

Introduction

Chapter Outline

- Introduction, 2
- Two-Minute Chapter, 3
- A Small Sample Database Application, 4
- Databases and Application Development, 6
- Components of a Database Management System, 10
 - Database Engine, 10*
 - Data Dictionary, 11*
 - Query Processor, 12*
 - Report Service, 13*
 - Forms Development, 14*
 - Management Utilities and Security, 15*
- Advantages of the Database Management System Approach, 16
 - Focus on Data, 17*
 - Data Independence, 18*
 - Data Independence and Web Applications, 19*
- Leading Commercial Database Systems, 20
- The Evolution of Database Management Systems, 21
 - Hierarchical Databases, 21*
 - Network Databases, 23*
 - Relational Databases, 23*
 - Object-Oriented Databases, 24*
- Key-Value Pairs: Cassandra, 28
- Drawbacks to Database Management Systems, 29
- Application Development, 30
- Introduction to this Book's Databases, 31
 - Sally's Pet Store, 31*
 - Corner Med, 32*
 - Rolling Thunder Bicycles, 33*
- Starting a Project: The Feasibility Study, 33
 - Costs, 33*
 - Benefits, 35*
- Summary, 36
- Key Terms, 37
- Review Questions, 37
- Exercises, 38
- Web Site References, 41
- Additional Reading, 41

What You Will Learn in This Chapter

- What is a database?
- What do database applications look like?
- How are databases used to build applications?
- What are the major components of a database management system?
- What are the advantages of using a database management system?
- What are the main database management systems?
- How have database management systems changed over time?
- What potential problems exist with a DBMS approach?
- What is an application?
- What databases are used with this book?
- What are the first steps to start a project?

A Developer's View

Miranda: My uncle just called me and said his company was desperate. It needs someone to build an application for the sales team. The company wants a laptop system for each salesperson to enter orders. The system needs to track the order status over time and generate notices and weekly reports. My uncle said that because I know a lot about computers, I should call and get the job. His company is willing to pay \$6,000, and I can work part-time.

Ariel: Wow! Sounds like a great job. What's the problem?

Miranda: Well, I know how to use basic computer tools, and I can program a little, but I'm not sure I can build a complete application. It could take a long time.

Ariel: Why not use a database management system? It should be easier than writing code from scratch.

Miranda: Do you really think so? What can a database system do? How does it work?

Getting Started

You need to choose a DBMS to use for exercises and projects. The Workbooks support Microsoft Access, Microsoft SQL Server, and Oracle. You can also use any SQL-based DBMS for most of the chapters. If necessary, install the DBMS on your computer—this step can take a few hours if you need to download the software. This chapter describes the basic features of a DBMS, but you will learn the details in the later chapters. You should read the descriptions of the sample databases, install them, and check out some of the data.

Introduction

What is a database? Do you want to build computerized business applications? Do you want to create business applications that operate in multiple locations? Do you want to conduct business on the Internet? Do you want to enable customers to place orders using the Web? If you are going to build a modern business application, you need a database management system.

Think about applications that you use: Almost any Web application, student course registration or billing systems, calendars, even games. All of them need to store data. If you are creating an application you have to identify the data to be stored and the best location to store it. You could use file read/write tools to save the data in a proprietary format that could only be read by your custom application. Or, you could store the data using a database management system. **database management system (DBMS)** is a software tool created to solve the common problems of sharing data among multiple users and applications. It has many features that make it easy to store and retrieve data efficiently.

The alternative to using a DBMS is to write file storage routines for every application that you create. For each application, you could store data in separate files, but only your application would know how to retrieve that data. Imagine

how difficult it would be for you to write a new application that uses data created by a program that someone else wrote ten years ago. Also, for every file-based application, you would have to rewrite the data-handling routines to deal with multiple users accessing the data at the same time (concurrent access). You would also have to provide security and data management routines. It is far easier and more reliable to use a database management system. It already has these features and more, so you can concentrate on building an application that meets the needs of the users.

Business applications often utilize common types of data—information about customers, employees, products, sales, purchases, and so on. A database system is one of the most powerful tools you can use to build business applications because it easily handles common business data, supports security controls, and sharing. Most systems also have query systems, powerful report writers, and application systems that make it easy to quickly build applications and retrieve data to support common business needs.

The most important features of a DBMS are the ability to define a database, store the data efficiently, and retrieve data with a query language. A **database** is a collection of data stored in a standardized format designed to be shared by multiple users. These concepts and the tools are discussed in detail in this book. The key point to remember now is that a database is independent of any specific application. Once you create a database within a DBMS, the data can be accessed with a variety of tools. The DBMS provides many tools to manage the data including security controls, data storage options, and backup facilities.

This chapter describes the basic role of a DBMS in application development. It also describes the major features of a DBMS and how you will use them in building business applications. It also summarizes the evolution of DBMS technology so you understand some of the background and can think about possible changes for future systems. This book focuses on building applications that use databases. It avoids detailed discussions of how database management systems are written.

Two-Minute Chapter

Chapter 1 is an introduction to databases and what they provide to developers of business applications. The main purpose is to explain their importance and outline how this book can be used to learn how to build and create applications using database systems. Database management systems are powerful tools that are used in almost all business applications. They solve many common problems for storing and accessing data and maintaining the integrity of the data with multiple users making changes. But, databases must be carefully designed to obtain these benefits. Relational databases are the most common tools in business and a set of normalization rules are used to identify exactly which columns belong in a table and when data need to be split into multiple tables with additional key columns. Chapters 2 and 3 focus on the rules for designing tables. One of the main strengths of a DBMS is the separation of the data from the application. By concentrating on the data storage, the data remains protected and accessible to almost any application. Tools and applications can change, and the data remains useful and accessible.

Newer key-value pair highly-distributed databases, such as Cassandra, are designed for massive Web sites with millions of users, where performance of a few specific queries takes priority over everything else. Database design is also critical for these tools, but it is not as rigid—which means that experience and experimentation are needed to determine how to optimize the storage and retrieval for

each specific problem. Chapter 13 explores the specific details of key-value pair databases, but it relies on basic understanding of relational design and queries—at least chapters 2 and 4.

Most DBMSs include the data storage engine and a query processor. SQL is a standard language used by most relational systems. Basic queries are straightforward, but SQL has powerful features to help answer complex business questions. Chapters 4 and 5 focus on constructing SQL queries to answer common business questions.

Ultimately, databases are not built in isolation are part of an application. The database is likely to be invisible to most users. Instead, users interact with the application through forms and reports. These tools can be created with desktop tools or as Web-based forms. The capabilities and tools for building forms and reports present the greatest differences between database tools. Eventually, developers need to learn to use at least one set of tools in depth. The basic skills are transferable to other tools, but a lack of standards requires learning picky details for each system. Chapters 6, 7, and 8 explain the process in general; but individual details for specific tools are covered in the accompanying workbooks.

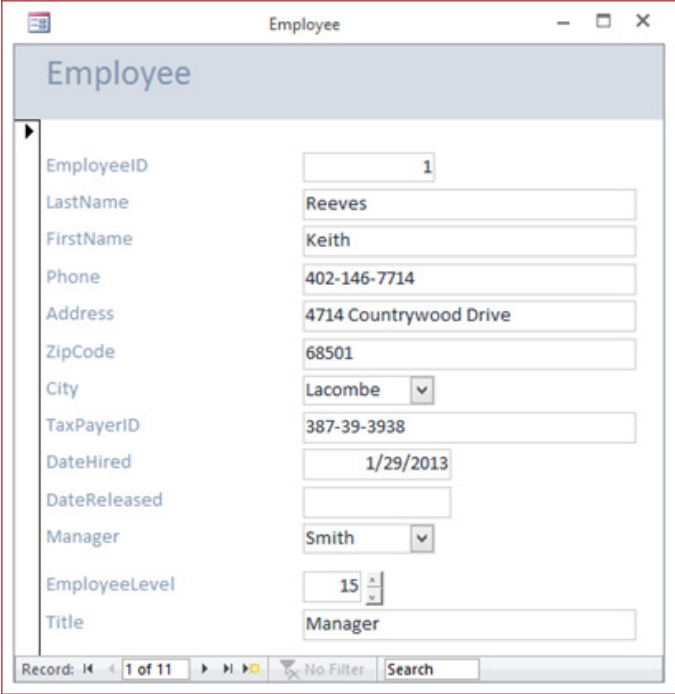
Increasingly, managers are looking to expand their use and understanding of a huge amount of data being collected. Much of this analysis requires statistical knowledge and tools that require additional background. However, the data storage needs are slightly different, so data warehouses and some basic analytical tools are covered in Chapter 9.

Chapter 10 examines some of the issues involved in setting up and maintaining a database. Security is important in all business applications today and DBMSs provide tools for assigning and monitoring various security conditions. Chapter 11 looks at the basic issues involved in distributed systems—where data is stored across multiple servers to improve performance and reliability. Chapter 12 is a bridge to computer science. DBMSs use specific methods to store and retrieve data. These topics are critical in computer science classes, and the chapter briefly shows how they are used to build the DBMS software; which highlights some of the strengths and weaknesses of the software.

A Small Sample Database Application

What do database applications look like? You have probably worked with several database-oriented applications and were not aware of the role of the database or the DBMS. In a business application, users do not care about the underlying data storage mechanism—they are only interested in the final application features and usability. But a DBMS can provide amazing functionality with less effort than programming for many applications. Before trying to explain the functions and benefits of a DBMS it is useful to look at some of the basic application features. This book uses several sample databases to illustrate the various capabilities and features. You should download them from the Web site and look through the applications. Note that the versions in Microsoft Access have more elements (forms and reports) than those in the server-based systems. These versions are also easier to examine—but you need a copy of Microsoft Access software.

A couple of examples from the Pet Store database are useful to understand how a database can be used to create business applications. The Pet Store database is a partially-completed application. It has all of the tables and a small amount of sample data. But only a couple of forms and reports were created. The Bicycle application is much larger and contains several useful and more complex forms.



The screenshot shows a web-based form titled "Employee" with a light blue header. The form contains the following fields and values:

| Field | Value |
|---------------|------------------------|
| EmployeeID | 1 |
| LastName | Reeves |
| FirstName | Keith |
| Phone | 402-146-7714 |
| Address | 4714 Countrywood Drive |
| ZipCode | 68501 |
| City | Lacombe |
| TaxPayerID | 387-39-3938 |
| DateHired | 1/29/2013 |
| DateReleased | |
| Manager | Smith |
| EmployeeLevel | 15 |
| Title | Manager |

At the bottom of the form, there is a record navigation bar showing "Record: 1 of 11" and a search box with the text "No Filter" and a "Search" button.

Figure 1.1

Sample Employee form. Users see a form with controls to help them enter and edit data. The data items are stored in the database but the form could be located on a single computer, a Web site, or even a mobile application.

The Bicycle case is good for examining detailed features of a mostly finished application. But it is easier to see the process of creating an application by looking at the partially finished Pet Store or CornerMed examples. Figure 1.1 shows a simple Employee form which is used to add data for new employees or edit values for existing workers. The form focuses attention on a single Employee at a time. It contains controls to make some tasks easier including selecting cities (or managers) from a list.

The Employee form is relatively simple because it shows only one concept: information about a person. Business applications are often more complex. Figure 1.2 shows a basic purchase order form, where the Pet Store is buying bulk items from a supplier. This form needs to display information about the order itself: Dates, the supplier, and total cost. It also needs to collect data on the individual items being purchased such as the cost and the quantity. The form is also capable of computing totals automatically. In this example, the repeating Value column, Subtotal, and Total values are computed based on arithmetic formulas that are programmed as properties in the form. Many additional features can be added such as filters to show all orders from a single supplier, orders within a range of dates, or orders that have not yet been received.

Applications usually have reports that can include tables, subtotals, and charts. Increasingly, these tools are interactive, where users can click buttons to compare totals across various categories or quickly create charts to see how values

MerchandiseOrder

PO Number: 1 Order Date: 3/6/2013

Employee: Hopkins Receive Date: 3/8/2013

Supplier: Rhodes

| ItemID | Description | Quantity | Cost | Value |
|--------|------------------------|----------|---------|----------|
| 27 | Aquarium Filter & Pump | 8 | \$24.65 | \$197.20 |
| 30 | Flea Collar-Dog-Medium | 208 | \$4.42 | \$919.36 |
| * | | | | |

Record: 1 of 2 No Filter Search

Subtotal: \$1,116.56

Shipping Cost: \$33.54

Total: \$1,150.10

Record: 1 of 25 No Filter Search

Figure 1.2

Sample Purchase Order form. The order form is more complex and handles data entry for the order itself as well as the individual items being purchased in the detail/repeating section.

change over time. These capabilities are often built into the DBMS tools and can sometimes be created in a few minutes by a skilled developer. In other cases, a programmer needs to write custom applications that provide tools to the users and interact with the database to store and retrieve the desired data.

The key thing to remember is that all of the data is handled by the DBMS. Even if the forms and reports are created with traditional programming tools, the DBMS stores and controls the data centrally. And the DBMS handles security controls, simultaneous access by multiple users, data backup, and data integrity issues—such as preventing negative values for prices.

Databases and Application Development

How are databases used to build applications? It is rare that someone would ask you to build a database and just use it to store and retrieve data. In almost all situations, someone asks you to build an application. The difference is that an application is used to perform specific tasks. In the process, you will use the DBMS to store and retrieve the data, but ultimately, you are building the database as part of the solution to the user's problem.

As shown in Figure 1.3, the database itself is just one element of the application. Developers define tables to hold the data in the DBMS. Application forms are screens displayed to the user to collect or display data. The data is stored in the underlying tables. Reports are structured displays of data from the tables—typically containing subtotals and charts. In new applications, these forms and reports are accessible using a Web browser. Applications built with older systems might require database components installed on each computer.

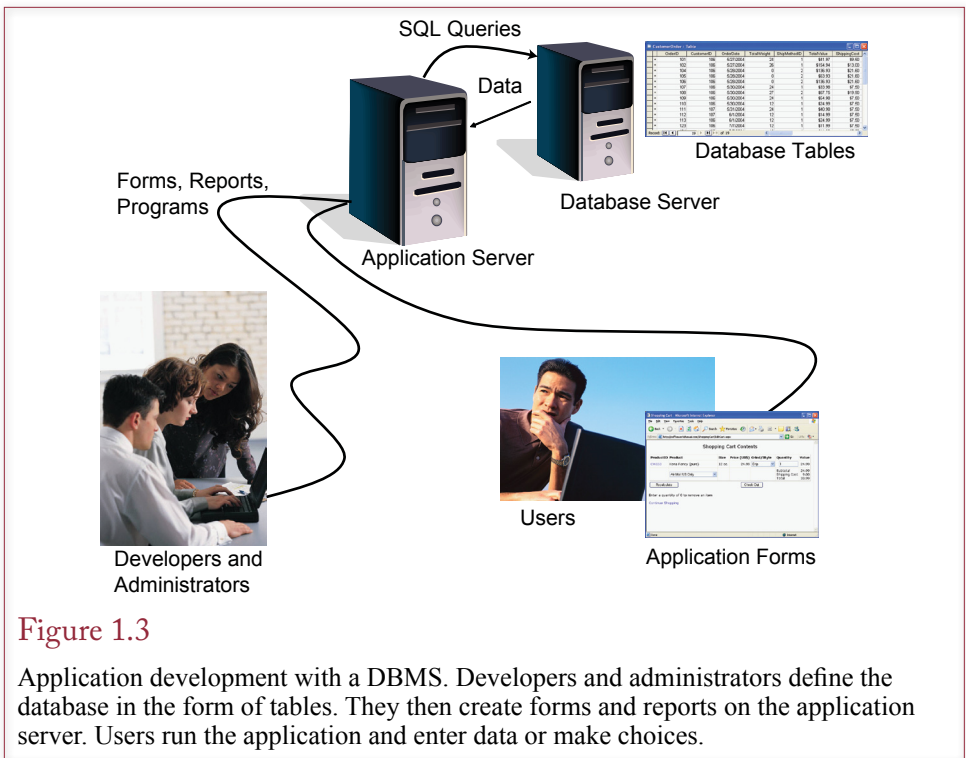


Figure 1.3

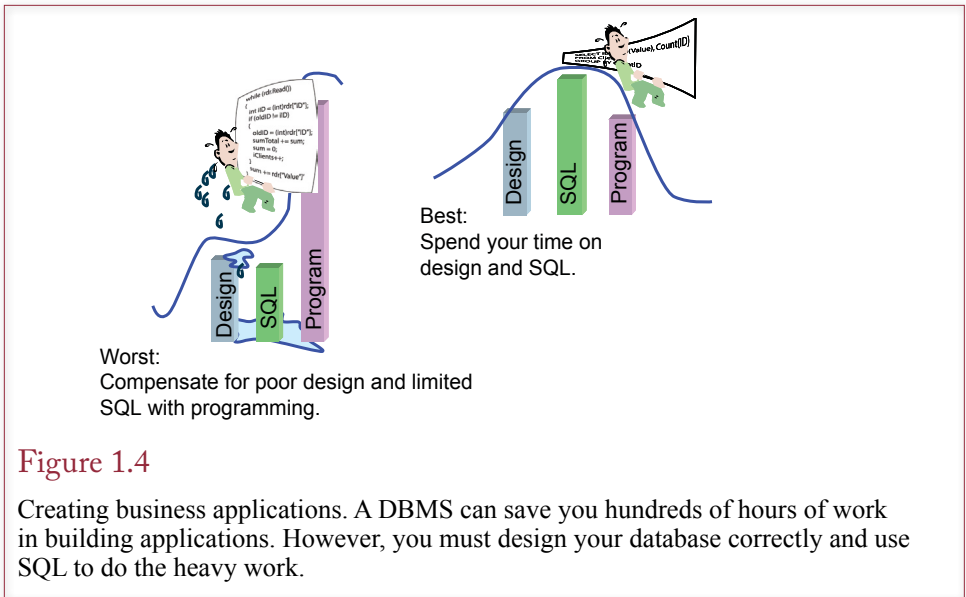
Application development with a DBMS. Developers and administrators define the database in the form of tables. They then create forms and reports on the application server. Users run the application and enter data or make choices.

Several different DBMS tools exist today and one of the first decisions you must make is to select the DBMS for the application being built. In many situations, this choice has already been made for you. For example, your instructor has probably already selected a DBMS, or within a large company, one tool is used as the standard platform. The Workbooks that accompany this textbook provide details about several specific DBMS platforms, including Microsoft Access, Microsoft SQL Server, and Oracle. Each of them uses different tools to create forms and reports, but the Workbooks provide examples and steps for using those tools.

Storing and retrieving data is relatively standard today—most of the DBMSs use the SQL query language. Designing the database tables and retrieving the data are critical tasks that are essentially the same regardless of the DBMS. Chapters 2 and 3 of this book focus on database design. Chapters 4 and 5 explore the power of SQL to retrieve and manipulate data. These chapters and tasks apply to almost any DBMS.

Building applications sometimes requires writing programming code. Some tools require more programming than others. Some tools have their own internal programming language, while others rely on standard languages (such as C++ or Java), and embed the database elements as extensions to the language. In any case, database applications will be easier to understand if you have already had at least one programming course.

The development process for a DBMS is somewhat different from traditional programming. One of the key changes is that your primary focus is on the data and how it is organized. Later, you can build forms and reports. To gain the advantages, data must be carefully organized. The query language is also a powerful component of a DBMS. It makes it easy to retrieve data—usually with a few



lines of simple commands. Once you understand the concepts of database design, queries, and application building, you will be able to create complex applications in a fraction of the time it would take with traditional programming techniques. Figure 1.4 illustrates the tradeoffs that you face in building applications. It is critical that you spend time and design your database correctly. You also need to use the query language (SQL) to do the heavy work in retrieving data. With these two tools, your application programming becomes easy and you can spend most of your time building forms and reports with automated tools. It still takes time and effort, but it is considerably easier than relying on detailed coding.

In the last few years, database systems have become the foundation of almost all application development projects. From large enterprise resource planning systems, to e-business Web sites, to standalone business applications, database systems store and retrieve data efficiently, provide security, and make it easier to build the applications. Today, when you build or modify an application, you will first create the database. To understand the capabilities of a DBMS and how you will use them to create applications, it is best to examine the process used to develop applications.

Organizations typically follow the basic steps outlined in Figure 1.5 when creating technology applications. Larger projects may require several people in each phase, whereas smaller projects might be created entirely by one or two developers. Organizations can rearrange the tasks that fall within each step, but all of the tasks must be completed for a project to be successful. The feasibility step defines the project and provides estimates of the costs. During the analysis phase, systems analysts collect data definitions, forms, and reports from users. These are used to design the database and all of the new forms, reports, and user interactions. During the development step, the forms, reports, and application features such as help files are created. Implementation generally consists of the transfer of data, installation, training, and review.

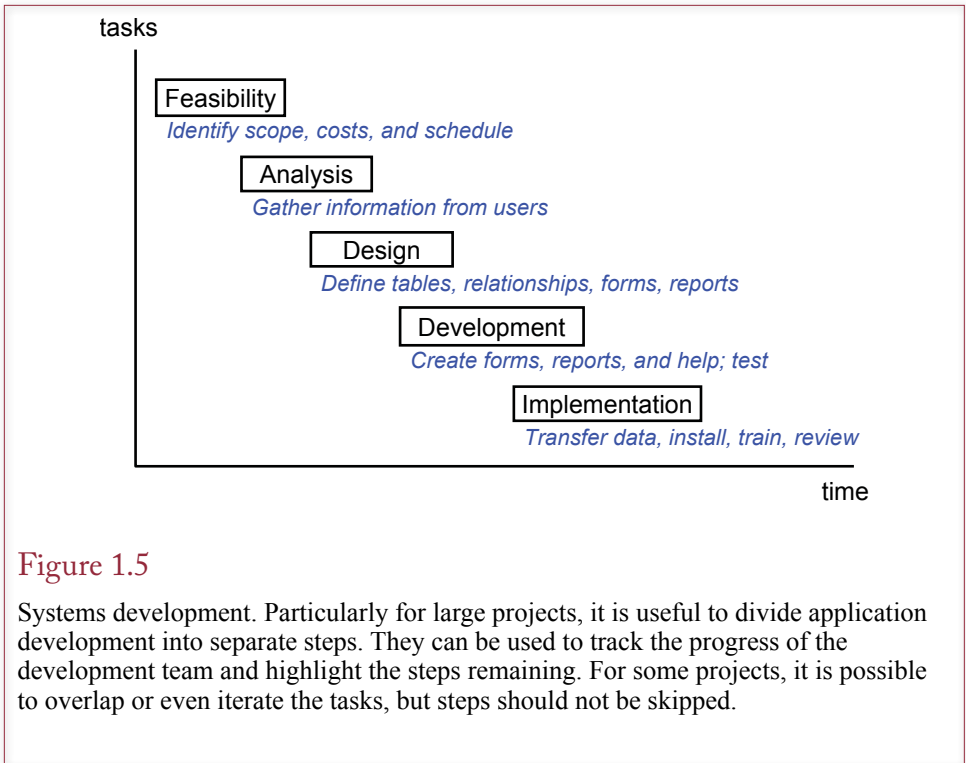
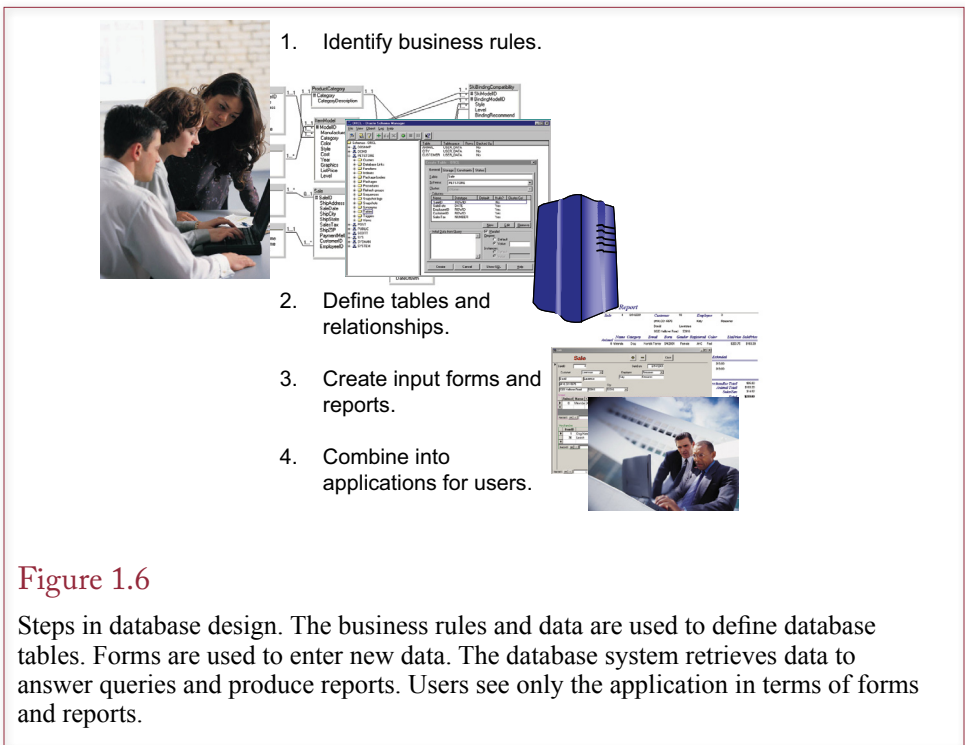


Figure 1.5

Systems development. Particularly for large projects, it is useful to divide application development into separate steps. They can be used to track the progress of the development team and highlight the steps remaining. For some projects, it is possible to overlap or even iterate the tasks, but steps should not be skipped.

For database-driven applications, the design stage is critical. Database systems and the associated development tools are incredibly powerful, but databases must be carefully designed to take advantage of this power. Figure 1.6 shows that the business rules and processes are converted into database tables and relationship definitions. Forms are defined that transfer data into the database, and reports use queries to retrieve and display data needed by users. These forms and reports, along with features such as menus and help screens, constitute applications. Users generally see only the application and not the underlying database or tables.

Designing the database tables and relationships is a key step in creating a database application. The process and rules for defining tables are detailed in Chapters 2 and 3. Using the database requires the ability to retrieve and manipulate the data. These tasks are handled by the query system, which is described in Chapters 4 and 5. With these foundations, it is relatively easy to use the tools to create forms and reports and build them into applications as discussed in Chapters 6, 7, and 8.



Components of a Database Management System

What are the major components of a database management system? To understand the value of a DBMS, it helps to see the components that are commonly provided. This basic feature list is also useful when you evaluate various products to determine which DBMS your company should use. Each DBMS has unique strengths and weaknesses. You can evaluate the various products according to how well they perform in each of these categories. The primary categories are the database engine, query processor, report service, forms development, management tools, and security.

Database Engine

The **database engine** is the heart of the DBMS. It is responsible for storing, retrieving, and updating the data. This component is the one that most affects the performance (speed) and the ability to handle large problems (scalability). The other components rely on the engine to store not only the application data but also the internal system data that defines how the application will operate. Figure 1.7 illustrates the primary relationship between the database engine and the data tables.

With some systems the database engine is a stand-alone component that can be purchased and used as an independent software module. For example, the Microsoft “jet engine” forms the foundation of Access. Similarly, the database engines for Oracle and Microsoft SQL Server can be purchased separately.

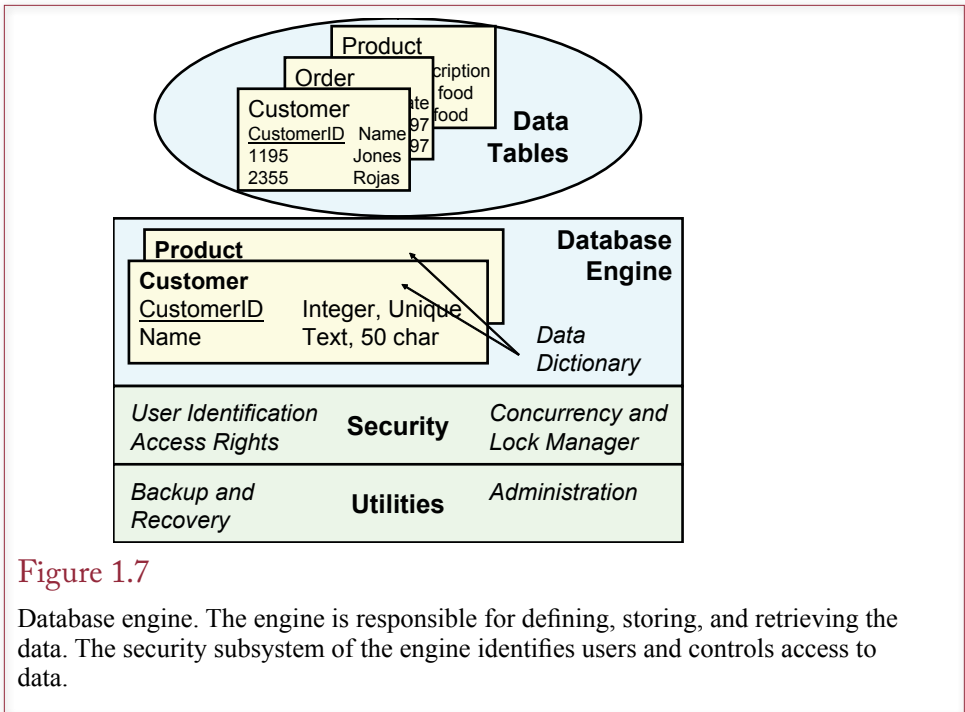


Figure 1.7

Database engine. The engine is responsible for defining, storing, and retrieving the data. The security subsystem of the engine identifies users and controls access to data.

The database engine is also responsible for enforcing business rules regarding the data. For example, most businesses would not allow negative prices to be used in the database. Once the designer creates that rule, the database engine will warn the users and prevent them from entering a negative value.

As shown in Figure 1.8, the database engine stores data in carefully designed tables. Tables are given names that reflect the data they hold. Columns represent simple attributes that describe the object, such as an employee's name, phone, and address. Each row represents one object in the table.

Database performance is an important issue. The speed of your application depends on the hardware, the DBMS software, the design of your database, and on how you choose to store your data. Chapter 12 discusses some popular methods, such as indexing, that improve the performance of a database application. Performance is also affected by how the software is written. Tools such as Microsoft Access have limitations on the size of the database and on how the data is processed. Similarly, free tools, including versions for Microsoft SQL Server, Oracle, and IBM, have limits on size and processing (such as support for only one processor). More expensive versions and other software tools remove these limitations.

Data Dictionary

The **data dictionary** holds the definitions of all of the data tables. It describes the type of data that is being stored, allows the DBMS to keep track of the data, and helps developers and users find the data they need. Most modern database systems hold the data dictionary as a set of system tables. For example, Microsoft Access keeps a list of all the tables in a hidden system table called `MsysObjects`. The larger systems like SQL Server and Oracle also have proprietary tables such as `sys.dba_tables` in Oracle. However, most of the vendors (except Oracle) have

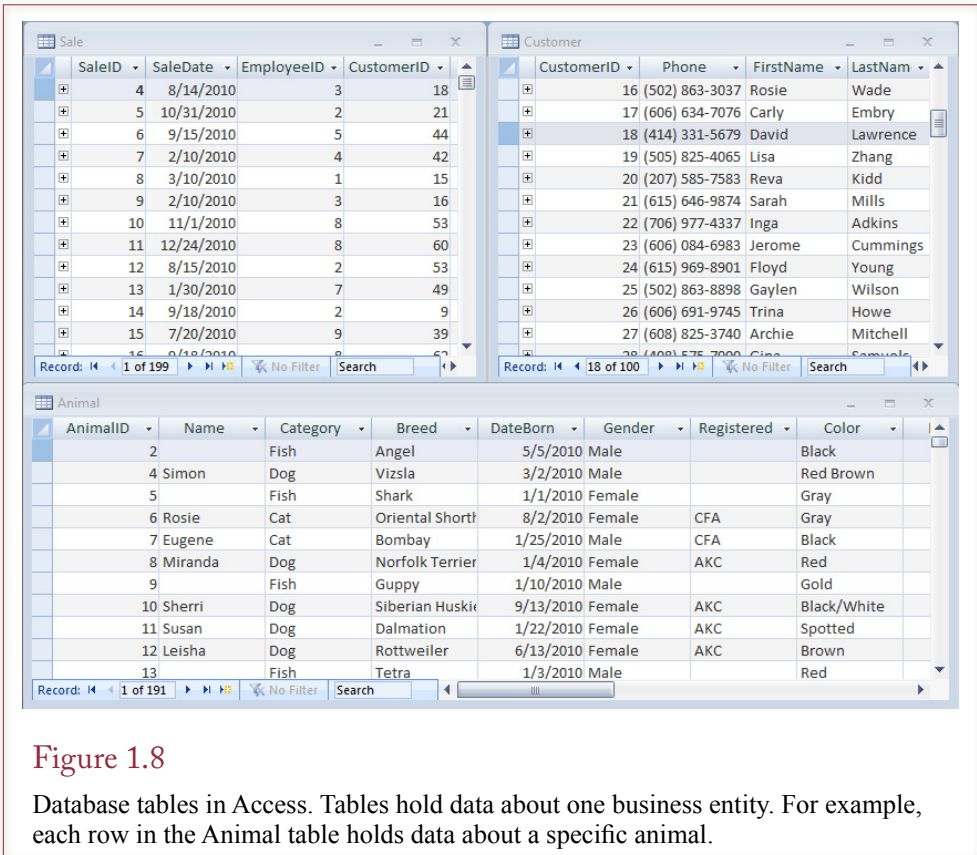


Figure 1.8

Database tables in Access. Tables hold data about one business entity. For example, each row in the Animal table holds data about a specific animal.

standardized on the Information_Schema queries, such as the Information_Schema.Tables view. These tools and related administrative issues are described in Chapter 10. If you need to install your own copy of the DBMS software now, you should read the basic steps in Chapter 1 of the associated Workbook. You can also skim through Chapter 10 if you want more detailed explanations.

These meta-data tables are used by the system, but most database systems also provide visually-oriented administration tools so you do not have to memorize commands. For example, it is relatively easy to obtain a list of tables using the basic administration tools for Access, SQL Server, Oracle, and DB2. For independent tools like MySQL, you will probably have to track down and install a separate management utility.

Query Processor

The query processor is a fundamental component of the DBMS. It enables developers and users to store and retrieve data. In some cases the query processor is the only connection you will have with the database. That is, all database operations can be run through the query language. Chapters 4 and 5 describe the features and power of query languages—particularly standard SQL.

Queries are derived from business questions. The query language is necessary because natural languages like English are too vague to trust with a query. To minimize communication problems and to make sure that the DBMS understands

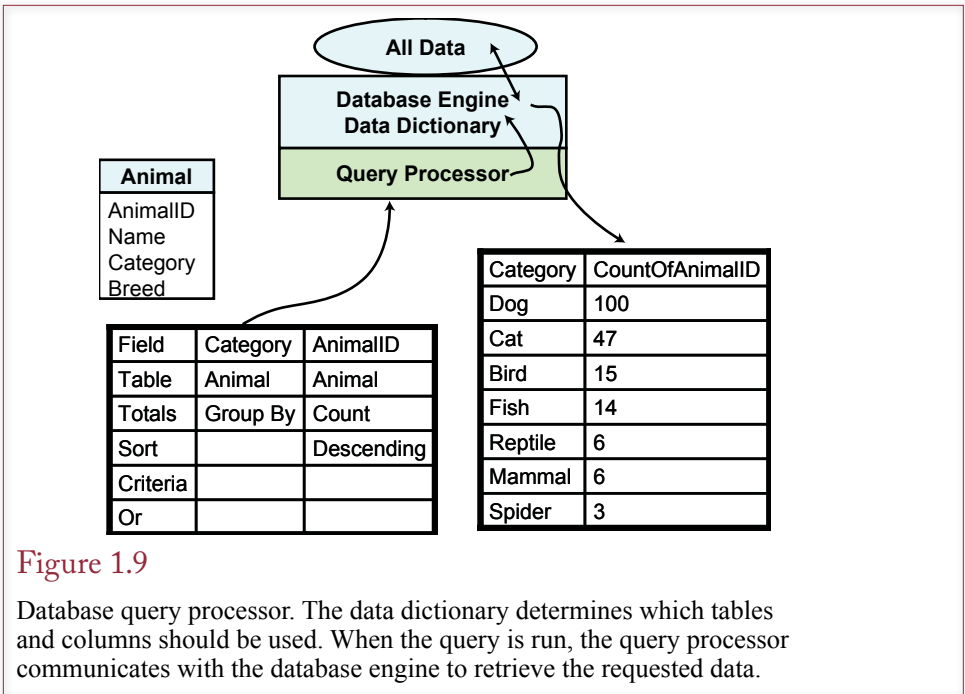


Figure 1.9

Database query processor. The data dictionary determines which tables and columns should be used. When the query is run, the query processor communicates with the database engine to retrieve the requested data.

your question, you should use a query language that is more precise than English. As shown in Figure 1.9, the DBMS refers to the data dictionary to create a query. When the query runs, the DBMS query processor works with the database engine to find the appropriate data. The results are then formatted and displayed on the screen.

Report Service

Most business users want to see summaries of the data in some type of report. Many of the reports follow common formats. A **report writer** enables you to set up the report on the screen to specify how items will be displayed or calculated. Most of these tasks are performed by dragging data onto the screen. Professional-level writers enable you to produce complex reports in a short time without writing any program code. Chapter 8 describes several of the common business reports and how they can be created with a database report writer. Increasingly, vendors are shipping **report services** that run as Web applications to make it easy to deliver your reports to users without relying on paper. Users can choose to explore the data interactively, or print the report on their own printer.

The process of exploring data interactively is increasingly important. The system demands for this type of application are quite different from traditional transactions and reporting systems. Consequently, most companies rely on separate report services and **online analytical processing (OLAP)**, and statistical **data mining** tools. Database designs and application tools are relatively new and substantially different from traditional database applications, so they are examined separately in Chapter 9.

The report writer can be integrated into the DBMS, or it can be a stand-alone application that the developer uses to generate code to create the needed report. As

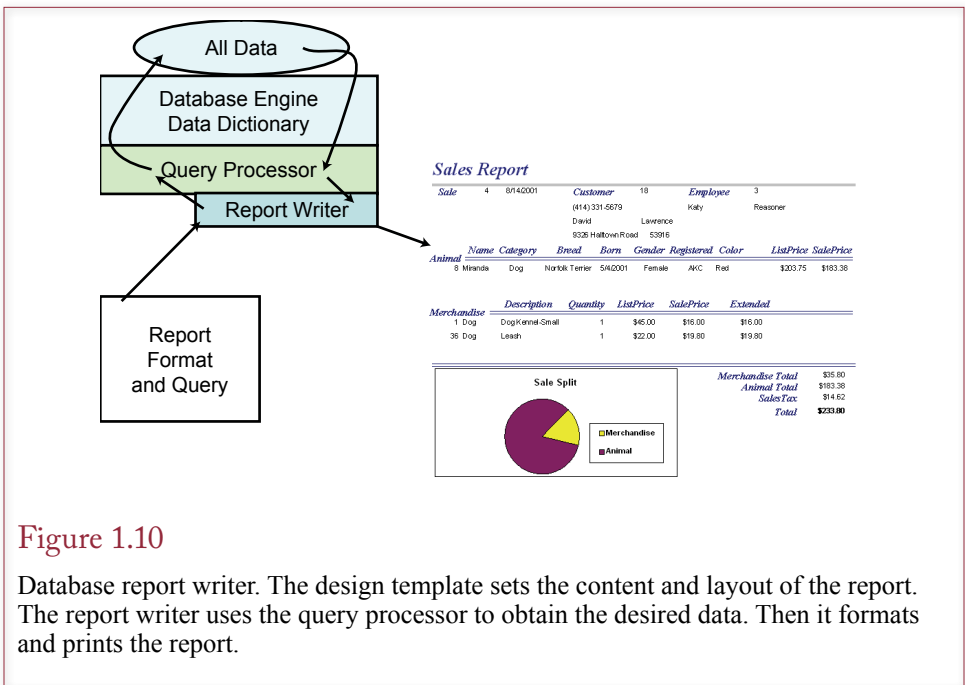


Figure 1.10

Database report writer. The design template sets the content and layout of the report. The report writer uses the query processor to obtain the desired data. Then it formats and prints the report.

shown in Figure 1.10, the developer creates a basic report design. This design is generally based on a query. When the report is executed, the report writer passes the query to the query processor, which communicates with the database engine to retrieve the desired rows of data. The report writer then formats the data according to the report template and creates the report complete with page numbers, headings and footers.

Figure 1.11 shows the report writer that Microsoft SQL Server provides with its Business Intelligence Reporting Services tool. The report writer generates reports that are posted to a Web site to be run by other users. You set up sections on the report and display data from the database. The report writer includes features to perform computations and format the columns. You also have control over colors, you can place images on the report (e.g., logos), and you can draw lines and other shapes to make the report more attractive or to call attention to specific sections.

Forms Development

A **forms builder** or input screen editor helps the developer create input forms. As described in Chapter 7, the goal is to create forms that represent common user tasks, making it easy for users to enter data. The forms can include graphs and images. The forms builder enables developers to create forms by dragging and dropping items on the screen. Figure 1.12 shows that forms make heavy use of the query processor to display data on the form.

Many database systems also provide support for traditional, third-generation languages (3GL) to access the database. The issues in writing programs and accessing data through these programs are directly related to the topics discussed in Chapter 7.

One of the most important questions you need to address for new projects is whether the application needs to be built as a Web site. Today, most new develop-

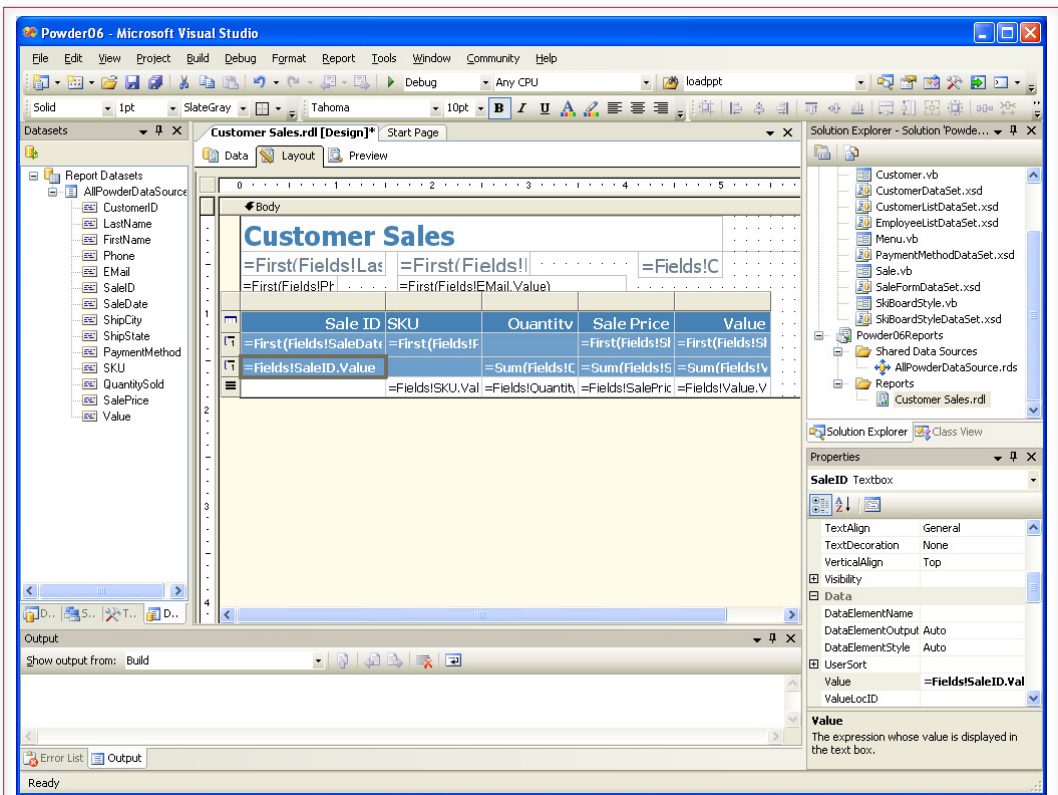


Figure 1.11

Oracle Reports report writer. The Data Model is used to create a query and select the data to be displayed. Then Reports creates the basic report layout. You can modify the layout and add features to improve the design or highlight certain sections.

ments are based on Web pages. However, the tools for building Web sites varies greatly depending on the underlying platform. For example, Microsoft uses its ASP.Net server to create and deliver pages, Java platforms (including Oracle) use the Java language and Java application servers, other tools use the open-source Apache server and often the PHP or Python programming languages to develop Web forms. The overall approach to building forms and reports is similar in these tools, but the methods and details are quite different.

Management Utilities and Security

Because data is so important to organizations, the DBMS includes several mechanisms and tools to protect the data and assign security permissions. As a result, someone needs to be in charge of assigning security, monitoring the database, and performing other management chores. A DBMS typically provides command-line tools as well as visual tools to help you perform these jobs. Chapter 10 describes the various tasks and introduces some of the commonly available tools. Typical features include backup and recovery, user management, data storage evaluation, and performance-monitoring tools.

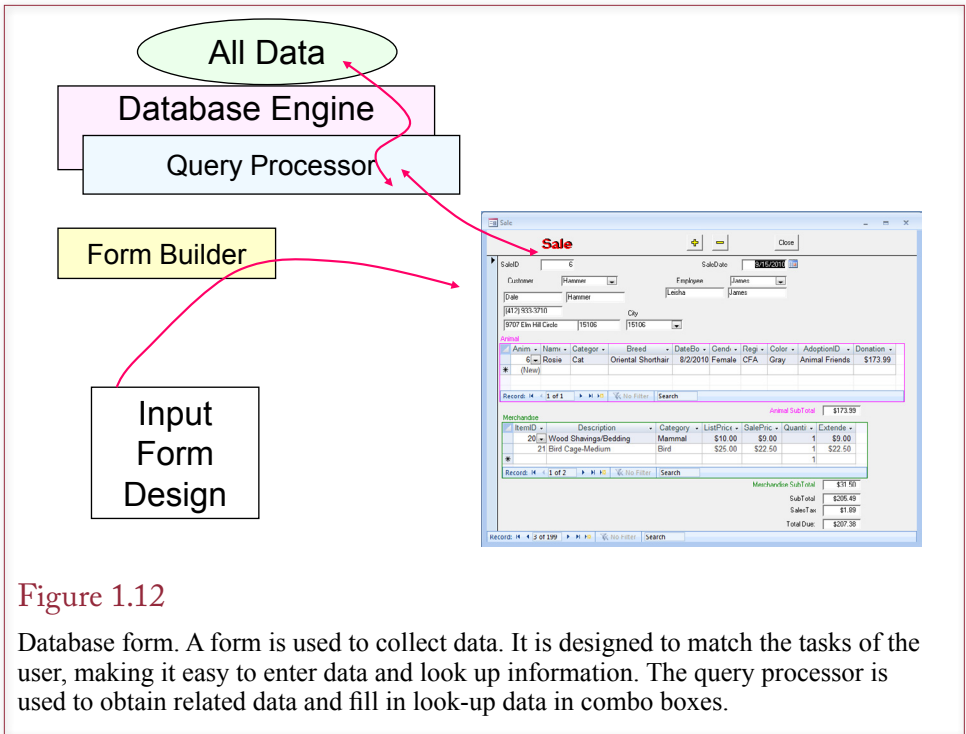


Figure 1.12

Database form. A form is used to collect data. It is designed to match the tasks of the user, making it easy to enter data and look up information. The query processor is used to obtain related data and fill in look-up data in combo boxes.

For security to work, it has to be embedded into the database engine. Consequently, you will encounter some security questions before you reach Chapter 10. In most cases, you will have a separate security account that has the permissions needed to complete most of the exercises in the book. However, if you need to share a database with dozens of other students, you might be denied the ability to perform some tasks, such as deleting data or tables. The challenge is even greater in Chapter 10. If you are serious about learning database administration tasks, you will need to install your own copy of the DBMS so that you have complete access and the ability to alter detailed elements.

Advantages of the Database Management System Approach

What are the advantages of using a database management system? Many business applications need the same features (efficient storage and retrieval of data, sharing data with multiple users, security, and so on). Rather than re-create these features within every application program, it makes more sense to purchase a database management system that includes these basic facilities. Then developers can focus on creating applications to solve business problems. The primary benefits provided by a DBMS are shown in Figure 1.13.

First, the DBMS stores data efficiently. As described in Chapters 2 and 3, if you set up your database according to a few basic rules, the data will be stored with minimal wasted space. Additionally, the data can be retrieved rapidly to answer any query. Although these two goals seem obvious, they can be challenging to handle if you have to write programs from scratch every time.

The DBMS also has systems to maintain data consistency with minimal effort. Most systems enable you to create basic business rules when you define the data. For example, price should always be greater than zero. These rules are enforced

- Minimal data redundancy
- Data consistency
- Integration of data
- Sharing of data
- Enforcement of standards
- Ease of application development
- Uniform security, privacy, and integrity
- Data independence

Figure 1.13

Advantages of a DBMS. The DBMS provides a solution to basic data storage and retrieval problems. By using a DBMS to handle data storage problems, programmers can concentrate on building applications—saving time and money in developing new systems and simplifying maintenance of existing applications.

for every form, user, or program that accesses the data. With traditional programs, you would have to force everyone to follow the same rules. Additionally, these rules would be stored in hundreds or thousands of separate programs—making them hard to find and hard to modify if the business changes.

The DBMS, particularly the query language, makes it easy to integrate data. For example, one application might collect data on customer sales. Another application might collect data on customer returns. If programmers created separate programs and independent files to store this data, combining the data would be difficult. In contrast, with a DBMS any data in the database can be easily retrieved, combined, and compared using the query system.

Focus on Data

With the old programming-file method, developers focused on the process and the program. Developers started projects by asking these kinds of questions: How should the program be organized? What computations need to be made? The database approach instead focuses on the data. Developers now begin projects by asking: What data will be collected? This change is more than just a technicality. It alters the entire development process.

Think about the development process for a minute. Which component changes the most: programs (forms and reports) or the data? Yes, companies collect new data all the time, but the structure of the data is relatively constant. And when it does change, the reason is usually that you are adding new elements—such as cellular phone numbers. In particular, business data is intentionally kept similar to enable comparisons over time. Sales, Costs, Inventory, and so on are stable numbers that are always collected. On the other hand, users constantly need modifications to forms and reports.

As shown in Figure 1.14, the database approach concentrates on the data. The DBMS is responsible for defining, storing, and retrieving the data. All requests for data must go through the database engine. Hence the DBMS is responsible for efficient data storage and retrieval, concurrency, data security, and so on. Once the data structure is carefully defined, additional tools like the report writer, forms generator, and query language make it faster and easier to develop business applications.

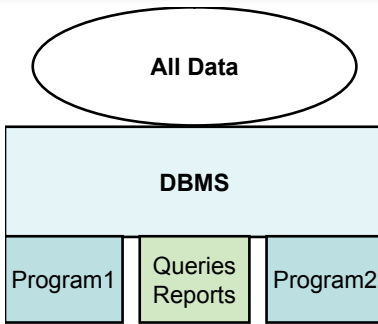


Figure 1.14

DBMS focus on data. First, define the data. Then all queries, reports, and programs access the data through the DBMS. The DBMS always handles common problems such as concurrency and security.

Data Independence

The other important feature of focusing on the data is the separation of the data definition from the program—known as **data independence**. Data independence enables you to change the data definition without altering the program. Similarly, data can be moved to new hardware or a completely different machine. Once the DBMS knows how to access the data, you do not have to alter the forms, reports, or programs that use that data. Similarly, you can alter individual programs without having to change the data definitions.

There are exceptions to this idealistic portrayal. Obviously, if you delete entire chunks of the database structure, some of your applications are not going to work properly. Similarly, if you make radical changes to the data definitions—such as changing phone number data storage from a numeric to a text data type—you will probably have to alter your reports and forms. However, a properly designed database rarely needs these radical changes.

Consider the problem of adding cell phone numbers to an Employee table. Figure 1.15 shows part of the data definition for employees. Regardless of how many forms, reports, or programs exist, the procedure is the same. Simply go to the

Figure 1.15

Adding cellular phone numbers to the Employee table. Adding a new element to a table does not affect the existing queries, reports, forms or programs.

| Field Name | Data Type | Description |
|------------|-----------|-----------------|
| EmployeeID | Number | Generated |
| TaxpayerID | Text | Federal ID |
| LastName | Text | |
| FirstName | Text | |
| ... | | |
| Phone | Text | |
| ... | | |
| CellPhone | Text | Cellular number |

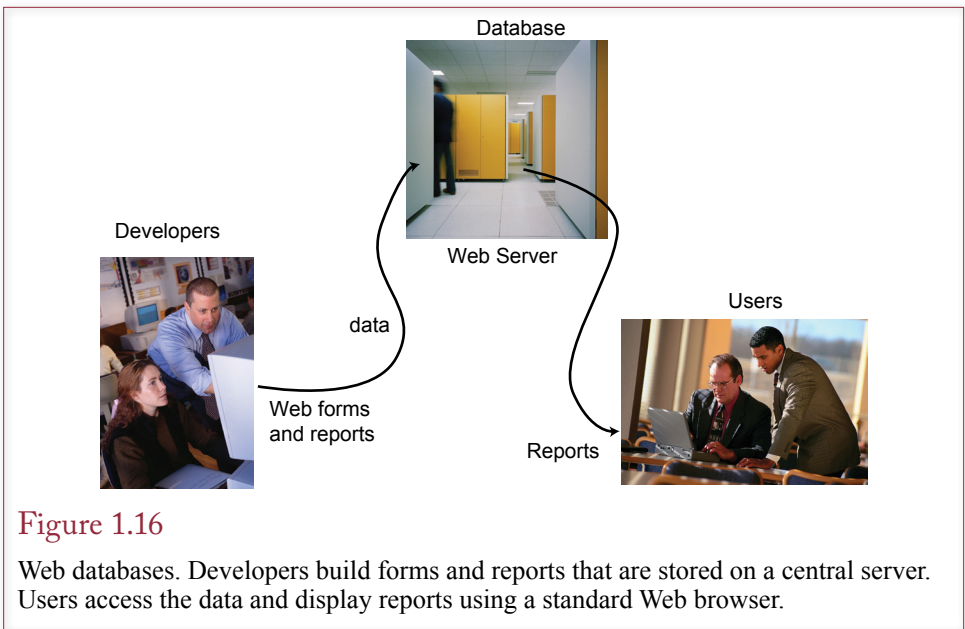


Figure 1.16

Web databases. Developers build forms and reports that are stored on a central server. Users access the data and display reports using a standard Web browser.

table definition and insert the entry for CellPhone. The existing queries, forms, reports, and programs will function exactly as they did before. Of course, they will ignore the new phone number entry. If you want to see the new values on a report, you will have to insert the new field onto the report. With a modern report writer, this change can be as simple as dragging the CellPhone item to the appropriate location on the form or report.

The focus on data and careful design enable database systems to avoid the problems experienced with traditional programming-file methods. The consolidation of common database functions within one application enables experts to create powerful database management systems and frees application programmers to focus on building applications that solve business problems.

Data Independence and Web Applications

For several years, business applications were built on a client-server model, where personal computers ran heavy applications such as DBMS and spreadsheet software. Data was shared with other users by placing it in a central database server and the individual applications connected across a network to retrieve and update the data. With the expanding use of the Web, the approach to business applications is changing. It is increasingly popular to build applications to run completely on centralized servers. With the Web approach, users only need access to a Web browser that can connect to the application server, which stores data in a database server. In many cases, the Web browser could run on simple inexpensive hardware, including cell phones. Although it raises new security issues, the Web approach also means that users can get access to the business data from almost any location.

More importantly, the Web approach makes it easier to modify the application. As shown in Figure 1.16, the data and the application forms and reports are all stored in a central location. It is easy to create new versions and change the software. In many cases, it is even possible to change the entire DBMS and the un-

| Vendor | Product |
|----------------|----------------------|
| Oracle | Oracle |
| Microsoft | SQL Server Access |
| IBM | DB2 Informix |
| Open source | PostgreSQL |
| MySQL (Oracle) | MySQL |

Figure 1.17

Commercial DBMS vendors. These are the leading DBMS products that you are likely to encounter. Many older systems exist, and dozens of smaller vendors provide complete systems and other tools.

derlying hardware. Instead of sending patches and new versions to hundreds or thousands of users, the developers simply update the single copy sitting on the application server.

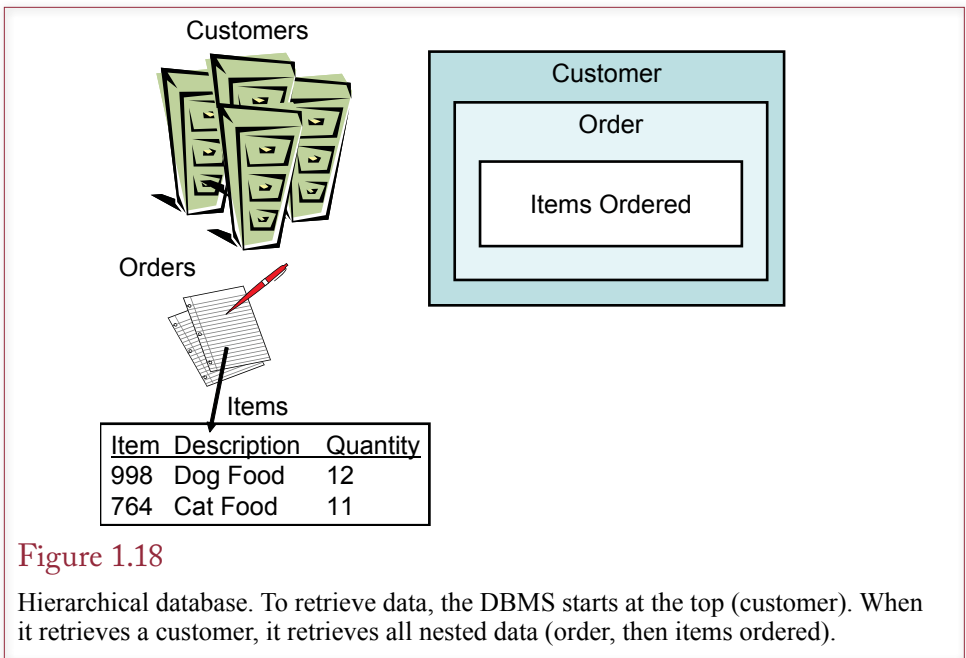
Developers can create new applications without altering the database. Similarly, they can expand the database or even move it to multiple servers, and the applications remain the same. Users continue to work with their familiar personal computer applications. Developers retain control over the data. The DBMS can monitor and enforce security and integrity conditions to protect the data, yet still give access to authorized users. Chapter 11 discusses the use of distributed database systems in more detail, including building client/server systems on the World Wide Web.

Leading Commercial Database Systems

What are the main database management systems? Figure 1.17 lists some of the leading database systems, including Oracle, DB2, and SQL Server. Many of the systems are available for multiple platforms. The PostgreSQL and MySQL tools are generally available free or at a low cost. Many other systems and tools exist, but these are the most common ones you will encounter. All of them have different strengths and weaknesses. Although the big three (Oracle, DB2, and SQL Server) can carry high price tags, the vendors can provide detailed support. All three of the vendors also provide inexpensive (or free) versions that are particularly useful for learning to use the systems. The free versions have various performance restrictions, but generally support fairly active smaller databases. You can download the free copies from the company Web sites.

Choosing a database system can be a major challenge. Many larger organizations standardize on a major vendor, negotiate reduced license costs, and make it available throughout the organization for all projects. However, if you need to choose a DBMS for a specific project, you want to carefully investigate the vendor options.

The premiere database systems are useful for large projects, offer extensive options and control over thousands of detailed features. However, these options make it difficult for beginners to understand the major concepts. It is generally best if you begin your studies with a simpler database system or stick with a smaller subset of options.



The Evolution of Database Management Systems

How have database management systems changed over time? Developers quickly realized that many business applications needed a common set of features for sharing data, and they began developing database management systems. Developers gradually refined their goals and improved their programming techniques. Many of the earlier database approaches still survive, partly because it is difficult to throw away applications that work. It is worth understanding some of the basic differences between these older methods. The following discussion simplifies the concepts and skips the details. The purpose is to highlight the differences between these various database systems—not to teach you how to design or use them.

The earliest database management systems were based on a hierarchical method of storing data. The early systems were an extension of the COBOL file structure. To provide flexible access, these systems were extended with network databases. However, the relational database approach originated by E. F. Codd eventually became the dominant method of storing and retrieving data. As programming methodologies changed to object-oriented techniques, developers started looking for ways to save internal object data in databases and some OO databases were created. With the high popularity of some Web sites such as social networks, new systems have been created to handle the huge amounts of specialized data. In particular, data is often stored in simple key-value pairs. A user uploads content, the application generates an ID value and the content is stored in a specialized DBMS to be quickly retrieved using the generated key value.

Hierarchical Databases

The **hierarchical database** approach begins by claiming that business data often exhibits a hierarchical relationship. For example, a small office without computers

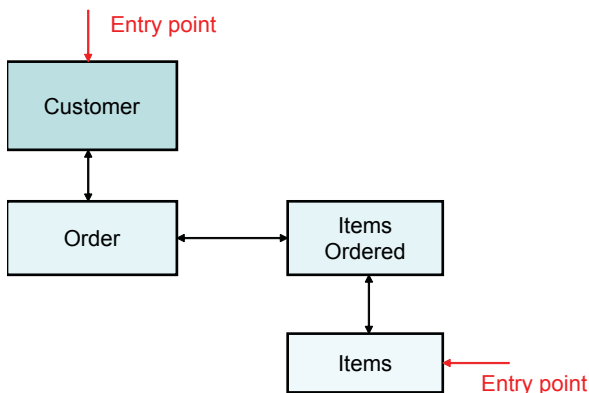
might store data in filing cabinets. The cabinets would be organized by customer. Each customer section would contain folders for individual orders, and the orders would list each item being purchased: Customer -> Orders -> Items. To store or retrieve data, the database system must start at the top—with a customer in this example. As shown in Figure 1.18, when the database stores the customer data, it stores the rest of the hierarchical data with it.

The hierarchical database approach is relatively fast—as long as you only want to access the data from the top. The most serious problem related to data storage is the difficulty of searching for items in the bottom or middle of the hierarchy. For example, to find all of the customers who ordered a specific item, the database would have to inspect each customer, every order, and each item.

The hierarchical model is an old concept in management. Many people are familiar with hierarchical objects and have a tendency to classify items using hierarchies. Consequently, hierarchical methods come into fashion every few years. However, the relational approach is substantially better at storing and retrieving data; so you have to be careful when you encounter new data formats. **Extensible markup language (XML)** is a good example. XML is a standard defined to support the transfer of data between diverse machines and companies. All data is marked with tags using angle brackets. The person or group transferring the data is free to create any labels or structure for the tags. For example, you might define an <Order> tag to transfer purchase order data. The structure of an XML file tends to be hierarchical instead of relational. It is designed to be parsed or searched from the top down. Most DBMSs have implemented the XML data type that enables you to store the raw XML file as a single unit within the database. This approach keeps the hierarchical structure of the XML file. If you use this approach, you need a way to search within the XML file. Many DBMSs support the standard **XQuery** tool for searching XML data. With this approach, you first use the relational database to locate a particular XML file, and then call XQuery to find individual items within that file. In effect, it squeezes a hierarchical dataset into a single cell in a relational database table. This approach has limitations but

Figure 1.19

Network database. All data sets must be connected with indexes as indicated by the arrows. Likewise, all entry points (starting point for a query) must be defined and created before the question can be answered.



works if you really need to keep the XML file in one piece. On the other hand, most DBMS vendors recommend that if you want to search the data received in an XML file, you should parse the data out and store the individual elements into relational database tables. Essentially, you should generally use XML as a transfer mechanism and not a storage method.

Network Databases

The **network database** has nothing to do with physical networks (e.g., local area networks). Instead, the network model is named from the network of connections between the data elements. The primary goal of the network model was to solve the hierarchical problem of searching for data from different perspectives.

Figure 1.19 illustrates the Customer, Order, and Item data components in a network model. First, notice that the items are now physically separated—typically stored in separate files. Second, note that they are connected by arrows. Finally, notice the entry points, which are indicated with arrows. The entry points are pre-defined items that can be searched. In all cases the purpose of the arrows is to show that once you enter the database, the DBMS can follow the arrows to find and display matching data. As long as there is an arrow, the database can make an efficient connection.

Although this approach seems to solve the search problem, the cost is high. All arrows must be physically implemented as indexes or embedded pointers. Essentially, an index duplicates every key data item in the associated data set and associates the item with a pointer to the storage location of the rest of the data. The problem with the network approach is that the indexes must be built before the user can ask a question. Consequently, the developer must anticipate every possible question that users might ask about the data. Worse, building and maintaining the indexes can require huge amounts of processor time and storage space.

Relational Databases

E. F. Codd originated the **relational database** approach in the 1970s, and within several years three elements came together to make the relational database the predominant method for storing data. First, theoreticians defined the basic concepts and illustrated the advantages. Second, programmers who built database management system software created efficient components. Third, hardware performance improved to handle the increased demands of the system.

Figure 1.20 illustrates how the four basic tables in the example are represented in a relational database. The key is that the tables (called “relations” by Codd) are sets of data. Each table stores attributes in columns that describe specific entities.

Figure 1.20

Relational database. Data is stored in separate sets of data. The tables are not physically connected; instead, data is linked between columns. For example, when retrieving an order, the database can match and retrieve the corresponding customer data based on CustomerID.

```
Customer(CustomerID, Name, ... )
Order(OrderID, CustomerID, OrderDate, ...)
ItemsOrdered(OrderID, ItemID, Quantity, ...)
Items(ItemID, Description, Price, ...)
```

These data tables are not physically connected to each other. The connections exist through the matching data stored in each table. For example, the Order table contains a column for CustomerID. If you find an order that has a CustomerID of 15, the database can automatically find the matching CustomerID and retrieve the related customer data.

The strength of the relational approach is that the designer does not need to know which questions might be asked of the data. If the data is carefully defined (see Chapters 2 and 3), the database can answer virtually any question efficiently (see Chapters 4 and 5). This flexibility and efficiency is the primary reason for the dominance of the relational model. Most of this book focuses on building applications for relational databases.

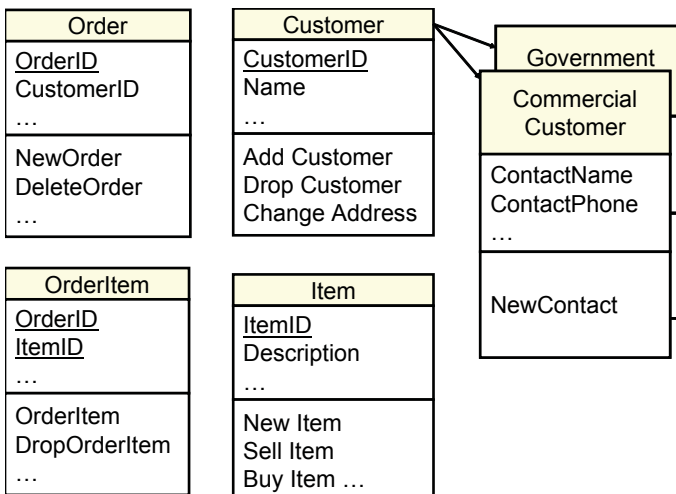
Object-Oriented Databases

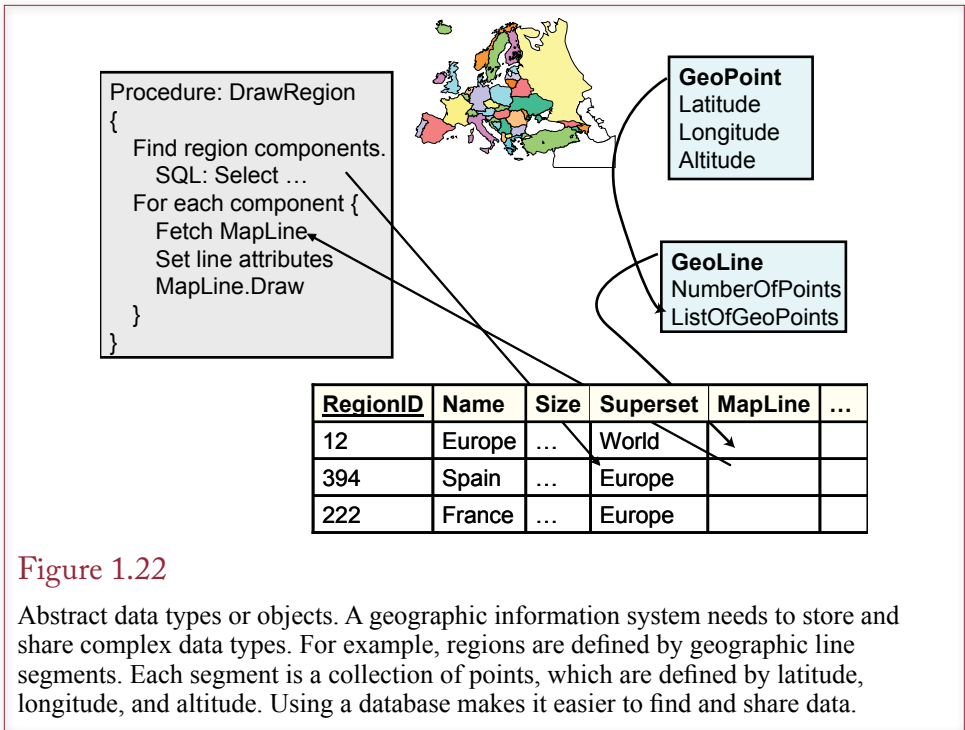
An **object-oriented (OO) database** is a different method of organizing data. The OO approach began as a new method to create programs. The goal is to define objects that can be reused in many programs—thus saving time and reducing errors. As illustrated in Figure 1.21, a class or object has three major components: a name, a set of properties or attributes, and a set of methods or functions. The properties describe the object—just as attributes describe an entity in the relational database. *Methods* are short programs that define the actions that each object can take. For example, the code to add a new

Note: This section contains a relatively detailed description of object-oriented databases and could be skipped for an introductory class. Or read it now and return to it later. Object features add a level of complexity to problems that can be confusing to beginners.

Figure 1.21

Object-oriented database. Objects have properties—just as relational entities have attributes—that hold data to describe the object. Objects have methods that are functions the objects can perform. Objects can be derived from other objects.





customer would be stored with the Customer object. The innovation is that these methods are stored with the object definition.

Figure 1.21 also hints at the power of the OO approach. Note that the base objects (Order, Customer, OrderItem, and Item) are the same as those for the relational approach. However, with the OO approach, new objects can be defined in terms of existing objects. For example, the company might create separate classes of customers for commercial and government accounts. These new objects would contain (inherit) all of the original Customer properties and methods, and also add variations that apply only to the new types of customers.

Two basic approaches are used to handle true object-oriented data: (1) extend the relational model to include typical OO features or (2) create a new object-oriented DBMS. Today, most commercially successful database systems follow the first approach by adding object features to the relational model.

The approach that adds OO features to the relational model is best exemplified by the American National Standards Institute (ANSI). Object-oriented features were a major component to the SQL 99 version. The SQL 2003 standard clarified some of the OO issues as well. In 1997 the SQL3 development group merged with the Object Database Management Group (ODMG). Three features are suggested to add OO capabilities: (1) abstract data types, (2) subtables, and (3) persistent stored modules. DBMS vendors have implemented most of these features.

Object Properties

The first issue involves defining and storing properties. In particular, OO programmers need the ability to create new composite properties that are built from other data types. SQL supports **abstract data types** to enable developers to create

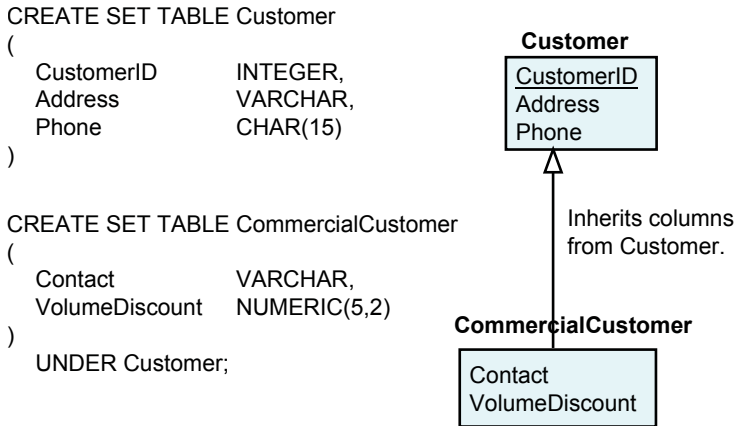


Figure 1.23

SQL subtables. A subtable inherits the columns from the selected supertable. Queries to the CommercialCustomer table will also retrieve data for the CustomerID, Address, and Phone columns inherited from the Customer table.

new types of data derived from existing types. This technique supports inheritance of properties. The type of data stored in a column can be a composite of several existing abstract types. Consider the example shown in Figure 1.22, which shows part of a database for a geographic information system (GIS). The GIS defines an abstract data type for location (GeoPoint) in terms of latitude, longitude, and altitude. Similarly, a line segment (e.g., national boundary), would be a collection of these location points (GeoLine). By storing the data in tables, the application can search and retrieve information based on user requirements. The database also makes it easier to share and to update the data. In the GIS example, the database handles the selection criteria (Region = Europe). The database can also match and retrieve demographic data stored in other tables. The advantage to this approach is that the DBMS handles the data storage and retrieval, freeing the developer to concentrate on the application details.

The abstract data type enables developers to create and store any data needed by the application. The abstract data type can also provide greater control over the application development. First, by storing the data in a DBMS, it simplifies and standardizes the way that all developers access the data. Second, the elements within the data type can be encapsulated. By defining the elements as private, application developers (and users) can only access the internal elements through the predefined routines. For example, developers could be prevented from directly modifying the latitude and longitude coordinates of any location by defining the elements as private.

SQL provides a second method to handle inheritance by defining subtables. A **subtable** inherits all the columns from a base table and provides inheritance similar to that of the abstract data types; however, all the data is stored in separate columns. The technique is similar to the method shown in Figure 1.21, which stores subclasses in separate tables. The difference is that the OO subtables will not need to include the primary key in the subtables. As indicated in Figure 1.23, inheritance is specified with an UNDER statement. You begin by defining the

highest level tables (e.g., Customer) in the hierarchy. Then when you create a new table (e.g., CommercialCustomer), you can specify that it is a subtable by adding the UNDER statement. If you use the unified modeling language (UML) triangle-pointer notation, or an IS-A icon for inheritance, it will be easy to create the tables in SQL. Just define the properties of the table and add an UNDER statement if there is a “pointer” to another table.

Do not worry about the details of the CREATE TABLE command. Instead, it is important to understand the difference between abstract data types and subtables. An abstract data type is used to set the type of data that will be stored in one column. With a complex data type, many pieces of data (latitude, longitude, etc.) will be stored within a single column. With a subtable the higher level items remain in separate columns. For example, a subtable for CommercialCustomer could be derived from a base Customer table. All the attributes defined by the Customer table would be available to the CommercialCustomer as separate columns.

Object Methods

Each abstract data type can also have methods or functions. In SQL, the routines are called **persistent stored modules**. They can be written as SQL statements. The SQL language is also being extended with programming commands—much like Oracle’s PL/SQL extensions. Routines are used for several purposes. They can be used as code to support triggers, which have been added to SQL. Persistent routines can also be used as methods for the abstract data types. Designers can define functions that apply to individual data types. For example, a GIS location data type could use a subtraction operator that computes the distance between two points.

To utilize the power of the database, each abstract data type should define two special functions: (1) to test for equality of two elements and (2) to compare elements for sorting. These functions enable the DBMS to perform searches and to sort the data. The functions may not apply to some data types (e.g., sound clips), but they should be defined whenever possible.

Object-Oriented Languages and Sharing Persistent Objects

The development of true OODBMS models was initiated largely in response to OO programmers who routinely create their own objects within memory. They needed a way to store and share those objects. Although the goals may appear similar to the modified-relational approach, the resulting database systems are unique.

Most OO development has evolved from programming languages. Several languages were specifically designed to utilize OO features. Common examples include C++, Smalltalk, and Java. Data variables within these languages are defined as objects. Each class has defined properties and methods. Currently, developers building applications in these languages must either create their own storage mechanisms or translate the internal data to a relational database.

Complex objects can be difficult to store within relational databases. Most languages have some facility for storing and retrieving data to files, but not to databases. For example, C++ libraries have a serialize function that transfers objects directly to a disk file. There are two basic problems with this approach: (1) it is difficult to search files or match data from different objects, and (2) the developer is responsible for creating all sharing, concurrency, and security operations. However, this approach causes several problems because data is now intrinsically tied to the programs and is no longer independent.

- InterSystems Caché
- Progress Software ObjectStore
- Objectivity
- McObject Perst
- Versant
- JADE

Figure 1.24

OODBMS vendors and products. Each tool has different features and goals. Contact the vendors for details or search the Web for user comments.

Essentially, OO programmers want the ability to create **persistent objects**, that is, objects that can be saved and retrieved at any time. Ideally, the database would standardize the definitions, control sharing of the data, and provide routines to search and combine data. The basic difficulty is that no standard theory explains how to accomplish all these tasks. Nonetheless, as shown in Figure 1.24, several OODBMS exist, and users have reportedly created many successful applications with these tools. But, most of these tools have minimal market share and may no longer exist.

The key to an OODBMS is that to the programmer it simply looks like extended storage. An object and its association links are treated the same whether the object is stored in RAM or shared through the DBMS. Clearly, these systems make development easier for OO programmers. The catch is that you have to be an OO programmer to use the system at all. In other words, if your initial focus is on OO programming, then a true OODBMS may be useful. If you started with a traditional relational database, you will probably be better off with a relational DBMS that has added OO features.

In theory, the 1997 agreements between ANSI and ODMG were designed to bring the SQL and OODBMS models closer to a combined standard. In practice, it could take a few years and considerable experimentation in the marketplace. For now, if you are serious about storing and sharing objects, you will have to make a choice based on your primary focus: OO programming or the relational database. As of 2010, the relational database approach with some OO extensions appears to have won out over pure OO database systems. Otherwise, programmers simply write data objects to individual files. If speed is an overriding issue, simple files are often the best answer.

Key-Value Pairs: Cassandra

The expanding use of highly-popular Web sites has created a need for high-performance, specialized data storage. With hundreds of millions of users uploading megabytes of data every hour (or minute), banks of servers are needed to handle the heavy demands. In many cases, the data to be stored is also non-traditional—it consists of complex objects including photos, raw files, blog entries, or various text items. Time is also a common feature, where people want to store multiple versions or organize data by the time an object was created. Initially, most big sites created proprietary storage methods to handle their unique situation. More recently, some people have started sharing portions of their ideas and works. One tool has gained some popularity and illustrates some of the features useful in these situations. The database system is called Cassandra and is available as an open

source project for several different hardware systems—although Linux-based systems are the most popular.

Storing data in Cassandra is unlike any of the earlier methods. Cassandra does borrow ideas from several advanced features of relational systems but trying to compare it with relational systems leads to confusion for beginners, so the details are not covered in this chapter. The two most important points are that data values are stored and accessed via a key item, and that the data is deliberately designed to be spread across multiple servers. By eliminating “relationships” data can be split into mostly independent pieces. Spreading these pieces across multiple servers enables the system to harness the power of massively parallel systems to perform storage, retrieval, and searches simultaneously on thousands of machines working independently. Chapter 13 explores the issues of design and querying for non-relational systems, including examples for Cassandra.

Drawbacks to Database Management Systems

What potential problems exist with a DBMS approach? The discussion of OO systems brings up the most common criticism of a DBMS: performance. The DBMS is a layer of complex software between the application and the data storage. Although this layer provides many useful features, it can slow down the storage and retrieval of huge amounts of data. And huge amounts of data are where the problems arise. Automated systems can easily generate gigabytes of data per hour or even faster. Writing massive amounts of data to a disk drive taxes the transfer capabilities even for fast servers. Pushing the data through a DBMS adds overhead for backup-and-recovery, concurrency controls, and indexes. It is possible for a DBMS to add two-to-three times the amount of data written for each byte of original data.

The high-end DBMSs provide tools to analyze data storage and to improve read/write times for large data transfers. But, you must always consider the possibility that it might be necessary to bypass the DBMS and store some data directly to the file system.

For example, it is possible to store images and even video data directly into a database table. However, in many applications—particularly Web-based ones—it is better to store the raw data in separate files and then store just the filename in the database. One reason this approach works is because this data rarely needs to be altered—once a file is stored, it is rarely edited, or the editing is not relevant to the DBMS. Consequently, there is little need to control for multiple users or to create repeated backups.

The problem of overhead created by the DBMS arises every few years in the industry. For example, around 2010 several writers began pushing the value of “non-SQL” database systems. The term is inaccurate because the query language SQL is not really the problem. Instead, some examples of extremely large databases create performance issues in terms of storage. Creating storage mechanisms for extremely large databases is difficult. But, before discarding the DBMS approach, you must seriously consider what features you are willing to give up. DBMS performance is slower than directly writing data to a disk largely because the DBMS provides safety through backup logs and concurrency controls. Alternative storage methods that bypass these features can provide faster performance, but you often give up the safety and security controls. When you evaluate alternatives, be sure that you understand exactly what features are being removed to pro-

vide performance increases. And then determine whether you need those features or how they might be provided through other methods.

The examples used in this book, and the Workbooks, are small enough that performance is not an issue. All of the data can be stored in individual tables. However, you should learn to recognize potential problems so that when you work on large-scale applications, you can choose the appropriate time to bypass the DBMS and store files directly to a server.

The other major drawback to a DBMS is the cost of the software. For relatively small projects, this cost can be small or even zero. Microsoft Access works for small projects. You can also obtain free copies of software for Oracle, SQL Server and IBM's DB2. These copies have size and performance limitations but work for many smaller projects. You can also obtain open-source software such as MySQL (now controlled by Oracle) and PostgreSQL. These two have been used for relatively large-scale commercial Web sites. However, keep in mind that "open-source" ultimately means that you pay for maintenance and support—either by paying a third party or by hiring more people. So, open-source is not actually a zero-cost option.

Application Development

What is an application? If you carefully examine Figures 1.16, 1.17, and 1.18, you will notice that they all have essentially the same data sets. This similarity is not an accident. Database design methods described in Chapters 2 and 3 should be followed regardless of the method used to implement the database. In other words, any database project begins by identifying the data that will be needed and analyzing that data to store it as efficiently as possible.

The second step in building applications is to identify forms and reports that the users will need. These forms and reports are based on queries, so you must create any queries or views that will be needed to produce the reports and forms as described in Chapters 4 and 5. Then you use the report writer and forms generator to create each report and form as described in Chapters 6 and 7. As described in Chapter 8, the next step is to combine the forms and reports into an application that handles all of the operations needed by the user. The goal is to create an application that matches the jobs of the users and helps them to do their work.

Chapter 7 describes how to deal with common problems in a multiuser environment to protect the integrity of the data and support transactions. Chapter 9 shows one more design method of storing data: the data warehouse. To deal with large databases, transaction processing systems use indexes and other features to optimize storage tasks. Today, managers want to retrieve and analyze the data. Data warehouses provide special designs and tools to support online analytical processing (OLAP).

When the application is designed and while it is being used, several database administration tasks have to be performed. Setting security parameters and controlling access to the data is one of the more important tasks. Chapter 10 discusses various administration and security issues.

As an organization grows, computer systems and applications become more complex. An important feature in modern organizations is the need for users to access and use data from many different computers throughout the organization. At some point you will need to increase the scope of your application so that it can be used by more people in different locations. Distributed databases discussed in Chapter 11 are a powerful way to create applications that remove the restrictions

of location. The Internet is rapidly becoming a powerful tool for building and implementing database applications that can be used by anyone around the world. The same technologies can be used for applications that are accessed only by in-house personnel. Systems that use Internet technology but limit access to insiders are called intranets.

Chapter 12 introduces the considerations of how the DBMS physically stores data. This chapter is particularly helpful for students who have a background in programming, but the topics are presented carefully so that non-programmers can understand the issues as well. The basic point is that high-end DBMSs allow the administrator to control how the data is stored in operating system files. This level of control is sometimes needed to improve the performance of large databases.

Introduction to this Book's Databases

What databases are used with this book? Several databases are used as examples in this book. The workbook has an additional database as well but it is described in the workbook. These databases are important because they provide concrete examples of various issues in database design, queries, and application development. You can study the databases to help you understand the topics discussed in this book. Bear in mind that the databases are not completed. In fact, each database is at a different level of completion so that you can see how an application is built in stages.

The main database in this book is Sally's Pet Store. The design is complete, and some forms and reports have been created, but many application features need to be added. The Corner Med database is newer and it is designed to be a smaller application and to provide more examples and exercises that illustrate some common issues in the healthcare industry. Rolling Thunder Bicycles is a relatively large database, in terms of design, application, and data. It was originally developed in Microsoft Access and contains many detailed forms. Scripts exist to build the data tables in other DBMSs, but the forms and reports have not been converted. The All Powder Board and Ski Shop in the Workbooks has similarities to Rolling Thunder, and many of the forms and reports have been developed for multiple DBMSs. Keep in mind that the purpose of the Workbook is to show you all of the steps in building an application.



Sally's Pet Store

A young lady with a love for animals is starting a new type of pet store. Sally wants to match pets with owners who will take good care of the animals. The Pet Store database was changed for this edition of the book. Specifically, the store no longer “sells” animals. Instead, animals are brought in for adoption by various local adoption organizations. Customers donate money to the organization to adopt an animal. The donations are handled by the store clerks at checkout time and the accumulated donations are paid to the adoption organizations. This approach cuts down on “backyard breeders,” and enables Sally and her customers to support local charitable organizations that help find homes for animals.

At the moment Sally has only one store, but she dreams of expanding into additional cities. She wants to hire and train workers to be “animal friends,” not salespeople. These friends will help customers choose the proper animal. They will answer questions about health, nutrition, and pet behavior. They will even be taught that some potential customers should be convinced not to buy an animal.

Because the workers will spend most of their time with the customers and animals, they will need technology to help them with their tasks. The new system will also have to be easy to use, since little time will be available for computer training.

Even based on a few short discussions with Sally, it is clear that the system she wants will take time to build and test. Fortunately, Sally admits that she does not need the complete system immediately. She has decided that she first needs a basic system to handle the store operations: sales, orders, customer tracking, and basic animal data. However, she emphasizes that she wants the system to be flexible enough to handle additional features and applications.

Details of Sally's Pet Store will be examined in other chapters. For now, you might want to visit a local pet store or talk to friends to get a basic understanding of the problems they face and how a database might help them.



Corner Med

Corner Med is currently a small medical office with big plans. Eventually, the owners want to franchise the concept and create a chain of walk-in medical offices that are affordable and accessible to customers in cities around the country. Currently, the company is run by a handful of physicians, supported by nurses and a few clerical staff. For the most part, the physicians focus on family practice and handle routine medical exams and common problems. More complex cases are referred to specialists, but the doctors at Corner Med are often responsible for the initial diagnosis and generally participate in the long-term care of the patients.

The company has a small medical lab and can perform simple tests, such as basic blood workups and routine x-rays. More complex tests and procedures such as CT scans and MRIs are handled by specialty firms available in every major city.

In terms of a business application, Corner Med wants to keep basic patient information in a database instead of thousands of paper folders. Of course, security and privacy issues become critical—particularly if the company eventually decides to centralize the data for multiple offices. One simplifying aspect of medical management and billing is that insurance companies along with various governments, and ultimately, the World Health Organization (WHO), have defined a common set of numbers used to define diagnoses (conditions) and procedures (treatments). The main reference is the International Disease Classification system. It is commonly referenced by its initials and the version number. The most commonly-used version is ICD-9. However, ICD-10 has been introduced and the U.S. government is currently stating that all medical organizations are supposed to switch to ICD-10 by 2013.

The version of the database for this edition has been modified to use the ICD-10 diagnosis and procedure codes (two sets). These codes are available for download from the U.S. government Web sites. The database also uses a DrugListing file which contains registered drugs directly from the FDA Web site. However, the original ICD-9 codes have been kept in the database tables to provide examples in converting data from the older codes to the newer ones.

The sample data for patient visits is derived from a U.S. physician survey. The names are fictional, but gender and diagnoses and procedures were created by randomly drawing data from that survey. The fees and payments are weak estimates and are likely to be inaccurate, but they illustrate the concepts.

Physicians are familiar with the diagnostic and procedure terms, but they rarely memorize the entire list of ICD codes. Many hospitals and large practices hire

medical encoders to translate the physician's descriptions into the proper codes. In some ways, the codes simplify the medical database. But, as you will see, they complicate the user interface because you have to find a way to make the list easy to use.



Rolling Thunder Bicycles

The Rolling Thunder Bicycle Company builds custom bicycles. Its database application is much more complete than the Pet Store application, and it provides an example of how the pieces of a database system fit together. This application also contains many detailed forms that illustrate the key concepts of creating a user interface. Additionally, most of the forms contain programming code that handles common business tasks. You can study this code to help you build your own applications. The Rolling Thunder application has a comprehensive help system that describes the company and the individual forms. The database contains realistic data for hundreds of customers and bicycles.

One of the most important tasks at the Rolling Thunder Bicycle Company is to take orders for new bicycles. Several features have been included to help non-experts select a good bicycle. As the bicycles are built, the employees record the construction on the Assembly form. When the bicycle is shipped, the customers are billed. Customer payments are recorded in the financial forms. As components are installed on bicycles, the inventory quantity is automatically decreased. Merchandise is ordered from suppliers, and payments are made when the shipments arrive.

The tasks performed at Rolling Thunder Bicycles are similar to those in any business. By studying the application and the techniques, you will be able to create solid applications for any business.

Starting a Project: The Feasibility Study

What are the first steps to start a project? Ideas for information systems can come from many sources: users, upper management, information system analysts, competitors, or firms in other industries. Ideas that receive initial support from several people might be proposed as new projects. If the project is small enough and easy to create, it might be built in a few days. Larger projects require more careful study. If the project is going to involve critical areas within the organization, require expensive hardware, or require substantial development time, then a more formal feasibility study is undertaken.

Feasibility studies are covered in detail within systems analysis texts. However, because of their unique nature, it is helpful to examine the typical costs and benefits that arise with the database approach.

The goal of a feasibility study is to determine whether a proposed project is worth pursuing. The study examines two fundamental categories: costs and potential benefits. As noted in Figure 1.25, costs are often divided into two categories: up-front or one-time costs and ongoing costs once the project is operational. Benefits can often be found in one of three categories: reduced operating costs, increased value, or strategic advantages that lock out competitors.

Costs

Almost all projects will entail similar up-front costs. The organization will often have to purchase additional hardware, software, and communication equipment (such as a Web server or expand a local area network). The cost of developing the

| Costs | Benefits |
|----------------------|---------------------------|
| Up-front/one-time | Cost savings |
| Software | Software maintenance |
| Hardware | Fewer errors |
| Communications | Less data maintenance |
| Data conversion | Less user training |
| Studies and design | Increased value |
| Training | Better access to data |
| Ongoing costs | Better decisions |
| Personnel | Better communication |
| Software upgrades | More timely reports |
| Supplies | Faster reaction to change |
| Support | New products and services |
| Hardware maintenance | Strategic advantages |
| | Lock out competitors |

Figure 1.25

Common costs and benefits from introducing a database management system. Note that benefits can be hard to measure, especially for tactical and strategic decisions. But it is still important to list potential benefits. Even if you cannot assign a specific value, managers need to see the complete list.

system is listed here, including the cost for all additional studies. Other one-time costs include converting data to the new system and initial training of users. Database management systems are expensive software items. For example, for larger projects, the cost for software such as Oracle can easily run to several million dollars. You will also have to purchase “maintenance” upgrades of the software at least on an annual basis.

Hardware and software costs can be estimated with the help of vendors. As long as you know the approximate size of the final system (e.g., number of users), vendors can provide reasonably accurate estimates of the costs. Data conversion costs can be estimated from the amount of data involved. The biggest challenge often lies in estimating the costs of developing the new system. If an organization has experience with similar projects, historical data can be used to estimate the time and costs based on the size of the project. Otherwise, the costs can be estimated based on the projected number of people and hours involved.

Once the project is completed and the system installed, costs will arise from several areas. For example, the new system might require additional personnel and supplies. Software and hardware will have to be modified and replaced—entailing maintenance costs. Additional training and support might be required to deal with employee turnover and system modifications. Again, most of these costs are straightforward to estimate—as long as you know the size of the project.

Unfortunately, information system (IS) designers have not been very successful at estimating the costs. For example, in January 1995 PC Week reported that 31 percent of new IS projects are canceled before they are completed. Additionally, 53 percent of those that are completed are 189 percent over budget. This pattern is not unique. A study published in MIS Quarterly in 2000 also estimated that 30-40 percent of projects “escalated” into late or over budget status. The greatest difficulty is in estimating the time it takes to design and develop new software. Every developer is different with large variations in programmer productivity. In

large projects, where the staff members are constantly changing, accurately predicting the amount of time needed to design and develop a new system is often impossible. Nonetheless, managers need to provide some estimate of the costs. On a related note, building anything new can be difficult to estimate in terms of time and cost. The Boeing 787 “Dreamliner” took several years longer and millions of extra dollars to design and build than originally anticipated. The problem in estimating information systems and physical systems is that you need to predict the future, and it is probably impossible to anticipate every possible problem that might arise.

Benefits

In many cases benefits are even more difficult to estimate. Some benefits are tangible and can be measured with a degree of accuracy. For instance, transaction processing systems are slightly easier to evaluate than a decision support system, since benefits generally arise from their ability to decrease operations costs. A system might enable workers to process more items, thus allowing the firm to expand without increasing labor costs. A database approach might reduce IS labor costs by making it easier for workers to create and modify reports. Finally, a new information system might reduce errors in the data, leading to improved decisions.

Many benefits are intangible and cannot be assigned specific monetary values. For instance, benefits can arise because managers have better access to data. Communication improves, better decisions are made, and managers can react faster to a changing environment. Similarly, the new system might enable the company to produce new products and services or to increase the sales of ancillary products to existing customers. Similarly, firms might implement systems that provide a competitive advantage. For example, an automated order system between a firm and its customers often encourages the customers to place more orders with the firm. Hence the firm gains an advantage over its competitors.

When information systems are built to automate operations-level tasks and the benefits are tangible, evaluating the economic benefits of the system is relatively straightforward. The effects of improving access to data are easy to observe and measure in decreased costs and increased revenue. However, when information systems are implemented to improve tactical and strategic decisions, identifying and evaluating benefits is more difficult. For instance, how much is it worth to a marketing manager to have the previous week’s sales data available on Monday instead of waiting until Wednesday?

In a database project benefits can arise from improving operations—which leads to cost savings. Additional benefits occur because it is now easier and faster to create new reports for users, so less programmer time will be needed to modify the system. Users can also gain better access to data through creating their own queries—instead of waiting for a programmer to write a new program.

Database projects can provide many benefits, but the organization will receive those benefits only if the project is completed correctly, on time, and within the specified budget. To accomplish this task, you will have to design the system carefully. More than that, your team will have to communicate with users, share work with each other, and track the progress of the development. You need to follow a design methodology.

Summary

One of the most important features of business applications is the ability to share data with many users at the same time. Without a DBMS sharing data causes several problems. For example, if data definitions are stored within each separate program, making changes to the data file becomes very difficult. Changes in one program and its data files can cause other programs to crash. Every application would need special code to provide data security, concurrency, and integrity features. By focusing on the data first, the database approach separates the data from the programs. This independence makes it possible to expand the database without crashing the programs.

A DBMS has many components. Required features include the database engine to store and retrieve the data and the data dictionary to help the DBMS and the user locate data. Other common features include a query language, which is used to retrieve data from the DBMS to answer business questions. Application development tools include a report writer, a forms generator, and an application generator to create features like menus and help files. Advanced database systems provide utilities to control secure access to the data, cooperate with other software packages, and communicate with other database systems.

Database systems have evolved through several stages. Early hierarchical databases were fast for specific purposes but provided limited access to the data. Network databases enabled users to build complex queries but only if the links were built with indexes in advance. The relational database is currently the leading approach to building business applications. Once the data is defined carefully, it can be stored and retrieved efficiently to answer any business question. The OO approach is a new technique for creating software. Object-oriented systems enable you to create your own new abstract data types. They also support subtables, making it easier to extend a class of objects without redefining everything from scratch. Recently, key-value databases are being developed to handle massive loads of complex data types but they are still evolving.

Regardless of the type of database implemented, application development follows similar steps. First, identify the user requirements, determine the data that needs to be collected, and define the structure of the database. Then, develop the forms and reports that will be used, and build the queries to support them. Next, combine the various elements into a polished application that ties everything together to meet the user needs. If necessary, distribute the database across the organization or through an Internet or intranet. Additional features can be provided by integrating the database with powerful analytical and presentation tools, such as spreadsheets, statistical packages, and word processors.

A Developer's View

For Miranda to start on her database project, she must first know the strengths of the tools she will use. At the starting point of a database project, you should collect information about the specific tools that you will use. Get the latest reference manuals. Install the latest software patches. Set up work directories and project space. For a class project, you should log on, get access to the DBMS, make sure you can create tables, and learn the basics of the help system.

Key Terms

| | |
|-----------------------------------|-------------------------------------|
| abstract data types | intranets |
| data dictionary | network database |
| data independence | object-oriented (OO) database |
| data mining | online analytical processing (OLAP) |
| database | persistent objects |
| database engine | persistent stored modules |
| database management system (DBMS) | relational database |
| extensible markup language (XML) | report services |
| feasibility study | report writer |
| forms development | subtable |
| hierarchical database | XQuery |

Review Questions

1. What features does a DBMS provide that make application development easier?
2. What are the basic components of a DBMS?
3. Why is data independence important and how is it achieved with a DBMS?
4. Why is the relational database approach better than earlier methods?
5. How do relational databases implement object-oriented features?
6. What potential drawbacks exist to a DBMS?
7. What are the main steps in application development with a database system?
8. What is the purpose of a feasibility study?
9. Why do many of the biggest Web sites use non-relational databases?

Exercises

1. Create a new database with the two tables shown in the figure. Feel free to add more data. Be sure to set a primary key for the underlined columns. Be sure to create a relationship that links the two tables. Use a report wizard to create the report shown. You should be able to use a visual tool to create the tables. Otherwise, check Chapter 3 for the syntax of the CREATE TABLE command.

Employee

| <u>EmployeeID</u> | LastName | FirstName | Address | DateHired |
|-------------------|----------|-----------|----------------|-----------|
| 332 | Ant | Adam | 354 Elm | 5/5/1964 |
| 442 | Bono | Sonny | 765 Pine | 8/8/1972 |
| 553 | Cass | Mama | 886 Oak | 2/2/1985 |
| 673 | Donovan | Michael | 421 Willow | 3/3/1971 |
| 773 | Moon | Keith | 554 Cherry | 4/4/1972 |
| 847 | Morrison | Jim | 676 Sandalwood | 5/5/1968 |


Client

| <u>ClientID</u> | LastName | FirstName | Balance | EmployeeID |
|-----------------|----------|-----------|---------|------------|
| 1101 | Jones | Joe | 113.42 | 442 |
| 2203 | Smith | Mary | 993.55 | 673 |
| 2256 | Brown | Laura | 225.44 | 332 |
| 4456 | Cieter | Jackie | 664.90 | 442 |
| 5543 | Wodkoski | John | 984.00 | 847 |
| 6673 | Sanchez | Paula | 194.87 | 773 |
| 7353 | Chen | Charles | 487.34 | 332 |
| 7775 | Hagen | Fritz | 595.55 | 673 |
| 8890 | Hauer | Marianne | 627.39 | 773 |
| 9662 | Nguyen | Suzie | 433.88 | 553 |
| 9983 | Martin | Mark | 983.31 | 847 |

Report

| | |
|----------------|----------|
| Ant, Adam | 5/5/1964 |
| Brown, Laura | 225.24 |
| Chen, Charles | 47.34 |
| | 712.58 |
| Bono, Sonny | |
| Dieter, Jackie | 664.90 |
| Jones, Joe | 114.32 |
| | 779.22 |

2. Read the documentation to your DBMS and write a brief outline that explains how to:
 - a) Create a table.
 - b) Create a simple query.
 - c) Create a report.
3. Describe two business or Web applications that could use a DBMS. Identify some of the main data elements that would be collected.

4. Find a reference or check Web sites so you can compare the specifications on three free DBMS products. At least one of the products should be from a major commercial vendor and one from an open-source or free source.
5. Describe how a university club or student organization could use a database to improve its service operations.
6. A company wants you to build a custom Web site to support sales of computer cables over the Internet. The company anticipates receiving an average of 100 orders per business day, with an average of \$57.19 per order. Gross profit margins (including credit card costs) are about 15 percent. Shipping costs are priced directly so do not affect profits. No new workers will be needed to package and ship the orders, but the company expects to hire a designer with a salary of about \$25,000, to keep the product list up to date with photos and descriptions. The initial costs include cost of the computer hardware and software (\$10,000) and the development cost (\$35,000). The ISP costs will be about \$400 per month. Annual maintenance costs are expected to be \$1000 per year. Assuming a project life of five years and an interest rate of 8 percent, compute the economic feasibility of the project. Compare the costs and benefits to the alternative of selling the items through Amazon instead.
-  7. You have just been hired by a company and have wandered around talking to people. A few accountants have developed a database-driven application to handle fixed-assets and track depreciation. They claim the system has enabled them to function with one less accounting clerk (\$30,000/year). A guy in finance has created a custom database in Microsoft Access to generate reports on a set of financial investments. Much of the data comes from brokerage firms and he notes that he is able to save 10 hours a week in clerical time (minimum wage). However, he complained about the difficulty of loading data from the main company database and says he spends 2 hours a week typing in data from printed reports. Two people in marketing have created separate databases to track survey and sales results for their projects. They claim the projects have not saved any labor costs, but enhance sales by at least \$1 million a year. Yesterday, your manager said that all of these people complained about their existing systems and the inability to get data from the corporate database. You need to define projects for each group, identify the cost of developing a new system and the potential benefits. Rank the projects by economic return and make a recommendation to management.
8. Find two companies that provide Web-based database hosting and compare the basic costs for running an online relational database with 500 GB of data, a medium-sized server, and 2 TB of monthly data transfer.
9. Find a company, government agency, or a Web site (perhaps through an article or blog) and briefly explain the data collected and how the database is used.
10. Use articles or blogs to identify a Web site that relies on a non-relational (NoSQL) database and briefly explain the benefits of using that approach over a relational DBMS in this situation.



Sally's Pet Store

11. Install the Pet Store database or find it on your local area network if it has already been installed. Print out (or write down) the list of the tables used in the database. Use the Help command to find the version number of Microsoft Access that you are using.
12. Visit a local pet store and make a list of 10 merchandise items and five animals for sale. Enter this data into the appropriate Pet Store database tables.
13. Identify the hardware and software that would be required to install this system in a typical Pet Store. Estimate the costs and the time required to build and install the system.
- ✓ 14. Outline the basic tasks that take place in running a pet store. Identify some of the basic data items that will be needed.
15. Find an online pet store and estimate the number of different products for sale. Assume the company has 500,000 customers and handles 1,000 sales a day with about 3 items per sale. Assume it takes 300 bytes to store data for one product, 80 bytes for a customer, and 500 bytes for a sale. Estimate the amount of data needed to be saved in one year.



Rolling Thunder Bicycles

16. Install the Rolling Thunder database or find it on your local area network if it has already been installed. Using the BicycleOrder form, create an entry for a new bicycle.
17. Use the Rolling Thunder Help system, or the Web site description, to briefly describe the firm and its major processes. Identify the primary business entities in the company.
18. Use Web sites or visit a local bike shop to find prices for at least two bicycles. Try to find the most expensive bicycle you can.
- ✓ 19. Refer to the relationship/class diagram to explain what a Component is and how it is connected to a Bicycle. Give an example from the data.
20. Use the Employee table and Excel to compute the company's total salary costs. What problems might you encounter if you tried a similar approach for a table with 5 million customers?



Corner Med

21. Install the Corner Med database if necessary. Use the Patient form to enter data for a new patient.
22. Use the Patient Visit form to enter data for a patient with at least one diagnostic code and one procedure code. Describe any usability or performance issues that might arise.
- ✓ 23. Make a list with a brief description of other items that the company might want to store in the database.

24. Check the U.S. government Web sites (e.g., Health and Human Services) to see when the conversion to ICD-10 is going to be required. Also, check the Web site to see what information is available for vendors and developers. What problems are likely to arise with the conversion from ICD9 to ICD10 codes?
25. Look at the design for the Corner Med database. Briefly explain the purpose of the three tables: VisitDiagnoses, VisitProcedures, and VisitMedications. Why is it necessary to have three separate tables instead of combining them into a single table?

Web Site References

| | |
|---|---|
| http://office.microsoft.com/en-us/access/ | Microsoft Access |
| http://www.microsoft.com/en-us/sqlserver/default.aspx | Microsoft SQL Server |
| http://www.oracle.com | Oracle |
| http://www.oracle.com/technetwork | Oracle technology network with software downloads |
| http://www-01.ibm.com/software/data/db2/ | IBM DB2 |
| http://www.mysql.com | Free DBMS now controlled by Oracle |
| http://www.postgresql.org | A better free DBMS |
| http://www.acm.org | Association for Computing Machinery |
| http://groups.google.com/groups/dir?sel=usenet%3Dcomp.databases http://dbforums.com http://dbasupport.com http://www.devx.com/outgoing/databasefeed.xml http://www.sqlservercentral.com | Newsgroups for database questions. |
| http://www.cms.gov/Medicare/Coding/ICD10/index.html | U.S. government Web site (HHS) with ICD-10 codes. |

Additional Reading

- Keil, M., J. Mann, and A. Rai, "Why Software Projects Escalate," *MIS Quarterly*, 24(4), 2000. [Estimates 30-40 percent of IT development projects escalate above budget.]
- Perry, J. and Post, G. *Introduction to Oracle 10g*, Englewood Cliffs: Prentice-Hall, 2007. [A step-by-step introduction to Oracle 10g with several databases.]
- Perry, J. and Post, G. *Introduction to SQL Server 2005*, Englewood Cliffs: Prentice-Hall, 2008. [A step-by-step introduction to SQL Server 2005 and Visual Studio 2005.]
- Zikopoulos, P.C. and R.B. Melnyk, *DB2: The Complete Reference*, McGraw-Hill, 2001. [One of few reference books on DB2 and written by IBM employees.]