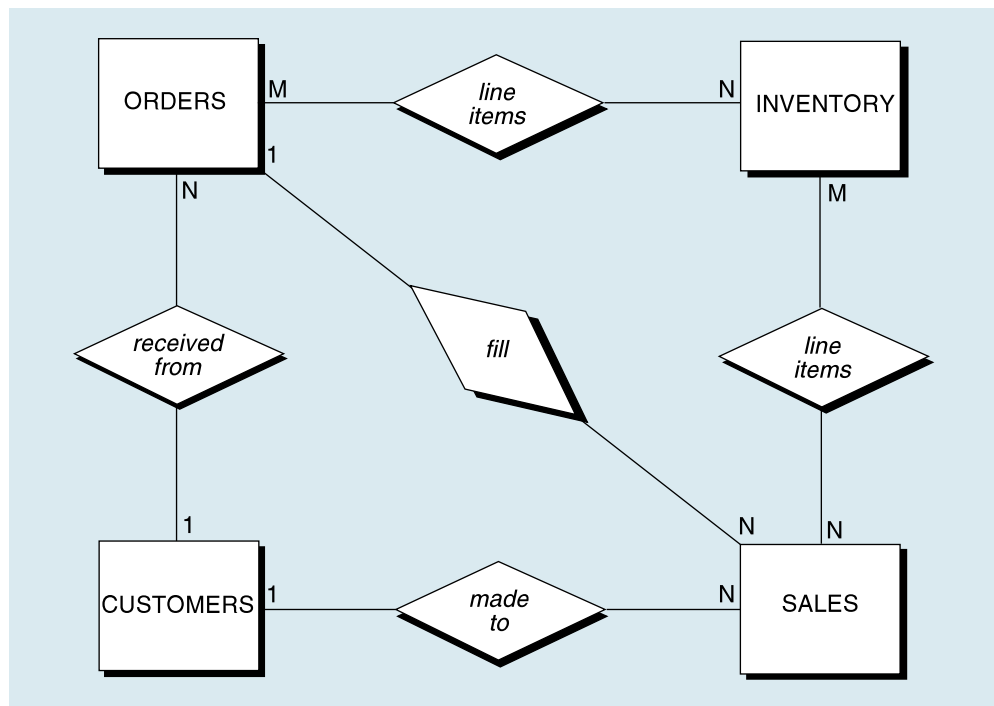
INVENTORY_ITEMS	
Item_Number	Item_Name
1000-1	20 in. Bicycle
1001-1	26 in. Bicycle
1002-1	24 in. Bicycle
1003-1	20 in. Bicycle
1280-054	Kickstand
2010-0050	Formed Handlebar
3050-2197	Pedal
3961-1010	Tire, 26 in.
3961-1041	Tire Tube, 26 in.
3965-1050	Spoke Reflector
3970-1011	Wheel, 26 in.

## Using Entity-Relationship Models

Although it is possible to create a workable database design using a bottom-up approach as previously described, most database professionals prefer to use a top-down approach, described in this section, as a first step in creating a new database. Although the bottom-up approach is useful for checking the results obtained with a top-down approach, and the bottom-up approach is easier to learn and understand, the top-down approaches usually result in a better database design.

A **data model** depicts user requirements for data stored in a database. There are a number of approaches to data modeling, any of which can be used to implement top-down database design. The most popular data modeling approach is **entity-relationship modeling**, in which the designer identifies the important things (called **entities**) about which information will be stored<sup>2</sup> and then identifies how the things are related to each other (called **relationships**). Then the designer draws a diagram of the relational model. Because this diagram includes entities and relationships, it is called an **entity-relationship model**. You will often see “entity-relationship” abbreviated as E-R. The

<sup>2</sup> Notice that this definition of entity—including events as it does—is broader than that introduced in Chapter 1.

**FIGURE 5.9** Entity-Relationship (E-R) Diagram

**E-R diagram** (also called an **entity-relationship diagram**) reflects the system's key entities and the relationships among those entities. The E-R diagram represents the data model. Database designers use standard symbols when creating E-R diagrams; however, several sets of standard symbols do exist. This book uses one of the more popular sets of symbols, but you should keep in mind that you might see E-R diagrams in other books or in your practice of accounting that differ from the diagrams used here. Figure 5.9 shows an E-R diagram in the form that we will be using in this book.

Most E-R diagrams use rectangles, connecting lines, and diamonds. The rectangles represent entities, and the connecting lines represent relationships. The diamonds are used to show the characteristics of relationships. In the E-R diagram shown in Figure 5.9, the entities include ORDERS, INVENTORY, CUSTOMERS, and SALES. The connecting lines show the five relationships in the data model. These relationships are between ORDERS and INVENTORY, ORDERS and CUSTOMERS, ORDERS and SALES, INVENTORY and SALES, and CUSTOMERS and SALES. The diamonds on the connecting lines provide some information about the relationships between the entities. For example, the relationship between ORDERS and CUSTOMERS occurs because orders are received from customers. Another database designer might have described this relationship as “place” because CUSTOMERS place ORDERS. Either description would be an acceptable way to describe this particular data model.

The first step in E-R modeling is to determine users' requirements for the database. This process is often incorporated into a process called *systems analysis* and is typically conducted by a person called a *systems analyst*. To discover user requirements, the systems analyst conducts interviews and observations and reviews existing system documentation. To document the existing system and the user's requirements for the new database, the analyst prepares narratives and diagrams, one of which is the E-R diagram.

Part of the information collected by the analyst is the structure of the data being used in the activities of the organization. This information leads the analyst through three steps. First, the analyst identifies the entities. Second, the analyst identifies the relationships between the entities and learns more about the characteristics of those relationships. Finally, the analyst uses the information about the entities and relationships (which is summarized in the E-R diagram) to create database tables and define connections among those tables.

### Identify Entities

The analyst first examines existing system documentation and talks with users to learn which entities are important to the business processes about which the database will store information. Any “thing” that is an important element in the business process can be modeled as an entity. If only one instance of a thing occurs in the process, it is not modeled as an entity. Entities become tables, and if there is only one instance of a thing, you do not need a table to store information about it. For example, if a small business has one sales manager, it does not need a table to store information about its sales managers. A larger company with several sales managers would probably want to use a table because it has multiple instances of the thing “sales manager.”

Accounting researchers have identified categories of entities that commonly occur in systems that track accounting information. These categories include resources, events, agents, and locations. **Resources** are assets (tangible or intangible) that the company owns. Resources include things such as inventory, equipment, and cash. **Events** are occurrences related to resources that are of interest to the business. Events include orders, sales, and purchases. **Agents** are people or organizations that participate in events. Agents can be part of the company, or they can be external to the company. Agents include customers, employees, and vendors. **Locations** are places or physical locations at which events occur, resources are stored, or agents participate in events. Many accounting systems researchers argue that a location owned by the company is a resource, but experts are not in complete agreement on that point. Locations can include countries, cities, stores, warehouses, and more precise categories such as bin or pallet.

Although many accountants find these categories to be useful organizing devices for thinking about entities, others simply identify entities without using these categories. The entity categories do not lead to any different treatment in the data modeling process. For example, an agent entity is modeled in an E-R diagram in the same way an event entity is.

### Identify the Relationships That Connect the Entities

Next, the analyst identifies the relationships between and among the entities. The *relationships* are shown in the E-R diagram as connecting lines with diamonds that describe the nature of the relationship. Figure 5.9 shows an E-R diagram of a simple sales and order entry database. In this figure, you can see that relationships can exist between the following:

- Two events, such as SALES *fill* ORDERS
- An agent and an event, such as ORDERS are *received from* CUSTOMERS or SALES are *made to* CUSTOMERS
- A resource and an event, such as ORDERS have *line items* INVENTORY or INVENTORY *line items* SALE

### Characteristics of Relationships

The diamond on the left side of the diagram in Figure 5.9 includes one characteristic of the relationship between ORDERS and CUSTOMERS: It shows that an order is

received from a customer. The formal notation for this relationship characteristic is “ORDERS are *received from* CUSTOMERS.”

A person reading the diagram could be confused by the diagram and want to interpret its meaning as “CUSTOMERS are *received from* ORDERS.” However, most users who are familiar with the business processes being modeled can usually interpret the diagram correctly because they know the context of the relationship.

In addition to the description of the relationship that appears in the diamond, each relationship has a characteristic, called a **cardinality** that shows the degree to which each entity participates in the relationship. A full discussion of cardinalities is complex and not really necessary for an understanding of E-R modeling. However, you should know that the discussion here is limited in scope and that you might see different notations used in practice to describe relationship cardinalities.

The **maximum cardinality** is a measure of the highest level of participation that one entity can have in another entity. In the cardinality notation used in this book, a maximum cardinality can have a value of “one” or “many.” A value of one is shown as the digit “1,” and a value of many is shown as the letter “N” or the letter “M.” For example, the letter “N” beneath the ORDERS rectangle in Figure 5.9 means that each customer may have many (more than one) orders, and the “1” above the CUSTOMERS rectangle means that each order is from only one customer. In reading this E-R diagram, you would say “the relationship between customers and orders is one-to-many.” You could also say “the relationship between orders and customers is many-to-one.”

In a relational database, a relationship between the entities is one of three types (common notation is shown in parentheses): one-to-many (1:N), many-to-many (M:N), or one-to-one (1:1). A many-to-one (N:1) relationship is the same as a one-to-many relationship but stated in reverse.

In Figure 5.9 (pg. 156), you can see that each inventory item can have many orders, and each order can have many inventory items. This relationship has a many-to-many (M:N) cardinality. The relationship between INVENTORY and SALES is also many-to-many (there can be many inventory items on a sale, and each sale can have many inventory items).

The third cardinality, one-to-one, means that an instance of an entity is related to one specific instance of another entity. This cardinality is easier to visualize using tables. Recall that each entity in the E-R model will become a table in the database. If two entities (tables) have a one-to-one relationship, each row in the first table will be related to one (and only one) row in the second table. An analyst that identifies a one-to-one relationship will, in most cases, simply combine the two tables into one.

The E-R diagram in Figure 5.9 reflects only part of an organization’s business processes. An organization would require more data than just the customers, orders, inventory, and sales shown in this diagram. It would require data related to purchases, payments, payroll, and so on. Before going on to the next step, the analyst would confirm the accuracy of the E-R diagram with the people in the organization who will use the database.

### Create Tables and Relationships

Having completed a data model that captures users’ requirements, the analyst continues the data modeling process by transforming the data model into a *logical* design for the database. This logical design takes the user requirements and converts them into a usable database. This database design includes a definition of each table in the database and how each table is related to other tables in the database.

Figure 5.10 is the logical design for our database. It is the data model in Figure 5.9 (pg. 156), implemented in a relational database. The analyst can create this logical model from the E-R diagram by following these five steps:



**FIGURE 5.10** Relational Database

Shaded\_Attribute(s) = Primary Key

CUSTOMERS				
Cust_Code	Cust_Name	Cust_City	Credit_Limit	Sales_YTD
ETC	Bikes Et Cetera	Elgin	10000.00	9561.55
IBS	Inter. Bicycle Sales	New York	5000.00	4191.18
RODEBYKE	Rodebyke Bic. & Mopeds	San Jose	2000.00	1142.50
STANS	Stan's Cyclery	Hawthorne	10000.00	8330.00
WHEEL	Wheelaway Cycle Center	Campbell	10000.00	6854.00

INVENTORY				
Item_Number	Item_Name	Qty_On_Hand	Unit_Cost	Unit_Price
1000-1	20 in. Bicycle	247	55.00	137.50
1001-1	26 in. Bicycle	103	60.00	150.00
1002-1	24 in. Bicycle	484	60.00	150.00
1003-1	20 in. Bicycle	4	24.37	60.93
1280-054	Kickstand	72	6.50	16.25
2010-0050	Formed Handlebar	90	4.47	11.25
3050-2197	Pedal	23	0.75	1.88
3961-1010	Tire, 26 in.	42	1.45	3.13
3961-1041	Tire Tube, 26 in.	19	1.25	3.13
3965-1050	Spoke Reflector	232	0.29	0.63
3970-1011	Wheel, 26 in.	211	10.50	25.00

ORDERS			
SO_Number	Cust_Code	Cust_Order_Number	SO_Date
1010	WHEEL	453	061205
1011	ETC	347	061205
1012	WHEEL	56-6	061205
1013	IBS	3422	061205
1014	ETC	778	061205
1015	WHEEL	5673	061206
1016	ETC	3345	061206

SALES			
Shipment_Number	Invoice_Number	SO_Number	Cust_Code
021207028	35	1011	ETC
021207042	36	1012	WHEEL

ORDERS line items INVENTORY			
SO_Number	Item_Number	Qty_Ordered	Sales_Price
1010	1000-1	5	137.50
1010	2010-0050	2	11.25
1011	1001-1	10	127.50
1011	1002-1	5	150.00
1012	1003-1	5	60.93
1012	1001-1	10	127.50
1013	1001-1	50	78.30
1014	1003-1	25	37.42
1015	1003-1	25	37.42
1016	1003-1	5	60.93
1016	3965-1050	50	33.00
1016	3961-1041	5	3.13
1016	1000-1	4	137.50

SALES line items INVENTORY		
Shipment_Number	Item_Number	Qty_Shipped
021207028	1001-1	8
021207028	1002-1	4
021207042	1001-1	5

1. Create a relational table for each entity. In Figure 5.10, CUSTOMERS, INVENTORY, ORDERS, and SALES are the entity tables.
2. Determine a *primary key* for each of the entity tables. The primary key must uniquely identify each row within the table, and it must contain a value in each row. A customer code or number (Cust\_Code), stock number (Item\_Number), and order number (SO\_Number) are commonly used to identify customers, items of inventory, and orders. The SALES table uses the date (YYMMDD) followed by a three-digit

- sequentially assigned serial number (the *unique* portion of the `Shipment_Number`) to represent each shipment record (row).
3. Determine the *attributes* for each of the entities. An attribute is sometimes called a *field* and is represented in a database table as a column. The accepted custom among relational database designers is to put the primary key attribute in the first column of the table. User requirements determine the other attributes. The other columns shown in these entity tables contain typical attributes, but you will see different attributes included in databases used by different organizations.
  4. Implement the relationships among the entities. This is accomplished by assuring that the primary key in one table also exists as an attribute in every table (entity) for which there is a relationship specified in the E-R diagram. Implementing the many-to-many relationships requires the creation of tables, called *relationship tables* or *junction tables*. **Relationship (junction) tables** are tables with *composite* primary keys that connect (join) tables in a many-to-many relationship. This is necessary because relational DBMSs do not have the capability to model a many-to-many relationship directly. Each many-to-many relationship must be modeled as a pair of one-to-many relationships. For example, the SALES and INVENTORY relationship (M:N) is modeled as two (1:N) relationships. One of these relationships is between SALES and the relationship table titled “SALES *line items* INVENTORY,” and the other is between INVENTORY and “SALES *line items* INVENTORY.” Relationship tables always have composite primary keys that are a combination of the primary keys of the entity tables that participate in the M:N relationship.
  5. Determine the *attributes*, if any, for each of the relationship tables. Some relationship tables only need the columns that make up their composite primary keys. Other relationship tables provide a way to store interesting information that depends on the combination of the attributes contained in their composite primary keys. The two relationship tables shown in Figure 5.10 each have additional attributes. For example, the “SALES *line items* INVENTORY” table includes the attribute `Qty_Shipped`. This attribute stores the quantity of each item on a particular sale. This value is determined jointly by the `Shipment_Number` and the `Item_Number`, which are the two attributes that make up the table’s composite primary key.

## Using DBMS and Intelligent Systems to AID Decision Makers

---

Using relational databases can help organizations track accounting information better, but a major benefit of having information stored in a database is that it is easier to access the information in new and creative ways. This type of information access can help managers make better decisions. Each day, hundreds of thousands of decisions are made in business organizations. In this section, you will learn about the information tools that can help decision makers: *decision support systems*, *executive information systems*, *group support systems*, *expert systems*, and *intelligent agents*.

As you learned earlier in this book, many decisions—particularly important decisions made by high-level management—are predominantly unstructured. Four levels of expertise can be applied to these decision situations:

- Managers can make the decision without assistance, using their expertise.
- The decision maker can be assisted by problem-solving aids such as manuals and checklists. For example, an information systems auditor will use an *internal control questionnaire* to evaluate controls to ensure a comprehensive review.

- The checklists and manuals might be automated. An automated internal control questionnaire can incorporate thousands of factors, relationships, and rules of thumb. This automated expertise can assist the auditor in arriving at a conclusion regarding the effectiveness of the controls.
- The system itself can replace the decision maker, as when an *expert system* monitors the activity in a production line and adjusts the machinery as required.

## Decision Support Systems, Executive Information Systems, and Group Support Systems

Many automated tools are available to assist or replace the decision maker. Technology Summary 5.3 (pg. 162–163) describes *decision support systems (DSS)* and *executive information systems (EIS)*, both of which assist the manager by combining current and historical facts, numerical data, and statistics—from both inside and outside the organization—and by converting these data into information useful in making the decision.

Here is a comparison that shows the differences between a DSS and an EIS. A manager could use a spreadsheet—a typical component of a DSS—to calculate variances and to compare them to variances from a previous period. This information might help the manager measure current performance against budget. With the DSS, the decision maker prepares a presentation in a format that is suitable for *him or her* for *this* decision at *this* point in time. In contrast, an EIS would have its presentation formats programmed in advance. An executive using an EIS turns on his or her computer each morning and views a screen that contains icons for EIS applications that are available. An executive who wants to examine sales trends would click the “sales trends” icon. The EIS might ask some questions, such as the period of time and the geographical area to be used in the analysis, but the EIS would follow its programmed instructions to retrieve data and present the output, most likely in the form of graphs depicting sales trends. This sales trend information might alert the executive to some problem, that is, the *intelligence* step in a decision. To determine what to do, the executive might successively request more detailed information, a process known as “drill down.”

DSS and EIS are similar in that neither tells the decision maker what to do; both simply provide views for interpreting the information. The knowledge and experience required to analyze the information, to make the judgments, and to take the actions required reside with the decision maker.

DSS and EIS help managers, who typically work alone, make decisions. **Group support systems (GSS)**, also called **group decision support systems (GDSS)**, are computer-based systems that support collaborative intellectual work such as idea generation, elaboration, analysis, synthesis, information sharing, and decision making. GSS use technology to solve the time and place dimension problems associated with group work. That is, a GSS creates a “virtual meeting” for a group. While “attending” this meeting, members of the group work toward completion of their tasks and achievement of the group’s objective(s).

**Groupware**, the software identified with GSS, focuses on such functions as e-mail, group scheduling, and document sharing. Technology Application 5.1 (pg. 164) describes PricewaterhouseCoopers’ use of distributed database technology to facilitate audit teamwork.

## Expert Systems

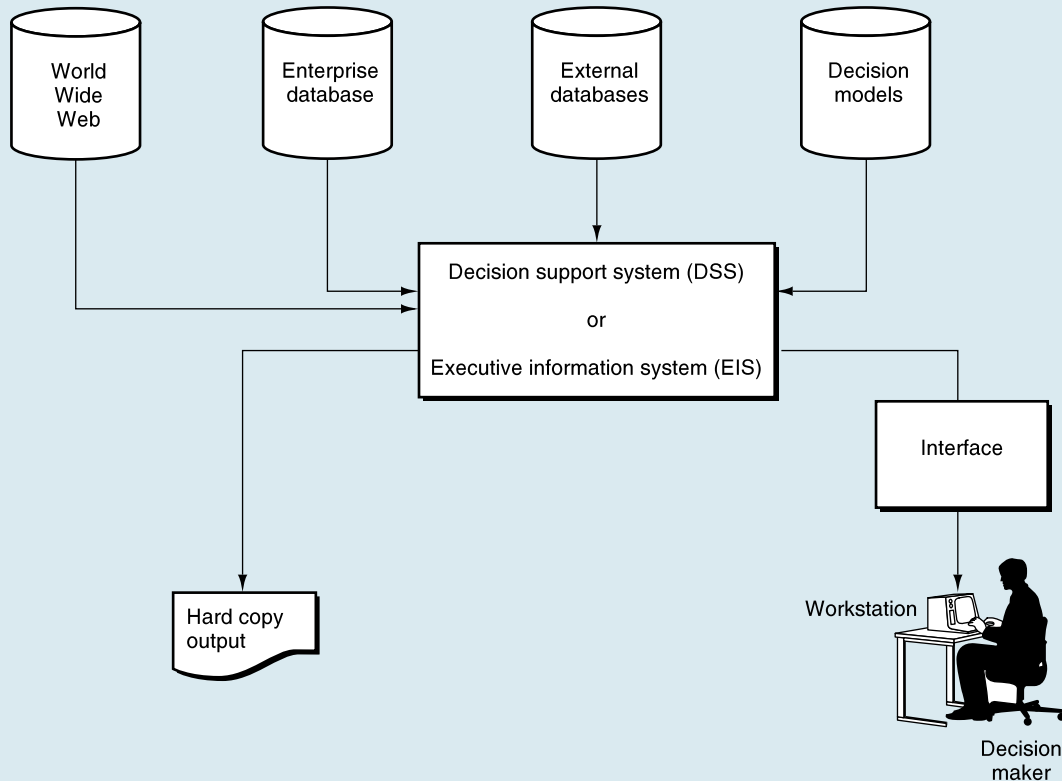
Many decision-making situations can benefit from an even higher level of support than that provided by the DSS, EIS, or GSS. Managers can use **expert systems (ES)** in these

## TECHNOLOGY SUMMARY 5.3

**DECISION SUPPORT SYSTEMS AND EXECUTIVE INFORMATION SYSTEMS**

**Decision support systems (DSS)** are information systems that assist managers with unstructured decisions by retrieving and analyzing data for purposes of identifying and generating useful information. A DSS possesses interactive capabilities, aids in answering ad hoc queries, and provides data and modeling facilities

(generally through the use of spreadsheet models and/or statistical analysis programs) to support non-recurring, relatively unstructured decision making. The main components of a DSS appear in the accompanying figure. Notice that the data made available to the decision maker include both internal data from the enterprise database, and data obtained from outside the organization, such as Dow Jones financial information.



A DSS can provide the required relevant data and can model, simulate, and perform “what-if” analysis. The DSS is superior to normal computer programs because a DSS can work on loosely defined tasks, in areas of high uncertainty, and in situations where user requirements and data are in a constant state of change.

Over the years, the term *DSS* has become synonymous with financial modeling and ad hoc querying. To support managers at the top echelon of the organization, **executive information systems (EIS)** have been developed (these also are called **executive support systems** or **ESS**). These systems, which can be considered a subset of DSS, combine information

from the organization and the environment, organize and analyze the information, and present the information to the manager in a form that assists in decision making.

Most EISs have graphical user interfaces (GUIs). The typical EIS presents output using text, graphics, and color; has multiple presentation formats; and can be tailored and customized for each executive. The complexity of these systems has greatly increased in recent years as they have expanded to include the support of crisis management and as sources of information for dealing with media questioning during such crises. This expansion in complexity has been

dovetailed with the development of data warehouses and can be used to search for data needed to answer questions during unexpected crises.

Although EIS originally were developed to support strategic-level managers and to relieve those managers of the burden of learning how to use a DSS, current EIS and ESS are virtually indistinguishable from DSS.

Rather than concentrating on the name (DSS, EIS, or ESS) or the system's nature (GUI, color, pre-programmed), it is probably more useful to distinguish these systems by their *users* and their *purpose*. For example, if a system is used exclusively by upper management, it is an executive system.

situations. Expert systems may be appropriate in situations that have the following characteristics:

- Decisions are extremely complex.
- Consistency of decision making is desirable.
- The decision maker wants to minimize time spent making the decision while *maximizing* the quality of the decision.
- Experts familiar with the knowledge and context of the decision are involved and their knowledge can be captured efficiently and modeled via computer software effectively.

Technology Application 5.2 (pg. 165) presents an example of an expert system used to assist in decision making. Companies sometimes use expert systems as a part of a downsizing strategy. In downsizing, much of an organization's collective knowledge and experience can be lost because employees with the most seniority and the highest pay are often the first to go. These employees often are exactly the people who have accumulated vast amounts of knowledge about the business and expertise in making decisions using that knowledge. The increasing complexity of the business organization and its operations along with the trend toward decentralization also prompt companies to implement expert systems. Expert systems can be used to:

- Capture and retain the expertise of the retiring employees.
- Distribute the expertise to the remaining employees.
- Distribute the expertise to the employees who cannot obtain timely access to the expert.
- Train new employees.
- Guide human experts by suggesting trends, asking questions, highlighting exceptions, and generally serving as an "electronic colleague."

The benefits derived—increased productivity, improved decision making, competitive advantage, and so on—from an expert system must exceed the costs of developing and maintaining the system. A company also must be able to identify and extract the expertise required and to enter that expertise into the expert system's knowledge base. Therefore, companies must carefully choose the areas in which they use expert systems.

**Neural networks (NN)** are computer hardware and software systems that mimic the human brain's ability to recognize patterns or predict outcomes using less-than-complete information. For example, NN are used to recognize faces, voices, and handwritten characters. NN also are used to sort good apples from bad, to detect fraudulent users of credit cards, and to manage investment funds.

Given a volume of data, an expert system makes a decision by using the knowledge it has acquired from outside experts. NN, on the other hand, derive their knowledge from the data. For example, an expert system designed to predict bankruptcy would

## TECHNOLOGY APPLICATION 5.1

## USING DISTRIBUTED DATABASE SYSTEMS TO AUTOMATE THE AUDIT AT PRICEWATERHOUSECOOPERS

At the heart of PricewaterhouseCoopers TeamMate software is a relational database that permits sorting and filtering of information by individual audit team members. The system captures, shares, and organizes audit-related information for the firm's auditors. In addition to the underlying database, TeamMate includes software modules developed at PricewaterhouseCoopers (PwC) along with spreadsheet and word processing programs. TeamMate runs on laptop computers used by PwC audit staff members.

At the start of the audit engagement, team members are given access rights to an intranet server that contains a master copy of the audit workpapers. These files include the audit programs—a listing of the work to be performed during the audit engagement. As auditors complete portions of the audit program, they update the shared set of audit workpapers, documenting the audit work performed, the conclusions reached, and any issues discovered. After the workpapers are updated, the system automatically records electronic information related to when and by whom the workpapers were updated. Through the intranet server—*regardless of their location anywhere in the world*—all audit team members have virtually instant access to the updated versions of the audit workpapers. These updated versions reflect changes made to the workpapers by *all members of the audit engagement team*. Audit partners and managers—again, regardless of their location—can review workpapers and monitor the progress of the audit. PwC reports that TeamMate has enhanced the *efficiency* and *effectiveness* of its audit

engagements. Additionally, PwC believes that TeamMate dramatically improves technical proficiency and accelerates career development of new staff. Key features of TeamMate include the following:

- TeamMate links elements of the workpapers together. For example, audit program steps are linked to the documentation of the work performed, and the work performed is linked together, regardless of its form—spreadsheet, word processing documents, images, and so on. The linking provides for complete, consistent, and accessible audit documentation.
- The status of the audit program (in progress, completed, and reviewed) is centrally reported to facilitate control by audit partners and managers. Additionally, the audit workpapers can be locked by the partners and managers to prohibit alteration after they have been reviewed.
- TeamMate records and classifies open issues (problems, recommended financial statement adjustments, potential consulting services). These are centrally maintained and reported to permit timely resolution.
- PwC's generic audit programs are stored on its intranet server. These programs are tailored for each audit engagement. Then, in each successive year, the previous audit files are automatically brought forward, updated, and used for the current year's audit.
- Client files and documents also can be downloaded or scanned to the auditor's laptop computer for analysis and inclusion in the audit workpapers.
- All relevant data are stored in the database.
- Paper documents and working papers are reduced or eliminated.

**Source:** PricewaterhouseCoopers, <http://www.pwc.com>, July 2006.

have a knowledge base that included the rules that experts have used to predict bankruptcy. A rule might be: "If the current ratio is less than X and interest has not been paid on long-term debt, then bankruptcy is likely." A neural network, on the other hand, would be given data on firms that have gone bankrupt and firms that have not (using an expert to decide which data are relevant). The neural network sifts through the data and decides how to determine whether a firm will go bankrupt. The neural network develops its own knowledge base and continues to learn as it processes

## TECHNOLOGY APPLICATION 5.2

**TAKE TWO ASPIRINS AND LOG IN IN THE MORNING?**

When you are sick, how do you know when you should see a health care provider? When feeling under the weather, do you prefer interacting with a doctor or computer? You rely on your computer for information to make decisions daily, throughout your life; it can also help you manage your health care.

The Virginia Tech, Schiffert Health Center, introduces its online Cold Self-Care program as “SELF-CARE FOR STUFFY HEAD, RUNNY NOSE, FLYING PHLEGM, SCRATCHY THROAT, & WATERY EYES SO YOU CAN FEEL BETTER AND RETURN TO CLASS (and life) WEB GUIDE!” The program asks basic questions regarding your temperature and various symptoms of your current illness. Based on your responses, you receive a recommendation, which may include calling for an appointment to see a

health care provider or taking over-the-counter medications.

How is such a system developed? It is based on rules, developed by expert health care providers.

By pooling their knowledge and experience, the providers decide, for example, at what point your temperature becomes something that can be treated by you versus a health professional. Such systems may provide you with better care by removing the “human factor” from the diagnosis (including frame of mind of the health care provider and variability between providers). Additionally, you can use such systems from home, which not only helps you but prevents you from exposing others to your condition.

The downside is that as human beings, when we are sick, we may *want* human interaction; in today’s world of technology, perhaps *instant messaging* can satisfy that need.

**Source:** “On-line Cold Clinic,” <http://www.healthcenter.vt.edu/>, June 2006.

additional data. This knowledge base includes an understanding of the patterns underlying the data and the logic necessary to reconstruct the patterns to solve future problems. Technology Application 5.3 (pg. 166) offers some examples of neural networks used in business today. The capability of NN to discover patterns in large quantities of data makes them useful in decision making and performing well in areas that are difficult for ES, DSS, or EIS.

**Intelligent Agents**

The greatest growth in intelligent systems currently underway is the development and application of *intelligent agents*. An **intelligent agent** is a software program that may be integrated into a DSS or other software tool (such as word processing, spreadsheet, or database packages).

Like any agent, an intelligent agent works on your behalf. Search engines frequently incorporate intelligent agents. For example, Google’s Froogle will search for products and prices from Internet vendors, keeping you from the time-consuming task of finding specific vendors and hunting down the products at their sites.

An intelligent agent might provide automated assistance or advice on the use of the software in to which it is embedded, identify factors that should be considered when using a system for decision making, or present a list of common responses made by other users. Most *intelligent agents* are designed to learn from the actions of the system’s user and to respond based on the user’s inputs or usage patterns.

Here is a summary of what you have read in this section regarding systems that provide intelligence-based assistance to the management decision maker:

## TECHNOLOGY APPLICATION 5.3

## USES OF NEURAL NETWORKS

These examples of NN can give you an understanding of how they operate and how useful they can be:

- At VISA, NN are used to prevent/detect fraud by comparing customer typical spending patterns with individual transactions.
- NN help manage several mutual funds, including the Standard & Poor's Neural Fair Value 25, a fund that since inception has consistently beat the Standard & Poor's 500 Stock Index.
- Real estate location and appraisal is aided by NN. The systems can review data from hundreds of houses and analyze the data in many different ways.
- NN can be used by auditors to forecast a client's earnings and expenses. By comparing the forecast to

actual results, the auditor can make a judgment as to the reasonableness of the actual results. The forecasted earnings also can indicate to the auditor if the client is likely to continue as a going concern.

- A cost accountant/consultant can use a neural network to determine optimal resource allocation and production schedules. The manipulation of the hundreds of variables and constraints has traditionally been undertaken using operations research models.

Neural networks have become so common that they are emerging as the tool of choice for fraud detection and order checking. Future applications will likely move toward more intelligent versions that will require even less user intervention.

**Source:** corporate.visa.com, August 2006; "A 'Neural' Approach to the Market", BusinessWeek Online, May 8, 2006, <http://www.businessweek.com>; for more information on these and other related topics, see the American Association of Artificial Intelligence at <http://www.aaai.org>.

- To overcome the roadblocks to quality decision making, managers use decision support systems (DSS), executive information systems (EIS), group support systems (GSS), expert systems (ES), neural networks (NN), and intelligent agents.
- A DSS structures the available data to provide information about alternative courses of action without offering a solution. DSS work well with unstructured or semi-structured problems that have a quantifiable dimension.
- An EIS uses menus, graphics, and color to provide a friendly interface to the DSS for executives who want to minimize their interaction with the system.
- A GSS facilitates group interaction and group consensus-building.
- An ES applies expertise extracted from a human expert to provide specific recommendations on problems or decisions.
- Both a DSS and an ES can assist a user in problem solving but in different ways. A DSS is a *passive tool*; it depends on the human user's knowledge and ability to provide the right data to the system's decision model. An ES is an *active* teacher or partner that can guide the user in deciding what data to enter and in providing hints about further actions that are indicated by the analysis to date.
- NN supplement the expert system in areas where expertise has not yet been captured. By examining the data, the NN can identify and replicate the patterns that exist.
- ES can automate portions of the decision-making activity. They can function independently and actually make the decision, or they can assist the decision maker and recommend a course of action. The goal of ES is not to replace people. These systems make it possible for valuable expertise to be available in multiple locations.
- Intelligent agents can be embedded in software to perform tasks for you or help you more effectively complete certain tasks.



## Knowledge Management

**Knowledge management** is the process of capturing, storing, retrieving, and distributing the knowledge of the individuals in an organization for use by others in the organization to improve the quality and efficiency of decision making across the firm. The primary enabler of *knowledge management* efforts is information technology, in particular, database technology.

Effective *knowledge management* means that an organization must be able to connect the knowledge of one individual with other individuals in the firm that need the same knowledge. This “capture and distribute” need is well served by databases. Employees can access a database to contribute knowledge or extract knowledge from anywhere in the world. Databases also provide a mechanism for orderly storage and retrieval of the captured knowledge.

### Storing Knowledge in Data Warehouses

At the heart of most knowledge-management systems is a series of interconnected databases. Two contemporary concepts that are driving many new database management systems implementations in organizations are *data warehousing* and *data mining*.

**Data warehousing** is the use of information systems facilities to focus on the collection, organization, integration, and long-term storage of entity-wide data. Its purpose is to provide users with easy access to large quantities of varied data from across the organization for the sole purpose of improving decision-making capabilities. A data warehouse is created by copying data periodically from the transaction databases into a separate database where it is stored. In this separate database, external data (non-organizational data, such as industry data or government statistics) may be included to improve the usefulness to decision makers. Managers can gain important insights by analyzing this data warehouse with multidimensional analytical tools and exploratory techniques. Such techniques, called **data mining**, are the exploration, aggregation, and analysis of large quantities of varied data from across the organization. *Data mining* is used to better understand an organization’s business processes, trends within these processes, and potential opportunities to improve the effectiveness and efficiency of the organization.

*Data warehousing* and *data mining* are dependent on the massive data integration and data independence made possible through database technology. At the same time, both may be limited in the future if well-designed database models that provide for future information needs are not effectively implemented. This starts with the information requirements analysis and successful attainment of an understanding of all users’ potential data and information needs.

## SUMMARY

In this chapter, you learned how organizations use databases to store information about business events such as sales, purchases, cash receipts, and cash disbursements. You learned how data from these events are recorded and processed in database systems.

You learned about the types of databases and the basic elements of database design that organizations use when they create databases for their accounting information, including normalization and entity-relationship data modeling. You learned that larger organizations store information in data warehouses and that managers can gain

important insights by analyzing the information in data warehouses. You learned that many companies combine their data resources with decision-support systems, executive information systems, group decision systems, and other advanced technology-based systems to improve decision making and operations.

## KEY TERMS

applications approach to business event processing	composite primary key classifying	E-R diagram
data redundancy	coding	entity-relationship diagram
database approach to business event processing	sequential coding	resources
database management system (DBMS)	serial coding	events
data independence	block coding	agents
schema	significant digit coding	locations
subschema	hierarchical coding	cardinality
query language	mnemonic coding	maximum cardinality
data manipulation language (DML)	self-checking digit code	relationship tables
hierarchical database model	normal forms	junction tables
child records	anomalies	decision support systems (DSS)
parent records	functionally dependent	executive information systems (EIS)
network database model	unnormalized table	executive support systems (ESS)
relational database model	first normal form (1NF)	group support systems (GSS)
object-oriented database model	update anomalies	group decision support systems (GDSS)
object-relational databases	partial dependency	groupware
tables	second normal form (2NF)	expert systems (ES)
queries	non-key attribute	neural networks (NN)
forms	transitive dependency	intelligent agent
reports	third normal form (3NF)	knowledge management
primary key	data model	data warehousing
	entity-relationship modeling	data mining
	entities	
	relationships	
	entity-relationship model	

## REVIEW QUESTIONS

- RQ 5-1** What is data redundancy? Explain why it is important in business information systems.
- RQ 5-2** How can storing the same facts in different computer files potentially affect the integrity of data?

- RQ 5-3 What are the most important limitations of the applications approach to business information system design?
- RQ 5-4 How are the applications and the database approaches to business event processing the same? How are they different?
- RQ 5-5 What are the main ways users can access information stored in a DBMS?
- RQ 5-6 What are the main advantages and disadvantages of using a database approach when designing and implementing business information systems?
- RQ 5-7 What are the four main elements in a relational database?
- RQ 5-8 What is a primary key? What is a composite primary key?
- RQ 5-9 What are the five primary schemes for coding data?
- RQ 5-10 What is the purpose of database normalization in a relational database?
- RQ 5-11 Explain the concept of functional dependence.
- RQ 5-12 What is a partial dependency?
- RQ 5-13 What is an entity in an entity-relationship model? How does it differ from the concept of “entity” used in creating data flow diagrams?
- RQ 5-14 What is the cardinality of a relationship in a relational database?
- RQ 5-15 Explain when a database designer might use a relationship table in constructing a relational database.
- RQ 5-16 What factors distinguish a DSS from an EIS?
- RQ 5-17 Describe the basic differences between an ES and an NN.
- RQ 5-18 What role do intelligent agents play in the operation of a decision-support system?
- RQ 5-19 Why have knowledge management systems become so important to businesses in recent years?
- RQ 5-20 Why are data warehouses important to decision makers?

## DISCUSSION QUESTIONS

- DQ 5-1 How has the technological availability and implementation of database management systems benefited decision makers in organizations?
- DQ 5-2 What is data independence? Why is it important in a comparison of the application and database approaches to storing data?
- DQ 5-3 What are the differences between a logical view and a physical view of a database? Which would be more important for accountants who are involved in the design of a database that will store business event information?
- DQ 5-4 What problems are solved by transforming a set of relational tables from second normal form to third normal form?
- DQ 5-5 “The database approach to data management is a good alternative to using enterprise systems such as ERP and CRM.” Do you agree? Discuss fully.
- DQ 5-6 Why have object-oriented databases not replaced relational databases in business information system applications?
- DQ 5-7 Demonstrate your understanding of some of the coding schemes discussed in Technology Summary 5.2 (pg. 149–150) by indicating which type of code is represented by each of the following. You should be prepared to explain and defend your answers.

- a. The student ID codes used at your college
- b. MICR codes used by the banking industry
- c. The customer codes used in Figure 5.3 (pg. 142).

**DQ 5-8** What are the comparative advantages of the various data *coding* types discussed in Technology Summary 5.2 (pg. 149–150) when applied to each of the following? Discuss fully.

- a. Employee ID numbers
- b. Customer ID numbers
- c. Vendor ID numbers
- d. The general ledger chart of accounts

## PROBLEMS

### Notes regarding Problems 5-1 through 5-5

These problems should be completed with a database software package, such as Access. For Problems 5-1 through 5-3, you may use data that you (or your instructor) have downloaded from an accounting database. Problem 5-4 provides an alternative to Problems 5-1 through 5-3 by using the database structure and sample data from Figure 5.10 (pg. 159). This problem also may be completed using the software of your choosing.

**P 5-1** Before starting this problem, you should consult the customer master data record layout in Figure 5.2 (pg. 138).

Using the database software indicated by your instructor:

- a. Create the “structure” for the records in the customer data. Use Figure 5.2 as a general guide to the data elements to be included in the customer records. However, observe the following specific requirements:
  - (1) For the customer address, use four separate fields, one each for street address, city, state, and ZIP code.
  - (2) Provide for two additional data elements that are not shown in Figure 5.2 (because they normally would be accessed from other files)—open sales orders and accounts receivable balance.
- b. If the software package supports a function to design input screens, create the screen format to be used for entering customer data.
- c. Create example customer records and enter them into the database. Use a variety of names, street addresses, states/ZIP codes, open sales order amounts, accounts receivable balances, and credit limits. (The number of records will be indicated by your instructor.)
- d. Obtain a printout of the database records.

**P 5-2** *Note:* This problem is a continuation of Problem 5-1.

- a. “Search” the database for all customers with a specific ZIP code (choose a code that is common to at least two, but not to all, of your customers). Obtain a printout of your search algorithm and a list of customers whose records met the search parameter.

- b. “Sort” the database in *descending* order of credit limit amounts. Obtain a printout of your sort algorithm and a printout of the sorted list of customers.
- c. Create a “Customer Status Report” (the report title). Observe the following specific requirements:
  - (1) Provide column headings, in left-to-right order, for customer name, credit limit, accounts receivable balance, and open orders.
  - (2) For each state, print subtotals of the accounts receivable balance and open orders columns.

P 5-3 *Note:* This problem is a continuation of Problem 5-1.

- a. Write a “program” to enter customer order amounts into the database and to have the system either warn the user if the new order places the customer over his or her credit limit or advise the user if the credit limit is not exceeded.
- b. Test the program developed in (a) by entering the amounts of customer order business event data (use a variety of order amounts and different customers, such that you test all possible combinations of variables involved in the credit-checking algorithm). (The number of order business event data will be indicated by your instructor.) Obtain hard copy evidence of the results of your testing.

P 5-4 Using the database structure and sample data in Figure 5.10 (pg. 159) as a starting point (rather than Figure 5.2, pg. 138), complete the requirements of Problems 5-1 through 5-3 (or whatever portions of those problems your instructor may indicate).

P 5-5 Use the database structure and sample data in Figure 5.10 (pg. 159) to:

- a. Combine the tables to obtain a complete record of orders and shipments. Obtain a printout of the algorithm(s) used to combine the tables and a printout of the list of these records.
- b. Write a query that selects the inventory items for which there is no order. Obtain printouts of the algorithm(s) you used and a list of the selected records.
- c. Select those orders that have not yet been shipped (that is, open orders). Obtain printouts of the algorithm(s) you used and a list of the selected records.
- d. Calculate the total value (price) of the inventory items that are on hand. Sort the items in descending order of value. Obtain printouts of the algorithm(s) you used and a list of the selected records.

P 5-6 A local accounting firm that is growing rapidly has asked for your help. The firm has four partners who are primarily responsible for developing new business. In addition to developing new business, the partners are very busy with their management tasks, so the partners need an easy way to record their new business development activities that does not take too much time or effort. The managing partner of the firm has asked you to develop a

database that will help the four partners track their new business development efforts. After talking with the managing partner, you decide that the following information needs to be included in the database:

1. Identity of the partner who is developing the new business lead, including the partner's first name, middle initial, last name, and four-digit employee ID number
2. Identity of the client or potential client for which the work would be done, including the company name, the key contact persons at the company, and the company's address (street address, city, state, and ZIP code)
3. Information about each new business lead, including the type of new business (the firm classifies its work into the following categories: audit, accounting, tax compliance, tax research, litigation support, and other consulting) and an estimate of the revenue the firm could derive from the new business
4. Information about each contact made to develop each new business lead, including the date of the contact, how much time the partner spent on the contact (in hours), and a brief summary of important points discussed

**Required:**

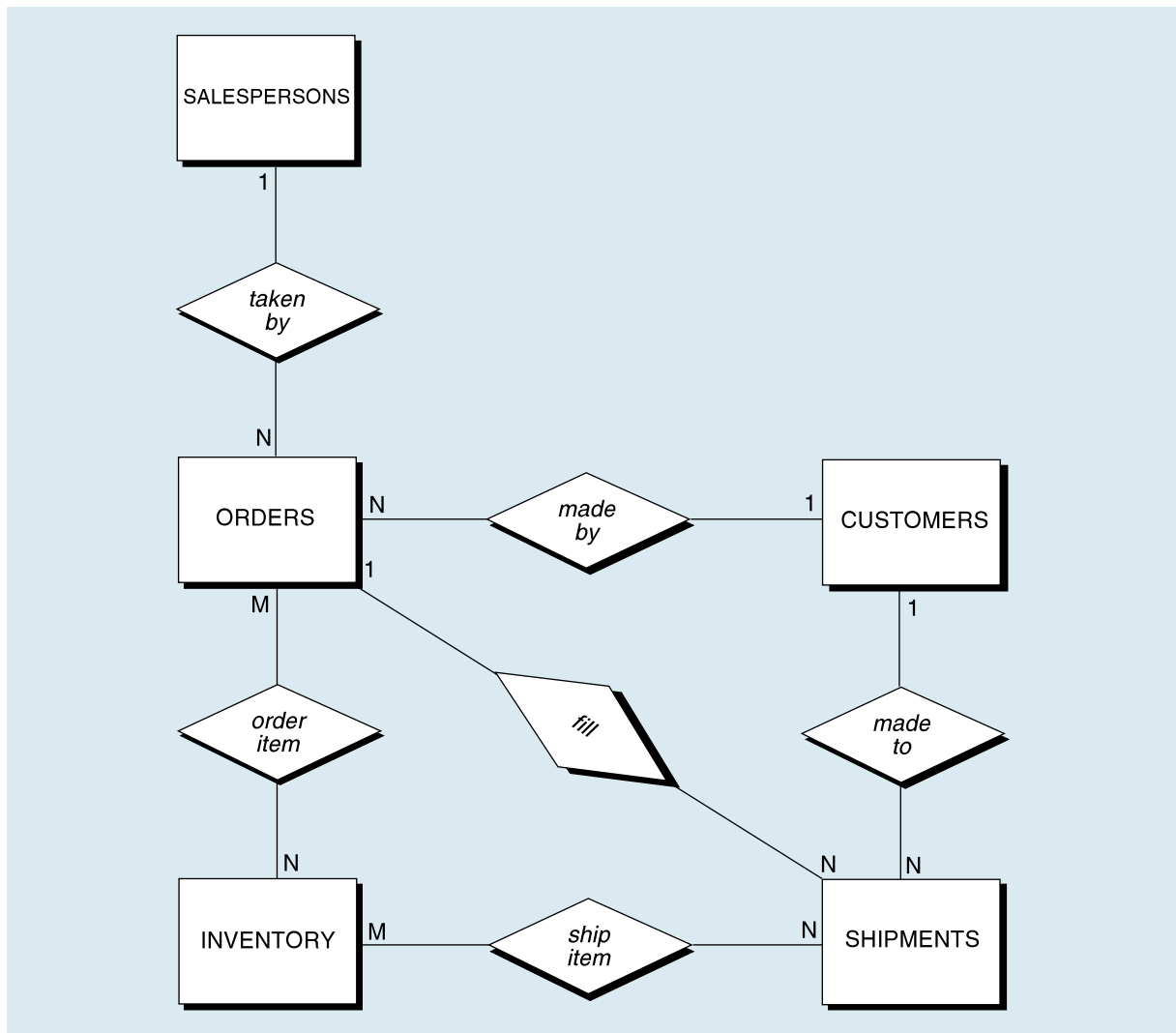
Using the bottom-up approach described in this chapter, design a set of relational database tables that will include all needed information. Be sure that the tables are in third normal form. Your instructor might have you create the tables and their relationships in a DBMS such as Microsoft Access and enter a few rows of data into each table.

P 5-7 Review the E-R diagram in Figure 5.11 and:

- a. List the resources, events, agents, and locations that are represented as entities in this diagram. (*Note:* You might not find all four types of entities in this diagram.)
- b. Write a description for each of the six relationships in the diagram. In your description, include the cardinalities. For example, you might describe the relationship between CUSTOMERS and ORDERS as: "Orders are made by customers. A customer may make many (N) orders, but each order is from only one (1) customer."

P 5-8 This problem asks you to research the literature for applications of intelligent systems. Your instructor will guide you regarding the number of pages required for each part.

- a. Develop a paper that outlines the use of expert systems in accounting and tax applications. Your paper should describe at least two expert systems and should describe the benefits and costs of the systems.
- b. Develop a paper that describes how executive information systems may be implemented (for example, using *digital dashboards*.) Include examples (including images) of the systems you find.
- c. Develop a paper that describes an accounting application of a neural network (not described in the chapter). Your paper should include at

**FIGURE 5.11** Entity-Relationship (E-R) Diagram for Problem 5-7

least one system and should include the positive and negative implications of using the application you selected.

- P 5-9** Transform the database structure that appears in Figure 5.12 (pg. 174) into third normal form (3NF). Be sure to show your intermediate steps of first and second normal form.
- P 5-10** Figure 5.13 (pg. 174) is a sample from a spreadsheet used to record donors for a small college. You have been asked to design and implement a database to allow easy input, updating, and reporting of contribution data.
- Transform the structure into third normal form (3NF) providing the intermediate steps of first and second normal form.
  - Use a DBMS to implement your structure.

**FIGURE 5.12** Unnormalized Relation for Problem 5-9

SOFTWARE							
PACKID	TAGNUM	COMPID	INSTDATE	SOFTCOST	EMPNUM	EMPNAME	LOCATION
AC01	32808	M759	9/13/06	754.95	611	Dinh, Melissa	Accounting
DB32	32808	M759	12/13/06	380.00	611	Dinh, Melissa	Accounting
	37691	B121	06/15/06	380.00	124	Alvarez, Ramon	Sales
DB33	57772	C007	05/27/06	412.77	567	Feinstein, Betty	Info Systems
WP08	37691	B121	06/15/06	227.50	124	Alvarez, Ramon	Sales
	57772	C007	05/27/06	170.24	567	Feinstein, Betty	Info Systems
WP09	59836	B221	10/30/06	35.00	124	Alvarez, Ramon	Home
	77740	M759	05/27/06	35.00	567	Feinstein, Betty	Home

**KEY:**

**PACKID = Software package identification code**

**TAGNUM = Fixed asset inventory tag number**

**COMPID = Computer model**

**INSTDATE = Date software was installed on the computer**

**SOFTCOST = Cost of the particular package installed**

**EMPNUM = Employee identification code**

**EMPNAME = Employee name**

**LOCATION = Location of the computer**

**Source:** Table and data adapted from Philip J. Pratt and Joseph J. Adamski, *Database Systems Management and Design*. 3rd ed. (Danvers, MA: Boyd & Fraser Publishing Company, 1994): 218 with permission of Boyd & Fraser Publishing Company. All rights reserved.

**FIGURE 5.13** Unnormalized Relation for Problem 5-10

GIFT						
DATE	DONOR NAME	DONOR NUMBER	RECEIPT NUMBER	FUND ID	FUND NAME	AMOUNT
12/5/2009	A. Eddy	109	1201	10	Academic Excellence	\$100
				40	Athletic Scholarships	\$100
12/17/2009	B. Lester	116	1317	99	Unrestricted Gift	\$500
12/17/2009	B. Green	102	1318	40	Athletic Scholarships	\$50
				60	Department of Accounting	\$100
				10	Academic Excellence	\$100
12/19/2009	R. Curtis	210	1411	10	Academic Excellence	\$500
				60	Department of Accounting	\$500
12/31/2009	G. Smith	221	1573	60	Department of Accounting	\$1,000
				40	Athletic Scholarships	\$25



- P 5-11 Use the Internet to research the database integration features of an ERP software package and a CRM software package. The number of pages will be indicated by your instructor.
- Learn about the ERP products of SAP and the CRM products of Oracle's Siebel. Select one specific product offered by each company and determine whether it includes its own database or must be used with an existing database. If the product requires an existing database, identify which DBMSs the product can use. Summarize your findings in a written report.
  - Discuss the advantages and disadvantages of stand-alone CRM and ERP packages versus integrated packages. Your sources should include information on CRM software packages that are stand-alone and integratable.
- P 5-12 Technology Summary 5.2 (pg. 149–150) uses examples of employee ID codes to illustrate five data coding types. Refer to those examples. Create *student* ID codes that illustrate each of the five coding schemes. Discuss the strengths and weaknesses of each example.