The Characteristics of Databases

The purpose of a database is to help people keep track of things. It does this by storing data in tables, where each table has rows and columns like those in a spreadsheet. A database usually has multiple tables, and each table contains data about a different type of thing. For example, Figure 1-1 shows a database with three tables: the STUDENT table has data about students, the GRADE table has data about grades, and the CLASS table has data about classes.

Each row of a table has data about a particular instance. For example, each row of the STUDENT table has data about one of four students: Cooke, Lau, Harris, and Greene. Similarly, each row of the CLASS table has data about a particular class. The columns of the table store characteristics about each of these instances. The first column of STUDENT stores StudentNumber, the second column stores StudentName, and so forth.

Although, in theory, you could switch the rows and columns by putting instances in the columns and characteristics in the rows, this is never done. Every database in this text, and 99.999999 percent of all databases throughout the world, store instances in rows and characteristics in columns, just as shown in Figure 1-1.

A Note on Conventions

In this text, table names appear in capital letters. This convention will help you to distinguish table names in explanations. However, you are not required to set table names in capital letters. Access and similar programs will allow you to write a table name as STUDENT, student, Student, stuDent, or in some other way.

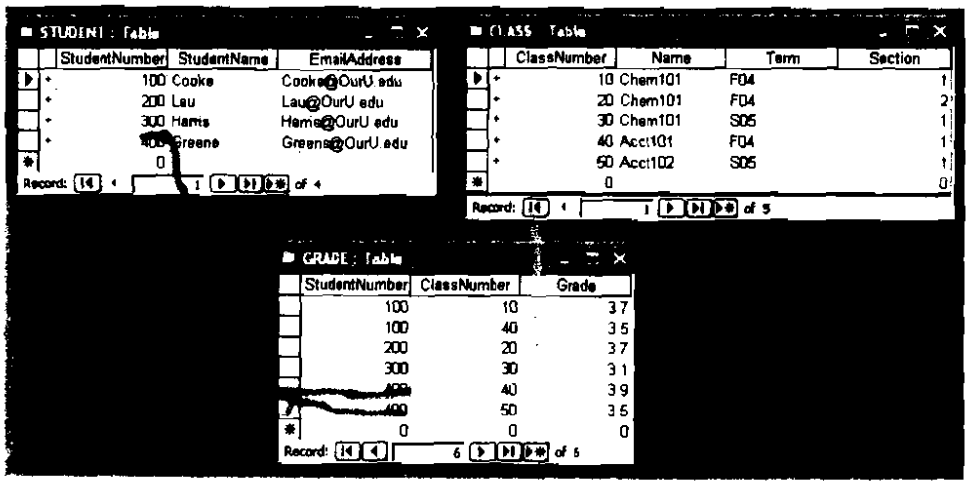
Additionally, in this text column names begin with a capital letter. Again, this is just a convention. You could write the column name Term as term, teRm, TERM, or in any other way. To ease readability, we will sometimes create compound column names in which the first letter of each element of the compound word is capitalized. Thus, in Figure 1-1, the STUDENT table has columns StudentNumber, StudentName, and EmailAddress. Again, this capitalization is just a convenient convention. However, following these or other consistent conventions will make interpretation of database structures easier. For example, you will always know that STUDENT is the name of a table and that Student is the name of a column of a table.

A Database Has Data and Relationships

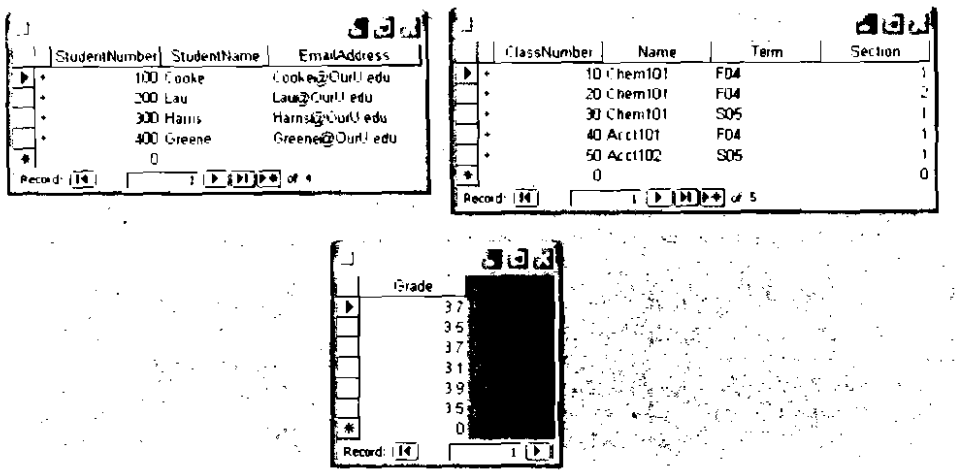
The database in Figure 1-1 shows data not only about students, classes, and grades, it also shows relationships among the rows in those tables. For example, StudentNumber

Figure 1-1

The Key Database Characteristic: Related Tables



Need for Relationships



400, who is Greene, earned a Grade of 3.9 in ClassNumber 40, which is Acct101. He also earned a Grade of 3.5 in ClassNumber 50, which is Acct102.

Figure 1-1 illustrates an important characteristic of database processing: Databases store not only rows of data, but also relationships among the rows of data. To see why this is important, examine Figure 1-2. In this figure, the database contains all of the basic data, but the relationship data are missing. In this format, the GRADE data are useless. It is like the joke about the sports commentator who announced: "Now for tonight's baseball scores: 2–3, 7–2, 1–0, and 4–5." The scores are useless without knowing the teams that earned them. Thus, a database contains both data and the relationships among the data.

Database Create Information

In your previous classes, you learned the difference between data and information. Data are recorded facts or figures. Information is knowledge derived from data. Or, using another common definition, information is data presented in a meaningful context.

Databases record facts and figures; they record data. They do so, however, in a way that enables them to produce information. The data in Figure 1-1 can be manipulated to produce a student's GPA, the average GPA for a class, the average number of students in a class, and so forth. In Chapter 2, you will be introduced to a language called Structured Query Language, or SQL, that you can use to produce information from database data.

To summarize, databases store data in tables, and they represent the relationships among the rows of those tables. They do so in a way that facilitates the production of information.

Database Examples

Today, database technology is part of almost every information system. This fact is not surprising when we consider that every information system needs to store data and the relationships among those data. Still, the vast array of applications that use this technology is staggering.

Inventory, Order Entry Applications

Consider, for example, the applications listed in Figure 1-3. The first application is used by a single salesperson to keep track of the customers she has called and the

Figure 1-3 Example Database Uses

Application	Example Users	Number of Users	Typical Size	Remarks
Sales contact manager	Salesperson	1	2,000 rows	Products such as GoldMine and Act! are database centric.
Patient appointment (doctor, dentist)	Medical office	15 to 50	100,000 rows	Vertical market software vendors incorporate databases into their software products.
Customer Resource Management (CRM)	Sales, marketing, or customer service departments	500	10 million rows	Major vendors such as Siebel and PeopleSoft build applications around the database.
Enterprise Resource Processing (ERP)	An entire organization	5,000	10 million+ rows	SAP uses a database as a central repository for ERP data.
E-commerce site	Internet users	Possibly millions	1 billion+ rows	Drugstore.com has a database that grows at the rate of 20 million rows per day!
Digital dashboard	Senior managers	500	100,000 rows	Extractions, summaries, and consolidations of operational databases.
Data mining	Business analysts	25	100,000 to millions+	Data are extracted, reformatted, cleaned, and filtered for use by statistical data mining tools.

contacts that she's had with them. Most salespeople do not build their own contact manager applications; instead, they license products such as GoldMine (see ***www.frontrange.com/goldmine***) or ACT! (see ***www.act.com***).

Multiuser Database Applications

The next applications in Figure 1-3 are those that involve more than one user. The patient-scheduling application, for example, may have 15 to 50 users. These users will be appointment clerks, office administrators, nurses, dentists, doctors, and so forth. A database like this one may have as many as 100,000 rows of data in perhaps 5 or 10 different tables.

When more than one user employs a database application, there is always the chance that one user's work may interfere with another's. Two appointment clerks, for example, might assign the same appointment to two different patients. Special concurrency-control mechanisms are used to coordinate activity against the database to prevent such conflict. You will learn about these mechanisms in Chapter 9.

The third row of Figure 1-3 shows an even larger database application. Customer Resource Management (CRM) is an information system that manages customer contacts from initial solicitation through acceptance, purchase, continuing purchase, support,

and so forth. CRM systems are used by salespeople, sales managers, customer service and support staff, and other personnel. A CRM database in a larger company might have 500 users and 10 million or more rows in perhaps 50 or more tables. According to Microsoft, in 2004 Verizon had a SQL Server customer database that contained more than 15 terabytes of data. If that data were published in books, a bookshelf 450 miles long would be required to hold them.

Enterprise Resource Planning (ERP) is an information system that touches every department in a manufacturing company. It includes sales, inventory, production planning, purchasing, and other business functions. SAP is the leading vendor of ERP applications, and a key element of their product is a database that integrates data from these various business functions. An ERP system may have 5,000 or more users and perhaps 100 million rows in several hundred tables.

E-Commerce Database Applications

E-commerce is another important database application. Databases are a key component of e-commerce order entry, billing, shipping, and customer support. Surprisingly, however, the largest databases at an e-commerce site are not order-processing databases. The largest databases are those that track customer browser behavior. Most of the prominent e-commerce companies, such as Amazon.com (***www.amazon.com***) and Drugstore.com (***www.drugstore.com***) keep track of the Web pages and the Web page components that they send to their customers. They also track customer clicks, additions to shopping carts, order purchases, abandoned shopping carts, and so forth.

E-commerce companies use Web activity databases to determine which Web page items are popular and successful and which are not. They also can conduct experiments to determine if a purple background generates more orders than a blue one, and so forth. Such Web usage databases are huge. For example, Drugstore.com adds 20 million rows to its Web log database each day!

Reporting and Data Mining Database Applications

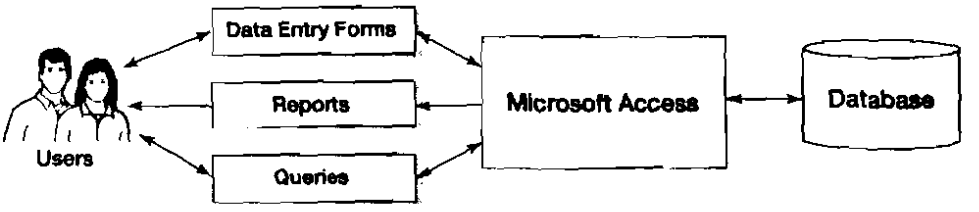
Two other example applications in Figure 1-3 are digital dashboards and data mining applications. These applications use the data generated by order processing and other operational systems to produce information to help manage the enterprise. Such applications do not generate new data, instead they summarize existing data to provide insights to management. Digital dashboards and other reporting systems assess past and current performance. Data mining applications predict future performance. We will consider such applications in Chapter 15. The bottom line is that database technology is used in almost every information system and involves databases ranging in size from a few thousand rows to many millions of rows.

B T W

Do not assume that just because a database is small that its structure is simple. For example, consider parts distribution for a company that sells $1 million in parts per year and parts distribution for a company that sells $100 million in parts per year. Despite the difference in sales, the companies have similar databases. Both have the same kinds of data, about the same number of tables of data, and the same level of complexity in data relationships. Only the amount of data varies from one to the other. Thus, although a database for a small business may be small, it is not necessarily simple.

Figure 1-4

Components of an Access Database System



Components of a Database System

Figure 1-4 shows the structure of a typical Access database system. Users interact with the application through data entry forms like the one shown in Figure 1-5. They also request reports and perform queries against the database data. Access then processes the forms, produces the reports, and runs the queries.

Microsoft Access is a low-end product intended for individuals and small workgroups. As such, Microsoft has done all that it can to hide the underlying database technology from the user. Although hiding the technology is an effective strategy for beginners working on small databases, it won't work for database professionals who work with applications such as most of those described in Figure 1-3. For larger, more complex databases, it is necessary to understand the technology and components that Microsoft hides.

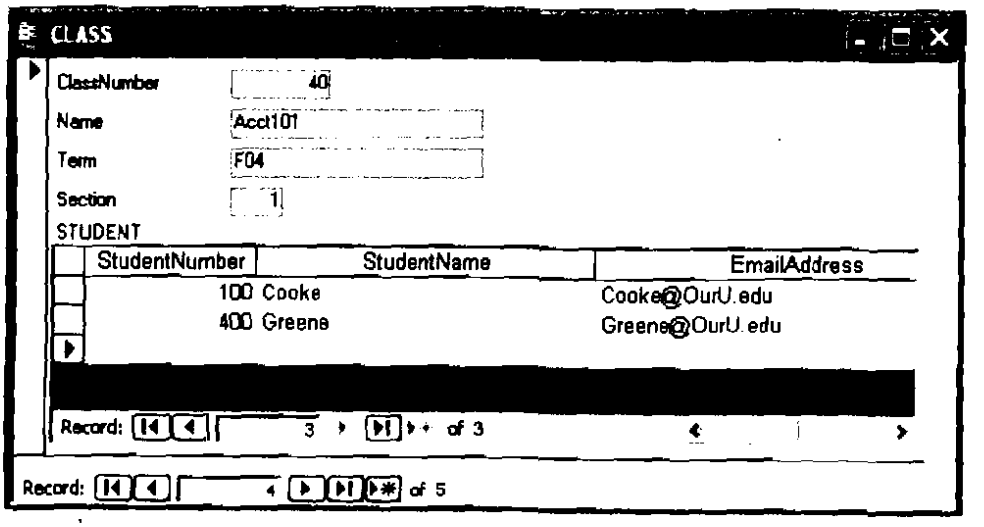
Applications, SQL, and the DBMS

Figure 1-6 is the same as Figure 1-5 except that the Access cover has been pulled off. As shown in the figure, databases have three components: applications, SQL, and the DBMS. **Applications** are computer programs that users interact with directly. Applications accept data from users, process it according to application-specific requirements, and use SQL to transfer data to and from the database. In the case of Access, applications create and process forms, reports, and queries. Other applications do far more, as you will learn.

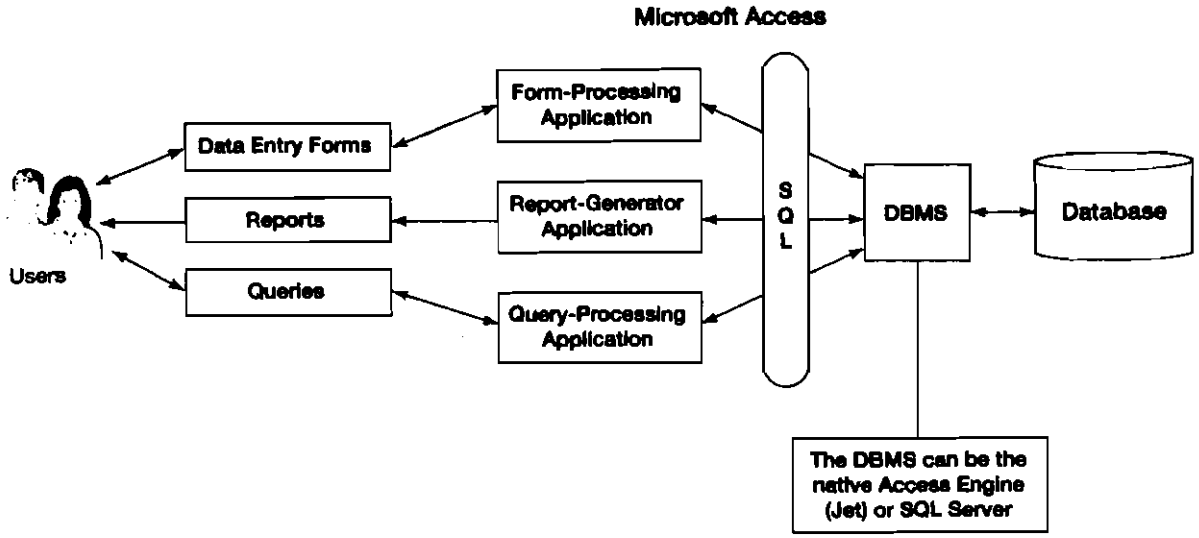
SQL, or **Structured Query Language**, is an internationally recognized standard language that is understood by all commercial database management system products.

Figure 1-5

An Example Data Entry Form



Under the Cover of an Access Application



To give you a taste of SQL, here is a sample SQL statement for processing the STUDENT table in Figure 1-1:

```
SELECT    StudentName, EmailAddress
FROM      STUDENT
WHERE     StudentNumber > 200;
```

This SQL statement will obtain the name and e-mail address of all students having a StudentNumber greater than 200. For the data in Figure 1-1, this statement will produce the StudentName and EmailAddress for students Harris and Greene.

As shown in Figure 1-6, the Access form, report, and query applications create SQL statements and pass them to the DBMS for processing. The **DBMS**, or ***database management system***, creates, processes, and administers the database. A DBMS is a large and complicated product and few organizations write their own DBMS program. Instead, DBMS products are licensed from vendors such as Microsoft, Oracle, and IBM.

Before continuing, we need to clear up a common misconception: Microsoft Access is *not* just a DBMS. Rather, it is a DBMS *plus* an application generator. Access contains a DBMS engine that creates, processes, and administers the database. It also contains form, report, and query components that are the Access application generator.

Internally, the application components hidden under the Access cover use SQL to call the DBMS that is also hidden under that cover. At Microsoft, the DBMS engine within Access is called Jet. You seldom hear about Jet because Microsoft does not sell Jet as a separate product.

B T W

With Microsoft Access 2000 and later versions, you can replace Jet with Microsoft's enterprise-class DBMS product—SQL Server. You would do this if you wanted to process a large database or if you needed the advanced functions and features of SQL Server.

To replace Jet with SQL Server, you need a computer that has both products installed. If your computer has Access 2003, select *File New Project using new data* and then select the option *Create a project using a new database*. If you are using Access 2000, select *File New/Project (New Database)* and then *Create a project using a new database*. If you are using Access XP, select *File New Project (New Database)* and then *Create a project using a new database*.

Once you make these selections, Access will use SQL Server rather than Jet as its DBMS. Jet will no longer be used. Also note that you can attach your project to a SQL Server database on another computer if you have processing rights on that computer.

Components of an Enterprise-class Database System

Figure 1-7 shows the components of an enterprise-class database system. Here, the applications and the DBMS are not under the same cover as they are in Access. Instead, the applications are separate from each other and separate from the DBMS.

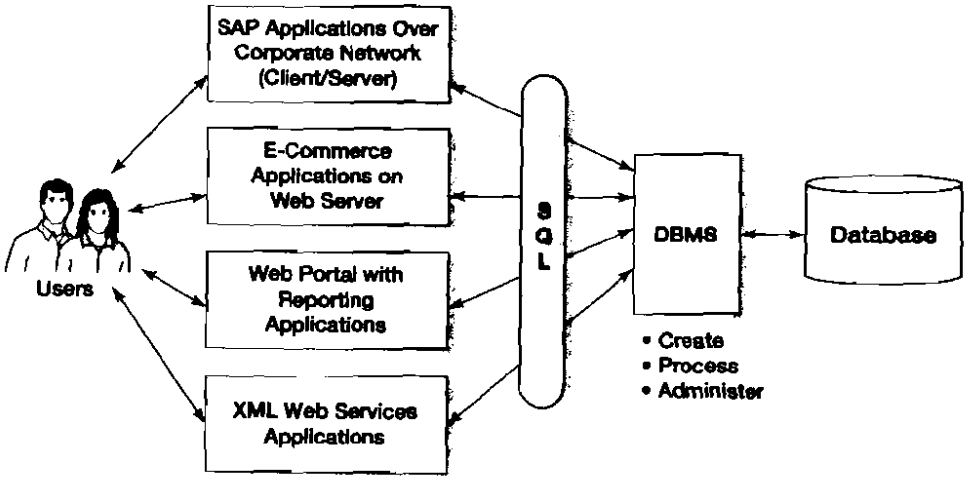
Database Applications

As exemplified by the list in Figure 1-3, dozens of different types of database applications are available. Figure 1-7 shows SAP applications that connect to the database over a corporate network. Such applications are sometimes called client/server applications because the application program is a client that connects to a database server. Client/server applications often are written in Visual Basic, C++, or Java.

A second category of applications in Figure 1-7 is e-commerce and other applications that run on a Web server. Users connect to such applications via Web browsers such as Internet Explorer and Netscape Navigator. Common Web servers include Microsoft's Internet Information Server (IIS) and Apache. Common languages for Web server applications are PHP, Java, and the Microsoft .NET languages such as C# and

Figure 1-7

Structure of an Enterprise-class Database Application



VB.NET. We will discuss some of the technology for such applications in Chapters 12, 13, and 14.

A third category of applications is reporting applications that publish the results of database queries on a corporate portal or other Web site. Such reporting applications are often created using third-party report generation and digital dashboard products from vendors such as Cognos and MicroStrategy. We will describe these applications in Chapter 15.

The last category of applications is XML Web services. These applications are at the leading edge of database processing. They use a combination of the XML markup language and other standards to enable program-to-program communication. In this way, the code that comprises an application is distributed over several different computers. Web services can be written in Java or any of the .NET languages. We will discuss this important new class of applications in Chapter 13.

All of these database applications get and put database data by sending SQL statements to the DBMS. These applications may create forms and reports, or they may send their results to other programs. They also may implement application logic that goes beyond simple form and report processing. For example, an order entry application uses application logic to deal with out-of-stock items and backorders.

The DBMS

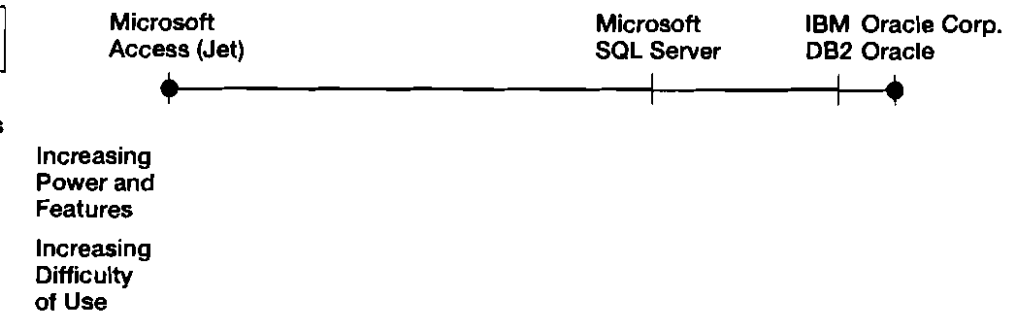
As stated earlier, the DBMS manages the database. It processes SQL statements and provides other features and functions for creating, processing, and administering the database. Figure 1-8 presents the four most prominent DBMS products. The products are shown in order of increasing power, features, and difficulty of use. Access (really Jet) is the easiest to use and the least powerful. SQL Server has far more power; it can process larger databases, faster, and it includes features for multiuser control, backup and recovery, and other administrative functions.

DB2 is a DBMS product from IBM. Most people would agree that it has faster performance than SQL Server, that it can handle larger databases, and that it is also more difficult to use. Finally, the fastest and most capable DBMS is Oracle from the Oracle Corporation. Oracle can be configured to offer very high performance on exceedingly large databases that operate 24/7, year after year. Oracle is also far more difficult to use and administer than SQL Server.

The Database

The last component in Figure 1-7 is the database. A **database** is a self-describing collection of integrated tables. Integrated tables are tables that store both data and the relationships among the data. The tables in Figure 1-1 are *integrated* because they store not just student, class, and grade data, but also data about the relationships among the rows of data.

A database is *self-describing* because it contains a description of itself. Thus, databases contain not only tables of user data, but also tables of data that describe that user



Common Professional View of DBMS Products

Figure 1-9

Typical Metadata Tables

USER_TABLES Table

TableName	NumberColumns	PrimaryKey
STUDENT	3	StudentNumber
CLASS	4	ClassNumber
GRADE	3	(StudentNumber, ClassNumber)

USER_COLUMNS Table

ColumnName	TableName	DataType	Length (bytes)
StudentNumber	STUDENT	Integer	4
StudentName	STUDENT	Text	50
EmailAddress	STUDENT	Text	50
ClassNumber	CLASS	Integer	4
Name	CLASS	Text	50
Term	CLASS	Text	5
Section	CLASS	SmallInteger	2
StudentNumber	GRADE	Integer	4
ClassNumber	GRADE	Integer	4
Grade	GRADE	Decimal	(3, 2)

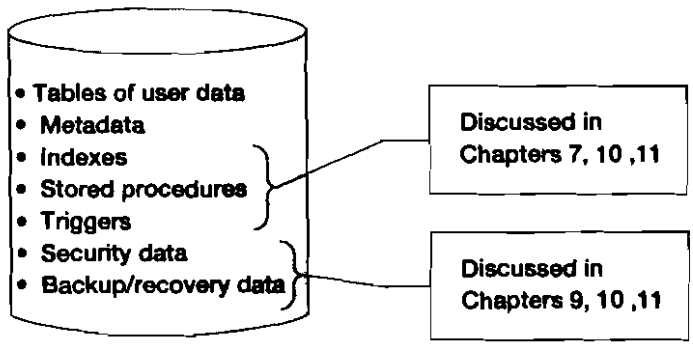
data. Such descriptive data is called ***metadata*** because it is data about data. The form and format of metadata varies from DBMS to DBMS. Figure 1-9 shows generic metadata tables that describe the tables and columns for the database in Figure 1-1.

You can examine metadata to determine if particular tables, columns, indexes, or other structures exist in a database. For example, the following statement queries the SQL Server metadata table SYSOBJECTS to determine if a user table (Type = 'U') named CLASS exists in the database. If it does, the table is dropped (removed) from the database.

```
If Exists
    (SELECT     *
    FROM        SYSOBJECTS
    WHERE       [Name] = 'CLASS'
        AND     Type = 'U')
    DROP TABLE CLASS
```

Figure 1-10

Database Contents



Do not be concerned with the syntax of this statement. You will learn what it means and how to write such statements yourself as we proceed. For now, just understand that this is one way that database administrators use metadata.

B T W

Because metadata is stored in tables, you can use SQL to query it as just illustrated. Thus, by learning how to write SQL to query user tables, you will also learn how to write SQL to query metadata. To do that, you just apply the SQL statements to metadata tables rather than user tables.

In addition to user tables and metadata, databases contain other elements, as shown in Figure 1-10. These other components will be described in detail in subsequent chapters. For now, however, understand that indexes are structures that speed the sorting and searching of database data. Triggers and stored procedures are programs that are stored within the database. Triggers are used to maintain database accuracy and consistency and to enforce data constraints. Stored procedures are used for database administration tasks and are sometimes part of database applications. You will learn more about these different elements in Chapters 7, 10, and 11.

Security data defines users, groups, and allowed permissions for users and groups. The particulars depend on the DBMS product in use. Finally, backup and recovery data are used to save database data to backup devices as well as to recover the database after failure. You will learn more about security and backup and recovery data in Chapters 9, 10, and 11.

Database Design

Database design is both difficult and important. Determining the proper structure of tables, the proper relationships among tables, the appropriate data constraints, and other structural components is challenging, and sometimes even daunting. Consequently, the world is full of poorly designed databases. Such databases do not perform well. They may require application developers to write overly complex and contrived SQL to get wanted data, they may be difficult to adapt to new and changing requirements, or they fail in some other way.

Figure 1-11

Three Types of Database Design

- From Existing Data (Chapters 3 and 4)
  - Analyze spreadsheets and other data tables
  - Extract data from other databases
  - Design using normalization principles
- New Systems Development (Chapters 5 and 6)
  - Create data model from application requirements
  - Transform data model into database design
- Database Redesign (Chapter 6)
  - Migrate databases to newer databases
  - Integrate two or more databases
  - Reverse engineer and design new databases using normalization principles and data model transformation

Note: Chapter 7 discusses database implementation using SQL. You need that knowledge before you can understand database redesign.

Because database design is both difficult and important, we will devote most of the first half of this text to the topic. As shown in Figure 1-11, there are three types of database design.

Database Design from Existing Data

The first type of database design involves databases that are constructed from existing data, as shown in Figure 1-12. In some cases, a development team is given a set of spreadsheets or a set of text files with tables of data. The team is required to design a database and import the data from those spreadsheets and tables into a new database.

Figure 1-12

Databases Originating from Existing Data

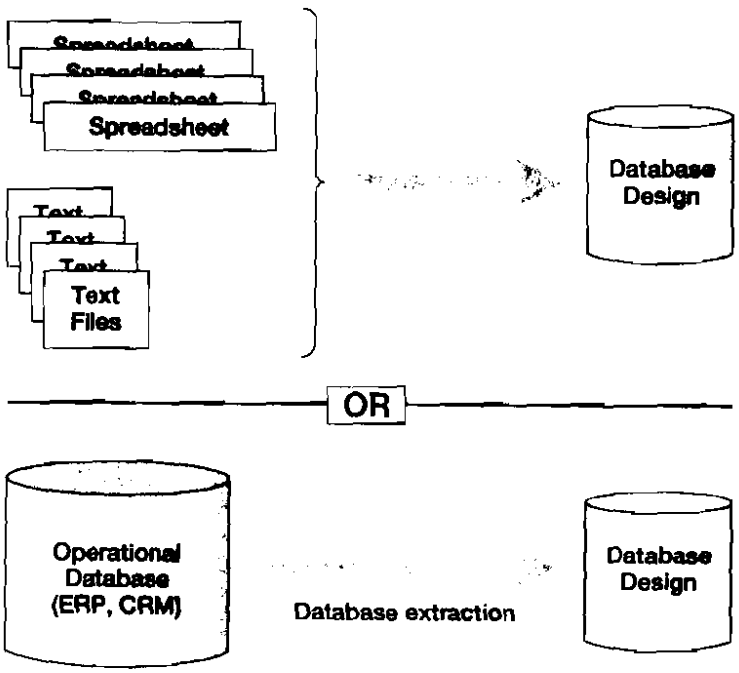


Figure 1-13 Data Import: One or Two Tables?

EmpNum	EmpName	DeptNum	DeptName
100	Jones	10	Accounting
150	Lau	20	Marketing
200	McCauley	10	Accounting
300	Griffin	10	Accounting

(a) One-Table Design

OR?

DeptNum	DeptName
10	Accounting
20	Marketing

EmpNum	EmpName	DeptNum
100	Jones	10
150	Lau	20
200	McCauley	10
300	Griffin	10

(b) Two-Table Design

Alternatively, databases can be created from extracts of other databases. This alternative is especially common in reporting and data mining applications. For example, data from an operational database, such as a CRM or ERP database, may be copied into a new database that will be used only for studies and analysis. As you will learn in Chapter 15, such databases are used in facilities called **data marts**. The data mart databases often are exported to other analytical tools, such as SAS's Enterprise Miner, SPSS's Clementine, or Insightful Corporation's I-Miner.

When creating a database from existing data, database developers must determine the appropriate structure for the new database. A common issue is how the multiple files or tables in the new database should be related. However, even the import of a single table can pose design questions. Figure 1-13 shows two different ways of importing a simple table of employees and their departments. Should this data be stored as one table or two?

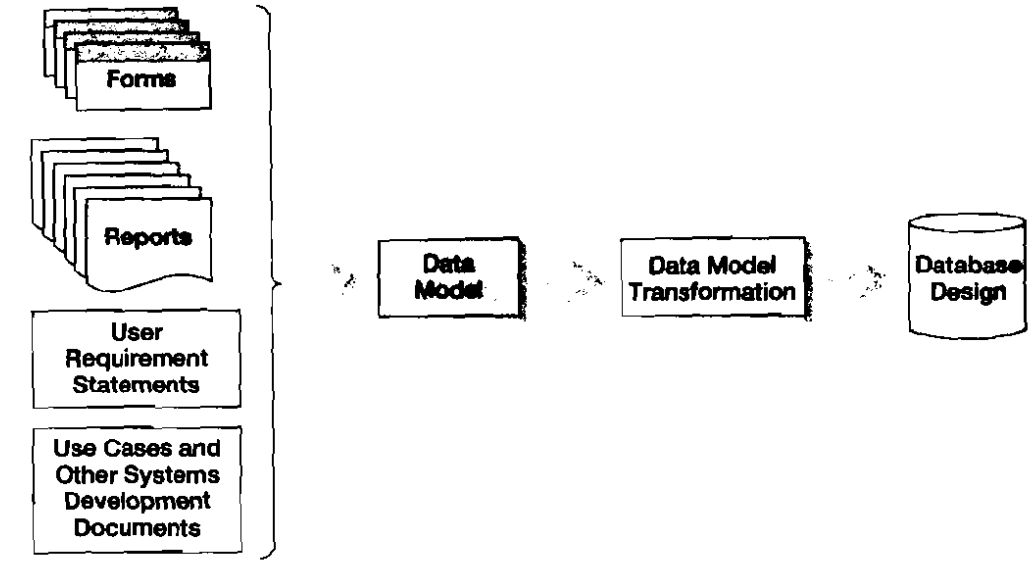
Decisions such as this are not arbitrary. Database professionals use a set of principles, collectively called **normalization**, or **normal forms**, to guide and assess database designs. You will learn those principles and their role in database design in Chapter 3.

Database Design from New Systems Development

A second way that databases are designed is from the development of new information systems. As shown in Figure 1-14, requirements for a new system, such as desired data entry forms and reports, user requirements statements, use cases, and other requirements, are analyzed to create the database design.

In all but the simplest system development projects, the step from user requirements to database design is too big. Accordingly, the development team proceeds in two steps. First, the team creates a **data model** from the requirements

Figure 1-14 Databases Originating from New Systems Development



statements and then transforms that data model into a database design. You can think of a data model as a blueprint that is used as a design aid on the way to a database design.

In Chapter 4, you will learn about the most popular data modeling technique—**entity-relationship data modeling**. You also will see how to use the entity-relationship model to represent a variety of common form and report patterns. Then, in Chapter 5, you will learn how to transform entity-relationship data models into database designs.

Database Design from Redesign

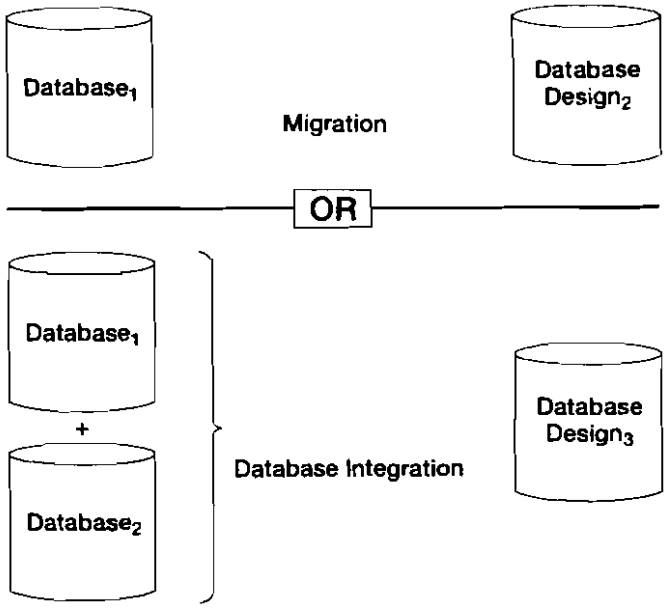
Database redesign is the third way that databases are designed. As shown in Figure 1-15, there are two common types of database redesign. In the first, a database is adapted to new or changing requirements. This process sometimes is called **database migration**. In the migration process, tables may be created, modified, or removed; relationships may be altered; data constraints may be changed; and so forth.

The second type of database redesign involves the integration of two or more databases. This type of redesign is common when adapting or removing legacy systems. It is also common for enterprise application integration, when two or more previously separate information systems are adapted to work with each other.

Database redesign is complicated. There is no getting around that fact. If this is your first exposure to database design, your instructor may skip this topic. If this is the case, after you have gained more experience, you should reread this material. In spite of its difficulty, database redesign is important.

To understand database redesign, you need to know SQL statements for defining database structures and more advanced SQL statements for querying and updating a database. Consequently, we will not address database redesign until after Chapter 7, which presents advanced SQL.

Databases Originating from Database Redesign



What You Need to Learn

In your career, you may work with database technology as either a user or as a database administrator. As a **user** you may be a *knowledge worker* who prepares reports, mines data, and does other types of data analysis or you may be a programmer who writes applications that process the database. Alternatively, you might be a **database administrator** who designs, constructs, and manages the database itself. Users are primarily concerned with constructing SQL statements to get and put the data they want. Database administrators are primarily concerned with the management of the database. The domains for each of these roles are shown in Figure 1-16.

Both users and database administrators need all of the knowledge in this text. However, the emphasis on each topic differs for the two groups. Figure 1-17 shows my opinion as to the relative importance of each topic to each group. Discuss this table with your instructor. He or she may have knowledge about your local job market that affects the relative importance of these topics.

Working Domains of Knowledge Workers, Programmers, and Database Administrators

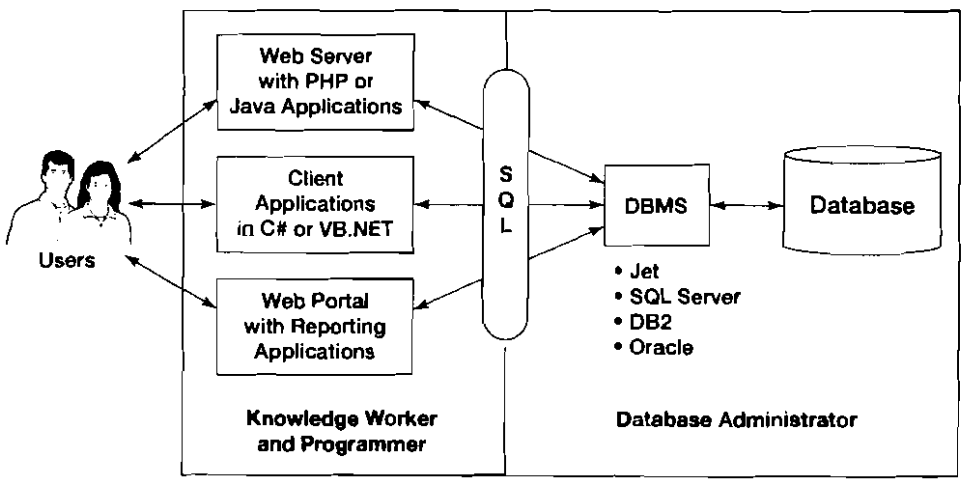


Figure 1-17 Priorities of What You Need to Know

Topic	Chapter	Importance to Knowledge Worker and Programmer	Importance to Database Administrator
Basic SQL	Chapter 2	1	1
Design via normalization	Chapter 3	2	1
Data modeling	Chapter 5	1	1
Data model transformation	Chapter 6	2	1
DDL SQL	Chapter 7	2	1
Constraint enforcement	Chapter 7	3	1
Database redesign	Chapter 8	3	2, but 1 for senior DBA
Database administration	Chapter 9	2	1
SQL Server, Oracle specifics	Chapter 10, 11	3	1
Database application technology	Chapter 12, 13, 14, and 15	1	3

1 = Very important; 2 = Important; 3 = Less important

Warning: Opinions vary, ask your instructor for his or hers.

B T

The most exciting and interesting jobs in technology are always those on the leading edge. If you live in the United States and are concerned about outsourcing, a recent study by the Rand Corporation[1] indicates that the most secure jobs in the United States involve the adaptation of new technology to solve business problems in innovative ways.

Right now, the leading edge involves the integration of XML, Web services, and database processing. You will need all of the fundamentals presented in this book, especially the material in Chapter 13, to work in this exciting new area. If I were starting out in database processing today, I would make learning about the integration of XML and database technology a top priority.

A Brief History of Database Processing

Database processing emerged around 1970 and has been continuously evolving and changing since then. This continual change has made it a fascinating and thoroughly enjoyable field in which to work. Figure 1-18 summarizes the major eras of database processing.

The Early Years

Prior to 1970, all data were stored in separate files, most of which were kept on reels of magnetic tape. Magnetic disks and drums (magnetic cylinders that are no longer used) were exceedingly expensive and very small. Today's 1.44 megabyte floppy disk has more

[1]Karoly, Lynn A., and Constantijn W. A. Panis. *The 21st Century at Work*. Santa Monica, CA: The Rand Corporation, 2004.

Database Processing

Fundamentals, Design, and Implementation

David M. Kroenke
University of Washington

Upper Saddle River, New Jersey 07458

Library of Congress Cataloging-in-Publication Data
Kroenke, David.
Database processing: fundamentals, design, and implementation / David M. Kroenke.—10th ed.
p. cm.
Includes bibliographical references and index.
ISBN 0-13-167267-3 (alk. paper)
1. Database management. I. Title.

QA76.9.D3K7365 2005
005.7'4—dc22 2004061616

VP/Editorial Director: Jeff Shelstad
AVP/Executive Editor: Bob Horan
Project Manager: Lori Cerreto
Senior Media Project Manager: Nancy Welcher
AVP/Executive Marketing Manager: Debbie Clare
Marketing Assistant: Joanna Sabella
Managing Editor: John Roberts
Production Editor: Suzanne Grappi
Permissions Supervisor: Charles Morris
Manufacturing Buyer: Diane Peirano
Design Director: Maria Lange
Art Director: Pat Smythe
Interior Designer: Karen Quigley
Cover Designer: Karen Quigley
Line Art: BookMasters, Inc.
Manager, Print Production: Christy Mahon
Composition: Integra Software Services
Full-Service Project Management: Jennifer Welsch/BookMasters, Inc.
Printer/Binder: Courier/Kendalville
Typeface: 10/12 Simoncini Garamond

Credits and acknowledgments borrowed from other sources and reproduced, with permission, in this textbook appear on appropriate page within text.

Microsoft® and Windows® are registered trademarks of the Microsoft Corporation in the U.S.A. and other countries. Screen shots and icons reprinted with permission from the Microsoft Corporarion. This book is not sponsored or endorsed by or affiliared with the Microsoft Corporation.

Pearson Education LTD.
Pearson Education Singapore, Pte. Ltd
Pearson Education, Canada, Ltd
Pearson Education–Japan
Pearson Educarion Australia PTY, Limited
Pearson Education North Asia Ltd
Pearson Educación de Mexico, S.A. de C.V.
Pearson Education Malaysia, Pte. Ltd

10 9 8 7 6 5 4 3 2 1
ISBN 0-13-167272-X