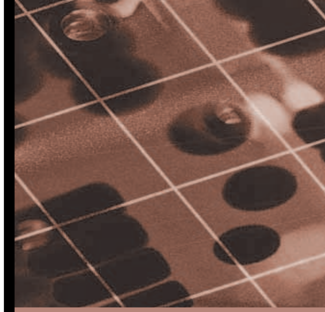


PART

I

DATABASE CONCEPTS

The Relational Revolution

Business Vignette

Today, in a world where scientists have sequenced the 3 billion base pairs that make up human DNA, we take for granted the benefits brought to us by relational databases: the ability to store, access, and change data quickly and easily on low-cost computers.

Yet until the late 1970s, databases stored large amounts of data in structures that were inflexible and difficult to navigate. Programmers needed to know what clients wanted to do with the data before the database was designed. Adding or changing the way the data were stored or analyzed was time-consuming and expensive.

In 1970, Edgar "Ted" Codd, a mathematician employed by IBM, published a groundbreaking article entitled "A Relational Model of Data for Large Shared Data Banks." At the time, nobody realized that Codd's theories would spark a technological revolution on par with the development of personal computers and the Internet. Don Chamberlin, coinventor of SQL, the most popular database query language today, explains, "There was this guy Ted Codd who had some kind of strange mathematical notation, but nobody took it very seriously."

Then Ted Codd organized a symposium, and Chamberlin listened as Codd reduced complicated five-page programs to one line. "And I said, 'Wow,' " Chamberlin recalls. The symposium convinced IBM to fund System R, a research project that built a prototype of a relational database and that would eventually lead to the creation of SQL and DB2. IBM, however, kept System R on the back burner for a number of years, which turned out to be a crucial decision, because the company had a vested interest in IMS, a reliable, high-end database system that had come out in 1968.

At about the same time System R started up, two professors from the University of California at Berkeley who had read Codd's work established a similar project called Ingres. The competition between the two tight-knit groups fueled a series of papers. Unaware of the market potential of this research, IBM allowed its staff to publish these papers publicly. Among those reading the papers was Larry Ellison, who had just founded a small company called Software Development Laboratories. Recruiting programmers from System R and Ingres and securing funding from the CIA and the Navy, Ellison was able to market the first SQL-based relational database in 1979, well before IBM. By 1983, the company had released a portable version of the database, had grossed over $5,000,000 annually, and had changed its name to Oracle.

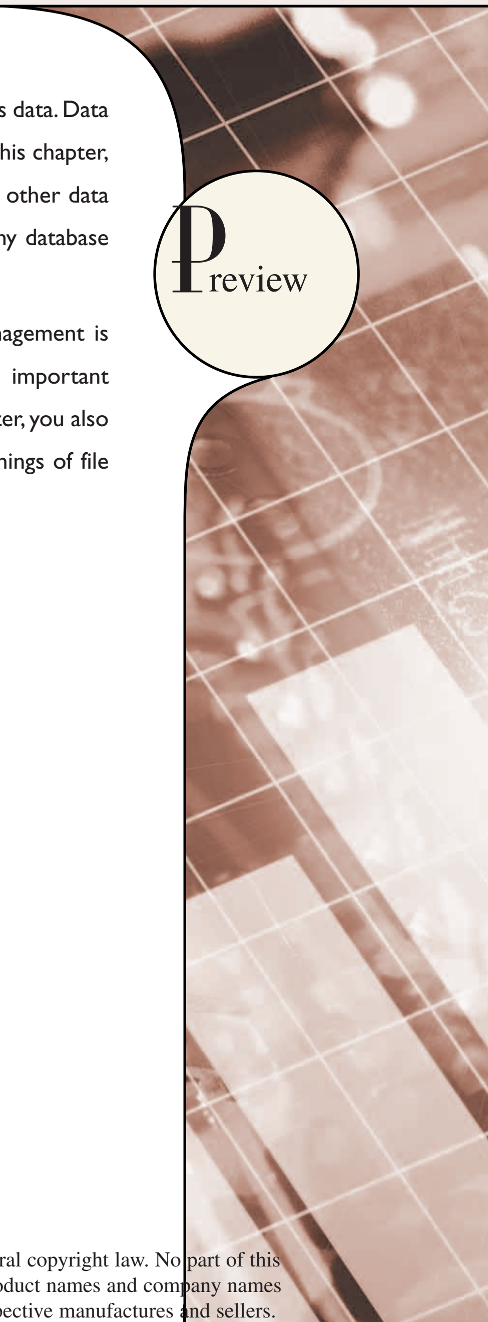
Spurred on by competition, IBM finally released SQL/DS, its first relational database, in 1980. By 2003, global sales of relational database management systems topped $7 billion.

ONE

1 DATABASE SYSTEMS

In this chapter, you will learn:

- The difference between data and information
- What a database is, what the different types of databases are, and why they are valuable assets for decision making
- The importance of database design
- How modern databases evolved from file systems
- About flaws in file system data management
- What the database system's main components are and how a database system differs from a file system
- The main functions of a database management system (DBMS)



Good decisions require good information that is derived from raw facts known as data. Data are likely to be managed most efficiently when they are stored in a database. In this chapter, you learn what a database is, what it does, and why it yields better results than other data management methods. You also learn about different types of databases and why database design is so important.

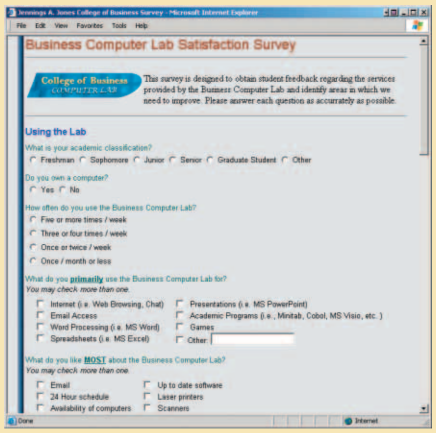
Databases evolved from computer file systems. Although file system data management is now largely outmoded, understanding the characteristics of file systems is important because they are the source of serious data management limitations. In this chapter, you also learn how the database system approach helps eliminate most of the shortcomings of file system data management.

1.1 DATA VS. INFORMATION

To understand what drives database design, you must understand the difference between data and information. **Data** are raw facts. The word *raw* indicates that the facts have not yet been processed to reveal their meaning. For example, suppose that you want to know what the users of a computer lab think of its services. Typically, you would begin by surveying users to assess the computer lab's performance. Figure 1.1, Panel A, shows the Web survey form that enables users to respond to your questions. When the survey form has been completed, the form's raw data are saved to a data repository, such as the one shown in Figure 1.1, Panel B. Although you now have the facts in hand, they are not particularly useful in this format—reading page after page of zeros and ones is not likely to provide much insight. Therefore, you transform the raw data into a data summary like the one shown in Figure 1.1, Panel C. Now it's possible to get quick answers to questions such as "What is the composition of our lab's customer base?" In this case, you can quickly determine that most of your customers are juniors (24.59%) and seniors (53.01%). Because graphics can enhance your ability to quickly extract meaning from data, you show the data summary bar graph in Figure 1.1, Panel D.

FIGURE 1.1 **Transforming raw data into information**

a) Initial Survey Screen


Jennings A. Jones College of Business Survey - Microsoft Internet Explorer
File Edit View Favorites Tools Help
College of Business COMPUTER LAB
Done
Internet

Business Computer Lab Satisfaction Survey

This survey is designed to obtain student feedback regarding the services provided by the Business Computer Lab and identify areas in which we need to improve. Please answer each question as accurrately as possible.

Using the Lab

What is your academic classification?

○ Freshman ○ Sophomore ○ Junior ○ Senior ○ Graduate Student ○ Other

Do you own a computer?

○ Yes ○ No

How often do you use the Business Computer Lab?

○ Five or more times / week
○ Three or four times / week
○ Once or twice / week
○ Once / month or less

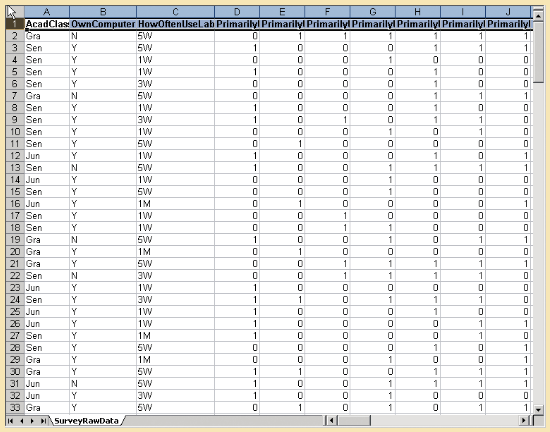
What do you primarily use the Business Computer Lab for?
You may check more than one.

☐ Internet (i.e. Web Browsing, Chat) ☐ Presentations (i.e. MS PowerPoint)
☐ Email Access ☐ Academic Programs (i.e., Minitab, Cobol, MS Visio, etc.)
☐ Word Processing (i.e. MS Word) ☐ Games
☐ Spreadsheets (i.e. MS Excel) ☐ Other:

What do you like MOST about the Business Computer Lab?
You may check more than one.

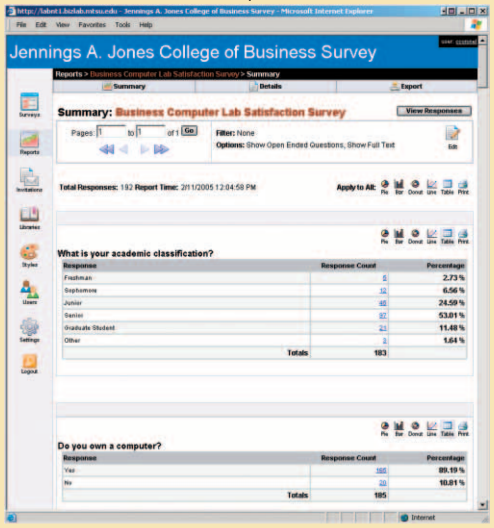
☐ Email ☐ Up to date software
☐ 24 Hour schedule ☐ Laser printers
☐ Availability of computers ☐ Scanners

b) Raw Data



	A	B	C	D	E	F	G	H	I	J
1	AcadClas	OwnComputer	HowOftenUseLab	Primarily	Primarily	Primarily	Primarily	Primarily	Primarily	Primarily
2	Gra	N	5W	0	1	1	1	1	1	1
3	Sen	Y	5W	1	0	0	0	1	1	1
4	Sen	Y	1W	0	0	0	1	0	0	0
5	Sen	Y	1W	1	0	0	0	1	0	0
6	Sen	Y	3W	0	0	0	0	1	0	0
7	Gra	N	5W	0	0	0	0	1	1	1
8	Sen	Y	1W	1	0	0	0	1	0	0
9	Sen	Y	3W	1	0	1	0	1	1	0
10	Sen	Y	1W	0	0	0	1	0	1	0
11	Sen	Y	5W	0	1	0	0	0	0	0
12	Jun	Y	1W	1	0	0	0	1	0	1
13	Sen	N	5W	1	0	0	1	1	1	1
14	Jun	Y	1W	0	0	0	1	0	0	0
15	Sen	Y	5W	0	0	0	1	0	0	0
16	Jun	Y	1M	0	1	0	0	0	0	1
17	Sen	Y	1W	0	0	1	0	0	0	0
18	Sen	Y	1W	0	0	1	1	0	0	0
19	Gra	N	5W	1	0	0	1	0	1	1
20	Gra	Y	1M	0	1	0	0	0	0	0
21	Gra	Y	5W	0	0	1	1	1	1	1
22	Sen	N	3W	0	0	1	1	1	1	0
23	Jun	Y	1W	1	0	0	0	0	0	0
24	Sen	Y	3W	1	1	0	1	1	1	0
25	Jun	Y	1W	1	0	0	0	1	0	0
26	Jun	Y	1W	1	0	0	0	0	1	1
27	Sen	Y	1M	1	0	0	0	0	1	0
28	Sen	Y	5W	0	0	0	0	1	0	1
29	Gra	Y	1M	0	0	0	1	0	0	1
30	Gra	Y	5W	1	1	0	0	1	1	1
31	Jun	N	5W	1	0	0	1	1	1	1
32	Jun	Y	3W	1	0	0	1	0	0	0
33	Gra	Y	5W	0	1	0	1	0	1	1

c) Information in Summary Format


http://labnt1.bizlab.mtsu.edu - Jennings A. Jones College of Business Survey - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Jennings A. Jones College of Business Survey
Reports > Business Computer Lab Satisfaction Survey > Summary
Summary
Details
Export
View Responses
Surveys
Reports
Invitations
Libraries
Styles
Users
Settings
Logout
Edit
Go
Apply to All:
Pie
Bar
Donut
Line
Table
Print
Internet

Summary: Business Computer Lab Satisfaction Survey

Pages: 1 to 1 of 1

Filter: None
Options: Show Open Ended Questions, Show Full Text

Total Responses: 192 Report Time: 2/11/2005 12:04:58 PM

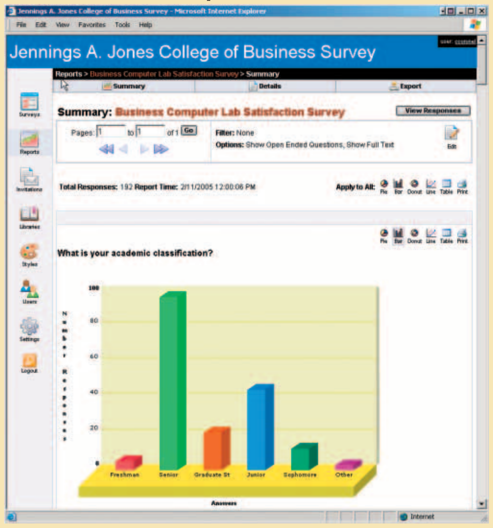
What is your academic classification?

Response	Response Count	Percentage
Freshman	5	2.73 %
Sophomore	12	6.56 %
Junior	45	24.59 %
Senior	97	53.01 %
Graduate Student	21	11.48 %
Other	3	1.64 %
Totals	183	

Do you own a computer?

Response	Response Count	Percentage
Yes	165	89.19 %
No	20	10.81 %
Totals	185	

d) Information in Graphic Format



Summary: Business Computer Lab Satisfaction Survey

Pages: 1 to 1 of 1

Filter: None
Options: Show Open Ended Questions, Show Full Text

Total Responses: 192 Report Time: 2/11/2005 12:00:06 PM

What is your academic classification?

Information is the result of processing raw data to reveal its meaning. Data processing may be as simple as organizing data to reveal patterns or as complex as making forecasts or drawing inferences using statistical modeling. Such information can then be used as the foundation for decision making. For example, the data summary for each question on the survey form can point out the lab's strengths and weaknesses, helping you to make informed decisions to better meet the needs of lab customers.

Keep in mind that raw data must be properly *formatted* for storage, processing, and presentation. For example, the student classification in Figure 1.1, Panel C is formatted to show the results based on the classifications Freshman, Sophomore, Junior, Senior, and Graduate Student. The respondents' yes/no responses may need to be converted to a Y/N format for data storage. More complex formatting is required when working with complex data types such as such as sounds, videos, or images.

In this "information age," production of accurate, relevant, and timely information is the key to good decision making. In turn, good decision making is the key to business survival in a global market. We are now said to be entering the "knowledge age."[1] Data are the foundation of information, which is the bedrock of **knowledge**—that is, the body of information and facts about a specific subject. Knowledge implies familiarity, awareness, and understanding of information as it applies to an environment. A key characteristic of knowledge is that "new" knowledge can be derived from "old" knowledge.

Let's summarize some key points:

- Data constitute the building blocks of information.
- Information is produced by processing data.
- Information is used to reveal the meaning of data.
- Accurate, relevant, and timely information is the key to good decision making.
- Good decision making is the key to organizational survival in a global environment.

Timely and useful information requires accurate data. Such data must be generated properly, and it must be stored in a format that is easy to access and process. And, like any basic resource, the data environment must be managed carefully. **Data management** is a discipline that focuses on the proper generation, storage, and retrieval of data. Given the crucial role that data plays, it should not surprise you that data management is a core activity for any business, government agency, service organization, or charity.

1.2 INTRODUCING THE DATABASE AND THE DBMS

Efficient data management typically requires the use of a computer database. A **database** is a shared, integrated computer structure that stores a collection of:

- End-user data, that is, raw facts of interest to the end user.
- **Metadata**, or data about data, through which the end-user data are integrated and managed.

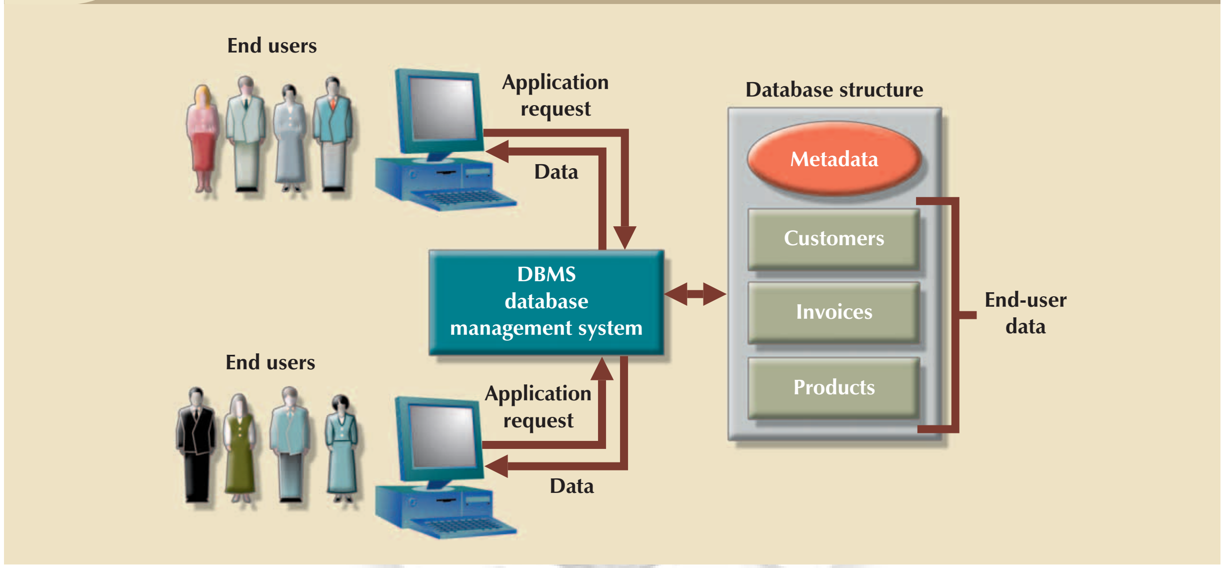
The metadata provide a description of the data characteristics and the set of relationships that link the data found within the database. In a sense, a database resembles a very well-organized electronic filing cabinet in which powerful software, known as a *database management system*, helps manage the cabinet's contents. A **database management system** (**DBMS**) is a collection of programs that manages the database structure and controls access to the data stored in the database.

[1] Peter Drucker coined the phrase "knowledge worker" in 1959 in his book *Landmarks of Tomorrow*. In 1994, Ms. Esther Dyson, Mr. George Gilder, Dr. George Keyworth, and Dr. Alvin Toffler introduced the concept of the "knowledge age."

1.2.1 Role and Advantages of the DBMS

Figure 1.2 illustrates that the DBMS serves as the intermediary between the user and the database. The DBMS receives all application requests and translates them into the complex operations required to fulfill those requests. The DBMS hides much of the database's internal complexity from the application programs and users. The application program might be written by a programmer using a programming language such as COBOL, Visual Basic, or C++, or it might be created through a DBMS utility program.

FIGURE 1.2 The DBMS manages the interaction between the end user and the database



Having a DBMS between the end user's applications and the database offers some important advantages. First, the DBMS enables the data in the database *to be shared* among multiple applications or users. Second, the DBMS *integrates* the many different users' views of the data into a single all-encompassing data repository.

Because data are the crucial raw material from which information is derived, you must have a good way of managing such data. As you will discover in this book, the DBMS helps make data management more efficient and effective. In particular, a DBMS provides advantages such as:

- *Improved data sharing*. The DBMS helps create an environment in which end users have better access to more and better-managed data. Such access makes it possible for end users to respond quickly to changes in their environment.
- *Better data integration*. Wider access to well-managed data promotes an integrated view of the organization's operations and a clearer view of the big picture. It becomes much easier to see how actions in one segment of the company affect other segments.
- *Minimized data inconsistency*. **Data inconsistency** exists when different versions of the same data appear in different places. For example, data inconsistency exists when a company's sales department stores a sales representative's name as "Bill Brown" and the company's personnel department stores that same person's name as "William G. Brown" or when the company's regional sales office shows the price of product "X" as $45.95 and its national sales office shows the same product's price as $43.95. The probability of data inconsistency is greatly reduced in a properly designed database.
- *Improved data access*. The DBMS makes it possible to produce quick answers to ad hoc queries. From a database perspective, a **query** is a specific request for data manipulation (for example, to read or update the data) issued to the DBMS. Simply put, a query is a question and an **ad hoc query** is a spur-of-the-moment question.

The DBMS sends back an answer (called the **query result set**) to the application. For example, end users, when dealing with large amounts of sales data, might want quick answers to questions (ad hoc queries) such as:
  - What was the dollar volume of sales by product during the past six months?
  - What is the sales bonus figure for each of our salespeople during the past three months?
  - How many of our customers have credit balances of $3,000 or more?
- *Improved decision making*. Better-managed data and improved data access make it possible to generate better quality information, on which better decisions are based.
- *Increased end-user productivity*. The availability of data, combined with the tools that transform data into usable information, empowers end users to make quick, informed decisions that can make the difference between success and failure in the global economy.

The advantages of using a DBMS are not limited to the few just listed. In fact, you will discover many more advantages as you learn more about the technical details of databases and their proper design.

1.2.2 Types of Databases

A DBMS can support many different types of databases. Databases can be classified according to the number of users, the database location(s), and the expected type and extent of use.

The number of users determines whether the database is classified as single-user or multiuser. A **single-user database** supports only one user at a time. In other words, if user A is using the database, users B and C must wait until user A is done. A single-user database that runs on a personal computer is called a **desktop database**. In contrast, a **multiuser database** supports multiple users at the same time. When the multiuser database supports a relatively small number of users (usually fewer than 50) or a specific department within an organization, it is called a **workgroup database**. When the database is used by the entire organization and supports many users (more than 50, usually hundreds) across many departments, the database is known as an **enterprise database**.

Location might also be used to classify the database. For example, a database that supports data located at a single site is called a **centralized database**. A database that supports data distributed across several different sites is called a **distributed database**. The extent to which a database can be distributed and the way in which such distribution is managed is addressed in detail in Chapter 12, "Distributed Database Management Systems."

The most popular way of classifying databases today, however, is based on how they will be used and on the time sensitivity of the information gathered from them. For example, transactions such as product or service sales, payments, and supply purchases reflect critical day-to-day operations. Such transactions must be recorded accurately and immediately. A database that is designed primarily to support a company's day-to-day operations is classified as an **operational database** (sometimes referred to as a **transactional** or **production database**). In contrast, a **data warehouse** focuses primarily on storing data used to generate information required to make tactical or strategic decisions. Such decisions typically require extensive "data massaging" (data manipulation) to extract information to formulate pricing decisions, sales forecasts, market positioning, etc. Most decision support data are based on historical data obtained from operational databases. Additionally, the data warehouse can store data derived from many sources. To make it easier to retrieve such data, the data warehouse structure is quite different from that of a transaction-oriented database. The design, implementation, and use of data warehouses are covered in detail in Chapter 13, "The Data Warehouse."

Table 1.1 compares features of several well-known database management systems.

TABLE 1.1 Types of Databases

PRODUCT	NUMBER OF USERS			DATA LOCATION		DATA USAGE	
	SINGLE USER	MULTIUSER		CENTRALIZED	DISTRIBUTED	OPERATIONAL	DATA WAREHOUSE
		WORKGROUP	ENTERPRISE				
MS Access	X	X		X		X	
MS SQL Server	X*	X	X	X	X	X	X
IBM DB2	X*	X	X	X	X	X	X
Oracle RDBMS	X*	X	X	X	X	X	X

* Vendor offers single-user/personal DBMS version.

NOTE

Most of the database design, implementation, and management issues addressed in this book are based on production (transaction) databases. The focus on production databases is based on two considerations. First, production databases are the databases most frequently encountered in common activities such as enrolling in a class, registering a car, buying a product, or making a bank deposit or withdrawal. Second, data warehouse databases derive most of their data from production databases, and if production databases are poorly designed, the data warehouse databases based on them will lose their reliability and value as well.

1.3 WHY DATABASE DESIGN IS IMPORTANT

Database design refers to the activities that focus on the design of the database structure that will be used to store and manage end-user data. A good database—that is, a database that meets all user requirements—does not just happen; its structure must be designed carefully. In fact, database design is such a crucial aspect of working with databases that most of this book is dedicated to the development of good database design techniques. Even a good DBMS will perform poorly with a badly designed database.

Proper database design requires the database designer to identify precisely the database's expected use. Designing a transactional database emphasizes accurate and consistent data and operational speed. The design of a data warehouse database recognizes the use of historical and aggregated data. Designing a database to be used in a centralized, single-user environment requires a different approach from that used in the design of a distributed, multiuser database. This book emphasizes the design of transactional, centralized, single-user, and multiuser databases. Chapters 12 and 13 also examine critical issues confronting the designer of distributed and data warehouse databases.

A well-designed database facilitates data management and generates accurate and valuable information. A poorly designed database is likely to become a breeding ground for difficult-to-trace errors that may lead to bad decision making—and bad decision making can lead to the failure of an organization. Database design is simply too important to be left to luck. That's why college students study database design, why organizations of all types and sizes send personnel to database design seminars, and why database design consultants often make an excellent living.

1.4 HISTORICAL ROOTS: FILES AND FILE SYSTEMS

Although file systems as a way of managing data are now largely obsolete, there are several good reasons for studying them in some detail:

- An understanding of the relatively simple characteristics of file systems makes the complexity of database design easier to understand.
- An awareness of the problems that plagued file systems can help you avoid those same pitfalls with DBMS software.
- If you intend to convert an obsolete file system to a database system, knowledge of the file system's basic limitations will be useful.

In the recent past, a manager of almost any small organization was (and sometimes still is) able to keep track of necessary data by using a manual file system. Such a file system was traditionally composed of a collection of file folders, each properly tagged and kept in a filing cabinet. Organization of the data within the file folders was determined by the data's expected use. Ideally, the contents of each file folder were logically related. For example, a file folder in a doctor's office might contain patient data, one file folder for each patient. All of the data in that file folder would describe only that particular patient's medical history. Similarly, a personnel manager might organize personnel data by category of employment (for example, clerical, technical, sales, and administrative). Therefore, a file folder labeled "Technical" would contain data pertaining to only those people whose duties were properly classified as technical.

As long as a data collection was relatively small and an organization's managers had few reporting requirements, the manual system served its role well as a data repository. However, as organizations grew and as reporting requirements became more complex, keeping track of data in a manual file system became more difficult. In fact, finding and using data in growing collections of file folders turned into such a time-consuming and cumbersome task that it became unlikely that such data could generate useful information. Consider just these few questions to which a retail business owner might want answers:

- What products sold well during the past week, month, quarter, or year?
- What is the current daily, weekly, monthly, quarterly, or yearly sales dollar volume?
- How do the current period's sales compare to those of last week, last month, or last year?
- Did the various cost categories increase, decrease, or remain stable during the past week, month, quarter, or year?
- Did sales show trends that could change the inventory requirements?

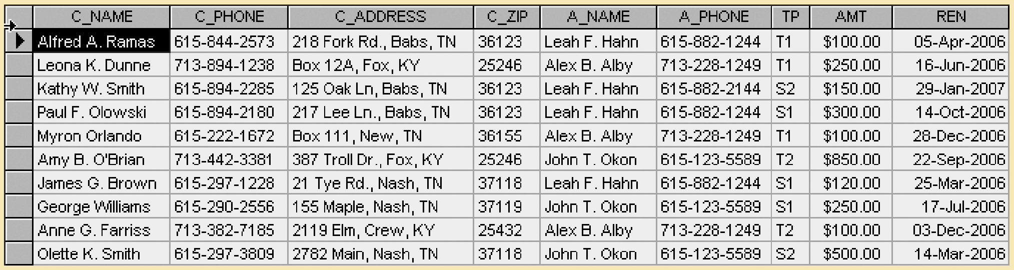
The list of questions tends to be long and to increase in number.

Unfortunately, report generation from a manual file system can be slow and cumbersome. In fact, some business managers faced government-imposed reporting requirements that required weeks of intensive effort each quarter, even when a well-designed manual system was used. Consequently, necessity called for the design of a computer-based system that would track data and produce required reports.

The conversion from a manual file system to a matching computer file system could be technically complex. (Because people are accustomed to today's relatively user-friendly computer interfaces, they have forgotten how painfully hostile computers used to be!) Consequently, a new kind of professional, known as a **data processing (DP) specialist**, had to be hired or "grown" from the current staff. The DP specialist created the necessary computer file structures, often wrote the software that managed the data within those structures, and designed the application programs that produced reports based on the file data. Thus, numerous homegrown computerized file systems were born.

Initially, the computer files within the file system were similar to the manual files. A simple example of a customer data file for a small insurance company is shown in Figure 1.3. (You will discover later that the file structure shown in Figure 1.3, although typically found in early file systems, is unsatisfactory for a database.)

FIGURE 1.3 Contents of the CUSTOMER file



C_NAME	C_PHONE	C_ADDRESS	C_ZIP	A_NAME	A_PHONE	TP	AMT	REN
Alfred A. Ramas	615-844-2573	218 Fork Rd., Babs, TN	36123	Leah F. Hahn	615-882-1244	T1	$100.00	05-Apr-2006
Leona K. Dunne	713-894-1238	Box 12A, Fox, KY	25246	Alex B. Alby	713-228-1249	T1	$250.00	16-Jun-2006
Kathy W. Smith	615-894-2285	125 Oak Ln, Babs, TN	36123	Leah F. Hahn	615-882-2144	S2	$150.00	29-Jan-2007
Paul F. Olowski	615-894-2180	217 Lee Ln., Babs, TN	36123	Leah F. Hahn	615-882-1244	S1	$300.00	14-Oct-2006
Myron Orlando	615-222-1672	Box 111, New, TN	36155	Alex B. Alby	713-228-1249	T1	$100.00	28-Dec-2006
Amy B. O'Brian	713-442-3381	387 Troll Dr., Fox, KY	25246	John T. Okon	615-123-5589	T2	$850.00	22-Sep-2006
James G. Brown	615-297-1228	21 Tye Rd., Nash, TN	37118	Leah F. Hahn	615-882-1244	S1	$120.00	25-Mar-2006
George Williams	615-290-2556	155 Maple, Nash, TN	37119	John T. Okon	615-123-5589	S1	$250.00	17-Jul-2006
Anne G. Farriss	713-382-7185	2119 Elm, Crew, KY	25432	Alex B. Alby	713-228-1249	T2	$100.00	03-Dec-2006
Olette K. Smith	615-297-3809	2782 Main, Nash, TN	37118	John T. Okon	615-123-5589	S2	$500.00	14-Mar-2006

C_NAME	**= Customer name**	**A_NAME**	**= Agent name**
C_PHONE	**= Customer phone**	**A_PHONE**	**= Agent phone**
C_ADDRESS	**= Customer address**	**TP**	**= Insurance type**
C_ZIP	**= Customer zip code**	**AMT**	**= Insurance policy amount, in thousands of $**
		REN	**= Insurance renewal date**

ONLINE CONTENT

The databases used in the chapters are available in the Student Online Companion for this book. Throughout the book, Online Content boxes highlight material related to chapter content located in the Student Online Companion.

The description of computer files requires a specialized vocabulary. Every discipline develops its own jargon to enable its practitioners to communicate clearly. The basic file vocabulary shown in Table 1.2 will help you understand subsequent discussions more easily.

TABLE 1.2 Basic File Terminology

TERM	DEFINITION
Data	"Raw" facts, such as a telephone number, a birth date, a customer name, and a year-to-date (YTD) sales value. Data have little meaning unless they have been organized in some logical manner. The smallest piece of data that can be "recognized" by the computer is a single character, such as the letter *A*, the number 5, or a symbol such as /. A single character requires 1 byte of computer storage.
Field	A character or group of characters (alphabetic or numeric) that has a specific meaning. A field is used to define and store data.
Record	A logically connected set of one or more fields that describes a person, place, or thing. For example, the fields that constitute a record for a customer named J. D. Rudd might consist of J. D. Rudd's name, address, phone number, date of birth, credit limit, and unpaid balance.
File	A collection of related records. For example, a file might contain data about vendors of ROBCOR Company, or a file might contain the records for the students currently enrolled at Gigantic University.

Using the proper file terminology given in Table 1.2, you can identify the file components shown in Figure 1.3. The CUSTOMER file shown in Figure 1.3 contains 10 records. Each record is composed of nine fields: C_NAME, C_PHONE, C_ADDRESS, C_ZIP, A_NAME, A_PHONE, TP, AMT, and REN. The 10 records are stored in a named file. Because the file in Figure 1.3 contains customer data, its filename is CUSTOMER.

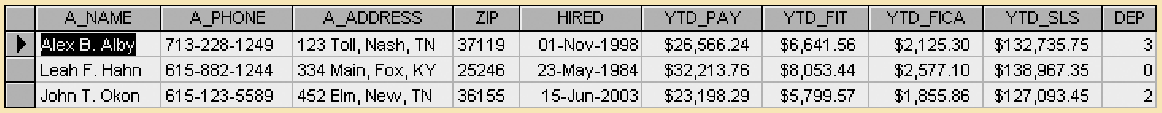
Using the CUSTOMER file's contents, the DP specialist wrote programs that produced very useful reports for the sales department:

- Monthly summaries that showed the types and amounts of insurance sold by each agent. (Such reports might be used to analyze each agent's productivity.)
- Monthly checks to determine which customers must be contacted for renewal.
- Reports that analyzed the ratios of insurance types sold by each agent.
- Periodic customer contact letters designed to summarize coverage and to provide various customer relations bonuses.

As time went on, additional programs were written to produce new reports. Although it took some time to specify the report contents and to write the programs that produced the reports, the sales department manager did not miss the old manual system—using the computer saved much time and effort. The reports were impressive, and the ability to perform complex data searches yielded the information needed to make sound decisions.

Then the sales department created a file named SALES, which helped track daily sales efforts. Additional files were created as needed to produce even more useful reports. In fact, the sales department's success was so obvious that the personnel department manager demanded access to the DP specialist to automate payroll processing and other personnel functions. Consequently, the DP specialist was asked to create the AGENT file shown in Figure 1.4. The data in the AGENT file were used to write checks, keep track of taxes paid, and summarize insurance coverage, among other tasks.

FIGURE 1.4 **Contents of the AGENT file**



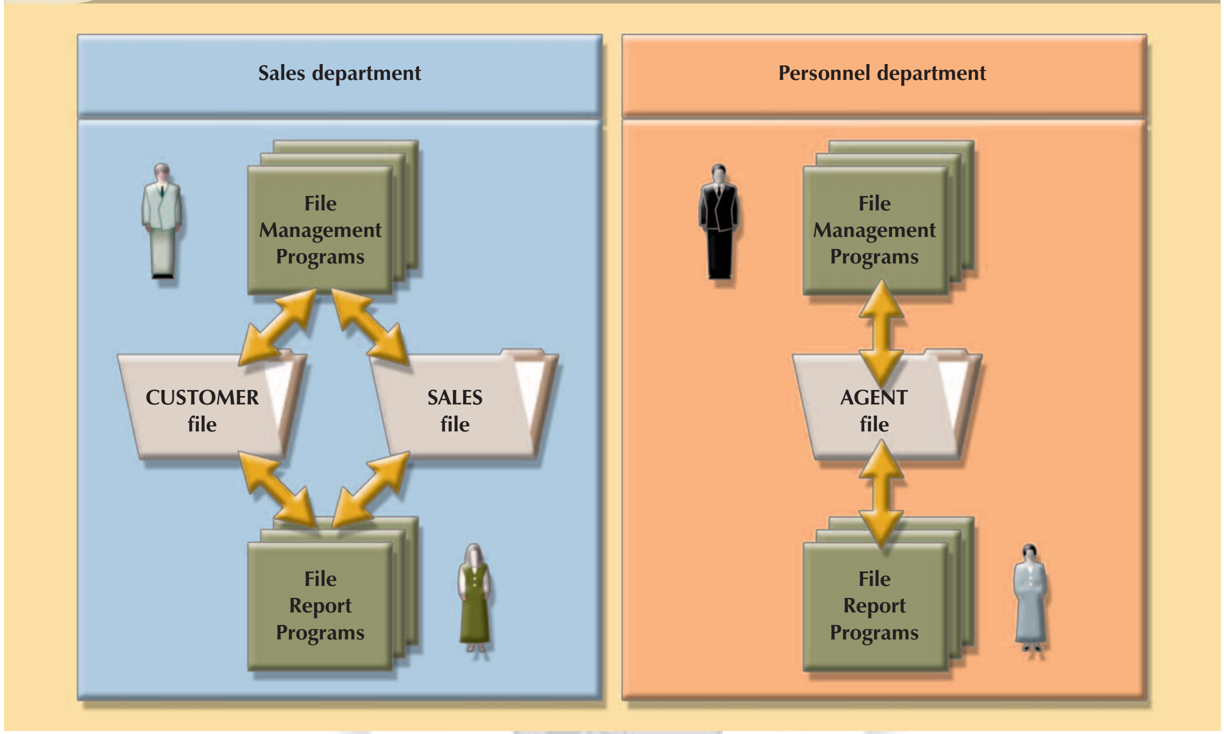
A_NAME	A_PHONE	A_ADDRESS	ZIP	HIRED	YTD_PAY	YTD_FIT	YTD_FICA	YTD_SLS	DEP
Alex B. Alby	713-228-1249	123 Toll, Nash, TN	37119	01-Nov-1998	$26,566.24	$6,641.56	$2,125.30	$132,735.75	3
Leah F. Hahn	615-882-1244	334 Main, Fox, KY	25246	23-May-1984	$32,213.76	$8,053.44	$2,577.10	$138,967.35	0
John T. Okon	615-123-5589	452 Elm, New, TN	36155	15-Jun-2003	$23,198.29	$5,799.57	$1,855.86	$127,093.45	2

A_NAME	**= Agent name**	**YTD_PAY**	**= Year-to-date pay**
A_PHONE	**= Agent phone**	**YTD_FIT**	**= Year-to-date federal income tax paid**
A_ADDRESS	**= Agent address**	**YTD_FICA**	**= Year-to-date Social Security taxes paid**
ZIP	**= Agent zip code**	**YTD_SLS**	**= Year-to-date sales**
HIRED	**= Agent date of hire**	**DEP**	**= Number of dependents**

As the number of files increased, a small file system, like the one shown in Figure 1.5, evolved. Each file in the system used its own application programs to store, retrieve, and modify data. And each file was owned by the individual or the department that commissioned its creation.

As the file system grew, the demand for the DP specialist's programming skills grew even faster, and the DP specialist was authorized to hire additional programmers. The size of the file system also required a larger, more complex computer. The new computer and the additional programming staff caused the DP specialist to spend less time programming and more time managing technical and human resources. Therefore, the DP specialist's job evolved into that of a **data processing** (**DP**) **manager**, who supervised a DP department. In spite of these organizational changes, however, the DP department's primary activity remained programming, and the DP manager inevitably spent much time as a supervising senior programmer and program troubleshooter.

FIGURE 1.5 A simple file system



1.5 PROBLEMS WITH FILE SYSTEM DATA MANAGEMENT

The file system method of organizing and managing data was a definite improvement over a manual system, and the file system served a useful purpose in data management for over two decades, a very long timespan in the computer era. Nonetheless, many problems and limitations became evident in this approach. A critique of the file system method serves two major purposes:

- Understanding the shortcomings of the file system enables you to understand the development of modern databases.
- Many of the problems are not unique to file systems. Failure to understand such problems is likely to lead to their duplication in a database environment, even though database technology makes it easy to avoid them.

The first and most glaring problem with the file system approach is that even the simplest data-retrieval task requires extensive programming in a **third-generation language** (**3GL**). A 3GL requires the programmer to specify what must be done and how it is to be done. Examples of 3GLs include Common Business-Oriented Language (COBOL), Beginner's All-purpose Symbolic Instruction Code (BASIC), C++, and FORmula TRANslation (FORTRAN). As you will learn in upcoming chapters, more modern databases use a **fourth-generation language** (**4GL**). A 4GL allows the user to specify what must be done without specifying how it must be done. Typically, 4GLs are used for data retrieval (such as query by example and report generator tools) and can work with different DBMSs. Studies have shown that a 4GL can save an average of 60 percent of lines of code compared to a typical 3GL. For example, Table 1.3 shows the code required to list customers who live in zip code 36123 in a generic 3GL vs. a 4GL. (We have used the sample data from the CUSTOMER file in Figure 1.3.)

TABLE 1.3 3GL vs. 4GL Sample Code

3GL (GENERIC CODE)	4GL (SQL CODE)
DO WHILE NOT EOF()	SELECT C_NAME, C_PHONE, C_ZIP
READ CUSTOMER	FROM CUSTOMER
IF CUSTOMER.C_ZIP = "36123"	WHERE CUSTOMER.C_ZIP = '36123';
THEN PRINT C_NAME, C_PHONE, C_ZIP;	
ENDDO;	

The need to write 3GL programs to produce even the simplest reports makes ad hoc queries impossible. Harried DP specialists and DP managers who work with mature file systems often receive numerous requests for new reports. They are often forced to say that the report will be ready "next week" or even "next month." If you need the information now, getting it next week or next month will not serve your information needs.

Another problem related to the need for extensive programming is that as the number of files in the system expands, system administration becomes more difficult. Each file must have its own file management system composed of programs that allow the user to:

- Add, delete, and modify file data.
- List the file contents and generate reports.

Making changes in an existing structure can be difficult in a file system environment. For example, changing just one field in the original CUSTOMER file requires a program that:

1. Opens the original file.
2. Reads a record from the original file.
3. Transforms the original data to conform to the new structure's storage requirements.
4. Writes the transformed data into the new file structure.
5. Deletes the original file.

Even a simple file system of only 20 files requires 5 × 20 = 100 file management programs. If each of the files is accessed by 10 different reporting programs, an additional 20 × 10 = 200 programs must be written. Because ad hoc queries are not possible, the file reporting programs can multiply quickly. And because each department in the organization "owns" its data by creating its own files, the number of files can multiply rapidly.

In fact, any file structure change, no matter how minor, forces modifications in all of the programs that use the data in that file. Modifications are likely to produce errors (bugs), and additional time can be spent using a debugging process to find those errors.

Another fault is that security features such as effective password protection, the ability to lock out parts of files or parts of the system itself, and other measures designed to safeguard data confidentiality are difficult to program and are, therefore, often omitted in a file system environment. Even when an attempt is made to improve system and data security, the security devices tend to be limited in scope and effectiveness.

To summarize the limitations of file system data management so far:

- It requires extensive programming.
- There are no ad hoc query capabilities.
- System administration can be complex and difficult.
- It is difficult to make changes to existing structures.
- Security features are likely to be inadequate.

Those limitations, in turn, lead to problems of structural and data dependency.

1.5.1 Structural and Data Dependence

A file system exhibits **structural dependence**; that is, access to a file is dependent on its structure. For example, adding a customer date-of-birth field to the CUSTOMER file shown in Figure 1.3 would require the five steps described in the previous section. Given this change, none of the previous programs will work with the new CUSTOMER file structure. Therefore, all of the file system programs must be modified to conform to the new file structure. In short, because the file system application programs are affected by change in the file structure, they exhibit structural dependence. Conversely, **structural independence** exists when it is possible to make changes in the file structure without affecting the application program's ability to access the data.

Even changes in file data characteristics, such as changing a field from integer to decimal, require changes in all programs that access the file. Because all data access programs are subject to change when any of the file's data storage characteristics change (that is, changing the data type), the file system is said to exhibit **data dependence**. Conversely, **data independence** exists when it is possible to make changes in the data storage characteristics without affecting the application program's ability to access the data.

The practical significance of data dependence is the difference between the **logical data format** (how the human being views the data) and the **physical data format** (how the computer "sees" the data). Any program that accesses a file system's file must tell the computer not only what to do, but also how to do it. Consequently, each program must contain lines that specify the opening of a specific file type, its record specification, and its field definitions. Data dependence makes the file system extremely cumbersome from a programming and data management point of view.

1.5.2 Field Definitions and Naming Conventions

At first glance, the CUSTOMER file shown in Figure 1.3 appears to have served its purpose well: requested reports usually could be generated. But suppose you want to create a customer phone directory based on the data stored in the CUSTOMER file. Storing the customer name as a single field turns out to be a liability because the directory must break up the field contents to list the last names, first names, and initials in alphabetical order. Or suppose you want to get a customer listing by area code. Including the area code in the phone number field is inefficient.

Similarly, producing a listing of customers by city is a more difficult task than is necessary. From the user's point of view, a much better (more flexible) record definition would be one that anticipates reporting requirements by breaking up fields into their component parts. Thus, the CUSTOMER file's fields might be listed as shown in Table 1.4.

TABLE 1.4 Sample Customer File Fields

FIELD	CONTENTS	SAMPLE ENTRY
CUS_LNAME	Customer last name	Ramas
CUS_FNAME	Customer first name	Alfred
CUS_INITIAL	Customer initial	A
CUS_AREACODE	Customer area code	615
CUS_PHONE	Customer phone	234-5678
CUS_ADDRESS	Customer street address or box number	123 Green Meadow Lane
CUS_CITY	Customer city	Murfreesboro
CUS_STATE	Customer state	TN
CUS_ZIP	Customer zip code	37130

Selecting proper field names is also important. For example, make sure that the field names are reasonably descriptive. In examining the file structure shown in Figure 1.3, it is not obvious that the field name REN represents the customer's insurance renewal date. Using the field name CUS_RENEW_DATE would be better for two reasons. First, the prefix CUS can be used as an indicator of the field's origin, which is the CUSTOMER file. Therefore, you know that the field in question yields a CUSTOMER property. Second, the RENEW_DATE portion of the field name is more descriptive of the field's contents. With proper naming conventions, the file structure becomes self-documenting. That is, by simply looking at the field names, you can determine which files the fields belong to and what information the fields are likely to contain.

Some software packages place restrictions on the length of field names, so it is wise to be as descriptive as possible within those restrictions. In addition, very long field names make it difficult to fit more than a few fields on a page, thus making output spacing a problem. For example, the field name CUSTOMER_INSURANCE_RENEWAL_DATE, while being self-documenting, is less desirable than CUS_RENEW_DATE.

Another problem in Figure 1.3's CUSTOMER file is the difficulty of finding desired data efficiently. The CUSTOMER file currently does not have a unique record identifier. For example, it is possible to have several customers named John B. Smith. Consequently, the addition of a CUS_ACCOUNT field that contains a unique customer account number would be appropriate.

The criticisms of field definitions and naming conventions shown in the file structure of Figure 1.3 are not unique to file systems. Because such conventions will prove to be important later, they are introduced early. You will revisit field definitions and naming conventions when you learn about database design in Chapter 4, "Entity Relationship (ER) Modeling" and in Chapter 6, "Advanced Data Modeling"; when you learn about database implementation issues in Chapter 9, "Database Design"; and when you see an actual database design implemented in Appendixes B and C ("The University Lab" design and implementation). Regardless of the data environment, the design—whether it involves a file system or a database—must always reflect the designer's documentation needs and the end user's reporting and processing requirements. Both types of needs are best served by adhering to proper field definitions and naming conventions.

ONLINE CONTENT

Appendixes A through L are available in the Student Online Companion for this book.

NOTE

No naming convention can fit all requirements for all systems. Some words or phrases are reserved for the DBMS's internal use. For example, the name ORDER generates an error in some DBMSs. Similarly, your DBMS might interpret a hyphen (-) as a command to subtract. Therefore, the field CUS-NAME would be interpreted as a command to subtract the NAME field from the CUS field. Because neither field exists, you would get an error message. On the other hand, CUS_NAME would work fine because it uses an underscore.

1.5.3 DATA REDUNDANCY

The file system's structure and lack of security make it difficult to pool data. The organizational structure promotes the storage of the same basic data in different locations. (Database professionals use the term **islands of information** for such scattered data locations.) Because it is unlikely that data stored in different locations will always be updated consistently, the islands of information often contain different versions of the same data. For example, in Figures 1.3 and 1.4, the agent names and phone numbers occur in both the CUSTOMER and the AGENT files. You need only one correct copy of the agent names and phone numbers. Having them occur in more than one place produces data redundancy. **Data redundancy** exists when the same data are stored unnecessarily at different places.

Uncontrolled data redundancy sets the stage for:

- *Data inconsistency*. Data inconsistency exists when different and conflicting versions of the same data appear in different places. For example, suppose you change an agent's phone number or address in the AGENT file.

If you forget to make corresponding changes in the CUSTOMER file, the files contain different data for the same agent. Reports will yield inconsistent results depending on which version of the data is used.

NOTE

Data that display data inconsistency are also referred to as data that lack data integrity. **Data integrity** is defined as "the condition in which all of the data in the database are consistent with the real-world events and conditions." In other words,

- Data are *accurate*—there are no data inconsistencies.
- Data are *verifiable*—the data will always yield consistent results.

Data entry errors are more likely to occur when complex entries (such as 10-digit phone numbers) are made in several different files and/or recur frequently in one or more files. In fact, the CUSTOMER file shown in Figure 1.3 contains just such an entry error: the third record in the CUSTOMER file has a transposed digit in the agent's phone number (615-882-2144 rather than 615-882-1244).

It is possible to enter a nonexistent sales agent's name and phone number into the CUSTOMER file, but customers are not likely to be impressed if the insurance agency supplies the name and phone number of an agent who does not exist. And should the personnel manager allow a nonexistent agent to accrue bonuses and benefits? In fact, a data entry error such as an incorrectly spelled name or an incorrect phone number yields the same kind of data integrity problems.

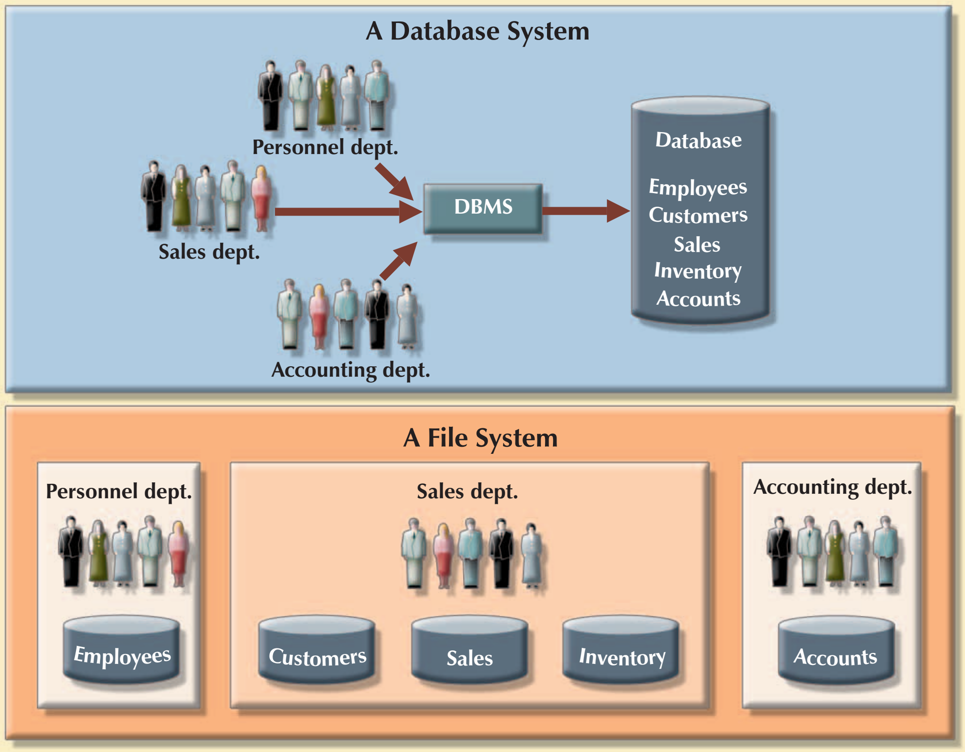
- *Data anomalies*. The dictionary defines *anomaly* as "an abnormality." Ideally, a field value change should be made in only a single place. Data redundancy, however, fosters an abnormal condition by forcing field value changes in many different locations. Look at the CUSTOMER file in Figure 1.3. If agent Leah F. Hahn decides to get married and move, the agent name, address, and phone are likely to change. Instead of making just a single name and/or phone/address change in a single file (AGENT), you also must make the change each time that agent's name, phone number, and address occur in the CUSTOMER file. You could be faced with the prospect of making hundreds of corrections, one for each of the customers served by that agent! The same problem occurs when an agent decides to quit. Each customer served by that agent must be assigned a new agent. Any change in any field value must be correctly made in many places to maintain data integrity. A **data anomaly** develops when all of the required changes in the redundant data are not made successfully. The data anomalies found in Figure 1.3 are commonly defined as follows:
  - *Update anomalies*. If agent Leah F. Hahn has a new phone number, that number must be entered in each of the CUSTOMER file records in which Ms. Hahn's phone number is shown. In this case, only three changes must be made. In a large file system, such changes might occur in hundreds or even thousands of records. Clearly, the potential for data inconsistencies is great.
  - *Insertion anomalies*. If only the CUSTOMER file existed, to add a new agent, you would also add a dummy customer data entry to reflect the new agent's addition. Again, the potential for creating data inconsistencies would be great.
  - *Deletion anomalies*. If you delete the customers Amy B. O'Brian, George Williams, and Olette K. Smith, you will also delete John T. Okon's agent data. Clearly, this is not desirable.

1.6 DATABASE SYSTEMS

The problems inherent in file systems make using a database system very desirable. Unlike the file system, with its many separate and unrelated files, the database consists of logically related data stored in a single logical data repository. (The "logical" label reflects the fact that, although the data repository appears to be a single unit to the end user, its contents may actually be physically distributed among multiple data storage facilities and/or locations.) Because the database's data repository is a single logical unit, the database represents a major change in the way end-user data are stored, accessed, and managed. The database's DBMS, shown in Figure 1.6, provides numerous advantages over file system management, shown in Figure 1.5, by making it possible to eliminate most of the file system's data inconsistency, data anomaly, data dependency, and structural dependency problems. Better yet, the current generation of DBMS software stores not only the data structures, but also the relationships between those structures and the access paths to those structures, all in a central location. The current generation of DBMS software also takes care of defining, storing, and managing all required access paths to those components.

Remember that the DBMS is just one of several crucial components of a database system. The DBMS may even be referred to as the database system's heart. However, just as it takes more than a heart to make a human being function, it takes more than a DBMS to make a database system function. In the sections that follow, you'll learn what a database system is, what its components are, and how the DBMS fits into the database system picture.

FIGURE 1.6 Contrasting database and file systems



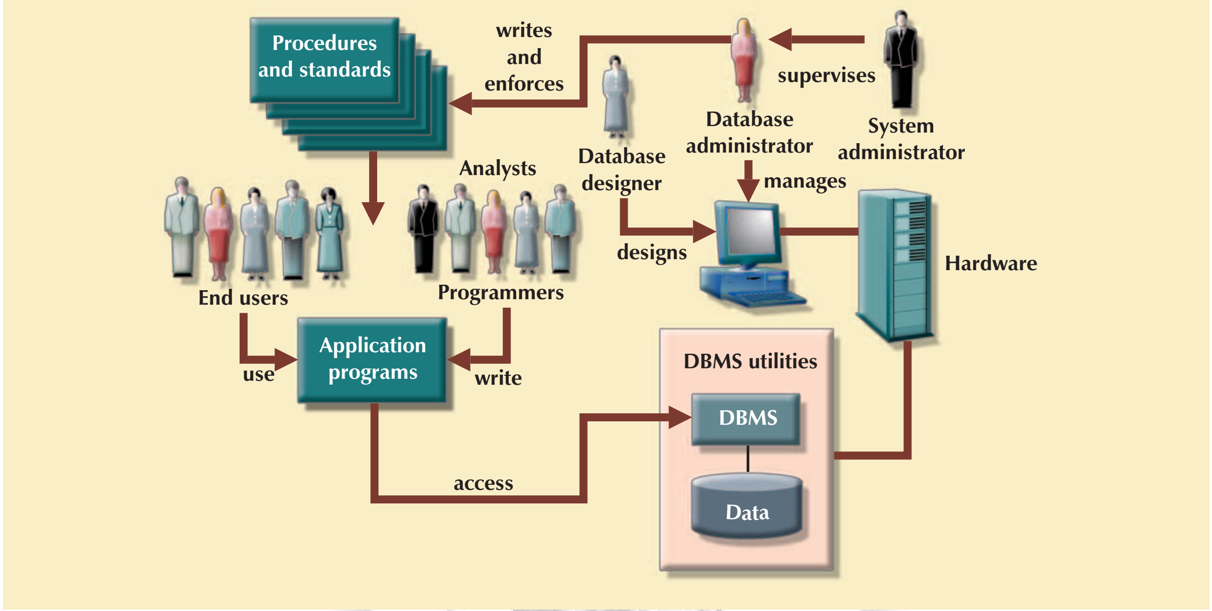
1.6.1 The Database System Environment

The term **database system** refers to an organization of components that define and regulate the collection, storage, management, and use of data within a database environment. From a general management point of view, the database system is composed of the five major parts shown in Figure 1.7: hardware, software, people, procedures, and data.

Let's take a closer look at the five components shown in Figure 1.7:

- *Hardware*. Hardware refers to all of the system's physical devices; for example, computers (microcomputers, mainframes, workstations, and servers), storage devices, printers, network devices (hubs, switches, routers, fiber optics), and other devices (automated teller machines, ID readers, etc.).

FIGURE 1.7 The database system environment



- *Software*. Although the most readily identified software is the DBMS itself, to make the database system function fully, three types of software are needed: operating system software, DBMS software, and application programs and utilities.
  - Operating system software manages all hardware components and makes it possible for all other software to run on the computers. Examples of operating system software include Microsoft Windows, Linux, Mac OS, UNIX, and MVS.
  - DBMS software manages the database within the database system. Some examples of DBMS software include Microsoft Access and SQL Server, Oracle Corporation's Oracle, and IBM's DB2.
  - Application programs and utility software are used to access and manipulate data in the DBMS and to manage the computer environment in which data access and manipulation take place. Application programs are most commonly used to access data found within the database to generate reports, tabulations, and other information to facilitate decision making. Utilities are the software tools used to help manage the database system's computer components. For example, all of the major DBMS vendors now provide graphical user interfaces (GUIs) to help create database structures, control database access, and monitor database operations.
- *People*. This component includes all users of the database system. On the basis of primary job functions, five types of users can be identified in a database system: systems administrators, database administrators, database designers, systems analysts and programmers, and end users. Each user type, described below, performs both unique and complementary functions.
  - *Systems administrators* oversee the database system's general operations.
  - *Database administrators*, also known as DBAs, manage the DBMS and ensure that the database is functioning properly. The DBA's role is sufficiently important to warrant a detailed exploration in Chapter 15, "Database Administration."
  - *Database designers* design the database structure. They are, in effect, the database architects. If the database design is poor, even the best application programmers and the most dedicated DBAs cannot produce a useful database environment. Because organizations strive to optimize their data resources, the database designer's job description has expanded to cover new dimensions and growing responsibilities.

  - *Systems analysts and programmers* design and implement the application programs. They design and create the data entry screens, reports, and procedures through which end users access and manipulate the database's data.
  - *End users* are the people who use the application programs to run the organization's daily operations. For example, salesclerks, supervisors, managers, and directors are all classified as end users. High-level end users employ the information obtained from the database to make tactical and strategic business decisions.
- *Procedures*. Procedures are the instructions and rules that govern the design and use of the database system. Procedures are a critical, although occasionally forgotten, component of the system. Procedures play an important role in a company because they enforce the standards by which business is conducted within the organization and with customers. Procedures also are used to ensure that there is an organized way to monitor and audit both the data that enter the database and the information that is generated through the use of that data.
- *Data*. The word *data* covers the collection of facts stored in the database. Because data are the raw material from which information is generated, the determination of what data are to be entered into the database and how that data are to be organized is a vital part of the database designer's job.

A database system adds a new dimension to an organization's management structure. Just how complex this managerial structure is depends on the organization's size, its functions, and its corporate culture. Therefore, database systems can be created and managed at different levels of complexity and with varying adherence to precise standards. For example, compare a local movie rental system with a national insurance claims system. The movie rental system may be managed by two people, the hardware used is probably a single microcomputer, the procedures are probably simple, and the data volume tends to be low. The national insurance claims system is likely to have at least one systems administrator, several full-time DBAs, and many designers and programmers; the hardware probably includes several mainframes at multiple locations throughout the United States; the procedures are likely to be numerous, complex, and rigorous; and the data volume tends to be high.

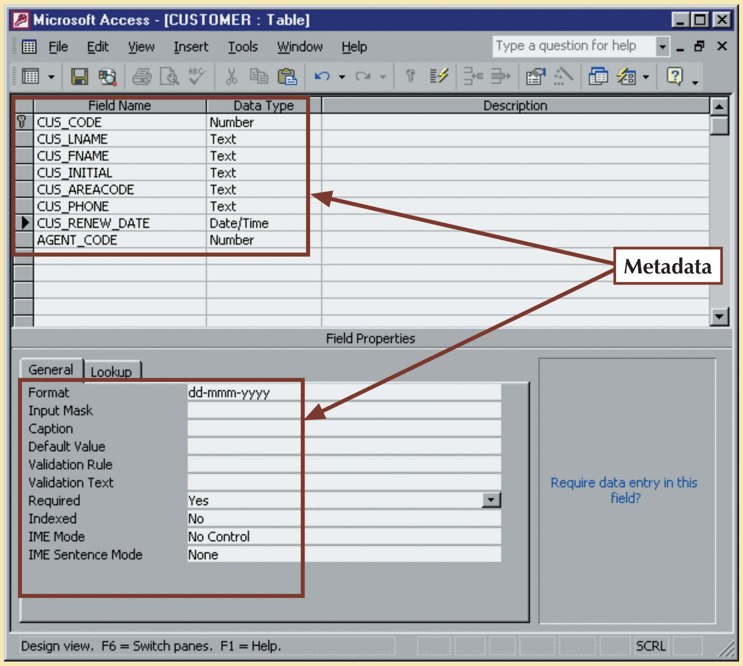
In addition to the different levels of database system complexity, managers must also take another important fact into account: database solutions must be cost-effective as well as tactically and strategically effective. Producing a million-dollar solution to a thousand-dollar problem is hardly an example of good database system selection or of good database design and management. Finally, the database technology already in use is likely to affect the selection of a database system.

1.6.2 DBMS Functions

A DBMS performs several important functions that guarantee the integrity and consistency of the data in the database. Most of those functions are transparent to end users, and most can be achieved only through the use of a DBMS. They include data dictionary management, data storage management, data transformation and presentation, security management, multiuser access control, backup and recovery management, data integrity management, database access languages and application programming interfaces, and database communication interfaces.

- *Data dictionary management*. The DBMS stores definitions of the data elements and their relationships (metadata) in a **data dictionary**. In turn, all programs that access the data in the database work through the DBMS. The DBMS uses the data dictionary to look up the required data component structures and relationships, thus relieving you from having to code such complex relationships in each program. Additionally, any changes made in a database structure are automatically recorded in the data dictionary, thereby freeing you from having to modify all of the programs that access the changed structure. In other words, the DBMS provides data abstraction and it removes structural and data dependency from the system. For example, Figure 1.8 shows an example of how Microsoft Access presents the data definition for the CUSTOMER table. Note the definition of the field properties for the CUS_RENEW_DATE.
- *Data storage management*. The DBMS creates and manages the complex structures required for data storage, thus relieving you from the difficult task of defining and programming the physical data characteristics. A modern DBMS system provides storage not only for the data, but also for related data entry forms or screen definitions, report definitions, data validation rules, procedural code, structures to handle video and picture

FIGURE 1.8 Illustrating metadata with Microsoft Access


Microsoft Access - [CUSTOMER : Table]
File Edit View Insert Tools Window Help
Type a question for help
Metadata
Field Properties
General
Lookup
Require data entry in this field?
Design view. F6 = Switch panes. F1 = Help.
SCRL

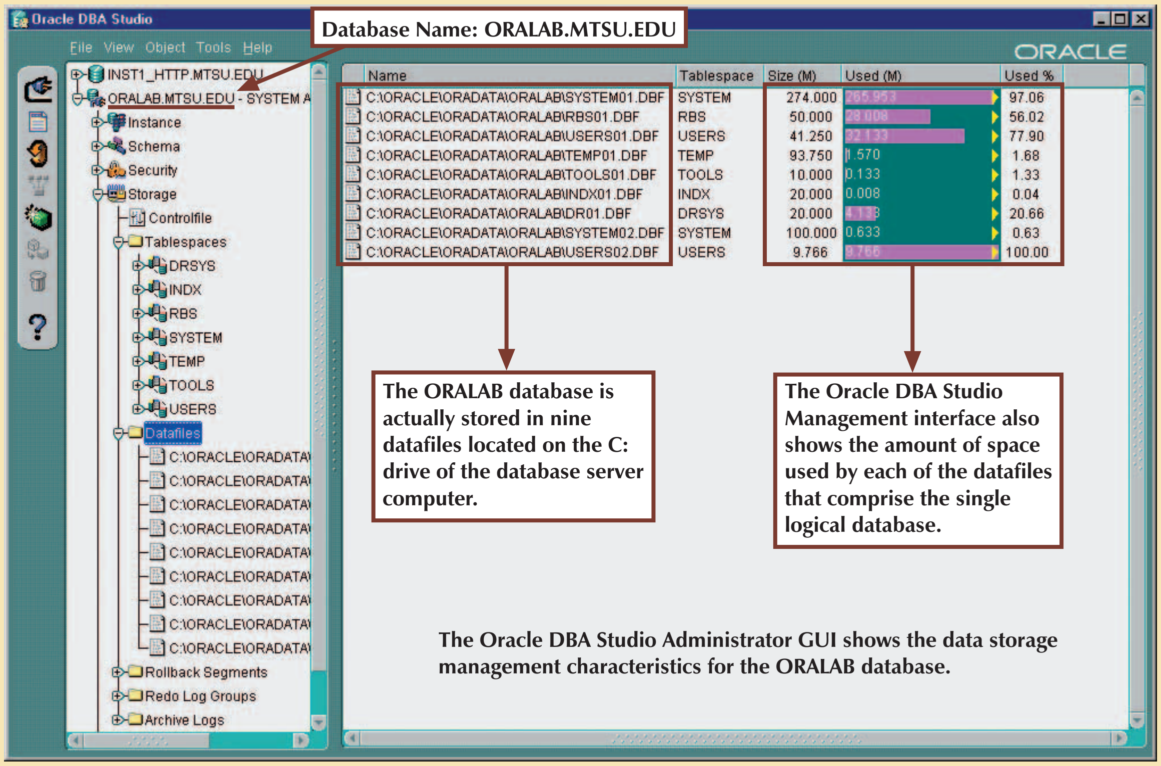
Field Name	Data Type	Description
CUS_CODE	Number	
CUS_LNAME	Text	
CUS_FNAME	Text	
CUS_INITIAL	Text	
CUS_AREACODE	Text	
CUS_PHONE	Text	
CUS_RENEW_DATE	Date/Time	
AGENT_CODE	Number	

Format	dd-mmm-yyyy
Input Mask	
Caption	
Default Value	
Validation Rule	
Validation Text	
Required	Yes
Indexed	No
IME Mode	No Control
IME Sentence Mode	None

formats, etc. Data storage management is also important for database performance tuning. **Performance tuning** relates to the activities that make the database perform more efficiently in terms of storage and access speed. Although the user sees the database as a single data storage unit, the DBMS actually stores the database in multiple physical data files. (See Figure 1.9.) Such data files may even be stored on different storage media. Therefore, the DBMS doesn't have to wait for one disk request to finish before the next one starts. In other words, the DBMS can fulfill database requests concurrently. Data storage management and performance tuning issues are addressed in Chapter 11, "Database Performance Tuning and Query Optimization."

- *Data transformation and presentation*. The DBMS transforms entered data to conform to required data structures. The DBMS relieves you of the chore of making a distinction between the logical data format and the physical data format. That is, the DBMS formats the physically retrieved data to make it conform to the user's logical expectations. For example, imagine an enterprise database used by a multinational company. An end user in England would expect to enter data such as July 11, 2006 as "11/07/2006." In contrast, the same date would be entered in the United States as "07/11/2006." Regardless of the data presentation format, the DBMS must manage the date in the proper format for each country.
- *Security management*. The DBMS creates a security system that enforces user security and data privacy. Security rules determine which users can access the database, which data items each user can access, and which data operations (read, add, delete, or modify) the user can perform. This is especially important in multiuser database systems where many users access the database simultaneously. Chapter 15, "Database Administration," examines data security and privacy issues in greater detail. All database users may be authenticated to the DBMS through a username and password or through biometric authentication such as a fingerprint scan. The DBMS uses this information to assign access privileges to various database components such as queries and reports.

FIGURE 1.9 Illustrating data storage management with Oracle



Database Name: ORALAB.MTSU.EDU

Name	Tablespace	Size (M)	Used (M)	Used %
C:\ORACLE\ORADATA\ORALAB\SYSTEM01.DBF	SYSTEM	274.000	265.953	97.06
C:\ORACLE\ORADATA\ORALAB\RBS01.DBF	RBS	50.000	28.008	56.02
C:\ORACLE\ORADATA\ORALAB\USERS01.DBF	USERS	41.250	32.133	77.90
C:\ORACLE\ORADATA\ORALAB\TEMP01.DBF	TEMP	93.750	1.570	1.68
C:\ORACLE\ORADATA\ORALAB\TOOLS01.DBF	TOOLS	10.000	0.133	1.33
C:\ORACLE\ORADATA\ORALAB\INDX01.DBF	INDX	20.000	0.008	0.04
C:\ORACLE\ORADATA\ORALAB\DR01.DBF	DRSYS	20.000	4.133	20.66
C:\ORACLE\ORADATA\ORALAB\SYSTEM02.DBF	SYSTEM	100.000	0.633	0.63
C:\ORACLE\ORADATA\ORALAB\USERS02.DBF	USERS	9.766	9.766	100.00

The ORALAB database is actually stored in nine datafiles located on the C: drive of the database server computer.

The Oracle DBA Studio Management interface also shows the amount of space used by each of the datafiles that comprise the single logical database.

The Oracle DBA Studio Administrator GUI shows the data storage management characteristics for the ORALAB database.

- *Multiuser access control*. To provide data integrity and data consistency, the DBMS uses sophisticated algorithms to ensure that multiple users can access the database concurrently without compromising the integrity of the database. Chapter 10, "Transaction Management and Concurrency Control," covers the details of the multiuser access control.
- *Backup and recovery management*. The DBMS provides backup and data recovery to ensure data safety and integrity. Current DBMS systems provide special utilities that allow the DBA to perform routine and special backup and restore procedures. Recovery management deals with the recovery of the database after a failure, such as a bad sector in the disk or a power failure. Such capability is critical to preserving the database's integrity. Chapter 15, "Database Administration," covers backup and recovery issues.
- *Data integrity management*. The DBMS promotes and enforces integrity rules, thus minimizing data redundancy and maximizing data consistency. The data relationships stored in the data dictionary are used to enforce data integrity. Ensuring data integrity is especially important in transaction-oriented database systems. Data integrity and transaction management issues are addressed in Chapter 7, "Introduction to Structured Query Language (SQL)," and Chapter 10, "Transaction Management and Concurrency Control."
- *Database access languages and application programming interfaces*. The DBMS provides data access through a query language. A **query language** is a nonprocedural language—one that lets the user specify what must be done without having to specify how it is to be done. The DBMS may also provide data access to programmers via procedural (3GL) languages such as COBOL, C, PASCAL, Visual Basic, and C++. The DBMS also provides administrative utilities used by the DBA and the database designer to create, implement, monitor, and maintain the database. **Structured Query Language** (**SQL**) is the de facto query language and data access standard supported by the majority of DBMS vendors. Chapter 7, "Introduction to Structured Query Language (SQL)," and Chapter 8, "Advanced SQL," address the use of SQL.

- *Database communication interfaces*. Current-generation DBMSs accept end-user requests via multiple, different network environments. For example, the DBMS might provide access to the database via the Internet through the use of Web browsers such as Mozilla Firefox or Microsoft Internet Explorer. In this environment, communications can be accomplished in several ways:
  - End users can generate answers to queries by filling in screen forms through their preferred Web browser.
  - The DBMS can automatically publish predefined reports on a Web site.
  - The DBMS can connect to third-party systems to distribute information via e-mail or other productivity applications.

Database communication interfaces are examined in greater detail in Chapter 12, "Distributed Database Management Systems," in Chapter 14, "Database Connectivity and Web Development," and in Appendix I, "Databases in Electronic Commerce." (Appendixes are found in the Student Online Companion.)

1.6.3 Managing the Database System: A Shift in Focus

The introduction of a database system provides a framework in which strict procedures and standards can be enforced. Consequently, the role of the human component changes from an emphasis on programming to a focus on the broader aspects of managing the organization's data resources and on the administration of the complex database software itself.

The database system makes it possible to tackle far more sophisticated uses of the data resources as long as the database is designed to make use of that available power. The kinds of data structures created within the database and the extent of the relationships among them play a powerful role in determining the effectiveness of the database system.

Although the database system yields considerable advantages over previous data management approaches, database systems do impose significant costs. For example:

- *Increased costs*. Database systems require sophisticated hardware and software and highly skilled personnel. The cost of maintaining the hardware, software, and personnel required to operate and manage a database system can be substantial.
- *Management complexity*. Database systems interface with many different technologies and have a significant impact on a company's resources and culture. The changes introduced by the adoption of a database system must be properly managed to ensure that they help advance the company's objectives. Given the fact that databases systems hold crucial company data that are accessed from multiple sources, security issues must be assessed constantly.
- *Maintaining currency*. To maximize the efficiency of the database system, you must keep your system current. Therefore, you must perform frequent updates and apply the latest patches and security measures to all components. Because database technology advances rapidly, personnel training costs tend to be significant.
- *Vendor dependence*. Given the heavy investment in technology and personnel training, companies may be reluctant to change database vendors. As a consequence, vendors are less likely to offer pricing point advantages to existing customers and those customers may be limited in their choice of database system components.

SUMMARY

- Data are raw facts. Information is the result of processing data to reveal its meaning. Accurate, relevant, and timely information is the key to good decision making, and good decision making is the key to organizational survival in a global environment.
- Data are usually stored in a database. To implement a database and to manage its contents, you need a database management system (DBMS). The DBMS serves as the intermediary between the user and the database. The database contains the data you have collected and "data about data," known as metadata.
- Database design defines the database structure. A well-designed database facilitates data management and generates accurate and valuable information. A poorly designed database can lead to bad decision making, and bad decision making can lead to the failure of an organization.
- Databases evolved from manual and then computerized file systems. In a file system, data are stored in independent files, each requiring its own data management programs. Although this method of data management is largely outmoded, understanding its characteristics makes database design easier to understand. Awareness of the problems of file systems can help you avoid similar problems with DBMSs.
- Some limitations of file system data management are that it requires extensive programming, system administration can be complex and difficult, making changes to existing structures is difficult, and security features are likely to be inadequate. Also, independent files tend to contain redundant data, leading to problems of structural and data dependency.
- Database management systems were developed to address the file system's inherent weaknesses. Rather than depositing data within independent files, a DBMS presents the database to the end user as a single data repository. This arrangement promotes data sharing, thus eliminating the potential problem of islands of information. In addition, the DBMS enforces data integrity, eliminates redundancy, and promotes data security.

KEY TERMS

ad hoc query, 7
centralized database, 8
data, 5
data anomaly, 17
data dependence, 15
data dictionary, 20
data inconsistency, 7
data independence, 15
data integrity, 17
data management, 6
data processing (DP) manager, 12
data processing (DP) specialist, 10
data redundancy, 16
data warehouse, 8
database, 6
database design, 9
database management system (DBMS), 6
database system, 18
database system, 6
desktop database, 8
distributed database, 8
enterprise database, 8
field, 11
file, 11
fourth-generation language (4GL), 13
information, 6
islands of information, 16
knowledge, 6
logical data format, 15
metadata, 6
multiuser database, 8
operational database, 8
performance tuning, 21
physical data format, 15
production database, 8
query, 7
query language, 22
query result set, 8
record, 11
single-user database, 8
structural dependence, 15
structural independence, 15
Structured Query Language (SQL), 22
third-generation language (3GL), 13
transactional database, 8
workgroup database, 8

ONLINE CONTENT

Answers to selected Review Questions and Problems for this chapter are contained in the Student Online Companion for this book.

REVIEW QUESTIONS

1. Discuss each of the following terms:
   a. data
   b. field
   c. record
   d. file
2. What is data redundancy, and which characteristics of the file system can lead to it?
3. Discuss the lack of data independence in file systems.
4. What is a DBMS, and what are its functions?
5. What is structural independence, and why is it important?
6. Explain the difference between data and information.
7. What is the role of a DBMS, and what are its advantages?
8. List and describe the different types of databases.
9. What are the main components of a database system?
10. What is metadata?
11. Explain why database design is important.
12. What are the potential costs of implementing a database system?

PROBLEMS

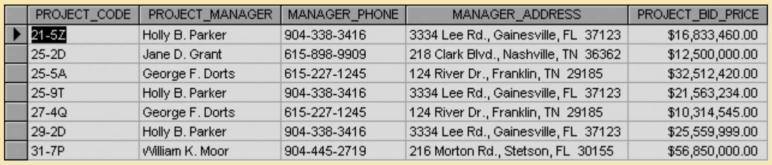
ONLINE CONTENT

The file structures you see in this problem set are simulated in a Microsoft Access database named **Ch01_ Problems**, available in the Student Online Companion for this book.

Given the file structure shown in Figure P1.1, answer Problems 1–4.

1. How many records does the file contain, and how many fields are there per record?
2. What problem would you encounter if you wanted to produce a listing by city? How would you solve this problem by altering the file structure?

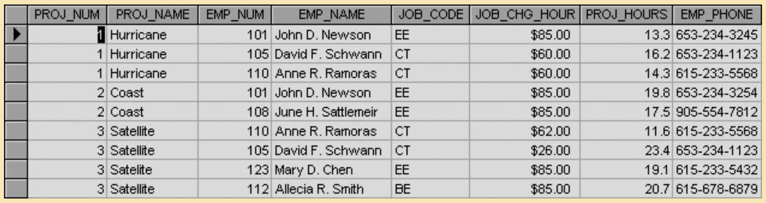
FIGURE P1.1 The file structure for Problems 1–4



PROJECT_CODE	PROJECT_MANAGER	MANAGER_PHONE	MANAGER_ADDRESS	PROJECT_BID_PRICE
21-5Z	Holly B. Parker	904-338-3416	3334 Lee Rd., Gainesville, FL 37123	$16,833,460.00
25-2D	Jane D. Grant	615-898-9909	218 Clark Blvd., Nashville, TN 36362	$12,500,000.00
25-5A	George F. Dorts	615-227-1245	124 River Dr., Franklin, TN 29185	$32,512,420.00
25-9T	Holly B. Parker	904-338-3416	3334 Lee Rd., Gainesville, FL 37123	$21,563,234.00
27-4Q	George F. Dorts	615-227-1245	124 River Dr., Franklin, TN 29185	$10,314,545.00
29-2D	Holly B. Parker	904-338-3416	3334 Lee Rd., Gainesville, FL 37123	$25,559,999.00
31-7P	William K. Moor	904-445-2719	216 Morton Rd., Stetson, FL 30155	$56,850,000.00

3. If you wanted to produce a listing of the file contents by last name, area code, city, state, or zip code, how would you alter the file structure?
4. What data redundancies do you detect, and how could those redundancies lead to anomalies?

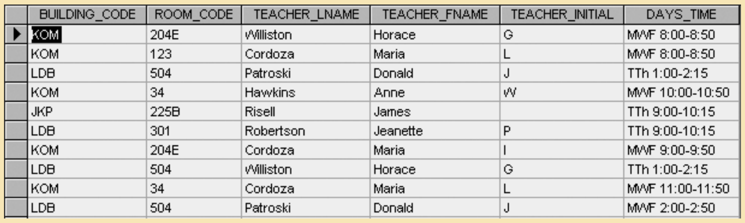
FIGURE P1.5 The file structure for Problems 5–8



PROJ_NUM	PROJ_NAME	EMP_NUM	EMP_NAME	JOB_CODE	JOB_CHG_HOUR	PROJ_HOURS	EMP_PHONE
1	Hurricane	101	John D. Newson	EE	$85.00	13.3	653-234-3245
1	Hurricane	105	David F. Schwann	CT	$60.00	16.2	653-234-1123
1	Hurricane	110	Anne R. Ramoras	CT	$60.00	14.3	615-233-5568
2	Coast	101	John D. Newson	EE	$85.00	19.8	653-234-3254
2	Coast	108	June H. Sattlemeir	EE	$85.00	17.5	905-554-7812
3	Satellite	110	Anne R. Ramoras	CT	$62.00	11.6	615-233-5568
3	Satellite	105	David F. Schwann	CT	$26.00	23.4	653-234-1123
3	Satelite	123	Mary D. Chen	EE	$85.00	19.1	615-233-5432
3	Satellite	112	Allecia R. Smith	BE	$85.00	20.7	615-678-6879

5. Identify and discuss the serious data redundancy problems exhibited by the file structure shown in Figure P1.5.
6. Looking at the EMP_NAME and EMP_PHONE contents in Figure P1.5, what change(s) would you recommend?
7. Identify the different data sources in the file you examined in Problem 5.
8. Given your answer to Problem 7, what new files should you create to help eliminate the data redundancies found in the file shown in Figure P1.5?

FIGURE P1.9 The file structure for Problems 9–10



BUILDING_CODE	ROOM_CODE	TEACHER_LNAME	TEACHER_FNAME	TEACHER_INITIAL	DAYS_TIME
KOM	204E	Williston	Horace	G	MWF 8:00-8:50
KOM	123	Cordoza	Maria	L	MWF 8:00-8:50
LDB	504	Patroski	Donald	J	TTh 1:00-2:15
KOM	34	Hawkins	Anne	W	MWF 10:00-10:50
JKP	225B	Risell	James		TTh 9:00-10:15
LDB	301	Robertson	Jeanette	P	TTh 9:00-10:15
KOM	204E	Cordoza	Maria	I	MWF 9:00-9:50
LDB	504	Williston	Horace	G	TTh 1:00-2:15
KOM	34	Cordoza	Maria	L	MWF 11:00-11:50
LDB	504	Patroski	Donald	J	MWF 2:00-2:50

9. Identify and discuss the serious data redundancy problems exhibited by the file structure shown in Figure P1.9. (The file is meant to be used as a teacher class assignment schedule. One of the many problems with data redundancy is the likely occurrence of data inconsistencies—two different initials have been entered for the teacher named Maria Cordoza.)
10. Given the file structure shown in Figure P1.9, what problem(s) might you encounter if building KOM were deleted?