4. Personal Versus Enterprise-Class Database Systems

We can divide database systems and DBMS products into two classes: personal database systems and enterprise-class database systems.

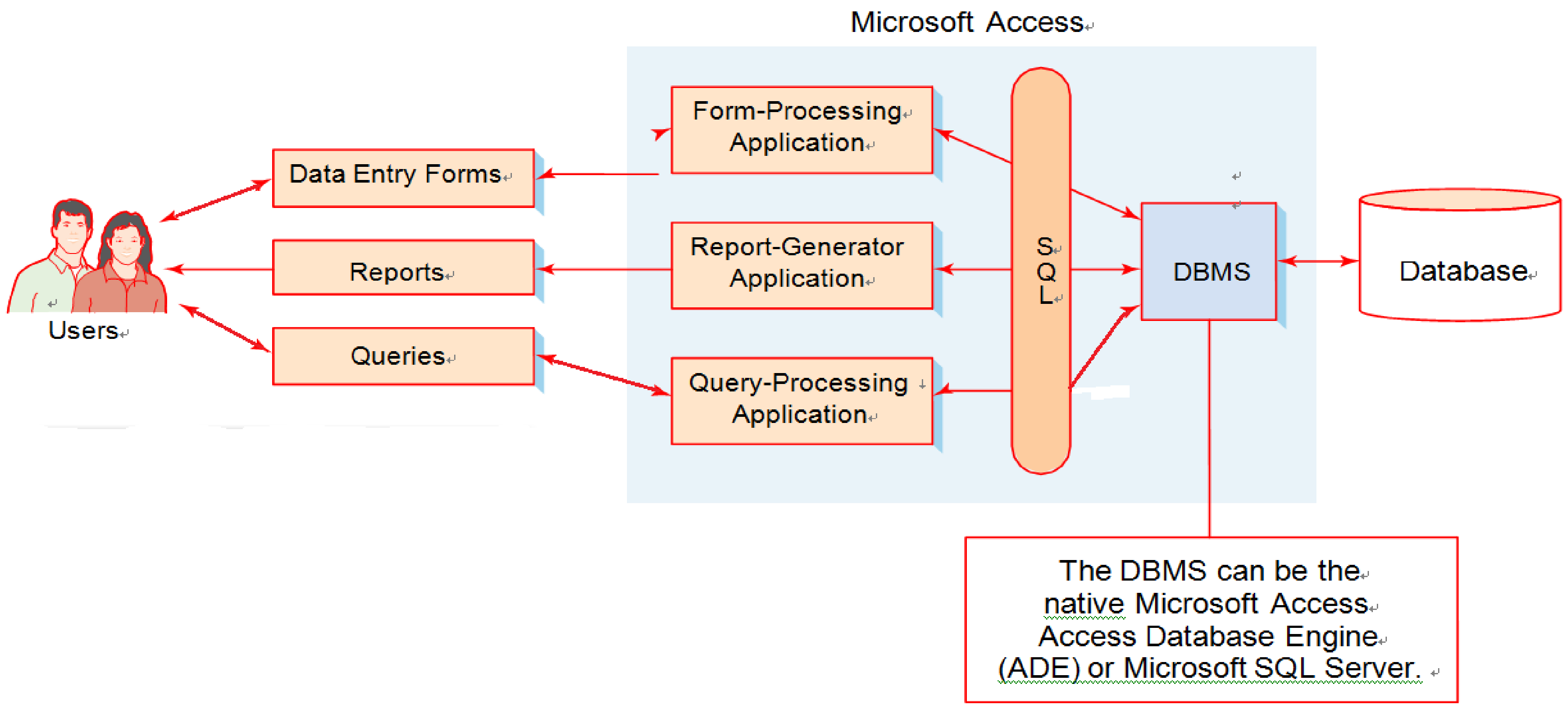
➢**What Is Microsoft Access?**

We need to clear up a common misconception: Microsoft Access is not just a DBMS. Rather, it is a personal database system: a DBMS plus an application generator. Although Microsoft Access contains a DBMS engine that creates, processes, and administers the database, it also contains form, report, and query components that are the Microsoft Access application generator. The components of Microsoft Access are shown in Figure 1-15, which illustrates that the Microsoft Access form, report, and query applications create SQL statements and then pass them to the DBMS for processing.

4. Personal Versus Enterprise-Class Database Systems

➢**What Is Microsoft Access?**

Figure 1-15 Components of a Microsoft Access Database System


Microsoft Access
Users
Data Entry Forms
Reports
Queries
Form-Processing Application
Report-Generator Application
Query-Processing Application
SQL
DBMS
Database
The DBMS can be the native Microsoft Access Access Database Engine (ADE) or Microsoft SQL Server.

4. Personal Versus Enterprise-Class Database Systems

➢What Is Microsoft Access?

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. Users interact with the application through data entry forms like the one shown in Figure 1-9. They also request reports and perform queries against the database data. Microsoft Access then processes the forms, produces the reports, and runs the queries. Internally, the application components hidden under the Microsoft Access cover use SQL to call the DBMS, which is also hidden under that cover. At Microsoft, the current DBMS engine within Microsoft Access is called the Access Database Engine (ADE). ADE is a Microsoft Office specific version of Microsoft’s Joint Engine Technology (JET or Jet) database engine. Jet was used as the Microsoft Access database engine until Microsoft Office 2007 was released. Jet itself is still used in the Microsoft Windows operating system, but you seldom hear about Jet because Microsoft does not sell Jet as a separate product.

By the way Although Microsoft Access is the best-known personal database system, it is not the only one. OpenOffice.org Base is a personal database system distributed as part of the OpenOffice.org software suite (which is available at www.openoffice. org), and the personal database system LibreOffice Base is distributed as part of the related LibreOffice software suite (which is available at www.libreoffice.org/).

4. Personal Versus Enterprise-Class Database Systems

➢**What Is Microsoft Access?**

Although hiding the technology is an effective strategy for beginners working on small databases, it will not work for database professionals who work with applications, such as most of those described in Figure 1-5. For larger, more complex databases, it is necessary to understand the technology and components that Microsoft hides.

Nonetheless, because Microsoft Access is included in the Microsoft Office suite, it is often the first DBMS used by students. In fact, you may have already learned to use Microsoft Access in other classes you have taken, and in this book we will provide some examples using Microsoft Access 2010. If you are not familiar with Microsoft Access 2010, you should work through Appendix A, "Getting Started with Microsoft Access 2010."

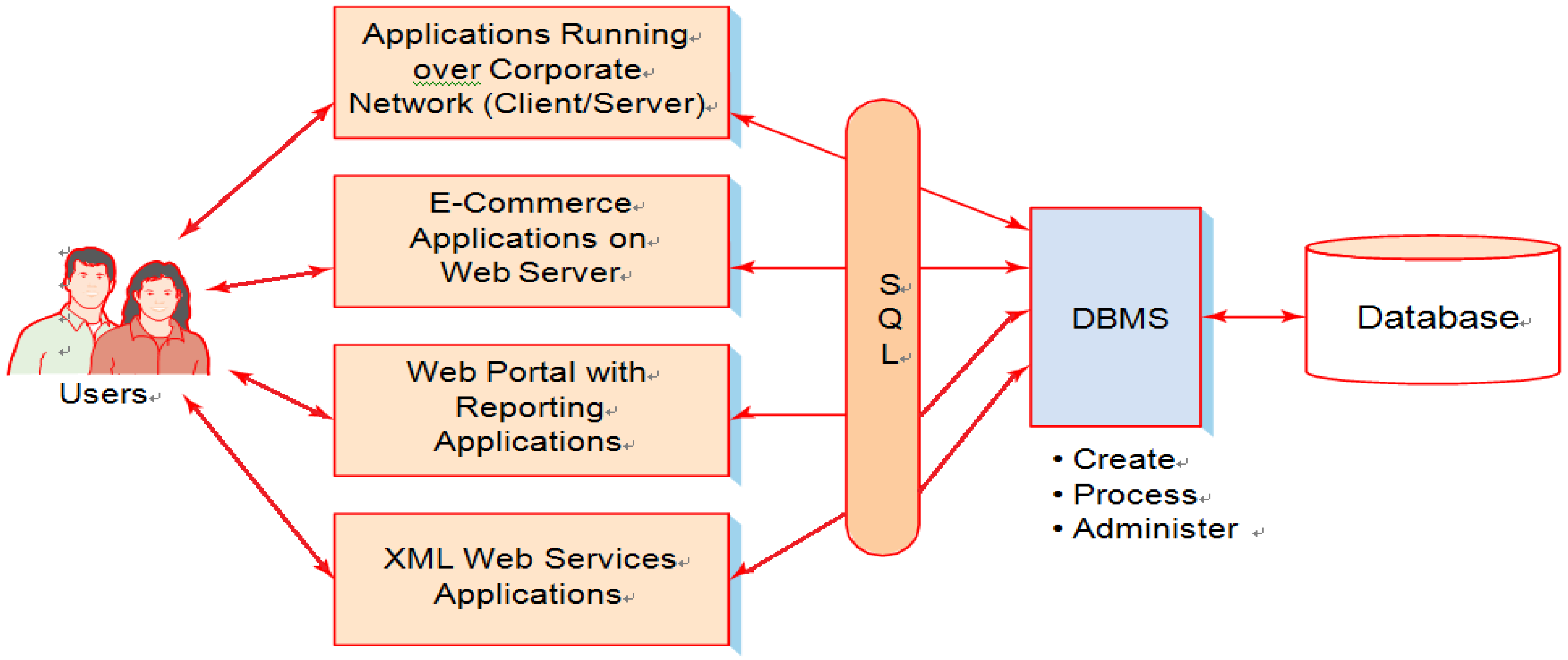
By the way With Microsoft Access 2000 and later versions, you can effectively replace the Micrsoft Access database engine (either Jet or ADE) with Microsoft's enterprise-class DBMS product—Microsoft SQL Server. You would do this if you wanted to process a large database or if you needed the advanced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

4. Personal Versus Enterprise-Class Database Systems

➢ **What Is an Enterprise-Class Database System?**

Figure 1-16 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 Microsoft Access. Instead, the applications are separate from each other and separate from the DBMS.

Figure 1-16 Components of an Enterprise-Class Database System


Applications Running over Corporate Network (Client/Server)
E-Commerce Applications on Web Server
Web Portal with Reporting Applications
XML Web Services Applications
Users
SQL
DBMS
Database
• Create
• Process
• Administer

4. Personal Versus Enterprise-Class Database Systems

➢Database Applications in an Enterprise-Class Database System

Earlier in this chapter, we discussed the basic functions of an application program, and these functions are summarized in Figure 1-8. However, as exemplified by the list in Figure 1-5, dozens of different types of database applications are available, and database applications in an enterprise-class database system introduce functions and features beyond the basics. For example, Figure 1-16 shows 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 programming languages such as VB.NET, C++, or Java.

A second category of applications in Figure 1-16 is e-commerce and other applications that run on a Web server. Users connect to such applications via Web browsers such as Microsoft Internet Explorer, Mozilla Firefox, and Google Chrome. Common Web serversinclude 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#.NET and VB.NET.

We will discuss some of the technology for such applications in Chapter 11.

4. Personal Versus Enterprise-Class Database Systems

- **Database Applications in an Enterprise-Class Database System**

  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 IBM (Cognos) and MicroStrategy (MicroStrategy 9). We will describe these applications in Chapter 13.

  The last category of applications is XML Web services. These applications 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 12.

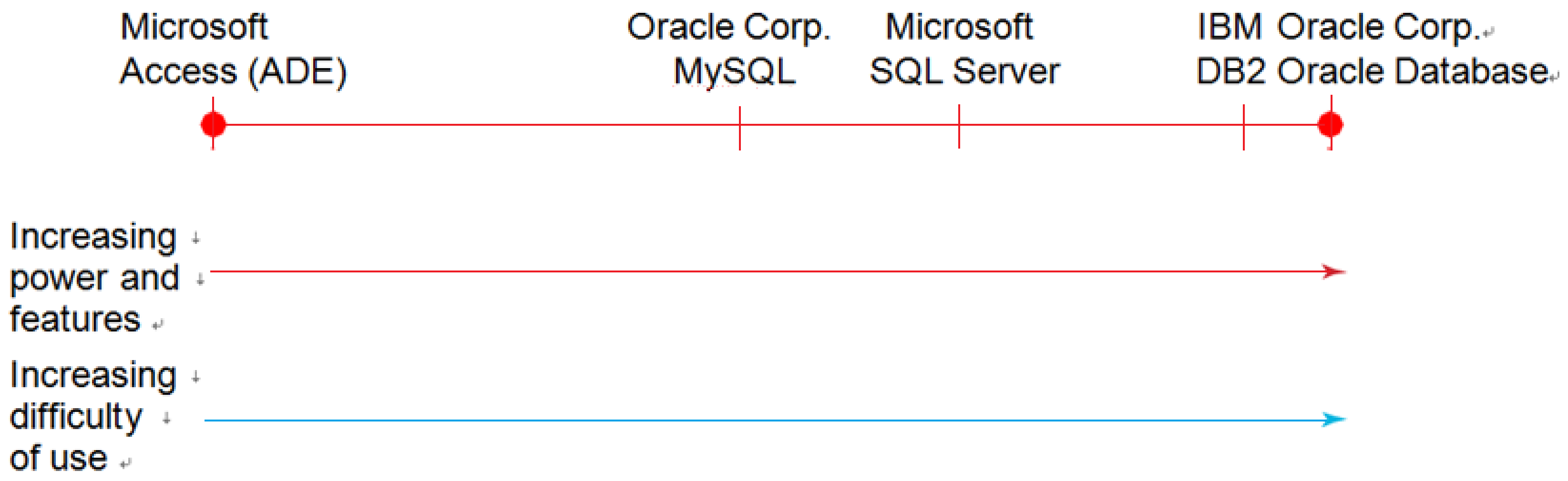
  All of these database applications get and put database data by sending SQL statements to the DBMS. These applications may create forms and reports, or they may send their results to other programs. They also may implement application logic that goes beyond simple form and report processing. For example,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

4. Personal Versus Enterprise-Class Database Systems

- **The DBMS in an Enterprise-Class Database System**

  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 1-17 presents the five most prominent DBMS products. The products are shown in order of increasing power, features, and difficulty of use.

Figure 1-17 Common Professional View of DBMS Products


Microsoft Access (ADE)
Oracle Corp. MySQL
Microsoft SQL Server
IBM DB2
Oracle Corp. Oracle Database
Increasing power and features
Increasing difficulty of use

4. Personal Versus Enterprise-Class Database Systems

- **The DBMS in an Enterprise-Class Database System**

Microsoft Access (really the Microsoft ADE) is the easiest to use and the least powerful. Oracle MySQL is a powerful, open source DBMS frequently chosen for Web applications. Microsoft SQL Server has far more power than its stablemate Microsoft Access—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 Database from Oracle Corporation. Oracle Database can be configured to offer very high performance on exceedingly large databases that operate 24/7, year after year. Oracle Database is also far more difficult to use and administer than Microsoft SQL Server.

5. Database Design

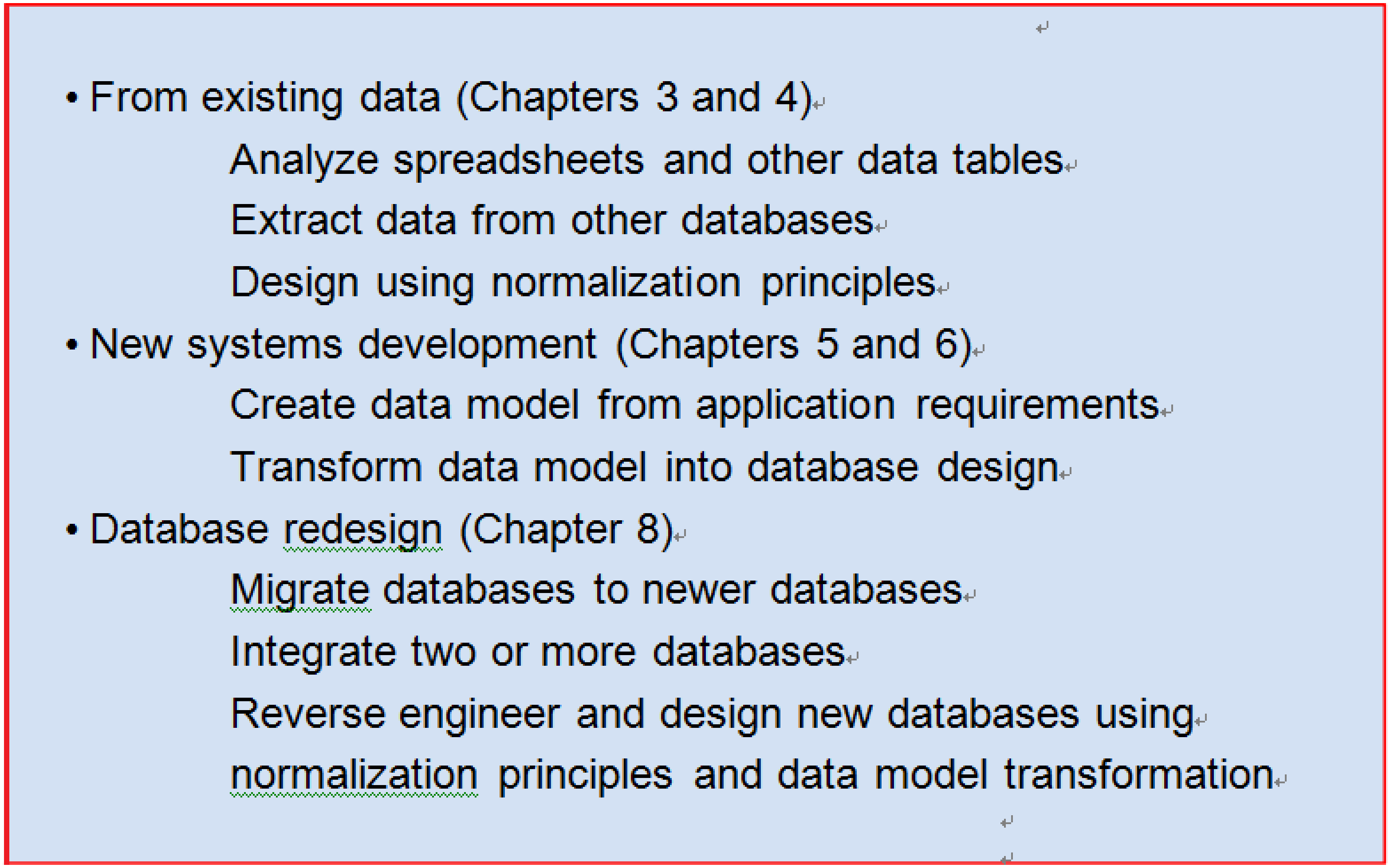
Database design is both difficult and important. Determining the proper structure of tables, the proper relationships among tables, the appropriate data constraints, and other structural components is challenging, and sometimes even daunting. Consequently, the world is full of poorly designed databases. 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.

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 1-18, there are three types of database design:

- Database design from existing data
- Database design for new systems development
- Database redesign of an existing database

5. Database Design

Figure 1-18 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 8)
  - Migrate databases to newer databases
  - Integrate two or more databases
  - Reverse engineer and design new databases using normalization principles and data model transformation

5. 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 1-19. 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.

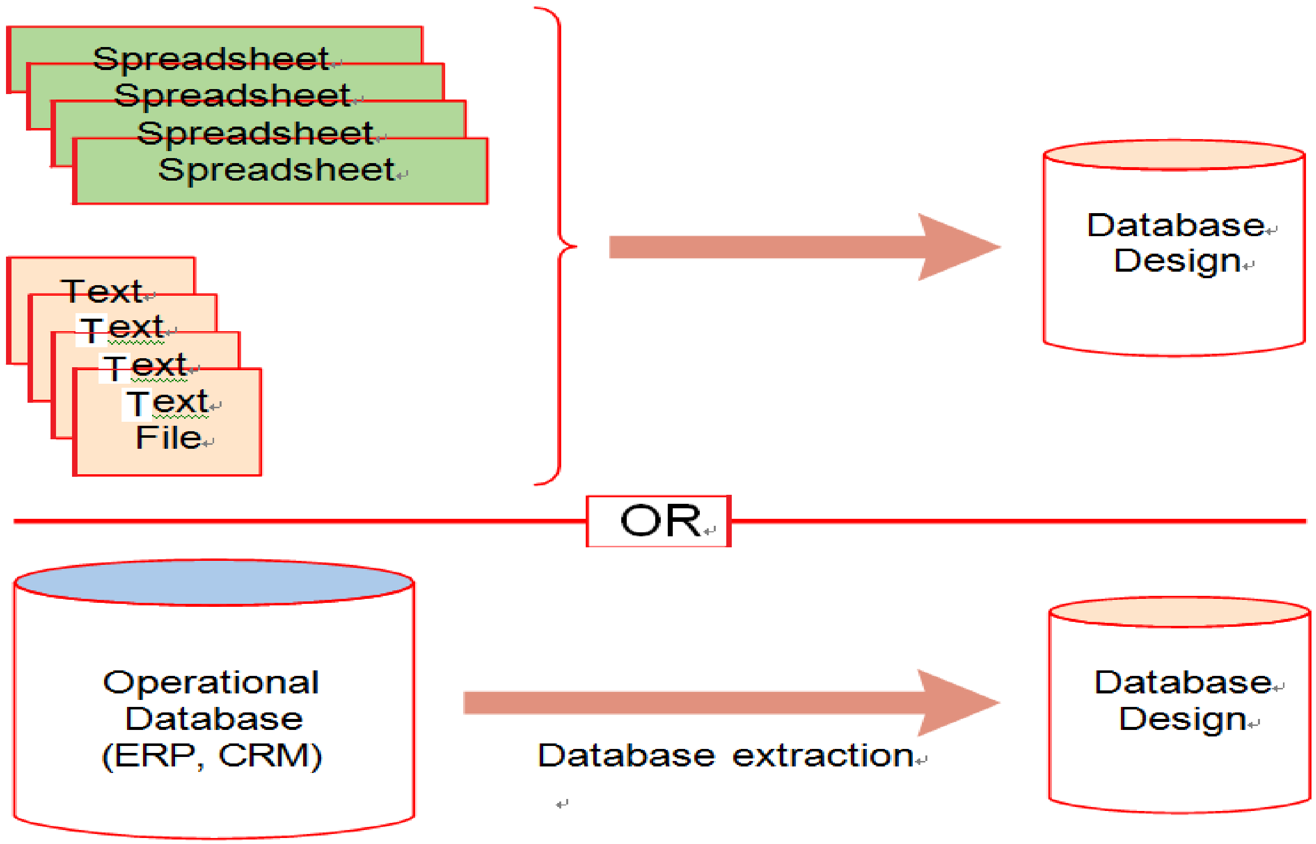
Alternatively, databases can be created from extracts of other databases. This alternative is especially common in business intelligence (BI) systems, which include reporting and data mining applications. For example, data from an operational database, such as a CRM or ERP database, may be copied into a new database that will be used only for studies and analysis. As you will learn in Chapter 13, such databases are used in facilities called data warehouses and data marts. The data warehouse and data mart databases store data specifically organized for research and reporting purposes, and these data often are exported to other analytical tools, such as SAS’ s Enterprise Miner, IBM's SPSS Data Modeler, or TIBCO's Spotfire Metrics.

5. Database Design

➢**Database Design from Existing Data**

Figure 1-19

Databases Originating from Existing Data



5. Database Design

➢ **Database Design from Existing Data**

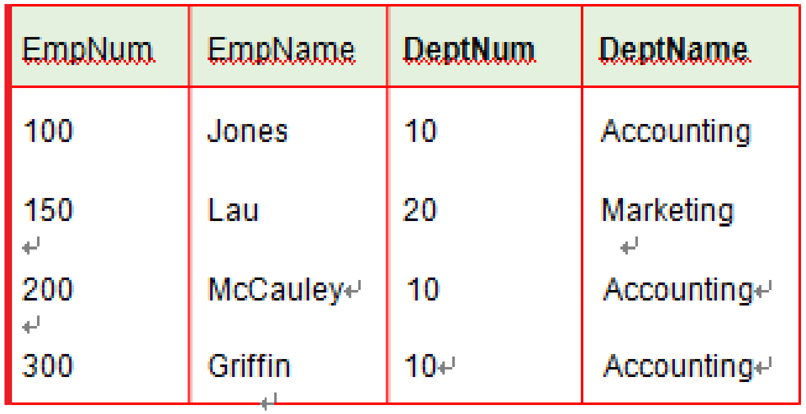
When creating a database from existing data, database developers must determine the appropriate structure for the new database. A common issue is how the multiple files or tables in the new database should be related. However, even the import of a single table can pose design questions. Figure 1-20 shows 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.

5. Database Design

➢ **Database Design from Existing Data**

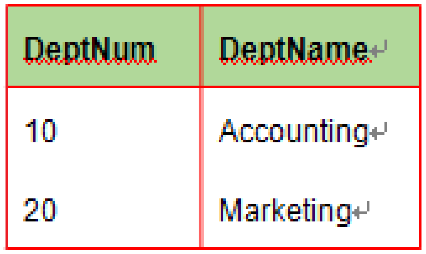
Figure 1-20 Data Import: One or Two Tables?



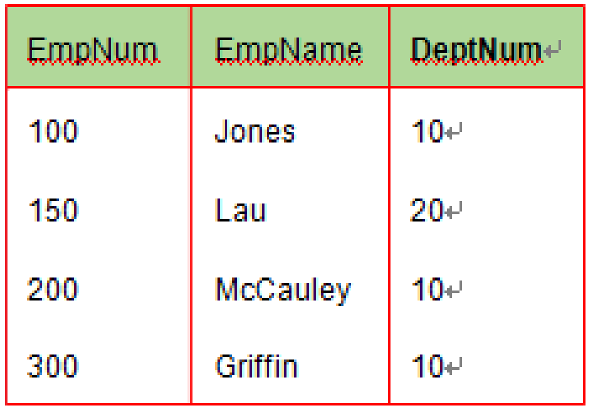
EmpNum	EmpName	DeptNum	DeptName
100	Jones	10	Accounting
150	Lau	20	Marketing
200	McCauley	10	Accounting
300	Griffin	10	Accounting

(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

5. Database Design

➢ Database Design for New Systems Development

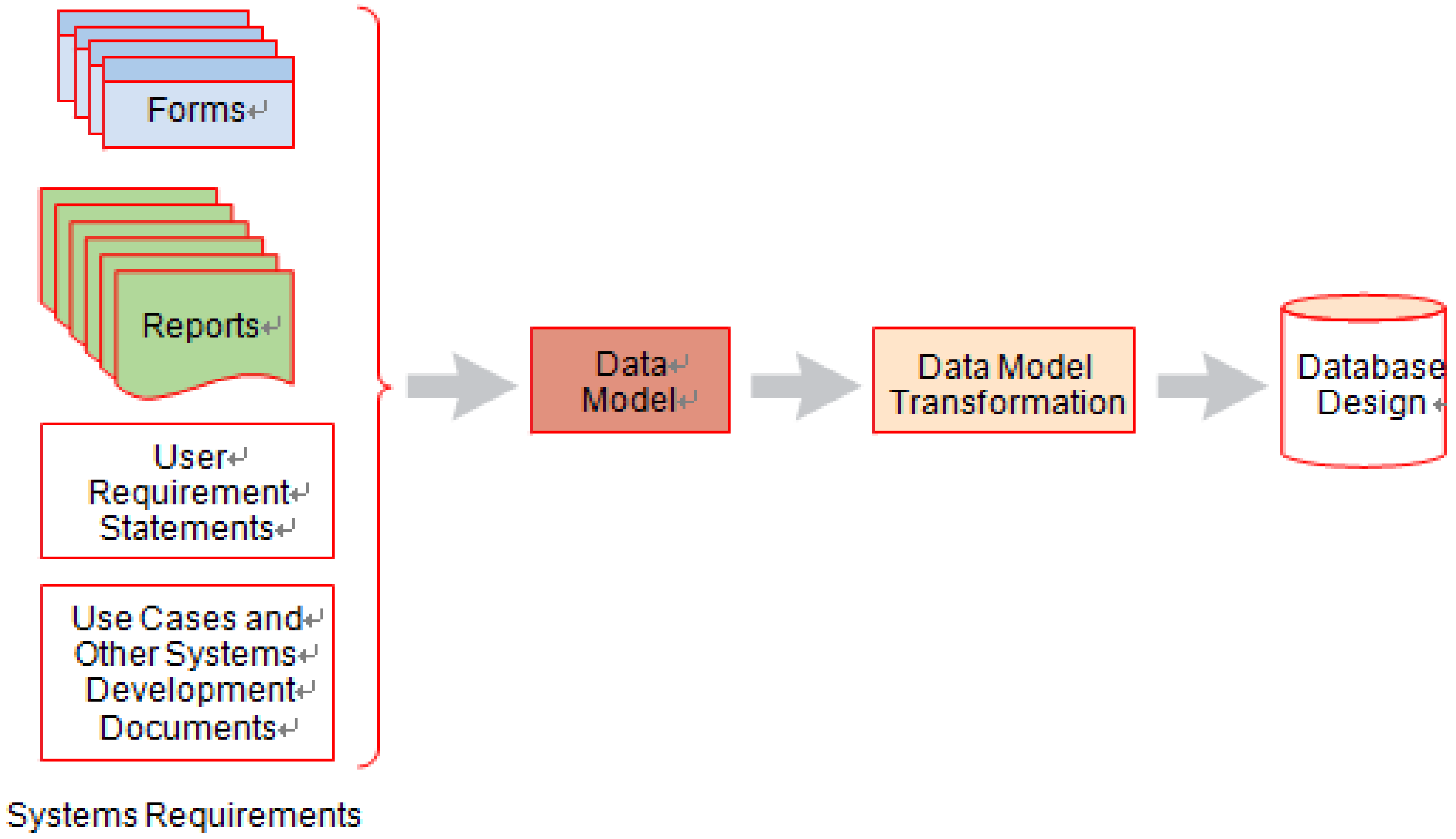
A second way that databases are designed is for the development of new information systems. As shown in Figure 1-21, requirements for a new system, such as desired data entry forms and reports, user requirements statements, use cases, and other requirements, are analyzed to create the database design.

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 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, which is the basis for constructing the actual database in a DBMS.

In Chapter 5, you will learn about the most popular data modeling technique—entityrelationship (ER) 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 6, you will learn how to transform entity relationship data models into database designs.

5. Database Design

➢ **Database Design for New Systems Development**


Forms
Reports
User Requirement Statements
Use Cases and Other Systems Development Documents
Systems Requirements
Data Model
Data Model Transformation
Database Design

5. Database Design

- **Database Redesign**

Database redesign also requires that databases are designed. As shown in Figure 1-22, 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.

5. Database Design

- **Database Redesign**

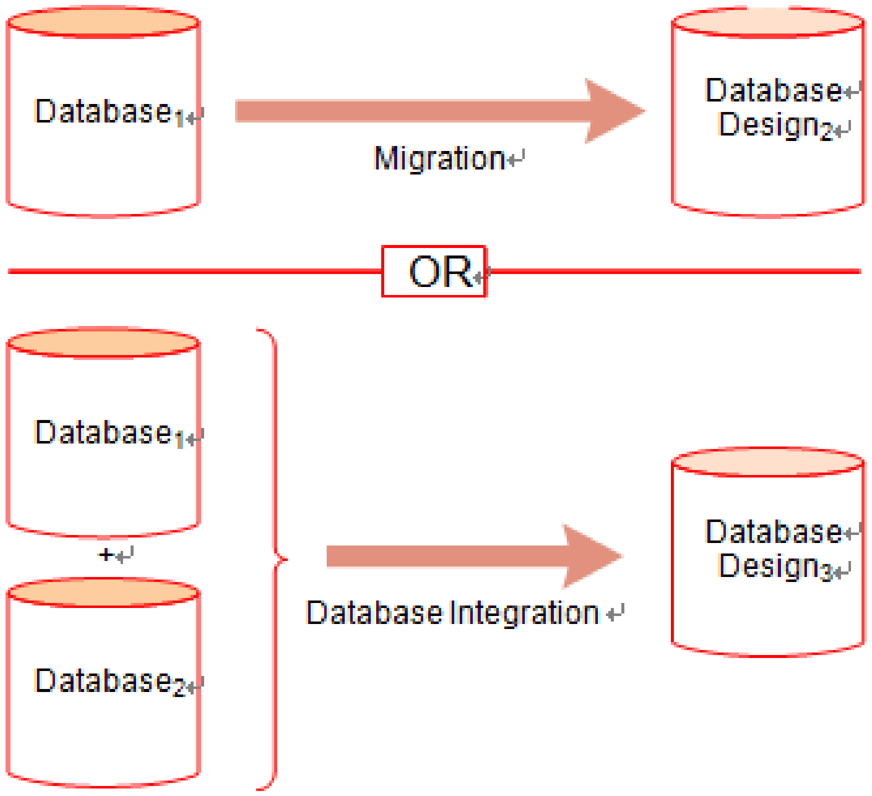
Database redesign is complicated. There is no getting around that fact. If this is your first exposure to database design, your instructor may skip this topic. If this is the case, after you have gained more experience you should reread this material. In spite of its difficulty, database redesign is 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 Chapter 8, after we present SQL statements and techniques for creating and altering the tables that make up a database in Chapter 7.

5. Database Design

➢**Database Redesign**

Figure 1-22 Databases Originating from Database Redesign



6.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 1-23.

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 1-24 shows our 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.

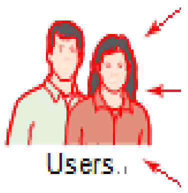
By the way The most exciting and interesting jobs in technology are always those on the leading edge. If you live in the United States and are concerned about outsourcing, a recent study by the Rand Corporation2 indicates that the most secure jobs in the United States involve the adaptation of new technology to solve business problems in innovative ways.

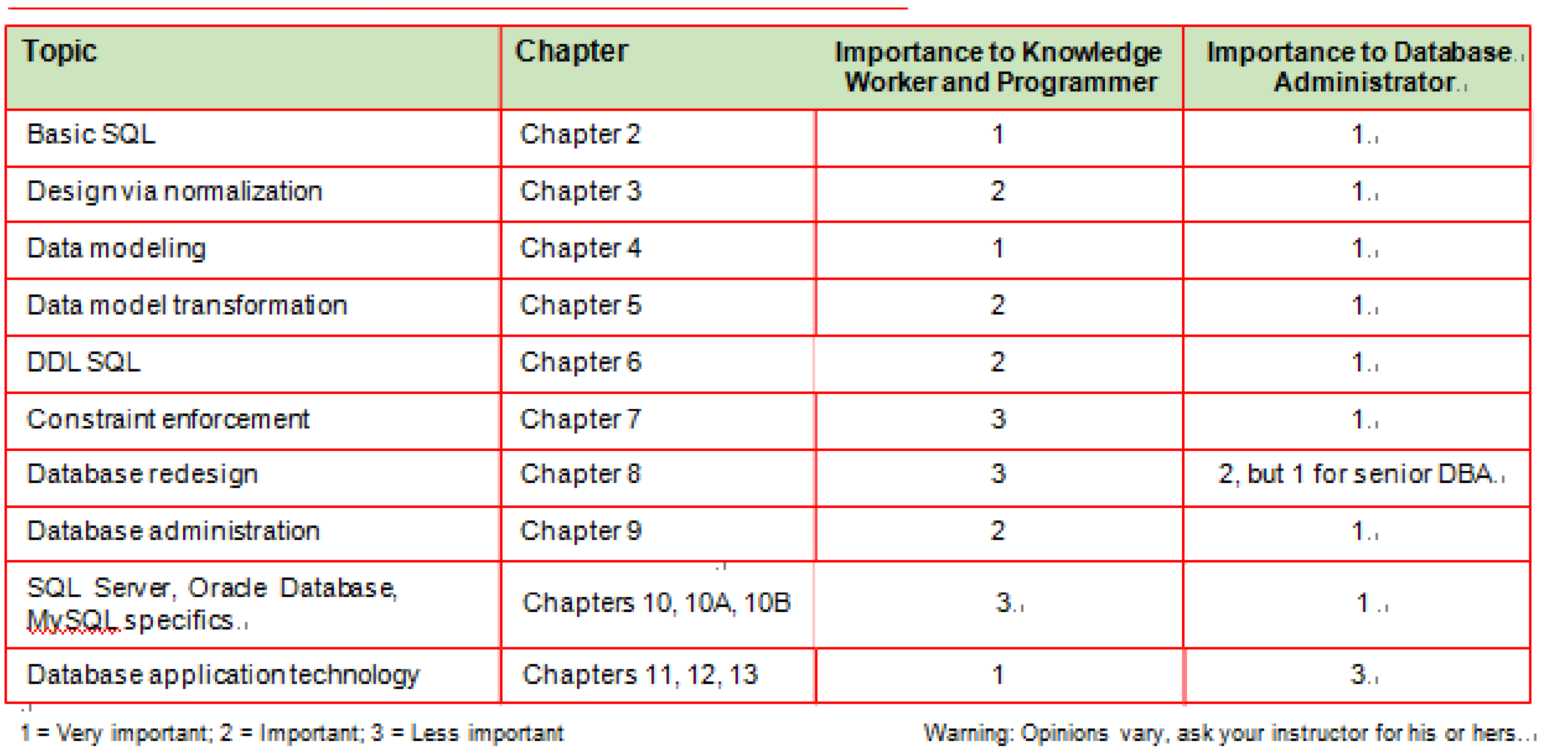
Right now, the leading edge involves the integration of XML, Web services, and database processing. You will need all of the fundamentals presented in this book, especially the material in Chapter 12, to work in this exciting new area.

6.What You Need to Learn

Figure 1-23 Working Domains of Knowledge Workers, Programmers, and Database Administrators

Figure 1-24 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 4	1	1
Data model transformation	Chapter 5	2	1
DDL SQL	Chapter 6	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 Database, MySQL specifics	Chapters 10, 10A, 10B	3	1
Database application technology	Chapters 11, 12, 13	1	3

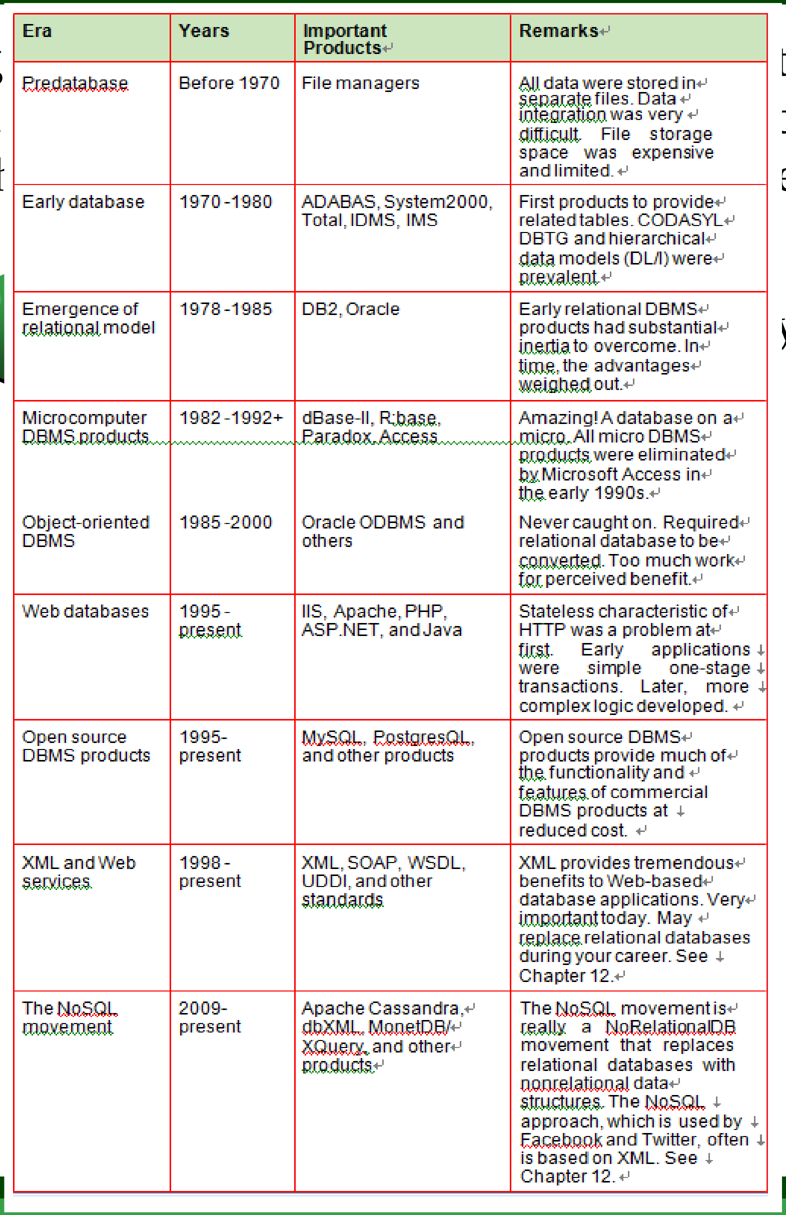
1 = Very important; 2 = Important; 3 = Less important

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

7.A Brief History of Database Processing

Database processing tinuously evolving and changing since then. cinating and thoroughly enjoyable field in wl e major eras of database processing.

y



Era	Years	Important Products	Remarks
Predatabase	Before 1970	File managers	All data were stored in separate files. Data integration was very difficult. File storage space was expensive and limited.
Early database	1970-1980	ADABAS, System2000, Total, IDMS, IMS	First products to provide related tables. CODASYL DBTG and hierarchical data models (DL/I) were prevalent.
Emergence of relational model	1978-1985	DB2, Oracle	Early relational DBMS products had substantial inertia to overcome. In time, the advantages weighed out.
Microcomputer DBMS products	1982-1992+	dBase-II, R:base, Paradox, Access	Amazing! A database on a micro. All micro DBMS products were eliminated by Microsoft Access in the early 1990s.
Object-oriented DBMS	1985-2000	Oracle ODBMS and others	Never caught on. Required relational database to be converted. Too much work for perceived benefit.
Web databases	1995-present	IIS, Apache, PHP, ASP.NET, and Java	Stateless characteristic of HTTP was a problem at first. Early applications were simple one-stage transactions. Later, more complex logic developed.
Open source DBMS products	1995-present	MySQL, PostgresQL, and other products	Open source DBMS products provide much of the functionality and features of commercial DBMS products at reduced cost.
XML and Web services	1998-present	XML, SOAP, WSDL, UDDI, and other standards	XML provides tremendous benefits to Web-based database applications. Very important today. May replace relational databases during your career. See Chapter 12.
The NoSQL movement	2009-present	Apache Cassandra, dbXML, MonetDB/XQuery, and other products	The NoSQL movement is really a NoRelationalDB movement that replaces relational databases with nonrelational data structures. The NoSQL approach, which is used by Facebook and Twitter, often is based on XML. See Chapter 12.

7.A Brief History 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 (which is now itself a limited use technology) has more capacity than many disks of that era. Memory was expensive as well. In 1969, we were processing payroll on a computer that had just 32,000 bytes of memory, while the computer on which this history is being written has 2 gigabytes of memory.

Integrated processing was an important but very difficult problem. An insurance company, for example, wanted to relate customer account data to customer claim data. Accounts were stored on one magnetic tape, and claims were stored on another. To process claims, the data on the two tapes had to be integrated somehow.

The need for data integration drove the development of the first database technology. By 1973, several commercial DBMS products had emerged. These products were in use by the mid-1970s. The first edition of this text, copyrighted 1977, featured the DBMS products ADABAS, System2000, Total, IDMS, and IMS. Of those five, only ADABAS and IMS are still in use, and neither of them has substantial market share today.

7.A Brief History of Database Processing

➢ **The Early Years**

Those early DBMS products varied in the way that they structured data relationships. One method, called Data Language/I (DL/I) used hierarchies or trees (see Appendix G) to represent relationships. IMS, which was developed and licensed by IBM, was based on this model. IMS had success at many organizations, particularly among large manufacturers, and is still in limited use today.

Another technique for structuring data relationships used data structures called networks. The CODASYL Committee (the group that developed the programming language COBOL) sponsored a subcommittee called the Database Task Group (DBTG). This subcommittee developed a standard data model that came to bear its name—the CODASYL DBTG model. It was an unnecessarily complicated model (everyone’s favorite idea made it into the committee’s design), but several successful DBMS products were developed using it. The most successful was IDMS, and its vendor, the Cullinane Corporation, was the first software company to be listed on the New York Stock Exchange. To the best of our knowledge, no IDMS database is in use today.

7.A Brief History of Database Processing

➢ **The Emergence and Dominance of the Relational Model**

In 1970, a then little-known IBM engineer named E. F. Codd published a paper in the Communications of the ACM3 in which he applied the concepts of a branch of mathematics called relational algebra to the problem of "shared data banks," as databases were then known. The results of this work are now the relational model for databases, and all relational database DBMS products are built on this model.

Codd's work was at first viewed as too theoretical for practical implementation. Practitioners argued that it was too slow and required so much storage that it would never be useful in the commercial world. However, the relational model and relational database DBMS products became adopted as the best way to create and manage databases.

The 1977 edition of this text featured a chapter on the relational model (which Codd himself reviewed). Many years later, Wayne Ratliff, the creator of the dBase series of products for personal computers, stated that he had the idea for dBase while reading that very chapter.4

By the way Today, there are as many opportunities for innovation as there were for Wayne Ratliff in 1977. Perhaps you can read Chapter 12 and develop an innovative product that integrates XML and DBMS processing in a new way, or join the NoSQL movement and help develop an alternative to relational database technology. Just as in 1977, no product has a lock on the future. Opportunity awaits you!

7.A Brief History of Database Processing

- **The Emergence and Dominance of the Relational Model**

The relational model, relational algebra, and, later, SQL made sense. They were not needlessly complicated; rather, they seemed to boil down the data integration problem to a few essential ideas. Over time, Codd convinced IBM management to develop relational-model DBMS products. The result was IBM’ s DB2 and its variants, which are still very popular today.

Meanwhile, other companies were considering the relational model as well, and by 1980 several more relational DBMS products had been released. The most prominent and important of those was Oracle Corporation’ s Oracle Database (the product was originally just named Oracle, but was renamed as Oracle Database after Oracle Corporation acquired other products and needed to distinguish their DBMS product from the others). Oracle Database achieved success for many reasons, one of which was that it would run on just about any computer and just about any operating system. (Some users complained, “Yes, and equally badly on all of them.” Another, when asked “Should we sell it to communist Russia?” responded, “Only as long as they have to take the documentation with it.”)

However, in addition to being able to run on many different types of machines, Oracle Database had, and continues to have, an elegant and efficient internal design. You will learn aspects of that design in the concurrency-control section in Chapter 10A. That excellent design, together with hard-driving and successful sales and marketing, has pushed Oracle Database to the top of the DBMS market.

7.A Brief History of Database Processing

- **The Emergence and Dominance of the Relational Model**

Meanwhile, Gordon Moore and others were hard at work at Intel. By the early 1980s, personal computers were prevalent, and DBMS products were developed for them. Developers of microcomputer DBMS products saw the advantages of the relational model and developed their products around it. dBase was the most successful of the early products, but another product, R:base, was the first to implement true relational algebra and other operations on the PC. Later, another relational DBMS product named Paradox was developed for personal computers. Eventually, Paradox was acquired by Borland.

Alas, it all came to an end when Microsoft entered the picture. Microsoft released Microsoft Access in 1991 and priced it at $99. No other PC DBMS vendor could survive at that price point. Microsoft Access killed R:base and Paradox, and then Microsoft bought a dBase “work-alike” product called FoxPro and used it to eliminate dBase. Microsoft has now stopped upgrading Microsoft FoxPro, now named Microsoft Visual FoxPro, but Microsoft will continue to support it until 2014 (see http://en.wikipedia.org/wiki/Visual_FoxPro).

7.A Brief History of Database Processing

- **The Emergence and Dominance of the Relational Model**

Thus, Microsoft Access is the only major survivor of that bloodbath of PC DBMS products. Today, the main challenge to Microsoft Access actually comes from Oracle Corporation and the open source software development community, who have taken over development of OpenOffice.org, a downloadable suite of free software products that includes the personal database OpenOffice.org Base (see www.openoffice.org), and its sister product LibreOffice (see www.libreoffice.org). LibreOffice is a related development of OpenOffice that was started when Oracle Corporation acquired Sun Microsystems in early 2010.

7.A Brief History of Database Processing

- **Post-Relational Developments**

In the mid-1980s, object-oriented programming (OOP) emerged, and its advantages over traditional structured programming were quickly recognized. By 1990, some vendors had developed object-oriented DBMS (OODBMS or ODBMS) products. These products were designed to make it easy to store the data encapsulated in OOP objects. Several specialpurpose OODBMS products were developed, and Oracle added OOP constructs to Oracle to enable the creation of a hybrid called an object-relational DBMS.

OODBMS never caught on, and today that category of DBMS products is fading away. There were two reasons for their lack of acceptance. First, using an OODBMS required that the relational data be converted from relational format to object-oriented format. By the time OODBMS emerged, billions upon billions of bytes of data were stored in relational format in organizational databases. No company was willing to undergo the expensive travail of converting those databases to be able to use the new OODBMS.

Second, object-oriented databases had no substantial advantage over relational databases for most commercial database processing. As you will see in the next chapter, SQL is not object oriented. But it works, and thousands of developers have created programs that use it. Without a demonstrable advantage over relational databases, no organization was willing to take on the task of converting their data to OODBMS format.

7.A Brief History of Database Processing

- **Post-Relational Developments**

Meanwhile, the Internet took off. By the mid-1990s, it was clear that the Internet was one of the most important phenomena in history. It changed, forever, the ways that customers and businesses relate to each other. Early Web sites were nothing more than online brochures,but within a few years dynamic Web sites that involved querying and processing databases began to appear.

However, one substantial problem existed. HTTP is a stateless protocol; a server receives a request from a user, processes the request, and then forgets about the user and the request. Many database interactions are multistage. A customer views products, adds one or more to a shopping cart, views more products, adds more to the shopping cart, and eventually checks out. A stateless protocol cannot be used for such applications.

Over time, capabilities emerged to overcome this problem. Web application developers learned to add SQL statements to their Web applications, and soon thousands of databases were being processed over the Web. You will learn more about such processing in Chapter 11. An interesting phenomenon was the emergence of open source DBMS products. Open source products generally make the source code widely available so that a group of programmers not bound to a single company can contribute to the program. Further, some forms of these products are usually offered as free downloads, although other forms or product support must be purchased from the company that owns the product.

7.A Brief History of Database Processing

- **Post-Relational Developments**

A good example of this is the MySQL DBMS (www.mysql.com). MySQL was originally released in 1995 by the Swedish company MySQL AB. In February 2008, Sun Microsystems bought MySQL AB, and in January 2010 Oracle Corporation completed its acquisition of Sun Microsystems. This means that Oracle Corporation now owns two major DBMS products: Oracle Database and Oracle MySQL. At present, MySQL continues to be available as an open source product, and the free MySQL Community Server edition can be downloaded from the MySQL Web site. MySQL has proven to be especially popular with Web site developers who need to run Web page queries against an SQL DBMS on a Web server running the Linux operating system. We will work with MySQL in Chapter 10B.

MySQL is not the only open source DBMS product—in fact, as this is being written there are 72 listed on the Wikipedia category page http://en.wikipedia.org/wiki/Category: Open_source_database_management_systems.

7.A Brief History of Database Processing

- **Post-Relational Developments**

One interesting outcome of the emergence of open source DBMS products is that companies that typically sell proprietary (closed source) DBMS products now offer free versions of their products. For example, Microsoft now offers SQL Server 2008 R2 Express (www.microsoft. com/express/Database), and Oracle Corporation makes its Oracle Database 10g Express Edition available for free (www.oracle.com/technetwork/database/express-edition/overview/index.html). Although neither of these products is as complete or as powerful (for example, in terms of maxi-mum data storage allowed) as some other versions the companies sell, they are useful for projects that require a small database. They are also ideal for students learning to use databases and SQL.

7.A Brief History of Database Processing

➢ **Post-Relational Developments**

In the late 1990s, XML was defined to overcome the problems that occur when HTML is used to exchange business documents. The design of the XML family of standards not only solved the problems of HTML, it also meant that XML documents were superior for exchanging views of database data. In 2002, Bill Gates said that "XML is the lingua-franca of the Internet Age." As you will learn in Chapter 12, however, two key problems that remain are (1) getting data from a database and putting it into an XML document and (2) taking data from an XML document and putting it into a database. In fact, this is where future application programmers can enter the picture.

XML database processing was given a further boost with the definition of XML Web service standards such as SOAP (not an acronym), WSDL (Web Services Description Language), UDDI (Universal Description, Discovery, and Integration), and others. Using Web services, it is possible to expose nuggets of database processing to other programs that use the Internet infrastructure. This means, for example, that in a supply chain management application a vendor can expose portions of its inventory application to its suppliers. Further, it can do so in a standardized way.

7.A Brief History of Database Processing

- **Post-Relational Developments**

The last row in Figure 1-25 brings us to the present. Built on the development of XML, the NoSQL movement has emerged in recent years, particularly following a 2009 conference organized around work on open source distributed databases (discussed in Chapter 9). This movement should really be called a NoRelational movement, because the work is really on databases that do not follow the relational model introduced in this chapter and discussed in Chapter 3. As discussed in Chapter 12, these databases are often based on XML and are finding wide acceptance in such applications as Facebook and Twitter.

The NoSQL movement brings us to the edge of the IT volcano, where the magma of new technology is just now oozing from the ground. What happens next will be, in part, up to you.

Summary

Applicasion	Example Users	Numbers 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 in corporate database into their software products.
Customer Relationship Management(CRM)	Sales, marketing, or customer service departments	500	10 million rows	Major vendors such as Siebel and PeopleSoft build applications around for database
Enterprise Resource Planning(ERP)	An entire orginization	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 database.
Data mining	Business analysts	25	100,000 to million +	Data are extracted, reformatted, cleaned, and filtered for use by statistical data mining tools.

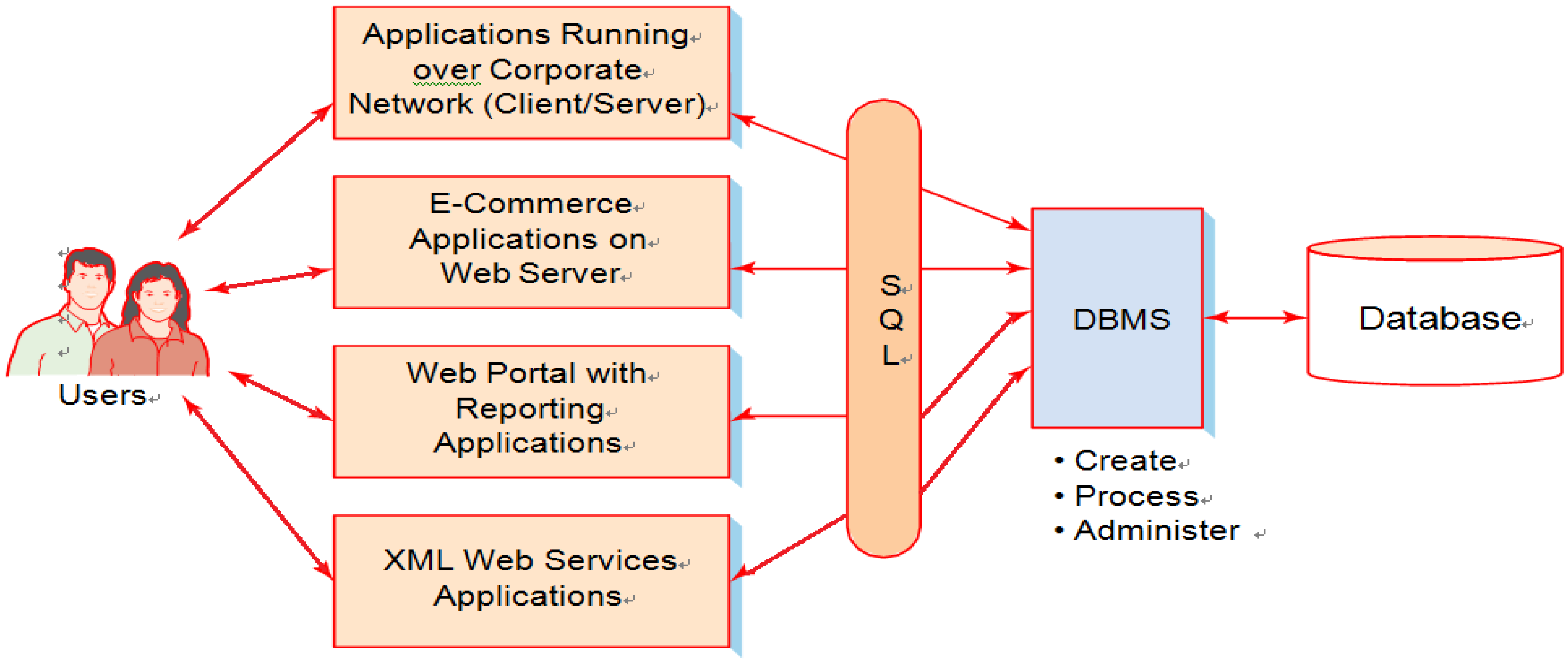
Summary

The functions of database applications are to create and process forms, to process user queries, and to create and process reports. Application programs also execute specific application logic and control the application. Users provide data and data changes and read data in forms, queries, and reports.

A DBMS is a large, complicated program used to create, process, and administer a database. DBMS products are almost always licensed from software vendors. Specific functions of a DBMS are summarized in Figure 1-12.

- Create database
- Create tables
- Create supporting structures (e.g., indexes)

A database is a self-descr lf-
describing collection of relate
relationships among rows of lumn. A
known
,
very
d
bases.
d,
ons,


Applications Running over Corporate Network (Client/Server)
E-Commerce Applications on Web Server
Web Portal with Reporting Applications
XML Web Services Applications
Users
S Q L
DBMS
Database
• Create
• Process
• Administer

Summary

The five most popular DBMS products, in order of power, features, and difficulty of use, are Microsoft Access, MySQL, SQL Server, DB2, and Oracle Database. Microsoft Access and SQL Server are licensed by Microsoft, DB2 is licensed by IBM, and Oracle Database and MySQL are licensed by Oracle Corporation.

Database design is both difficult and important. Most of the first half of this text concerns database design. New data-bases arise in three ways: from existing data, from new systems development, and from database redesign. Normalization is used to guide the design of databases from existing data. Data models are used to create a blueprint from system requirements. The blueprint is later transformed into a database design. Most data models are created using the entity-relationship model. Database redesign occurs when an existing database is adapted to support new or changed requirements or when two or more databases are integrated.

With regards to database processing, you can have one of two roles: user or database administrator. You may be a user of a database/DBMS as a knowledge worker or as an application programmer. Alternatively, you might be a database administrator who designs, constructs, and manages the database itself. The domains of each role are shown in Figure 1-23, and the priorities as to what you need to know for each role are shown in Figure 1-24.

Summary

The history of database processing is summarized in Figure 1-25. In the early years, prior to 1970, database processing did not exist, and all data were stored in separated files. The need for integrated processing drove the development of early DBMS products. The CODASYL DBTG and DL/I data models were prevalent. Of the DBMS products used at that time, only ADABAS and IMS are still in use.

The relational model rose to prominence in the 1980s. At first, the relational model was judged to be impractical, but over time relational products such as DB2 and Oracle Database achieved success. During this time, DBMS products were developed for personal computers as well. dBase, R:base, and Paradox were all PC DBMS products that were eventually consumed by the success of Microsoft Access.

Object-oriented DBMS products were developed in the 1990s but never achieved commercial success. More recently, Web-based databases have been developed to support e-commerce. Open source DBMS products are readily available, forcing commercial DBMS vendors to offer limited-capacity free versions of their enterprise products. Features and functions have been implemented to overcome the stateless nature of HTTP. XML and XML Web services databases are at the leading edge of database processing, as are the databases in the NoSQL movement.

LOGO

Thank You!