CHAPTER

1

The Database Environment and Development Process

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

Learning Objectives

After studying this chapter, you should be able to:

- Concisely define each of the following key terms: **data, database, database management system, data model, information, metadata, enterprise data model, entity, relational database, enterprise resource planning (ERP) system, database application, data warehouse, data independence, repository, user view, enterprise data modeling, systems development life cycle (SDLC), prototyping, agile software development, computer-aided software engineering (CASE), conceptual schema, logical schema,** and **physical schema.**
- Name several limitations of conventional file processing systems.
- Explain at least ten advantages of the database approach, compared to traditional file processing.
- Identify several costs and risks of the database approach.
- List and briefly describe nine components of a typical database environment.
- Identify four categories of applications that use databases and their key characteristics.
- Describe the life cycle of a systems development project, with an emphasis on the purpose of database analysis, design, and implementation activities.
- Explain the prototyping and agile-development approaches to database and application development.
- Explain the roles of individuals who design, implement, use, and administer databases.
- Explain the differences between external, conceptual, and internal schemas and the reasons for the three-schema architecture for databases.

DATA MATTER!

The world has become a very complex place. The advantage goes to people and organizations that collect, manage, and interpret information effectively. To make our point, let's visit Continental Airlines. A little over a decade ago, Continental was in real trouble, ranking at the bottom of U.S. airlines in on-time performance, mishandled baggage, customer complaints, and overbooking. Speculation was that Continental would have to file for bankruptcy for the third time. In the past

10 years, Continental had had 10 CEOs. Could more effective collection, management, and interpretation of Continental's data and information help Continental's situation? The answer is a definite yes. Today Continental is one of the most respected global airlines and has been named the Most Admired Global Airline on *Fortune* magazine's list of Most Admired Global Companies annually since 2004. It was recognized as Best Airline Based in North America and the airline with the Best Airline Finance Deal by the 2008 OAG Airline of the Year awards.

Continental's former chairman of the board and CEO, Larry Kellner, points to the use of real-time business intelligence as a significant factor in Continental's turnaround. How? Implementation of a real-time or "active" data warehouse has supported the company's business strategy, dramatically improving customer service and operations, creating cost savings, and generating revenue. Fifteen years ago, Continental could not even track a customer's travel itinerary if more than one stop was involved. Now, employees who deal with travelers know if a high-value customer is currently experiencing a delay in a trip, where and when the customer will arrive at the airport, and the gate where the customer must go to make the next airline connection. High-value customers receive letters of apology if they experience travel delays on Continental and sometimes a trial membership in the President's Club.

Following is a list of some of the wins that came from integrating revenue, flight schedule, customer, inventory, and security data as part of the data warehousing project:

1. Better optimization of airfares using mathematical programming models that are able to adjust the number of seats sold at a particular fare using real-time sales data
2. Improvement of customer relationship management focused on Continental's most profitable customers
3. Immediate availability of customer profiles to sales personnel, marketing managers, and flight personnel, such as ticket agents and flight attendants
4. Support for union negotiations, including analysis of pilot staffing that allows management and union negotiators to evaluate the appropriateness of work assignment decisions
5. Development of fraud profiles that can be run against the data to identify transactions that appear to fit one of over 100 fraud profiles

To emphasize this last win, Continental's ability to meet Homeland Security requirements has been greatly aided by the real-time data warehouse. During the period immediately following the terrorist attacks of September 11, 2001, Continental was able to work with the FBI to determine whether any terrorists on the FBI watch list were attempting to board Continental flights. The data warehouse's ability to identify fraudulent activity and monitor passengers contributes significantly to Continental's goal of keeping all its passengers and crew members safe (Anderson-Lehman et al., 2004).

Continental's turnaround has been based on its corporate culture, which places a high value on customer service and the effective use of information through the integration of data in the data warehouse. Data do, indeed, matter. The topics covered in this textbook will equip you with a deeper understanding of data and how to collect, organize, and manage data. This understanding will give you the power to support any business strategy and the deep satisfaction that comes from knowing how to organize data so that financial, marketing, or customer service questions can be answered almost as soon as they are asked. Enjoy!

INTRODUCTION

Over the past two decades there has been enormous growth in the number and importance of database applications. Databases are used to store, manipulate, and retrieve data in nearly every type of organization, including business, health care, education, government, and libraries. Database technology is routinely used by individuals on personal computers, by workgroups accessing databases on network

servers, and by employees using enterprise-wide distributed applications. Databases are also accessed by customers and other remote users through diverse technologies, such as automated teller machines, Web browsers, smartphones, and intelligent living and office environments. Most Web-based applications depend on a database foundation.

Following this period of rapid growth, will the demand for databases and database technology level off? Very likely not! In the highly competitive environment of the early 2000s, there is every indication that database technology will assume even greater importance. Managers seek to use knowledge derived from databases for competitive advantage. For example, detailed sales databases can be mined to determine customer buying patterns as a basis for advertising and marketing campaigns. Organizations embed procedures called *alerts* in databases to warn of unusual conditions, such as impending stock shortages or opportunities to sell additional products, and to trigger appropriate actions.

Although the future of databases is assured, much work remains to be done. Many organizations have a proliferation of incompatible databases that were developed to meet immediate needs rather than based on a planned strategy or a well-managed evolution. Enormous amounts of data are trapped in older, "legacy" systems, and the data are often of poor quality. New skills are required to design and manage data warehouses and to integrate databases with Internet applications. There is a shortage of skills in areas such as database analysis, database design, data administration, and database administration. We address these and other important issues in this textbook to equip you for the jobs of the future.

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

In this chapter, we introduce the basic concepts of databases and database management systems (DBMSs). We describe traditional file management systems and some of their shortcomings that led to the database approach. Next, we consider the benefits, costs, and risks of using the database approach. We review of the range of technologies used to build, use, and manage databases, describe the types of applications that use databases—personal, two-tier, three-tier, and enterprise—and describe how databases have evolved over the past five decades.

Because a database is one part of an information system, this chapter also examines how the database development process fits into the overall information systems development process. The chapter emphasizes the need to coordinate database development with all the other activities in the development of a complete information system. It includes highlights from a hypothetical database development process at Pine Valley Furniture Company. Using this example, the chapter introduces tools for developing databases on personal computers and the process of extracting data from enterprise databases for use in stand-alone applications.

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

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

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

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

BASIC CONCEPTS AND DEFINITIONS

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

Database
An organized collection of logically related data.

A database may be of any size and complexity. For example, a salesperson may maintain a small database of customer contacts—consisting of a few megabytes of data—on her laptop computer. A large corporation may build a large database consisting of several terabytes of data (a *terabyte* is a trillion bytes) on a large mainframe computer that is used for decision support applications (Winter, 1997). Very large data warehouses contain more than a petabyte of data. (A *petabyte* is a quadrillion bytes.) (We assume throughout the text that all databases are computer based.)

Data

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

The traditional definition of data now needs to be expanded to reflect a new reality: Databases today are used to store objects such as documents, e-mails, maps, photographic images, sound, and video segments in addition to structured data. For example, the salesperson's database might include a photo image of the customer contact. It might also include a sound recording or video clip about the most recent product. This type of data is referred to as unstructured data, or as multimedia data. Today structured and unstructured data are often combined in the same database to create a true multimedia environment. For example, an automobile repair shop can combine structured data (describing customers and automobiles) with multimedia data (photo images of the damaged autos and scanned images of insurance claim forms).

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

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

Data Versus Information

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

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

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

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

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

FIGURE 1-1 Converting data to information

(a) Data in context

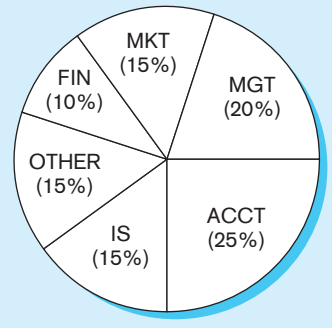
Class Roster

Course: MGT 500 Business Policy Semester: Spring 2010

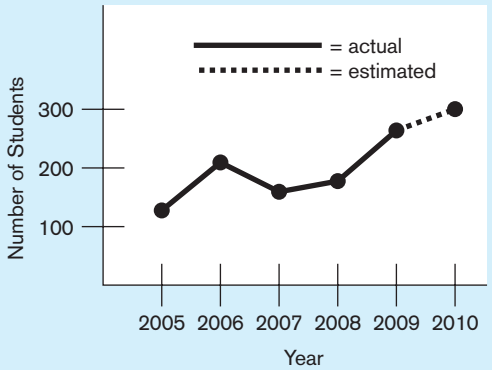
Section: 2

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

(b) Summarized data



Percent Enrollment by Major (2010)



Enrollment Projections

Another way to convert data into information is to summarize them or otherwise process and present them for human interpretation. For example, Figure 1-1b shows summarized student enrollment data presented as graphical information. This information could be used as a basis for deciding whether to add new courses or to hire new faculty members.

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

Metadata

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

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

Some sample metadata for the Class Rster (Figure 1-1a) are listed in Table 1-1. For each data item that appears in the Class Roster, the metadata show the data item name, the data type, length, minimum and maximum allowable values (where appropriate), a brief description of each data item, and the source of the data (sometimes called the *system of record*). Notice the distinction between data and metadata. Metadata are once removed from data. That is, metadata describe the properties of data but are separate from that data. Thus, the metadata shown in Table 1-1 do not include any sample data from the Class Roster of Figure 1-1a. Metadata enable database designers and users to understand what data exist, what the data mean, and how to distinguish between data items that at first glance look similar. Managing metadata is at least as crucial as managing the associated data because data without clear meaning can be confusing, misinterpreted, or erroneous. Typically, much of the metadata are stored as part of the database and may be retrieved using the same approaches that are used to retrieve data or information.

Data can be stored in files or in databases. In the following sections we examine the progression from file processing systems to databases and the advantages and disadvantages of each.

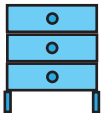
TABLE 1-1 Example Metadata for Class Roster

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

TRADITIONAL FILE PROCESSING SYSTEMS

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

As business applications became more complex, it became evident that traditional file processing systems had a number of shortcomings and limitations (described next). As a result, these systems have been replaced by database processing systems in most business applications today. Nevertheless, you should have at least some familiarity with file processing systems since understanding the problems and limitations inherent in file processing systems can help you avoid these same problems when designing database systems.



File Processing Systems at Pine Valley Furniture Company

Early computer applications at Pine Valley Furniture (during the 1980s) used the traditional file processing approach. This approach to information systems design met the data processing needs of individual departments rather than the overall information needs of the organization. The information systems group typically responded to users' requests for new systems by developing (or acquiring) new computer programs for individual applications such as inventory control, accounts receivable, or human resource management. No overall map, plan, or model guided application growth.

Three of the computer applications based on the file processing approach are shown in Figure 1-2. The systems illustrated are Order Filling, Invoicing, and Payroll. The figure also shows the major data files associated with each application. A file is a collection of related records. For example, the Order Filling System has three files: Customer Master, Inventory Master, and Back Order. Notice that there is duplication of some of the files used by the three applications, which is typical of file processing systems.

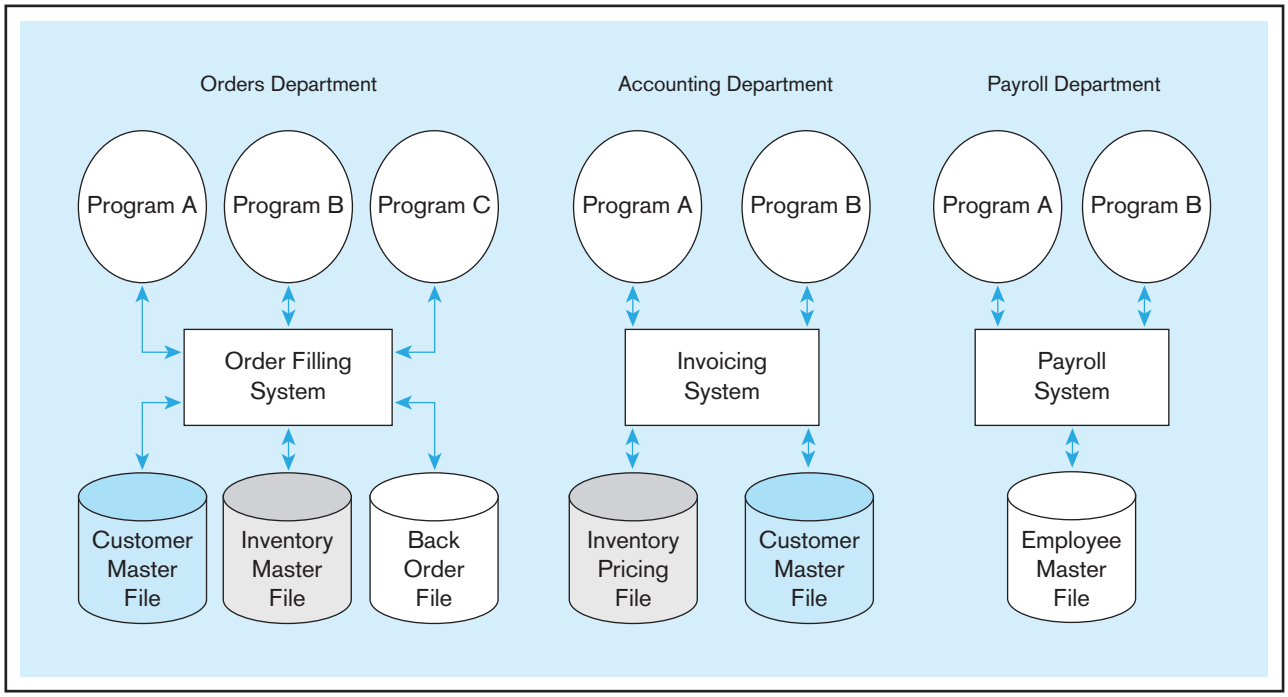


FIGURE 1-2 Old file processing systems at Pine Valley Furniture Company

Disadvantages of File Processing Systems

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

TABLE 1-2 Disadvantages of File Processing Systems

Program-data dependence
Duplication of data
Limited data sharing
Lengthy development times
Excessive program maintenance

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

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

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

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

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

LENGTHY DEVELOPMENT TIMES With traditional file processing systems, each new application requires that the developer essentially start from scratch by designing new file formats and descriptions and then writing the file access logic for each new program. The lengthy development times required are inconsistent with today's fast-paced business environment, in which time to market (or time to production for an information system) is a key business success factor.

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

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

THE DATABASE APPROACH

So, how do we overcome the flaws of file processing? No, we don't call Ghostbusters, but we do something better: We follow the database approach. We first begin by defining some core concepts that are fundamental in understanding the database approach to managing data. We then describe how the database approach can overcome the limitations of the file processing approach.

Data Models

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

Designing a database properly is fundamental to establishing a database that meets the needs of the users. **Data models** capture the nature of and relationships among data and are used at different levels of abstraction as a database is conceptualized and designed. The effectiveness and efficiency of a database is directly associated with the structure of the database. Various graphical systems exist that convey this structure and are used to produce data models that can be understood by end users, systems analysts, and database designers. Chapters 2 and 3 are devoted to developing your understanding of data modeling, as is Chapter 13, which addresses a different approach using object-oriented data modeling. A typical data model is made up entities, attributes, and relationships and the most common data modeling representation is the entity-relationship model. A brief description is presented below. More details will be forthcoming in Chapters 2 and 3.

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

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

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

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

Relational Databases

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

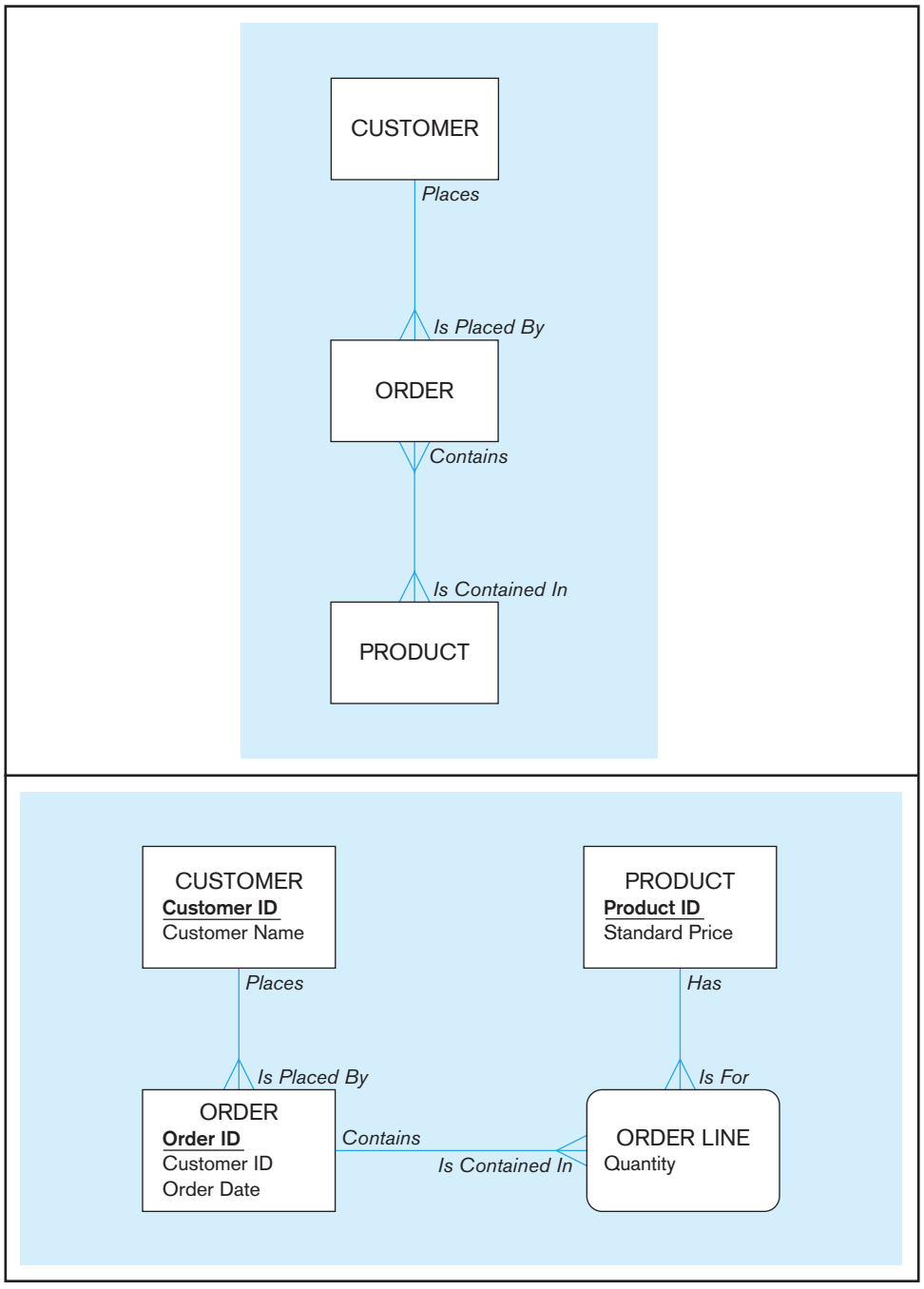


FIGURE 1-3 Comparison of enterprise and project level data models
(a) Segment of an enterprise data model

(b) Segment of a project data model

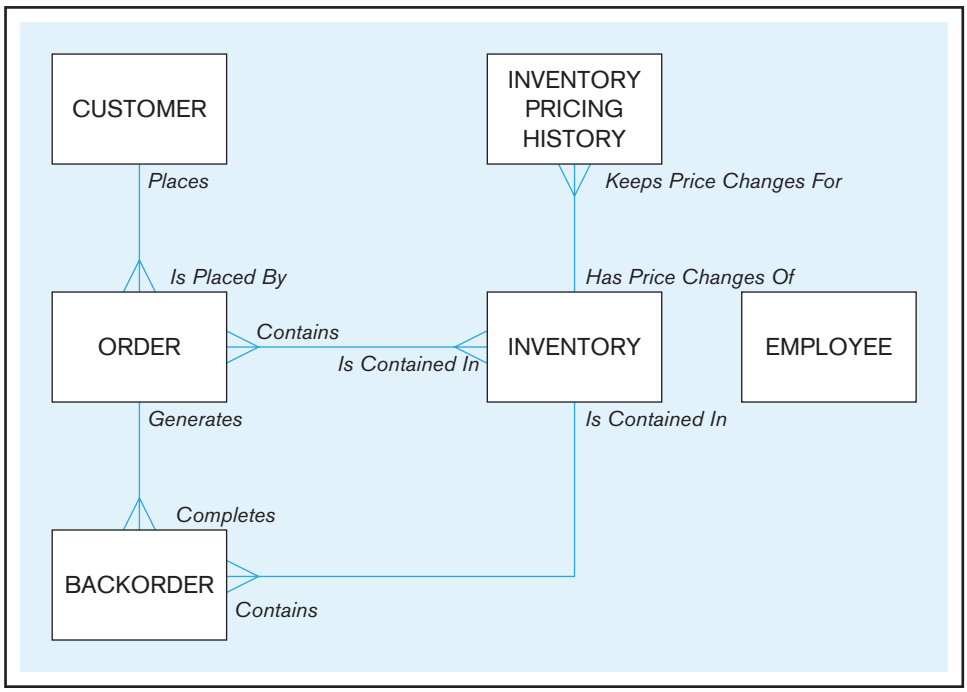
customer number with the customer's order. Thus, a customer's identification number is included in the file (or relation) that holds customer information such as name, address, and so forth. Every time the customer places an order, the customer identification number is also included in the relation that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

Database Management Systems

A **database management system (DBMS)** is a software system that enables the use of a database approach. The primary purpose of a DBMS is to provide a systematic method of creating, updating, storing, and retrieving the data stored in a database. It enables end users and application programmers to share data, and it enables data to be shared among multiple applications rather than propagated and stored in new files for every new application (Mullins, 2002). A DBMS also provides facilities for controlling data

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

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



access, enforcing data integrity, managing concurrency control, and restoring a database. We describe these DBMS features in detail in Chapter 11.

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

Advantages of the Database Approach

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

TABLE 1-3 Advantages of the Database Approach

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

PROGRAM-DATA INDEPENDENCE The separation of data descriptions (metadata) from the application programs that use the data is called **data independence**. With the database approach, data descriptions are stored in a central location called the *repository*. This property of database systems allows an organization's data to change and evolve (within limits) without changing the application programs that process the data.

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

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

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

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

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

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

1. Assuming that the database and the related data capture and maintenance applications have already been designed and implemented, the application developer can concentrate on the specific functions required for the new application, without having to worry about file design or low-level implementation details.
2. The database management system provides a number of high-level productivity tools, such as forms and report generators, and high-level languages that automate some of the activities of database design and implementation. We describe many of these tools in subsequent chapters.
3. Significant improvement in application developer productivity, estimated to be as high as 60 percent (Long, 2005), is currently being realized through the use of Web services, based on the use of standard Internet protocols and a universally accepted data format (XML). Web services and XML are covered in Chapter 8.

ENFORCEMENT OF STANDARDS When the database approach is implemented with full management support, the database administration function should be granted single-point authority and responsibility for establishing and enforcing data standards. These standards will include naming conventions, data quality standards, and uniform procedures for accessing, updating, and protecting data. The data repository provides database administrators with a powerful set of tools for developing and enforcing these standards. Unfortunately, the failure to implement a strong database administration function is

perhaps the most common source of database failures in organizations. We describe the database administration (and related data administration) functions in Chapter 11.

IMPROVED DATA QUALITY Concern with poor quality data is a common theme in strategic planning and database administration today. In fact, a recent report by The Data Warehousing Institute (TDWI) estimated that data quality problems currently cost U.S. businesses some $600 billion each year (**www.tdwi.org/research/display.asp?ID=6589**). The database approach provides a number of tools and processes to improve data quality. Two of the more important are the following:

Constraint
A rule that cannot be violated by database users.

1. Database designers can specify integrity constraints that are enforced by the DBMS. A **constraint** is a rule that cannot be violated by database users. We describe numerous types of constraints (also called "business rules") in Chapters 2 and 3. If a customer places an order, the constraint that ensures that the customer and the order remain associated is called a "relational integrity constraint," and it prevents an order from being entered without specifying who placed the order.
2. One of the objectives of a data warehouse environment is to clean up (or "scrub") operational data before they are placed in the data warehouse (Jordan, 1996). Do you ever receive multiple copies of a catalog? The company that sends you three copies of each of its mailings could recognize significant postage and printing savings if its data were scrubbed, and its understanding of its customers would also be enhanced if it could determine a more accurate count of existing customers. We describe data warehouses in Chapter 9 and the potential for improving data quality in Chapter 10.

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

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

The language used in this query is called Structured Query Language, or SQL. (You will study this language in detail in Chapters 6 and 7.) Although the queries constructed can be *much* more complex, the basic structure of the query is easy for even novice, nonprogrammers to grasp. If they understand the structure and names of the data that fit within their view of the database, they soon gain the ability to retrieve answers to new questions without having to rely on a professional application developer. This can be dangerous; queries should be thoroughly tested to be sure they are returning accurate data before relying on their results, and novices may not understand that challenge.

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

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

IMPROVED DECISION SUPPORT Some databases are designed expressly for decision support applications. For example, some databases are designed to support customer

relationship management, whereas others are designed to support financial analysis or supply chain management. You will study how databases are tailored for different decision support applications and analytical styles in Chapter 9.

Cautions About Database Benefits

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

Costs and Risks of the Database Approach

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

TABLE 1-4 Costs and Risks of the Database Approach

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

NEW, SPECIALIZED PERSONNEL Frequently, organizations that adopt the database approach need to hire or train individuals to design and implement databases, provide database administration services, and manage a staff of new people. Further, because of the rapid changes in technology, these new people will have to be retrained or upgraded on a regular basis. This personnel increase may be more than offset by other productivity gains, but an organization should recognize the need for these specialized skills, which are required to obtain the most from the potential benefits. We discuss the staff requirements for database management in Chapter 11.

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

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

NEED FOR EXPLICIT BACKUP AND RECOVERY A shared corporate database must be accurate and available at all times. This requires that comprehensive procedures be developed and used for providing backup copies of data and for restoring a database when damage occurs. These considerations have acquired increased urgency in today's security-conscious environment. A modern database management system normally automates many more of the backup and recovery tasks than a file system. We describe procedures for security, backup, and recovery in Chapter 11.

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

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

COMPONENTS OF THE DATABASE ENVIRONMENT

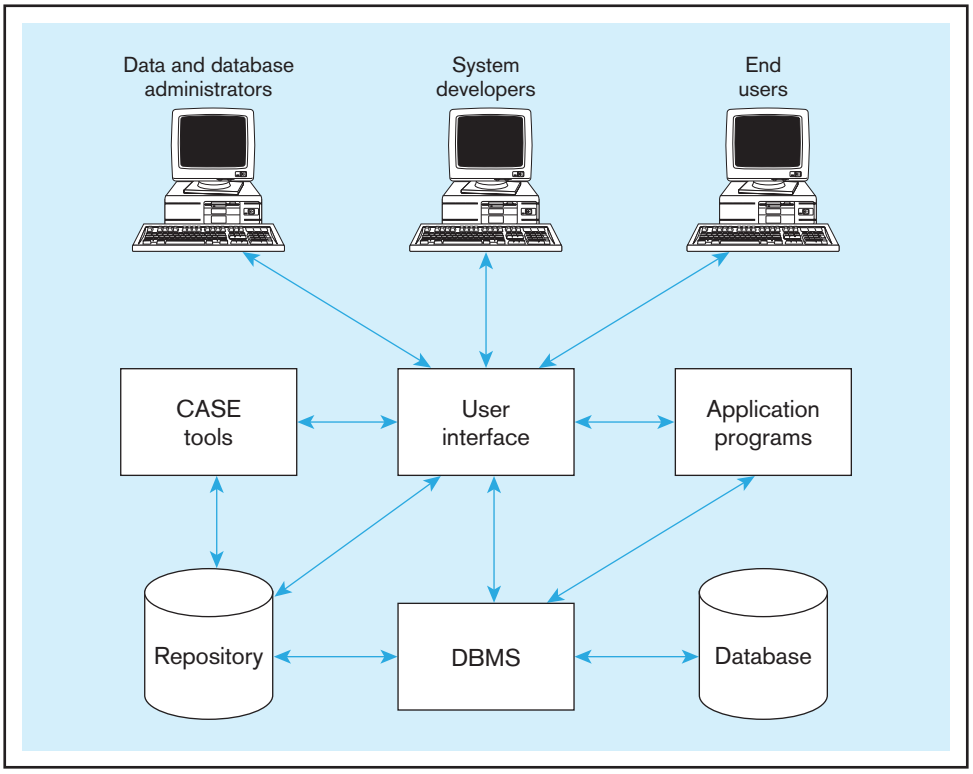
Now that you have seen the advantages and risks of using the database approach to managing data, let us examine the major components of a typical database environment and their relationships (see Figure 1-5). You have already been introduced to some (but not all) of these components in previous sections. Following is a brief description of the nine components shown in Figure 1-5:

Computer-aided software engineering (CASE) tools
Software tools that provide automated support for some portion of the systems development process.

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

1. ***Computer-aided software engineering (CASE) tools*** **CASE tools** are automated tools used to design databases and application programs. These tools help with creation of data models and in some cases can also help automatically generate the "code" needed to create the database. We reference the use of automated tools for database design and development throughout the text.
2. ***Repository*** A **repository** is a centralized knowledge base for all data definitions, data relationships, screen and report formats, and other system components. A repository contains an extended set of metadata important for managing databases as well as other components of an information system. We describe the repository in Chapter 11.

FIGURE 1-5 Components of the database environment



3. ***DBMS*** A DBMS is a software system that is used to create, maintain, and provide controlled access to user databases. We describe the functions of a DBMS in Chapter 11.
4. ***Database*** A database is an organized collection of logically related data, usually designed to meet the information needs of multiple users in an organization. It is important to distinguish between the database and the repository. The repository contains definitions of data, whereas the database contains occurrences of data. We describe the activities of database design in Chapters 4 and 5 and of implementation in Chapters 6 through 9.
5. ***Application programs*** Computer-based application programs are used to create and maintain the database and provide information to users. Key database-related application programming skills are described in Chapters 6 through 9 and Chapter 14.
6. ***User interface*** The user interface includes languages, menus, and other facilities by which users interact with various system components, such as CASE tools, application programs, the DBMS, and the repository. User interfaces are illustrated throughout this text.
7. ***Data and database administrators*** Data administrators are persons who are responsible for the overall management of data resources in an organization. Database administrators are responsible for physical database design and for managing technical issues in the database environment. We describe these functions in detail in Chapter 11.
8. ***System developers*** System developers are persons such as systems analysts and programmers who design new application programs. System developers often use CASE tools for system requirements analysis and program design.
9. ***End users*** End users are persons throughout the organization who add, delete, and modify data in the database and who request or receive information from it. All user interactions with the database must be routed through the DBMS.

In summary, the database operational environment shown in Figure 1-5 is an integrated system of hardware, software, and people, designed to facilitate the storage, retrieval, and control of the information resource and to improve the productivity of the organization.

THE RANGE OF DATABASE APPLICATIONS

What can databases help us do? Figure 1-5 shows that there are several methods for people to interact with the data in the database. First, users can interact directly with the database using the user interface provided by the DBMS. In this manner users can issues commands (called *queries*) against the database and examine the results or potentially even store this inside a Microsoft Excel spreadsheet or Word document. This method of interaction with the database is referred to ad-hoc querying and requires a level of understanding the query language on the part of the user.

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

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

Personal Databases

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

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

Two-Tier Client/Server Databases

As noted above, the utility of a personal (single-user) database is quite limited. Often, what starts off as a single-user database evolves into something that needs to be shared among several users. A workgroup is a relatively small team of people (typically fewer than 25 persons) who collaborate on the same project or application or on a group of similar projects or applications. These persons might be engaged (for example) with a construction project or with developing a new computer application and need to share data amongst the group.

The most common method of sharing data for this type of need is based on creating a two-tier client/server application as shown in Figure 1-6. Each member of the workgroup has a computer, and the computers are linked by means of network (wired or wireless LAN). In most cases, each computer has a copy of a specialized application (client) which provides the user interface as well as the business logic through which the data is manipulated. The database itself and the DBMS are stored on a central device

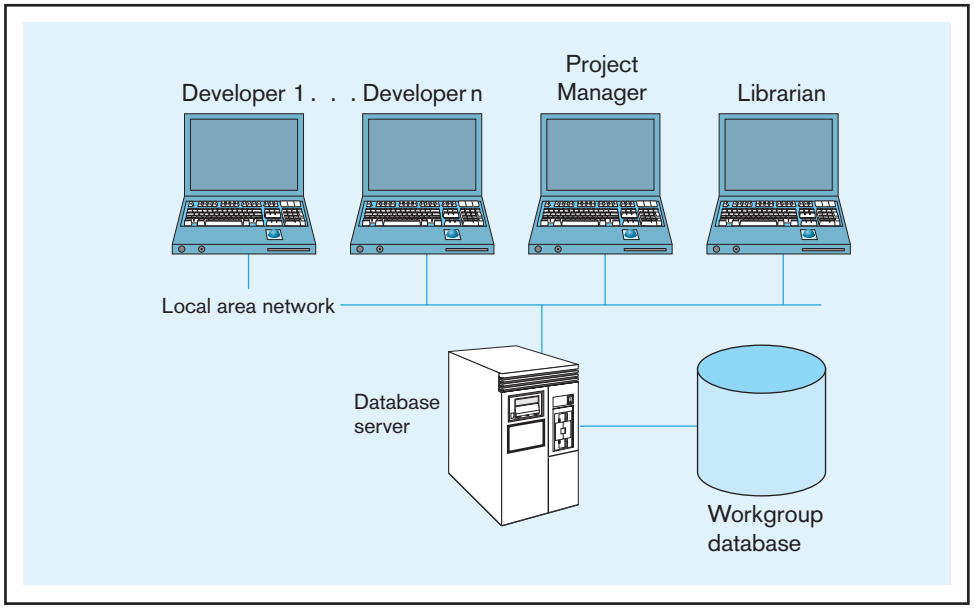


FIGURE 1-6 Two-tier database with local area network

called the "database server," which is also connected to the network. Thus, each member of the workgroup has access to the shared data. Different types of group members (e.g., developer or project manager) may have different user views of this shared database. This arrangement overcomes the principal objection to PC databases, which is that the data are not easily shared. This arrangement, however, introduces many data management issues not present with personal (single-user) databases, such as data security and data integrity when multiple users attempt to change and update data at the same time.

Multitier Client/Server Databases

One of the drawbacks of the two-tier database architecture is that the amount of functionality that needs to be programmed into the application on the users' computer can be pretty significant because it needs to contain both the user interface logic as well as the business logic. This, of course, means that the client computers need to be powerful enough to handle the programmed application. Another drawback is that each time there is a change to either the business logic or user interface, each client computer that has the application needs to be updated.

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

An example of a company that has several multitier applications is shown in Figure 1-7. In a three-tiered architecture, the user interface is accessible on the individual users' computer. This user interface may either be Web browser based or written using programming languages such as Visual Basic.NET, Visual C#, or Java. The application layer/Web server layer contains the business logic required to accomplish the business transactions requested by the users. This layer in turn talks to the database server. The

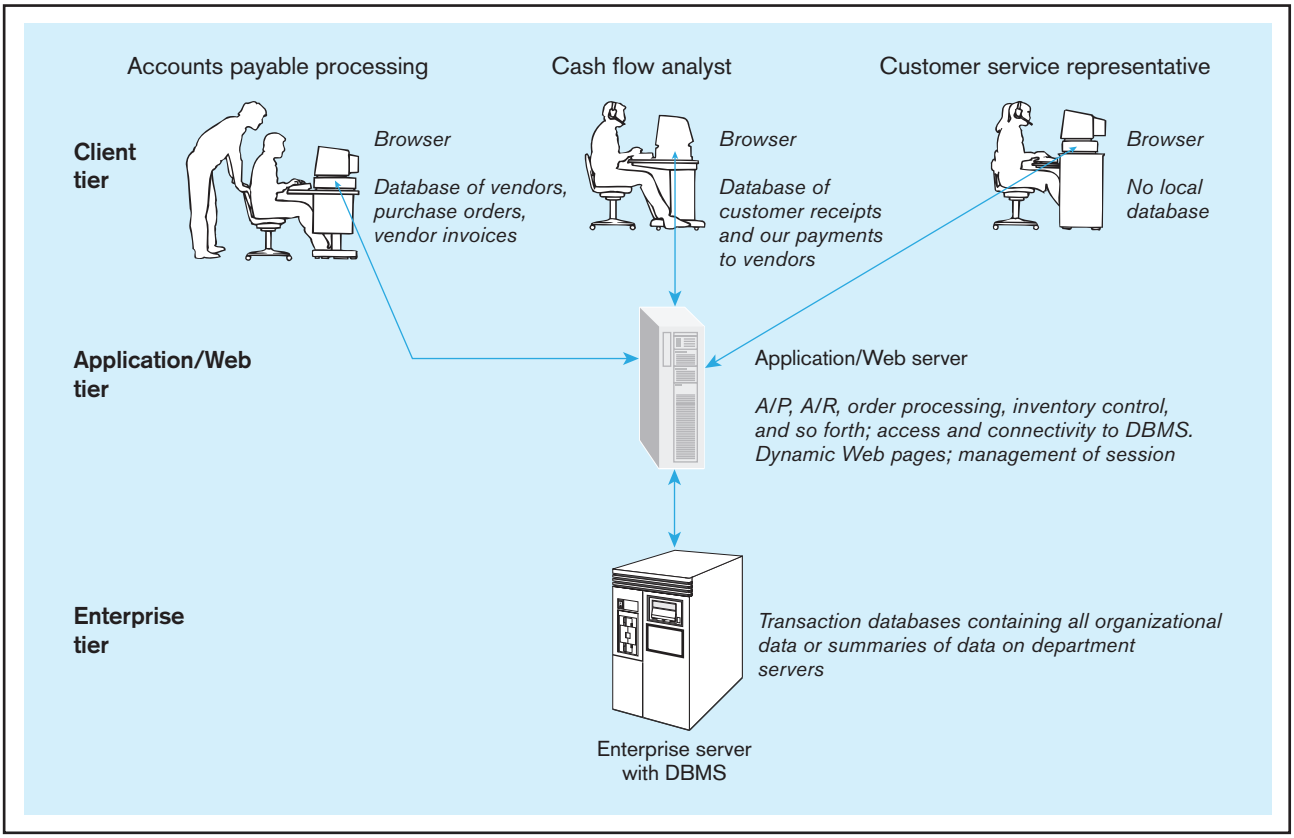


FIGURE 1-7 Three-tiered client/server database architecture

most significant implication for database development from the use of multitier client/server architectures is the ease of separating the development of the database and the modules that maintain the data from the information systems modules that focus on business logic and/or presentation logic. In addition, this architecture allows us to improve performance and maintainability of the application and database. We will consider both two and multitier client/server architectures in more detail in Chapter 8.

Enterprise Applications

An enterprise (that's small "e", not capital "E," as in *Starship*) application/database is one whose scope is the entire organization or enterprise (or, at least, many different departments). Such databases are intended to support organization-wide operations and decision making. Note that an organization may have several enterprise databases, so such a database is not inclusive of all organizational data. A single operational enterprise database is impractical for many medium to large organizations due to difficulties in performance for very large databases, diverse needs of different users, and the complexity of achieving a single definition of data (metadata) for all database users. An enterprise database does, however, support information needs from many departments and divisions. The evolution of enterprise databases has resulted in two major developments:

1. Enterprise resource planning (ERP) systems
2. Data warehousing implementations

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

Enterprise resource planning (ERP) systems have evolved from the material requirements planning (MRP) and manufacturing resource planning (MRP-II) systems of the 1970s and 1980s. These systems scheduled the raw materials, components, and subassembly requirements for manufacturing processes, and also scheduled shop floor and product distribution activities. Next, extension to the remaining business functions resulted in enterprise-wide management systems, or ERP systems. All ERP systems are heavily dependent on databases to store the integrated data required by the ERP applications. In addition to ERP systems, there are several specialized applications, such as customer relationship management (CRM) systems and supply chain management (SCM) systems, that also are dependent on data stored in databases.

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

Whereas ERP systems work with the current operational data of the enterprise, **data warehouses** collect content from the various operational databases, including personal, workgroup, department, and ERP databases. Data warehouses provide users with the opportunity to work with historical data to identify patterns and trends and answers to strategic business questions. We describe data warehouses in detail in Chapter 9.

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

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

TABLE 1-5 Summary of Database Applications

Type of Database / Application	Typical Number of Users	Typical Size of Database
Personal	1	Megabytes
Two-tier	5–100	Megabytes–gigabytes
Three-tier	100–1000	Gigabytes
Enterprise resource planning	>100	Gigabytes–terabytes
Data warehousing	>100	Terabytes–petabytes

Allowing such access to a business database raises data security and integrity issues that are new to the management of information systems, where data have traditionally been closely guarded and secured within each company. These issues are covered in more detail in Chapters 8 and 10.

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

EVOLUTION OF DATABASE SYSTEMS

Database management systems were first introduced during the 1960s and have continued to evolve during subsequent decades. Figure 1-8a sketches this evolution by highlighting the database technology (or technologies) that were dominant during each decade. In most cases, the period of introduction was quite long, and the technology was first introduced during the decade preceding the one shown in the figure. For example, the relational model was first defined by E. F. Codd, an IBM research fellow, in a paper published in 1970 (Codd, 1970). However, the relational model did not realize widespread commercial success until the 1980s. For example, the challenge of the 1970s where programmers needed to write complex programs to access data was addressed by the introduction of the Structured Query Language (SQL) in the 1980s.

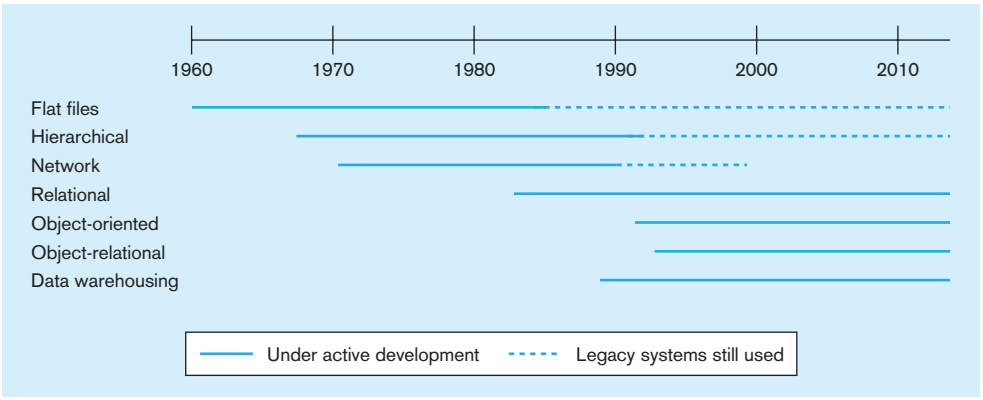
Figure 1-8b shows a visual depiction of the organizing principle underlying each of the major database technologies. For example, in the hierarchical model, files are organized in a top-down structure that resembles a tree or genealogy chart, whereas in the network model, each file can be associated with an arbitrary number of other files. The relational model (the primary focus of this book) organizes data in the form of tables and relationships among them. The object-oriented model is based on object classes and relationships among them. As shown in Figure 1-8b, an object class encapsulates attributes and methods. Object-relational databases are a hybrid between object-oriented and relational databases. Finally, multidimensional databases, which form the basis for data warehouses, allow us to view data in the form of cubes or a star schema; we discuss this in more detail in Chapter 9. Database management systems were developed to overcome the limitations of file processing systems, described in a previous section. To summarize, some of the following four objectives generally drove the development and evolution of database technology:

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

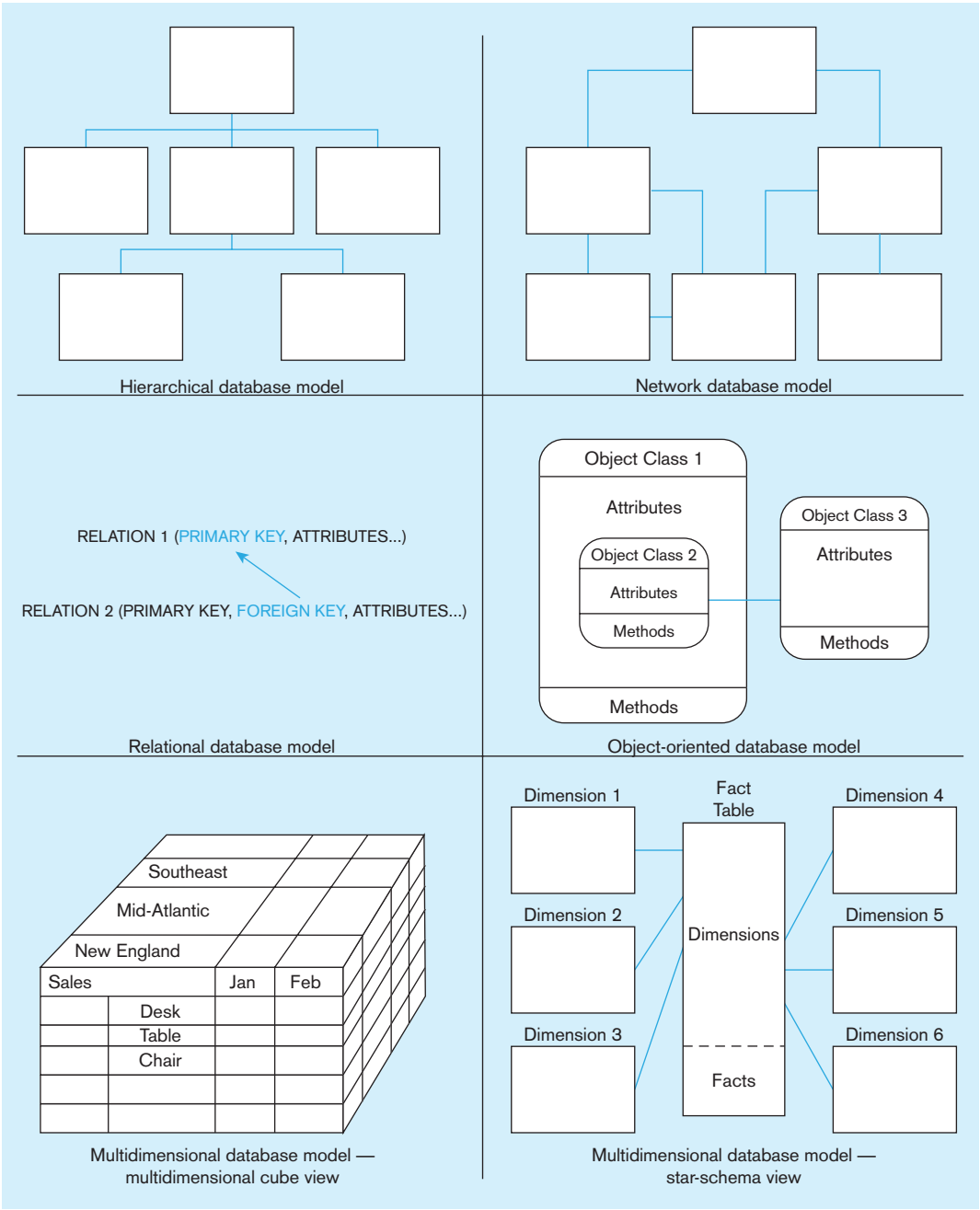
1960s

File processing systems were still dominant during this period. However, the first database management systems were introduced during this decade and were used

FIGURE 1-8 The range of database technologies: past and present



(a) Evolution of database technologies



(b) Database architectures

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

1970s

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

1980s

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

1990s

The 1990s ushered in a new era of computing, first with client/server computing, and then with data warehousing and Internet applications becoming increasingly important. Whereas the data managed by a DBMS during the 1980s were largely structured (such as accounting data), multimedia data (including graphics, sound, images, and video) became increasingly common during the 1990s. To cope with these increasingly complex data, object-oriented databases (considered third generation) were introduced during the late 1980s (Grimes, 1998).

Because organizations must manage a vast amount of structured and unstructured data, both relational and object-oriented databases are of great importance today. In fact, some vendors are developing combined object-relational DBMSs that can manage both types of data. We describe object-relational databases in Chapter 13.

2000 and Beyond

Currently, the major type of database that is still most widely used is the relational database. However, object-oriented and object-relational databases are also garnering some attention, especially as the growth in unstructured content continues. This growth is partially fueled by Web 2.0 applications such as blogs, wikis, and social networking sites (Facebook, MySpace, Twitter, LinkedIn, etc.) and partially by how easy it has become to create unstructured data such as pictures and images. Developing effective database practices to deal with these diverse types of data is going to continue to be of prime importance as we move into the next decade. As larger computer memory chips become cheaper, new database technologies to manage in-memory databases are emerging. This trend opens up new possibilities for even faster database processing.

Recent regulations such as Sarbanes-Oxley, HIPAA, and the Basel Convention have highlighted the importance of good data management practices and the ability to reconstruct historical positions has gained prominence. This has led to developments in

computer forensics with increased emphasis and expectations around discovery of electronic evidence. The importance of good database administration capabilities also continues to rise because effective disaster recovery and adequate security are mandated by these regulations.

THE DATABASE DEVELOPMENT PROCESS

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

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

Figure 1-3a showed a segment of an enterprise data model for Pine Valley Furniture Company, using a simplified version of the notation you will learn in Chapters 2 and 3. Besides such a graphical depiction of the entity types, a thorough enterprise data model would also include business-oriented descriptions of each entity type and a compendium of various statements about how the business operates, called business rules, which govern the validity of data. Relationships between business objects (business functions, units, applications, etc.) and data are often captured using matrixes and complement the information captured in the enterprise data model. Figure 1-9 shows an example of such a matrix.

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

A typical bottom-up database development project usually focuses on the creation of one database. Some database projects concentrate only on defining, designing, and implementing a database as a foundation for subsequent information systems development. In

FIGURE 1-9 Example business function-to-data entity matrix

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

X = data entity is used within business function

most cases, however, a database and the associated information processing functions are developed together as part of a comprehensive information systems development project.

Systems Development Life Cycle

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

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

The various steps in the SDLC and their associated purpose are depicted in Figure 1-10 (Hoffer et al., 2010). The process appears to be circular and is intended to convey the iterative nature of systems development projects. The steps may overlap in time, they may be conducted in parallel, and it is possible to backtrack to previous steps

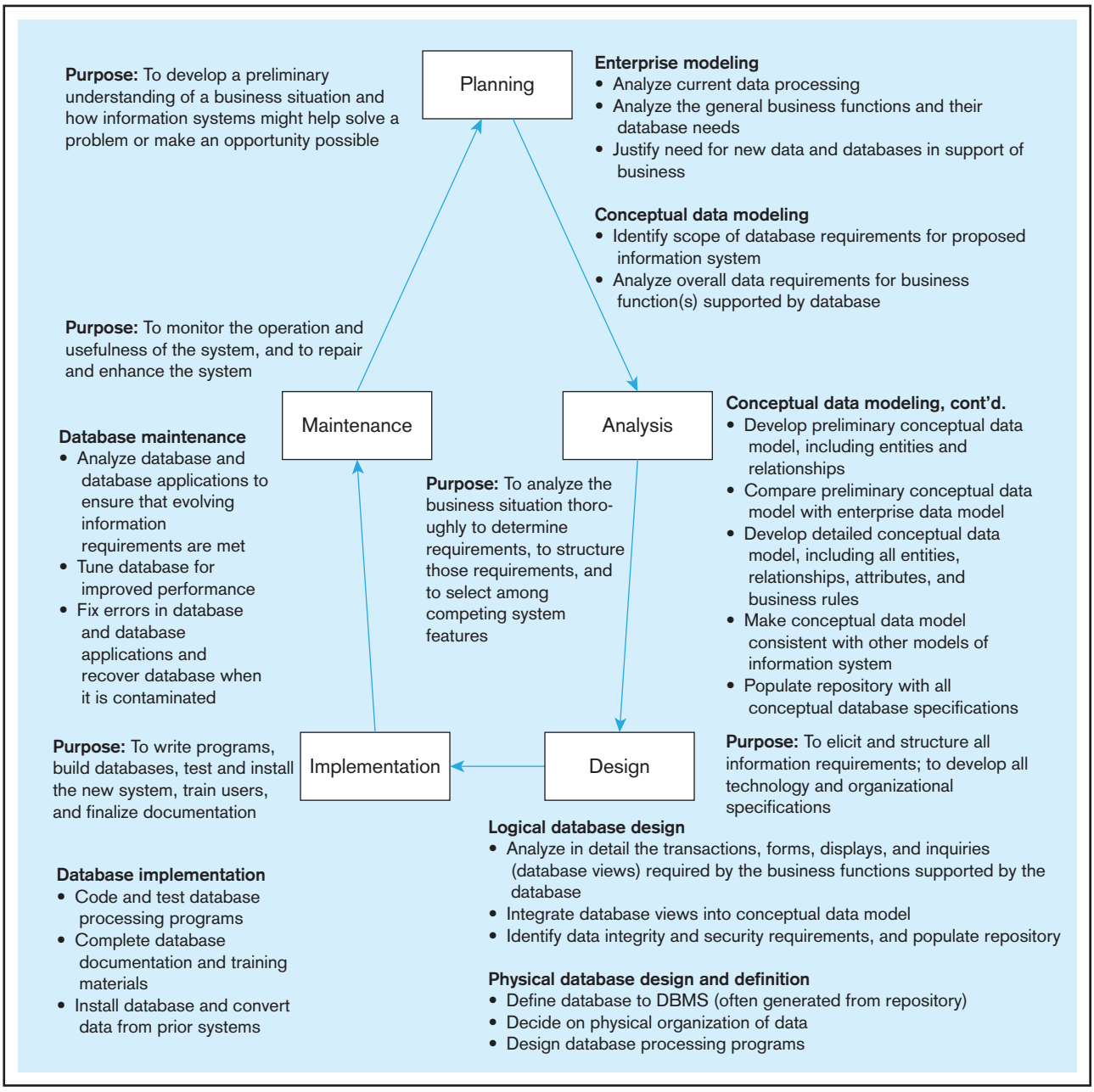


FIGURE 1-10 **Database development activities during the systems development life cycle (SDLC)**

when prior decisions need to be reconsidered. Some feel that the most common path through the development process is to cycle through the steps depicted in Figure 1-10, but at more detailed levels on each pass, as the requirements of the system become more concrete.

Figure 1-10 also provides an outline of the database development activities typically included in each phase of the SDLC. Note that there is not always a one-to-one correspondence between SDLC phases and database development steps. For example, conceptual data modeling occurs in both the Planning and the Analysis phases. We will briefly illustrate each of these database development steps for Pine Valley Furniture Company later in this chapter.

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

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

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

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

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

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

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

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

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

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

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

MAINTENANCE—DATABASE MAINTENANCE The database evolves during database maintenance. In this step, the designer adds, deletes, or changes characteristics of the structure of a database in order to meet changing business conditions, to correct errors in database design, or to improve the processing speed of database applications. The designer might also need to rebuild a database if it becomes contaminated or destroyed due to a program or computer system malfunction. This is typically the

longest step of database development, because it lasts throughout the life of the database and its associated applications. Each time the database evolves, view it as an abbreviated database development process in which conceptual data modeling, logical and physical database design, and database implementation occur to deal with proposed changes.

Alternative IS Development Approaches

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

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

One of the most popular RAD methods is **prototyping**, which is an iterative process of systems development in which requirements are converted to a working system that is continually revised through close work between analysts and users. Figure 1-11 shows the prototyping process. This figure includes annotations to indicate roughly which database development activities occur in each prototyping phase. Typically, you make only a very cursory attempt at conceptual data modeling when the information system problem is identified. During the development of the initial prototype, you simultaneously design the displays and reports the user wants while

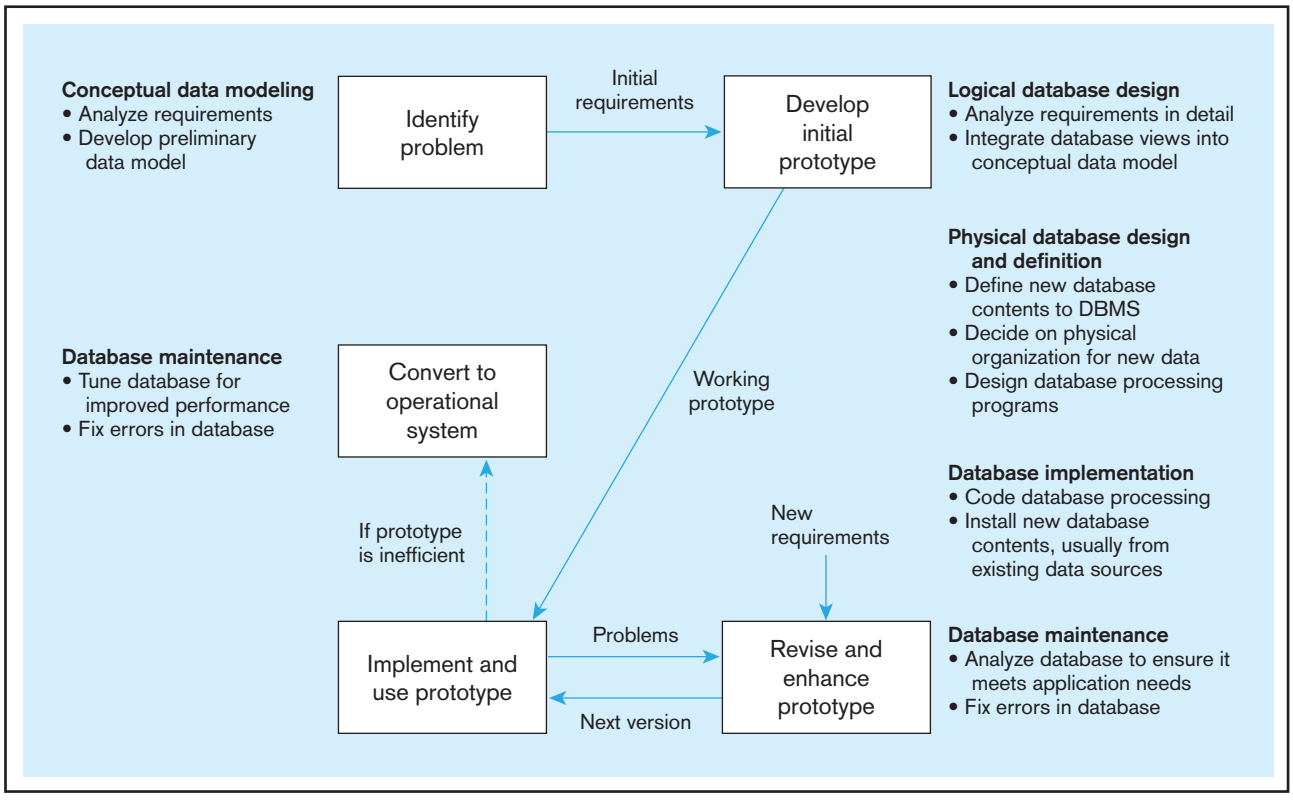


FIGURE 1-11 The prototyping methodology and database development process

understanding any new database requirements and defining a database to be used by the prototype. This is typically a new database, which is a copy of portions of existing databases, possibly with new content. If new content is required, it will usually come from external data sources, such as market research data, general economic indicators, or industry standards.

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

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

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

> *Individuals and interactions* over processes and tools
> *Working software* over comprehensive documentation
> *Customer collaboration* over contract negotiation, and
> *Responding to change* over following a plan

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

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

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

Three-Schema Architecture for Database Development

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

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

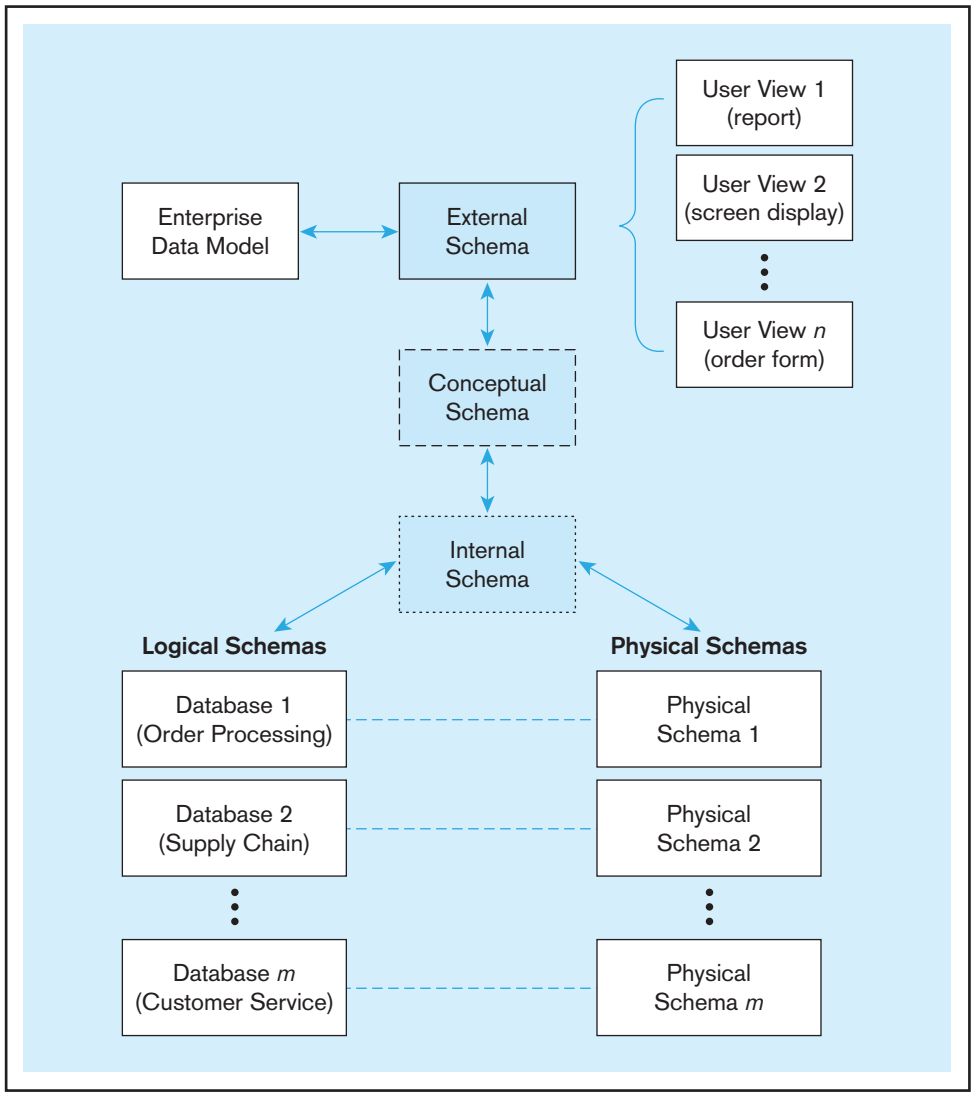


FIGURE 1-12 **Three-schema architecture**

- Logical schema (during the Logical Design phase)
- Physical schema (during the Physical Design phase)

In 1978, an industry committee commonly known as ANSI/SPARC published an important document that described three-schema architecture—external, conceptual and internal schemas—for describing the structure of data. Figure 1-12 shows the relationship between the various schemas developed during the SDLC and the ANSI three-schema architecture. It is important to keep in mind that all these schemas are just different ways of visualizing structure of the same database by different stakeholders.

The three schemas as defined by ANSI (depicted down the center of Figure 1-12) are as follows:

1. ***External schema*** This is the view (or views) of managers and other employees who are the database users. As shown in Figure 1-12, the external schema can be represented as a combination of the enterprise data model (a top-down view) and a collection of detailed (or bottom-up) user views.
2. ***Conceptual schema*** This schema combines the different external views into a single, coherent, and comprehensive definition of the enterprise's data. The conceptual schema represents the view of the data architect or data administrator.
3. ***Internal schema*** As shown in Figure 1-12, an internal schema today really consists of two separate schemas: a logical schema and a physical schema. The logical schema is the representation of data for a type of data management technology

(e.g., relational). The physical schema describes how data are to be represented and stored in secondary storage using a particular DBMS (e.g., Oracle).

Managing the People Involved in Database Development

Isn't it always ultimately about people working together? As implied in Figure 1-10, a database is developed as part of a project. A **project** is a planned undertaking of related activities to reach an objective that has a beginning and an end. A project begins with the first steps of the Project Initiation and Planning phase and ends with the last steps of the Implementation phase. A senior systems or database analyst will be assigned to be project leader. This person is responsible for creating detailed project plans as well as staffing and supervising the project team.

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

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

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

It is the responsibility of the project leader to select and manage all of these people as an effective team. See Hoffer et al. (2010) for details on how to manage a systems development project team. See Henderson et al. (2005) for a more detailed description of career paths and roles in data management. The emphasis on people rather than roles when agile development processes are adopted means that team members will be less likely to be constrained to a particular role. They will be expected to contribute and collaborate across these roles, thus using their particular skills, interests, and capabilities more completely.

DEVELOPING A DATABASE APPLICATION FOR PINE VALLEY FURNITURE COMPANY

Pine Valley Furniture Company was introduced earlier in this chapter. By the late 1990s, competition in furniture manufacturing had intensified, and competitors seemed to respond more rapidly than Pine Valley Furniture to new business opportunities. While there were many reasons for this trend, managers felt that the computer information systems they had been using (based on traditional file processing)

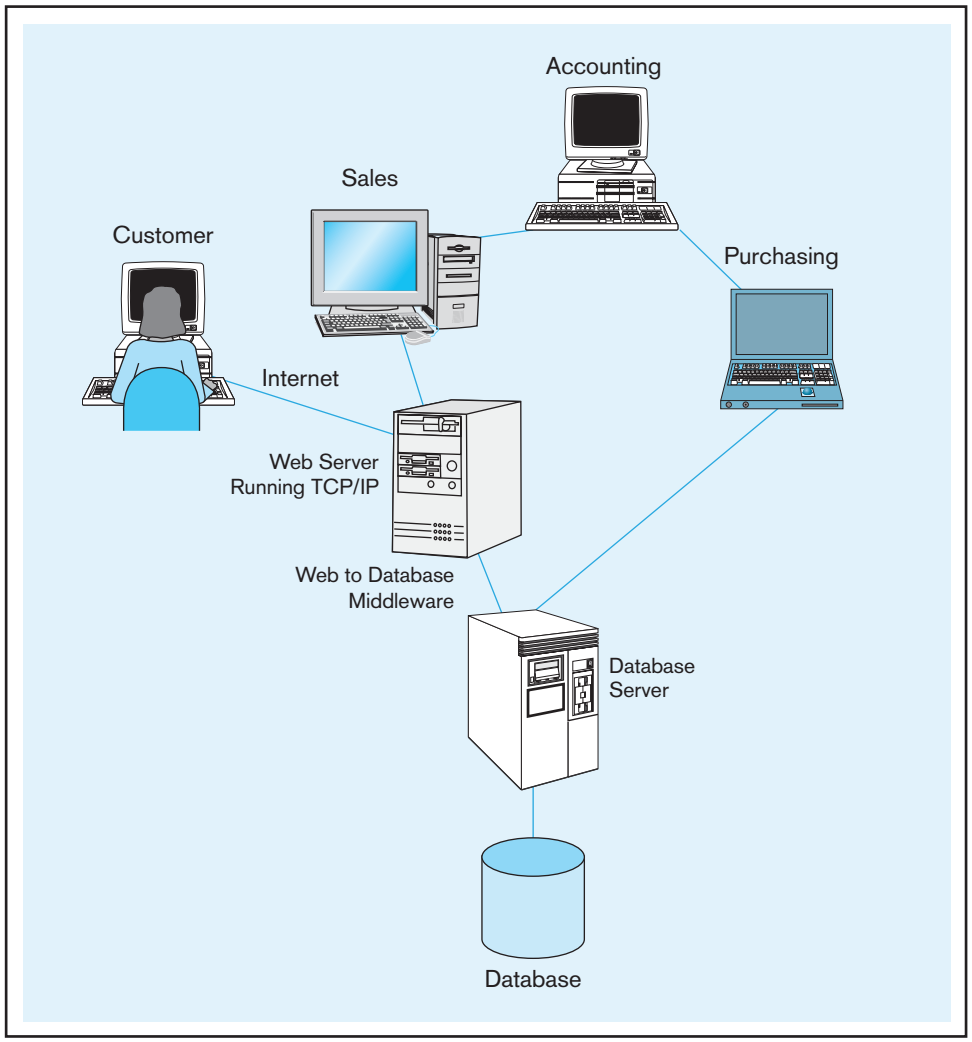


FIGURE 1-13 Computer System for Pine Valley Furniture Company

had become outdated. After attending an executive development session led by Fred McFadden and Jeff Hoffer (we wish!), the company started a development effort that eventually led to adopting a database approach for the company. Figure 1-13 displays a general schematic of the computer network within Pine Valley Furniture Company.

Data previously stored in separate files have been integrated into a single database structure. Also, the metadata that describe these data reside in the same structure. The DBMS provides the interface between the various database applications for organizational users and the database (or databases). The DBMS allows users to share the data and to query, access, and update the stored data.

Before addressing a request that has been received for direct access to sales data from Helen Jarvis, product manager for home office furniture, let's review the process that Pine Valley Furniture Company followed as they originally moved into a database environment. Pine Valley Furniture Company's first step in converting to a database approach was to develop a list of the high-level entities that support the business activities of the organization. You will recall that an entity is an object or concept that is important to the business. Some of the high-level entities identified at Pine Valley Furniture were the following: CUSTOMER, PRODUCT, EMPLOYEE, CUSTOMER ORDER, and DEPARTMENT. After these entities were identified and defined, the company proceeded to develop an enterprise data model. Remember that an enterprise data model is a graphical model that shows the high-level entities for the organization and associations among those entities.

The results of preliminary studies convinced management of the potential advantages of the database approach. After additional data modeling steps had been

completed, the company decided to implement a modern relational database management system that views all data in the form of tables. (We cover relational databases in more detail in Chapter 4.) A simplified segment of the project data model used is discussed next.

Simplified Project Data Model Example

A segment of the project data model containing four entities and three pertinent associations is shown in Figure 1-3b. The entities shown in this model segment are the following:

CUSTOMER	A person or an organization that buys or may potentially buy products from Pine Valley Furniture
ORDER	The purchase of one or more products by a customer
PRODUCT	The items Pine Valley Furniture makes and sells
ORDER LINE	Details about each product sold on a particular customer order (such as quantity and price)

The three associations (called *relationships* in database terminology) shown in the Figure 1-3b (represented by the three lines connecting entities) capture three fundamental business rules, as follows:

1. Each CUSTOMER *Places* any number of ORDERs. Conversely, each ORDER *Is Placed By* exactly one CUSTOMER.
2. Each ORDER *Contains* any number of ORDER LINEs. Conversely, each ORDER LINE *Is Contained In* exactly one ORDER.
3. Each PRODUCT *Has* any number of ORDER LINEs. Conversely, each ORDER LINE *Is For* exactly one PRODUCT.

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

Notice the following characteristics of the project data model:

1. It is a model of the organization that provides valuable information about how the organization functions, as well as important constraints.
2. The project data model focuses on entities, relationships, and business rules. It also includes attribute labels for each piece of data that will be stored in each entity. Many entities would include more attributes than we list in Figure 1-3b, but we have included a sufficient number to help you begin to understand how the data will be stored in a database.

Figure 1-14 shows the following four tables with sample data: Customer, Product, Order, and OrderLine. Notice that these tables represent the four entities shown in the project data model (Figure 1-3b). Each column of a table represents an attribute (or characteristic) of an entity. For example, the attributes shown for Customer are CustomerID and CustomerName. Each row of a table represents an instance (or occurrence) of the entity. An important property of the relational model is that it represents relationships between entities by values stored in the columns of the corresponding tables. For example, notice that CustomerID is an attribute of both the Customer table and the Order table. As a result, we can easily link an order to its associated customer. For example, we can determine that OrderID 1003 is associated with CustomerID 1. Can you determine which ProductIDs are associated with OrderID 1004? In subsequent chapters, you will learn how to retrieve data from these tables by using a powerful query language, SQL, which exploits these linkages.

To facilitate the sharing of data and information, Pine Valley Furniture Company uses a local area network (LAN) that links employee workstations in the various departments to a database server, as shown in Figure 1-13. During the early 2000s, the company mounted a two-phase effort to introduce Internet technology. First, to improve intracompany communication and decision making, an intranet was installed

FIGURE 1-14 **Four relations (Pine Valley Furniture Company)**

(a) Order and Order Line Tables

Order_T

OrderID	OrderDate	CustomerID
1001	10/21/2010	
1002	10/21/2010	
1003	10/22/2010	
1004	10/22/2010	
1005	10/24/2010	
1006	10/24/2010	
1007	10/27/2010	
1008	10/30/2010	
1009	11/5/2010	
1010	11/5/2010	

Record: 1 of 10 No Filter Search

OrderLine_T

OrderID	ProductID	OrderedQuantity
1001	1	2
1001	2	2
1001	4	1
1002	3	5
1003	3	3
1004	6	2
1004	8	2
1005	4	4
1006	4	1
1006	5	2
1006	7	2
1007	1	3
1007	2	2
1008	3	3
1008	8	3
1009	4	2
1009	7	3
1010	8	10
*	0	0

Record: 1 of 18 No Filter Search

Customer_T

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

(b) Customer table

Product_T

ProductID	ProductStandardPrice
1	$175.00
2	$200.00
3	$375.00
4	$650.00
5	$325.00
6	$750.00
7	$800.00
8	$250.00
* (New)	$0.00

(c) Product table

that allows employees fast Web-based access to company information, including phone directories, furniture design specifications, e-mail, and so forth. In addition, Pine Valley Furniture Company also added a Web interface to some of its business applications, such as order entry, so that more internal business activities that require access to data in the database server can also be conducted by employees through its intranet. However, most applications that use the database server still do not have a Web interface and require that the application itself be stored on employees' workstations.

Although the database quite adequately supports daily operations at Pine Valley Furniture Company, managers soon learned that the same database is often inadequate for decision support applications. For example, following are some types of questions that cannot be easily answered:

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

To answer these and other questions, an organization often needs to build a separate database that contains historical and summarized information. Such a database is usually called a *data warehouse* or, in some cases, a *data mart*. Also, analysts need specialized decision support tools to query and analyze the database. One class of tools used for this purpose is called online analytical processing (OLAP) tools. We describe data warehouses, data marts, and related decision support tools in Chapter 9. There you will learn of the interest in building a data warehouse that is now growing within Pine Valley Furniture Company.

A Current Pine Valley Furniture Company Project Request

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

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

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

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

Project Planning

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

Chris does two quick analyses before talking with Helen again. First, he identifies all of the databases that contain data associated with the data entities Helen mentioned. From these databases, Chris makes a list of all of the data attributes from these data entities that he thinks might be of interest to Helen in her analyses of the home office furniture market. Chris's previous involvement in projects that developed Pine Valley's standard sales tracking and forecasting system and cost accounting system helps him to speculate on the kinds of data Helen might want. For example, the objective to exceed sales goals for each product finish category of office furniture suggests that Helen wants product annual sales goals in her system; also, the objective of achieving at least an 8 percent annual sales growth means that the prior year's orders for each product need to be included. He also concludes that Helen's database must include all products, not just those in the office furniture line, because she wants to compare her line to others. However, he is able to eliminate many of the data attributes kept on each data entity. For example, Helen does not appear to need various customer data such as address, phone number, contact person, store size, and salesperson. Chris does, though, include a few additional attributes, customer type and zip code, which he feels might be important in a sales forecasting system.

Second, from this list, Chris draws a graphic data model that represents the data entities with the associated data attributes, as well as the major relationships between these data entities. Chris's hope is that he can reduce the time for the analysis phase of the systems development process (and hence the time to do conceptual data modeling) by presenting this data model to Helen. A graphic of the data model for the preliminary database that Chris produces appears in Figure 1-15. The data attributes

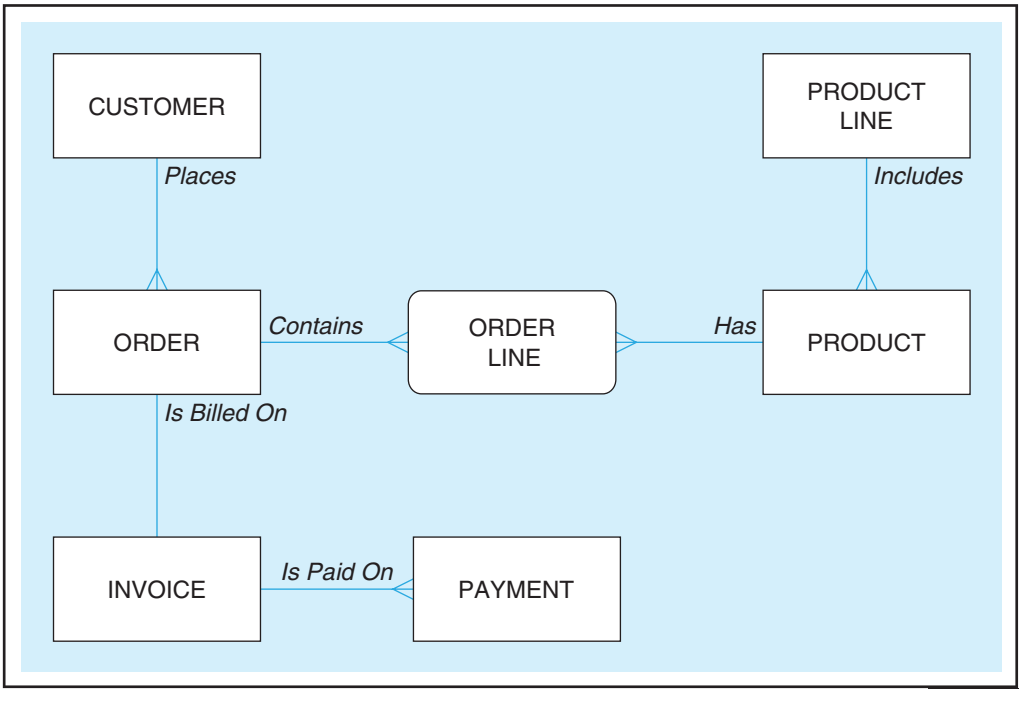


FIGURE 1-15 Preliminary data model for Home Office product line marketing support system

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

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

of each entity Chris thinks Helen wants for the system are listed in Table 1-6. Chris lists in Table 1-6 only basic data attributes from existing databases, because Helen will likely want to combine these data in various ways for the analyses she will want to do.

Analyzing Database Requirements

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

the duration of certain project steps. For this reason, Chris's boss has decided that this project should be billed to Helen's department on a consulting time basis, not at a fixed cost.

Chris does more of the talking at this second meeting, but he pays close attention to Helen's reactions to his initial ideas for the database application. He methodically walks through each data entity in Figure 1-15, explaining what it means, what each data attribute associated with it (in Table 1-6) means, and what business policies and procedures are represented by each line between entities. For example, Chris explains that each order is billed on one invoice and each invoice is a bill for exactly one order. An Order Number uniquely identifies each order, and an order is placed by one customer. Other data about an order Chris thinks Helen might want to know include the date when the order was placed and the date when the order was filled. (This would be the latest shipment date for the products on the order.) Chris also explains that the Payment Date attribute represents the most recent date when the customer made any payments, in full or partial, for the order.

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

Designing the Database

Because Chris is following a prototyping methodology, and because the first two sessions with Helen quickly identified the data Helen might need, Chris is able to immediately begin to build the prototype. First, Chris extracts from the corporate databases the data entities and attributes that Helen suggested. Chris is able to create all of these files using the SQL query language. Some of the data Helen wants are computed from raw, operational data (e.g., Customer Years), but SQL makes it easy for Chris to specify these calculations. This extracting results in a single ASCII file for each data entity; each row in a file contains all of the data attributes associated with that data entity in the data model, and the rows are different instances of the entity. For example, each row of the ASCII file for the PRODUCT LINE data entity contains data for product line names and the annual sales goals for the past and current years.

Second, Chris translates the final data model from his discussion with Helen into a set of tables for which the columns are data attributes and the rows are different sets of values for those attributes. Tables are the basic building blocks of a relational database, which is the database style for Microsoft Access. The definitions of the ProductLine and Product tables Chris created, including associated data attributes, are shown in Figures 1-16 and 1-17. The tables are defined using SQL. It is customary to add the suffix _T to a table name. Also note that because relational databases do not allow for spaces between names, the individual words in the attributes from the data model have now been concatenated. Hence, Product Description in the data model has become ProductDescription in the table. Chris did this translation so that each table had an attribute, called the table's "primary key," which will be distinct for each row in the table. The other major properties of each

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

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

*Changes from preliminary list of attributes appear in italics.

table are that there is only one value for each attribute in each row, and if we know the value of the identifier, there can be only one value for each of the other attributes. For example, for any product line, there can be only one value for the current year's sales goal.

The design of the database includes specifying the format, or properties, for each attribute (MS Access calls attributes *fields*). These design decisions were easy in this

FIGURE 1-16 SQL definition of ProductLine table

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

FIGURE 1-17 SQL definition of Product table

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

case because most of the attributes were already specified in the corporate data dictionary.

The other major decision Chris has to make about database design is how to physically organize the database to respond fastest to the queries Helen will write. Because the database will be used for decision support, neither Chris nor Helen can anticipate all of the queries that will arise; thus, Chris must make the physical design choices from experience rather than precise knowledge of the way the database will be used. The key physical database design decision that SQL allows a database designer to make is on which attributes to create indexes. (An index is like a card catalog in the library, through which rows with common characteristics can be quickly located.) All primary key attributes (like OrderNumber for the Order_T table)—those with unique values across the rows of the table—are indexed. In addition to this, Chris uses a general rule of thumb: Create an index for any attribute that has more than 10 different values and that Helen might use to segment the database. For example, Helen indicated that one of the ways she wants to use the database is to look at sales by product finish. Thus, it might make sense to create an index on the Product_T table using the Product Finish attribute.

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

Figure 1-18 shows the prototype project data model developed by Chris for the home office marketing database. Each box represents one table in the database; the attributes of a table are listed inside the associated box. The project data model shows how the relations will be linked by including common fields, or foreign keys. The foreign keys have a dashed underline in the model. For example, the attribute Product Line is the primary identifier in the PRODUCT LINE relation, and is also included in the PRODUCT relation. This linkage makes it possible to compare current sales to sales goals by product line.

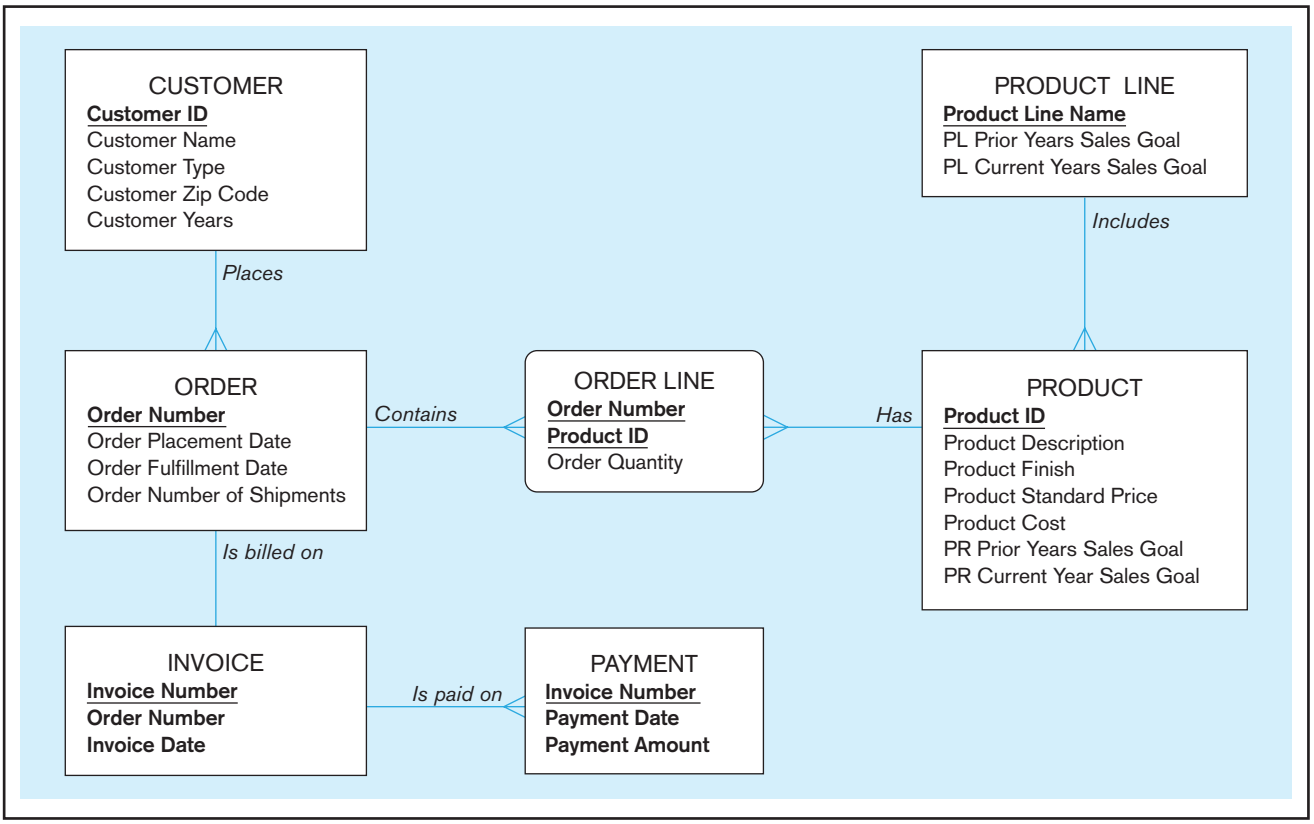


FIGURE 1-18 **Project data model for Home Office product line marketing support system**

Using the Database

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

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

The query to produce this list of products appears in Figure 1-19, with sample output in Figure 1-20. The query in Figure 1-19 uses SQL. You can see three of the six standard SQL clauses in this query: SELECT, FROM, and WHERE. SELECT indicates which attributes will be shown in the result. One calculation is also included and given the label "Sales to Date." FROM indicates which tables must be accessed to retrieve data. WHERE defines the links between the tables and indicates that results from only the Home Office product line are to be included. Only limited data are included for this example, so the Total Sales results in Figure 1-20 are fairly small, but the format is the result of the query in Figure 1-19.

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

FIGURE 1-19 SQL query for Home Office sales-to-goal comparison

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

FIGURE 1-20 Home Office product line sales comparison

Home Office Sales to Date : Select Query

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

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

Administering the Database

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

Summary

Over the past two decades there has been enormous growth in the number and importance of database applications. Databases are used to store, manipulate, and retrieve data in every type of organization. In the highly competitive environment of the 2000s, there is every indication that database technology will assume even greater importance. A course in modern database management is one of the most important courses in the information systems curriculum.

A database is an organized collection of logically related data. We define *data* as stored representations of objects and events that have meaning and importance in the

user's environment. Information is data that have been processed in such a way that the knowledge of the person who uses the data increases. Both data and information may be stored in a database.

Metadata are data that describe the properties or characteristics of end-user data and the context of that data. A database management system (DBMS) is a software system that is used to create, maintain, and provide controlled access to user databases. A DBMS stores metadata in a repository, which is a central storehouse for all data definitions, data relationships, screen and report formats, and other system components.

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

Database applications can be arranged into the following categories: personal databases, two-tier databases, multitier, and enterprise databases. Enterprise databases include data warehouses and integrated decision support databases whose content is derived from the various operational databases. Enterprise resource planning (ERP) systems rely heavily on enterprise databases. A modern database and the applications that use it may be located on multiple computers. Although any number of tiers may exist (from one to many), three tiers of computers relate to the client/server architecture for database processing: (1) the client tier, where database contents are presented to the user; (2) the application/Web server tier, where analyses on database contents are made and user sessions are managed; and (3) the enterprise server tier, where the data from across the organization are merged into an organizational asset.

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

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

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

Chapter Review

Key Terms

Agile software development *29*
Computer-aided software engineering (CASE) tools *16*
Conceptual schema *26*
Constraint *14*
Data *5*
Data independence *13*
Data model *10*
Data warehouse *20*
Database *5*
Database application *9*
Database management system (DBMS) *11*
Enterprise data modeling *24*
Enterprise resource planning (ERP) *20*
Entity *10*
Information *6*
Logical schema *26*
Metadata *7*
Physical schema *27*
Project *31*
Prototyping *28*
Relational database *10*
Repository *16*
Systems development life cycle (SDLC) *25*
User view *13*

Review Questions

1. Define each of the following terms:
   a. data
   b. information
   c. metadata
   d. database application
   e. data warehouse
   f. constraint
   g. database
   h. entity
   i. database management system
   j. client/server architecture
   k. systems development life cycle (SDLC)
   l. agile software development
   m. enterprise data model
   n. conceptual data model
   o. logical data model
   p. physical data model
2. Match the following terms and definitions:

____data	a. data placed in context or summarized
____database application	b. application program(s)
____constraint	c. facts, text, graphics, images, etc.
____repository	d. a graphical model that shows the high-level entities for the organization and the relationships among those entities
____metadata	e. organized collection of related data
____data warehouse	f. includes data definitions and constraints
____information	g. centralized storehouse for all data definitions
____user view	h. separation of data description from programs
____database management system	i. a business management system that integrates all functions of the enterprise
____data independence	j. logical description of portion of database
____database	k. a software application that is used to create, maintain, and provide controlled access to user databases
____enterprise resource planning (ERP)	l. a rule that cannot be violated by database users
____systems development life cycle (SDLC)	m. integrated decision support database
____prototyping	n. consist of the enterprise data model and multiple user views
____enterprise data model	o. a rapid approach to systems development
____conceptual schema	p. consists of two data models: a logical model and a physical model
____internal schema	q. a comprehensive description of business data
____external schema	r. a structured, step-by-step approach to systems development

3. Contrast the following terms:
   a. data dependence; data independence
   b. structured data; unstructured data
   c. data; information
   d. repository; database
   e. entity; enterprise data model
   f. data warehouse; ERP system
   g. two-tier databases; multitier databases
   h. systems development life cycle; prototyping
   i. enterprise data model; conceptual data model
   j. prototyping; agile software development
4. List five disadvantages of file processing systems.
5. List the nine major components in a database system environment.
6. How are relationships between tables expressed in a relational database?
7. What does the term *data independence* mean, and why is it an important goal?
8. List 10 potential benefits of the database approach over conventional file systems.
9. List five costs or risks associated with the database approach.
10. Define a three-tiered database architecture.
11. In the three-tiered database architecture, is it possible for there to be no database on a particular tier? If not, why? If yes, give an example.
12. Name the five phases of the traditional systems development life cycle, and explain the purpose and deliverables of each phase.
13. In which of the five phases of the SDLC do database development activities occur?
14. Are there procedures and processes that are common to the use of SDLC, prototyping, and agile methodologies? Explain any that you can identify and then indicate why the methodologies are considered to be different even though fundamental procedures and processes are still included.
15. Explain the differences between user views, a conceptual schema, and an internal schema as different perspectives of the same database.
16. In the three-schema architecture:
    a. The view of a manager or other type of user is called the _____schema.
    b. The view of the data architect or data administrator is called the _____schema.
    c. The view of the database administrator is called the _____schema.
17. Why might Pine Valley Furniture Company need a data warehouse?
18. As the ability to handle large amounts of data improves, describe three business areas where these very large databases are being used effectively.

Problems and Exercises

1. For each of the following pairs of related entities, indicate whether (under typical circumstances) there is a one-to-many or a many-to-many relationship. Then, using the shorthand notation introduced in the text, draw a diagram for each of the relationships.
   a. STUDENT and COURSE (students register for courses)
   b. BOOK and BOOK COPY (books have copies)
   c. COURSE and SECTION (courses have sections)
   d. SECTION and ROOM (sections are scheduled in rooms)
   e. INSTRUCTOR and COURSE
2. Reread the definitions for *data* and *database* in this chapter. Database management systems only recently began to include the capability to store and retrieve more than numeric and textual data. What special data storage, retrieval, and maintenance capabilities do images, sound, video, and other advanced data types require that are not required or are simpler with numeric and textual data?
3. Table 1-1 shows example metadata for a set of data items. Identify three other columns for these data (i.e., three other metadata characteristics for the listed attributes) and complete the entries of the table in Table 1-1 for these three additional columns.
4. In the section "Disadvantages of File Processing Systems," the statement is made that the disadvantages of file processing systems can also be limitations of databases, depending on how an organization manages its databases. First, why do organizations create multiple databases, not just one all-inclusive database supporting all data processing needs? Second, what organizational and personal factors are at work that might lead an organization to have multiple, independently managed databases (and, hence, not completely follow the database approach)?
5. Consider a student club or organization in which you are a member. What are the data entities of this enterprise? List and define each entity. Then, develop an enterprise data model (such as Figure 1-3a) showing these entities and important relationships between them.
6. A driver's license bureau maintains a database of licensed drivers. State whether each of the following represents data or metadata. If it represents data, state whether it is structured or unstructured data. If it represents metadata, state whether it is a fact describing a property of data or a fact describing the context of data.
   a. Driver's name, address, and birth date
   b. The fact that the driver's name is a 30-character field
   c. A photo image of the driver
   d. An image of the driver's fingerprint
   e. The make and serial number of the scanning device that was used to scan the fingerprint
   f. The resolution (in megapixels) of the camera that was used to photograph the driver
   g. The fact that the driver's birth date must precede today's date by at least 16 years
7. Great Lakes Insurance would like to implement a relational database for both its in-house and outside agents. The outside agents will use notebook computers to keep track of customers and policy information. Based on what you have learned in this chapter, what type (or types) of database(s) would you recommend for this application?

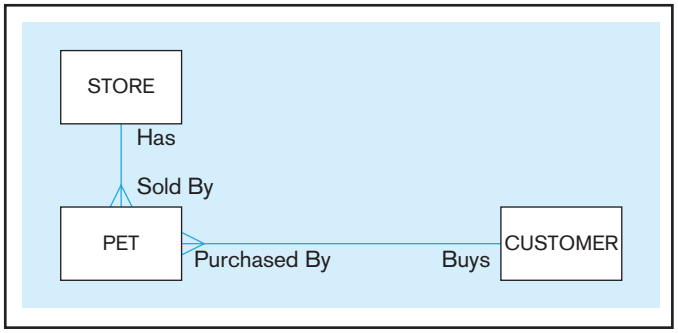


FIGURE 1-21 Data model for Problem and Exercise 8

8. Figure 1-21 shows an enterprise data model for a pet store.
   a. What is the relationship between Pet and Store (one-to-one, many-to-many, or one-to-many)?
   b. What is the relationship between Customer and Pet?
   c. Do you think there should be a relationship between Customer and Store?
9. Consider Figure 1-7, which depicts a hypothetical three-tiered database architecture. Identify potential duplications of data across all the databases listed on this figure. What problems might arise because of this duplication? Does this duplication violate the principles of the database approach outlined in this chapter? Why or why not?
10. What is your reaction to the representation of the systems development life cycle included in this chapter? Explain any problems you have with it.
11. List three additional entities that might appear in an enterprise data model for Pine Valley Furniture Company (Figure 1-3a).
12. Consider your business school or other academic unit as a business enterprise.
    a. Define several major data entity types and draw a preliminary enterprise data model (similar in notation to Figure 1-3a).
    b. Would your business school or academic unit benefit from a multiple-tiered architecture for data? Why or why not?
13. Contrast the top-down nature of database development during conceptual data modeling with the bottom-up nature of database development during logical database design. What major differences exist in the type of information considered in each of these two database development steps?
14. The objective of the prototyping systems development methodology is to rapidly build and rebuild an information system as the user and systems analyst learn from use of the prototype what features should be included in the evolving information system. Because the final prototype does not have to become the working system, where do you think would be an ideal location to develop a prototype: on a personal computer, department server, or enterprise server? Does your answer depend on any assumptions?
15. Explain the differences between an enterprise data model and a conceptual data model. How many databases does each represent? What scope of the organization does each address? What are other salient differences?

16. Is it possible that during the physical database design and creation step of database development you might want to return to the logical database design activity? Why or why not? If it is possible, give an example of what might arise during physical database design and creation that would cause you to want to reconsider the conceptual and external database designs from prior steps.
17. Consider an organization with which you frequently interact, such as a bank, credit card company, university, or insurance company, from which you receive several computer-generated messages, such as monthly statements, transaction slips, and so forth. Depict the data included in each message you receive from the organization as its own user view; use the notation of Figure 1-3a to represent these views. Now, combine all of these user views together into one conceptual data model, also using the notation of Figure 1-3a. What did you observe about the process of combining the different user views? Were there inconsistencies across the user views? Once you have created the conceptual data model, would you like to change anything about any of the user views?
18. Consider Figure 1-15. Explain the meaning of the line that connects ORDER to INVOICE and the line that connects INVOICE to PAYMENT. What does this say about how Pine Valley Furniture Company does business with its customers?
19. Answer the following questions concerning Figures 1-16 and 1-17:
    a. What will be the field size for the ProductLineName field in the Product table? Why?
    b. In Figure 1-17, how is the ProductID field in the Product table specified to be required? Why is it a required attribute?
    c. In Figure 1-17, explain the function of the FOREIGN KEY definition.
20. Consider the SQL query in Figure 1-19.
    a. How is Sales to Date calculated?
    b. How would the query have to change if Helen Jarvis wanted to see the results for all of the product lines, not just the Home Office product line?
21. Helen Jarvis wants to determine the most important customers for Home Office products. She requests a listing of total dollar sales year-to-date for each customer who bought these products, as revealed by invoiced payments. The list is to be sorted in descending order, so that the largest customer heads the list.
    a. Look at Figure 1-18 and determine what entities are required to produce this list.
    b. Which entities will be involved in the SQL query that will give Helen the information she needs?
22. In this chapter, we described four important data models and their properties: enterprise, conceptual, logical, and physical. In the following table, summarize the important properties of these data models by entering a Y (for Yes) or an N (for No) in each cell of the table.

Table for Problem and Exercise 22

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

Field Exercises

For Questions 1 through 7, choose an organization with a fairly extensive information systems department and set of information system applications. You should choose one which you are familiar, possibly your employer, your university, or an organization where a friend works. Use the same organization for each question.

1. Investigate whether the organization follows more of a traditional file processing approach or the database approach to organizing data. How many different databases does the organization have? Try to draw a figure, similar to Figure 1-2, to depict some or all of the files and databases in this organization.
2. Talk with a database administrator or designer from the organization. What type of metadata does this organization maintain about its databases? Why did the organization choose to keep track of these and not other metadata? What tools are used to maintain these metadata?
3. Determine the company's use of intranet, extranet, or other Web-enabled business processes. For each type of process, determine its purpose and the database management system that is being used in conjunction with the networks. Ask what the company's plans are for the next year with regard to using intranets, extranets, or the Web in their business activities. Ask what new skills they are looking for in order to implement these plans.
4. Consider a major database in this organization, such as one supporting customer interactions, accounting, or manufacturing. What is the architecture for this database? Is the organization using some form of client/server architecture? Interview information systems managers in this organization to find out why they chose the architecture for this database.
5. Interview systems and database analysts at this organization. Ask them to describe their systems development process. Does it resemble more the systems development life cycle or prototyping? Do they use methodologies similar to both? When do they use their different methodologies? Explore the methodology used for developing applications to be used through the Web. How have they