PART I

BASIC DATABASE CONCEPTS

CHAPTER 1

THE DATABASE APPROACH

CHAPTER OBJECTIVES

- Understand how the database approach is different and superior to earlier data systems
- Examine how information demand and technology explosion drive database systems
- Trace the evolution of data systems and note how we have arrived at the database approach
- Comprehend the benefits of database systems and perceive the need for them
- Survey briefly various data models, types of databases, and the database industry

Consider the following scenarios:

- You meet someone in a computer store. As a knowledgeable IT professional, you want to help this person. He says he is looking for *database* software to keep the names and addresses of his customers to do his mailings and billings. But what he really needs is a mail-merge program.
- You call your travel agent to make your airline reservations for the vacation you have been waiting for all year. The agent responds by saying that she cannot do that just now because the *database* is down. She really means that the reservations computer system is not working.
- Here is one more. You call your cellular phone company to complain about errors on the latest billing statement. The phone company representative says

Database Design and Development: An Essential Guide for IT Professionals by Paulraj Ponniah
ISBN 0-471-21877-4

that the *database* must have printed some incorrect numbers. What the representative really implies is that the billing application has miscalculated the charges.

In our modern society most people know the term *database* without understanding its full and clear meaning. Even in information technology circles, not everyone knows the concepts in reasonable detail. What is a *database*? Is it data? Is it software? Is it the place where you store data? Is there something special about the way you store data? Is it how you store and retrieve data? What exactly is a database system? What are the features and functions? Many more such questions arise.

Today, almost all organizations depend on their database systems for the crucial information they need to run their business. In every industry across the board, from retail chain stores to financial institutions, from manufacturing enterprises to government departments, and from airline companies to utility businesses, database systems have become the norm for information storage and retrieval. Database systems form the centerpiece of the growing and maturing electronic commerce. Database and Web technologies have merged.

The Information Technology department of today's organization has a primary responsibility: The department has to support and keep the database systems running. In this transformed computing environment, knowledge of database systems is no longer confined only to specialists such as data analysts and database administrators. Are you are a systems analyst, programmer, project leader, or network specialist? Then you also need to know the basics of database systems. You also need to grasp the significance of the database approach. All IT professionals need to study the basic principles and techniques of database design and development.

First, let us begin to understand how we got to this stage where most organizations depend on their database systems for running the business. Let us trace the evolution of data systems and see the essential need for the database approach. Let us understand what exactly the database approach is. Let us briefly survey the database industry and grasp the significance of the developments.

EVOLUTION OF DATA SYSTEMS

How were companies running their business before computers came into use? Even at that time, organizations needed information to execute the business processes, sell goods and services, and satisfy the needs of customers. Manual files supported business operations. Accounting personnel performed manual calculations and prepared invoices. Payroll departments manually wrote the checks. Business operations were reasonably satisfactory.

So, what happened? How did we get to the computer database systems of today? When computers were introduced in the 1960s, computer file systems replaced the manual files. This marked a significant leap in the way data was stored and retrieved for business operations. What has been really happening from that time until now, when database systems have become the norm? What prompted the progress toward database systems?

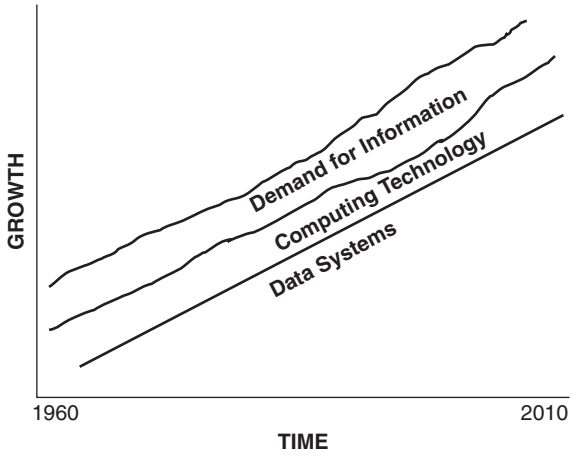

Figure 1-1 Technology growth and demand for information.

From the 1970s onward, two striking and remarkable phenomena were distinctly observed. Refer to Figure 1-1 indicating these two major developments.

First, demand for information has escalated in every organization. Organizations have steadily become global and widespread. Organizations have to contend with fierce competitive pressures. They need vast and complex information to stay in business and make a profit. Second, the past three decades have witnessed a huge, explosive growth in information technology. Processors have become faster, cheaper, and smaller. Operating systems have become powerful and robust. Data storage media have expanded tremendously in capacity; data storage prices have tumbled. Network and communication technology can now connect any remote site without difficulty. Application programming and people-machine interface have dramatically improved.

The escalating demand for information and the explosive growth in information technology have worked hand in hand to bring about the evolution to database systems. Ever-increasing demand for information drives the need for better methods of storing and retrieving data, for faster ways of processing data, and for improved methods of providing information. The demand for more and better information drove the technology growth. Progress in technology, in turn, spurred the capability to provide different types of information, not just to run day-to-day operations of an organization, but also to make strategic decisions.

Let us first examine the pertinent aspects of the technology explosion as related to data systems, because these are what we are specifically interested in. Then let us discuss the escalating demand for information that has prompted better and improved data systems.

Technology Explosion

If you have been in the information technology area for 5–10 years, you are certainly an eyewitness to the explosive growth. Growth is not confined to any one

sector. All aspects of the technology have been improving tremendously. Here are some specifics:

- Twenty-five years ago, there were only 50,000 computers in the whole world; now more than 500,000 are installed every day.
- More than 60% of American households have at least one computer; more than 50% have e-mail and Internet access.
- Growth of the Internet and the use of the Web have overshadowed the PC breakthrough of the 1970s; at the beginning of 2000, about 50 million households worldwide were estimated to be using the Internet; by the end of 2005, this number is expected to grow 10-fold.
- About 7 years ago, there were only 50 websites; now 100,000 are added every hour.
- Databases in the terabyte range are becoming common; a few years ago, even the gigabyte range was unusual.
- In the mid-1960s, programmers in large corporations had to write programs that had to run on 12K machines; today even your personal computer at home has 10,000 times larger memory.

Growth has not been isolated here and there in hardware and software. We notice explosive growth in all sectors of information technology. Let us proceed further to look at specific areas of information technology that are related to data systems.

Data Storage Devices Have you seen an 80-column card that very early computer systems used to store data? Each column in a card had holes punched to represent a single character. So a card could hold up to 80 characters. Keypunch operators typed data and program code into the cards. In the next stage, computer systems stored data on magnetic tapes. Initially, magnetic tapes of 800 BPI (bytes per inch) were used. Then we moved on to higher densities of 1600 BPI and 6250 BPI. For a brief while, paper tapes with punched holes were used as the storage medium. Special-purpose paper tape readers were used to read data from paper tapes.

It was a large leap forward when disk drives began to replace the earlier data storage media. Disk drives in mainframes consist of sets of large circular disks arranged in parallel with a common spindle. Sophisticated disk drives have come to stay as the common storage device of choice. Today's data servers use RAID (redundant array of inexpensive disks) technology as the advanced fault-tolerant storage system. Data storage devices have progressed tremendously from the primitive punched cards to the sophisticated RAID systems.

Three-and-a-Half-Inch Disk Drives You are very familiar with the three-and-a-half-inch disk drives in your home computer system. Just review the progress in the capacities of these disk drives. See how the capacities kept doubling every year. Note the following details:

1992	1 gigabyte
1993	2 gigabytes

1994	4 gigabytes
1995	9 gigabytes
1997	18 gigabytes
2000	50 gigabytes

Computer Applications Over the years, the types of computer applications have changed and progressed from mere bookkeeping applications to multimedia and data mining applications. Some of you might remember the days when the computer department was known as the data processing department. Applications in those days just processed data in elementary ways to produce some reports. The technology explosion resulted in a grand transition of computer usage from simple to increasing sophistication. Review the following details.

Data Processing Applications (DP). In the early days of computing, computer departments built applications just to replace clerical labor. Mostly, these applications performed simple accounting and financial functions. These applications produced straightforward reports. Speed and accuracy of the computer in performing calculations were the primary factors. Computer systems stored and retrieved data from magnetic tapes and earlier versions of disk drives. Applications used sequential or flat files to organize data.

Management Information Systems (MIS). In the next stage, growth of technology manifested itself in applications that went beyond accounting and finance to supporting the entire core business of an organization. Applications began to appear to process orders, manage inventory, bill customers, pay employees, and so on. Organizations depended on their management information systems for their day-to-day business. Storage and retrieval of data mostly depended on hard disks. Many applications adopted the use of database technology.

Decision-Support Systems (DSS). Further technology growth in processor speed, storage media, systems software, and database techniques pushed the application types to systems that supported strategic decision making. These applications are not meant for supporting day-to-day operations of a business but for providing information to executives and managers to make strategic decisions. In which markets should the company expand? Where should the next distribution warehouse be built? Which product lines should be discontinued? Which ones should be boosted? These applications dealt with sales analysis, profitability analysis, and customer support. Decision-support systems made use of improved storage facilities and newer features of database technology.

Data Warehousing (DW) and Data Mining (DM) Systems. In recent years, with the enormous progress in processor scalability, mass storage, and database methods, organizations are able to forge ahead with their applications, especially in building data warehousing and data mining systems. These recent decision-support systems, much more sophisticated than earlier attempts, require large volumes of data and complex analytical techniques. These systems need large databases specially designed and built separately from the databases that support the day-to-day operational systems.

Data Systems What is the effect of the technology explosion on the way data is organized? Over the years, how were businesses organizing data? We just looked at the way applications have progressed from simpler types toward increasing sophistication. What about data systems?

Manual-Type Records. Very early computer applications worked with data stored on punched cards and paper tapes. Keypunch operators prepared data on these primitive media from manual files and records. Computer applications read data from cards and tapes to prepare reports.

Sequential Files. Improved storage media such as magnetic tapes and early disk drives enabled application developers to organize data as sequential (or flat) files. Each file contained data records of the same type arranged sequentially one after the other, usually in the order in which they were created. Sorting techniques allowed data records to be resorted in a different sequence.

Databases. Increased sophistication in data storage techniques on hard disk drives and enhancements to operating systems enabled random and quick access of data. Data systems moved to a wholly new level. Applications were able to store data in databases and retrieve data sequentially and randomly.

Demand for Information

Of the two major factors that mutually contributed to the database approach to computing, so far we have considered the explosive growth of technology. Let us now turn our attention to the other factor, namely, the escalating demand for information. It is not just more information that organizations need. The demand for information includes several dimensions.

Consider how billing requirements and sales analysis have changed. In the early years of computing, organizations were happy if they could bill their customers once a month and review total sales by product quarterly. Now it is completely different. Organizations must bill every sale right away to keep up the cash flow. They need up-to-date customer balance and daily and cumulative sales totals by products. What about inventory reconciliation? Earlier systems provided reports to reconcile inventory or to determine profitability only at the end of each month. Now organizations need daily inventory reconciliation to manage inventory better, daily profitability analysis to plan sales campaigns, and daily customer information to improve customer service.

In the earlier period of computing, organizations were satisfied with information showing only current activity. They could use the information to manage day-to-day business and make operational decisions. In the changed business climate of globalization and fierce competition, this type of information alone is no longer adequate. Companies need information to plan and shape their future. They need information, not just to run day-to-day operations, but to make strategic decisions as well.

What about the delivery of information now compared to the early days of computing? Today, online information is the norm for most companies. Fast response times and access to large volumes of data have become essential. Earlier computer

systems just provided reports, mostly once a month, a few once a week, and a small number once a day.

Organizations have come to realize that information is a key asset to be carefully managed and used for greater profitability. In summary, demand for information by today's enterprises contains the following attributes:

- More information
- Newer purposes
- Different information types
- Integrated information
- Information to be shared
- Faster access to information

Waves of Evolution

As we have seen so far, information technology, along with and because of the escalating demand for information, has made giant strides in the past few decades. Evolution to higher levels is evident in every aspect of information technology. The evolution has taken place in distinct waves. Refer to Figure 1-2.

Note carefully the evolution of information technology in the three major areas. First note how the very methods of computing technology have progressed from mainframes to client/server architecture. The centrally administered mainframes have made room for the client/server configuration in which each set of machines can perform specialized tasks.

What about the way in which humans interface with computers? In earlier days, we punched data on cards and fed them to be read by the early computers. Then came the CRTs (cathode-ray terminals) where textual data could be typed into the computer through the use of keyboards. Point-and-click GUIs (graphical user interfaces) proved to be a major improvement. Now, interfacing with computers through

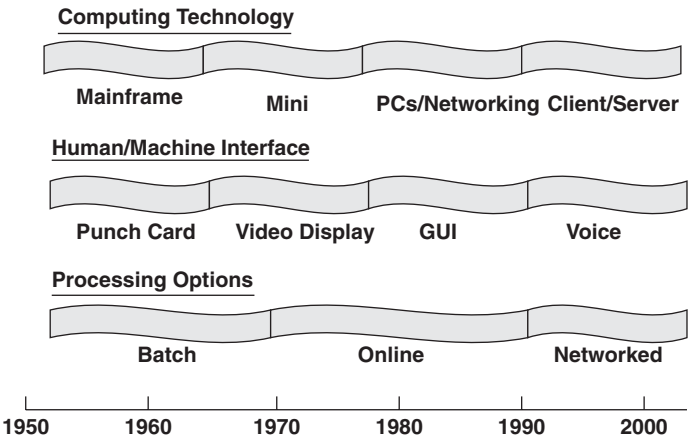

Figure 1-2 Information technology: waves of evolution.

the human voice is gaining ground. What a major transition from punch cards to direct voice input!

In the early days of computing, transactions were batched together for processing at the end of a stipulated period. For example, you could not invoice each sale as it happened. You had to collect and batch all the sales for a month and run the batched sales through the invoicing application. We moved to online transaction processing in the next wave. Now transactions are transmitted and processed over LANs (local area networks) and WANs (wide area networks).

File-Oriented Data Systems

As the demand for information continued to increase, organizations began to adopt improved file systems to store and access data. The file-oriented data systems essentially mimicked the manual file systems. The computer files corresponded to the paper files. In a filing cabinet, you store file folders and each file folder contains file records. Similarly, the computer systems use electronic files containing records. For example, a customer file would contain records, with each record containing data about a single customer. In the beginning, these computer files were primarily used for accounting applications. As we shall discuss in some detail, file-oriented systems have serious limitations. Therefore, organizations needed to go to better and improved methods for data storage and access.

File-oriented systems started out by using sequential or flat files. When you need to retrieve the 100th record from a sequential file, you have to read and bypass the first 99 records before you can get to record number 100. This is a very serious shortcoming. Therefore, slightly better methods of retrieval evolved. This was the transition to improved file-oriented data systems. Let us review how storage and retrieval methods apply to a customer file.

Sequential File. Customer records are stored in the sequence in which they are entered into the file. Record retrieval is sequential. For the file to be processed in any other sequence, it must be sorted in the required sequence.

ISAM File. This is the indexed-sequential access method. The customer records in the data file are stored sequentially, similar to the sequential file method. However, another index file is created, for example, with the customer numbers and the physical addresses of the records. When the record of a specific customer is needed, as done previously, you do not have to read the records of the data file one after the other until you find the record you are looking for. You can read the smaller index file, find the record you are looking for, and then use the physical address of the data record stored in the index file.

VSAM File. This is based on virtual storage access method, a major improvement over ISAM files. VSAM files provide for indexed access. Also, this method provides for storing and retrieving records directly from the customer file without an index file. In the direct method, the address where a customer record is stored may be calculated from the customer number with a specialized algorithm.

WHY DATABASE SYSTEMS?

We traced the evolution of data systems. We grasped the essentials of the explosive growth of information technology. We noted the escalating demand of organizations for information. We observed how growth in information technology and the increased demand for information worked hand in hand. Increasing demand for information spurred the growth of information technology. Growth of information technology, in turn, enabled organizations to satisfy the increasing demand for information.

Let us summarize the driving forces for organizations to adopt database systems. A major reason is the inadequacy of the earlier file-oriented data systems. We shall review the limitations and see how database systems overcome the limitations and provide significant benefits.

The Driving Forces

Among others, four major forces drove organizations to adopt database systems. Figure 1-3 illustrates these four major forces.

Information as a Corporate Asset. Today, companies strongly realize that information is a corporate asset similar to other assets such as cash, plant and equipment, or inventory. Proper management of key assets is essential for success. Companies understand that it is essential to manage information as a key asset. They understand the need to find improved methods for storing, retrieving, and using information.

Explosive Growth of Computer Technology. Computer technology, especially data storage and retrieval systems, has grown in a phenomenal manner. Without growth in this sector, it is unlikely that we could have progressed to database systems that need sophisticated ways of data storage and retrieval.

Escalating Demand for Information. We have noted the increase in demand for information by organizations, not only in volume but in the types of information as

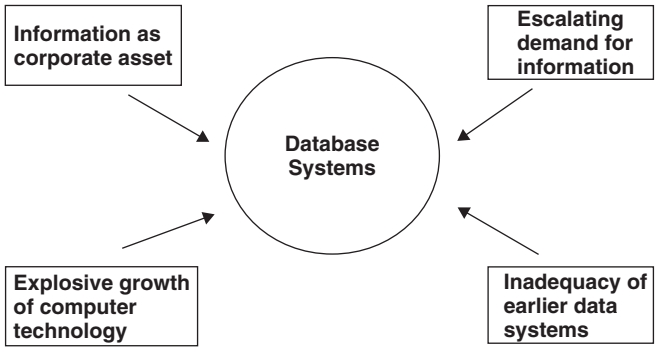

Figure 1-3 Database systems: the driving forces.

well. If companies did not need more and newer types of information, there would have been no impetus for development of database systems. The earlier data systems might have been satisfactory.

Inadequacy of Earlier Data Systems. Suppose the earlier data systems were able to meet the escalating demand for information. Then why bother to find better methods? But the fact is that these earlier systems were grossly inadequate to meet the information demands. Storage and management of large volumes of data were not adequate. Finding and retrieving information were extremely difficult. Protecting the information asset of a company was nearly impossible with the earlier data systems. Why was this so? How were the earlier systems inadequate? In what ways could they not meet the information demands? Understanding the limitations will give you a better appreciation for database systems.

Inadequacy of Earlier Data Systems

Assume that you work for a company called Progressive Book Distributors in the early 1970s. Your company purchases books from various publishers and sells them to retail bookstores and book clubs. The computer applications in your company would work with sequential files because those are the types of data systems available at that time. Here is a list of possible sequential computer files in your company.

Customer master file Every time a new customer comes on board, a record is created in the file, with a new customer number, in the order in which the customers are added.

Book master file As each new book is added to the inventory, a record is created in the file, with ISBN identifying each book.

Salesperson file As each new salesperson is hired, the person is given an identification number and data about him or her is added to the file.

Sale transaction file Each sale is recorded with the date of the sale.

Publisher file As each new publisher is included, a record is created in the file, with a new publisher number, in the order in which the publishers are added.

Payment transaction file Payments received from customers are recorded with the date of the payment.

Refer to Figure 1-4 showing the fields and sample data for each file.

What types of information is your company's staff looking for? Do they need to print an invoice for customer Allbooks Book Store for the sale on January 10, 2002? Do they want the total sales of all books from publishers Ron Fairchild during the month of December 2001? Do they want a list of all customers in New York state? In today's computing environment, no one will think of these requests as difficult or impossible. This was not the case with file-oriented applications of the early 1970s. File-oriented data systems have serious limitations.

Let us take just one specific example of printing statements to customers for sales. This had to be done in a batch mode at the end of a reasonable interval such as at

CUSTOMER MASTER FILE

CustNo	CustName	Address	Country
1000	Allbooks Book Store	5757 Westheimer, Houston, TX 77057	U.S.A.
1010	Akito Books	Chiyoda-ku, Tokyo 100	Japan
1040	Robert Smith Ltd.	10 Bonds St., London W1A 2AA	U.K.
2050	Sally Orobetz	8 Hazelton Ave., Toronto, Ontario M5R 2E1	Canada

BOOK MASTER FILE

ISBN	Title	Author	PubId
2093356790	DB Design	Carey	100
2101155897	DW Fundamentals	McMillan	200
1558712215	Art Appreciation	Stewart	300
3456765432	Existentialism	Ernst	400

PUBLISHER FILE

PubId	PublisherName	Country
100	Ron Fairchild	USA
200	Crosley	U.K.
300	Summer Hill	Canada
400	Ully Wille	Germany

SALESPERSON FILE

SalRepId	SalRepName	Office	Comm%
10	Williams	Chicago	13
20	Harreld	London	12
30	Swamy	Toronto	9
40	Katzman	Munich	10

SALE TRANSACTION FILE

SalDate	CustNo	SalRepId	ISBN	Qty	Amount
6-Apr	1000	10	2093356790	10	799.50
10-Apr	1010	20	2101155897	10	699.50
19-Apr	1040	30	1558712215	20	1,000.50
24-Apr	2050	40	3456765432	10	750.00

PAYMENT TRANSACTION FILE

PymtDte	CustNo	PayMethod	Amount
13-Apr	1000	Amex	600.00
22-Apr	1010	Visa	500.00
24-Apr	1040	Check	800.00
30-Apr	2050	Visa	650.00

Figure 1-4 Progressive Book Distributors: files.

the end of a month. All the sales transactions and payment transactions for the month had to be batched together and processed in a batch mode.

Figure 1-5 indicates a flowchart of the jobs that must be run to produce the billing statements.

Even an initial review of the flowchart reveals that there are too many sorts for just producing simple billing statements. That is because each file is kept in a sequence that is not useful for the processing logic of the entire application. The whole concept of batch processing is very inflexible. Suppose in the middle of the month you need the billing statements for just a few select customers. There is no easy way of doing this. You have to run the complete batch process in the middle of the month and then separate out the statements you need.

File-oriented systems are inadequate to face the challenges of increasing demand for information. Especially when companies care for information as a key asset, the earlier file-oriented data systems possess severe limitations. Let us discuss the important limitations so that we can appreciate how database systems overcome these shortcomings.

Uncontrolled Data Redundancy In file-oriented systems, each application has its own set of files. Each application creates and stores data just for the use of that application. For example, in a bank, there could be three separate applications—one for checking accounts, one for savings accounts, and another one for loan

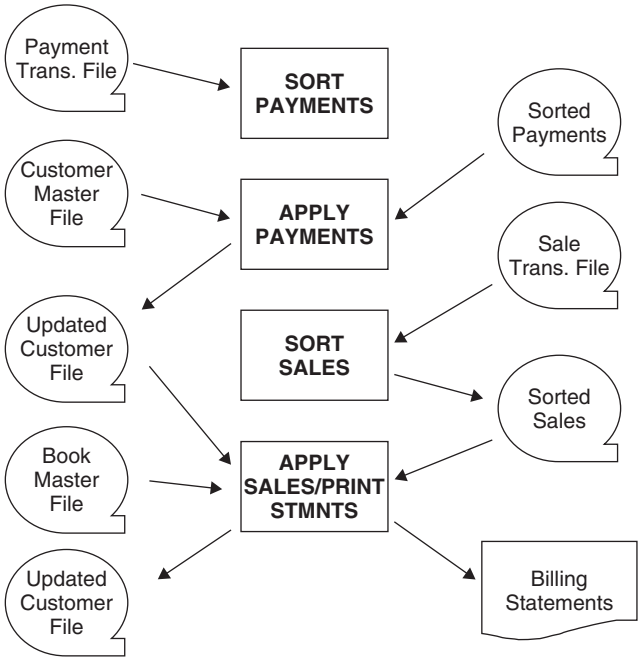

Figure 1-5 Flowchart for billing application.

accounts. If a bank customer has a checking account, a savings account, and a loan account, data about that customer such as name and address are kept separately in each of these applications. There is unnecessary and uncontrolled duplication of the same data in the bank's computer files. Similarly, the possibility of data duplication exists in the inpatient and outpatient accounts of a medical center. In an auction business, data duplication is possible in sellers' and buyers' accounts.

Obviously, data duplication results in wasted storage space. In the bank example, it is very likely that many customers have both checking and savings accounts. In auction businesses, dealers of art and other such items are customers recorded in both sellers' and buyers' account applications. In these cases, the wastage of storage space can be enormous. Multiple input of same data item also requires extra time and resources that could be applied for other, useful purposes.

Inconsistent Data Data redundancy or duplication of data in your computer files can cause serious inconsistency in the data. Suppose a bank customer has both checking and savings accounts. If that person's name, address, or both are different in the two accounts, then the data about that customer are inconsistent in the bank's files. Which set of data is correct? It is possible that the name of that customer in one system is correct and the address as recorded in the other application is correct. Inconsistency of data is a direct result of data duplication.

Field sizes and formats of the same data item might be different in the various applications. In the checking and savings accounts, there could be just one long field for the address to be recorded in textual format. The loan account, being a later application, could have separate fields for street address, city, state, and zip code.

Such variations are likely causes for data integrity problems. Variations in names and addresses may cause erroneous printing of names and addresses on documents mailed to customers.

Inflexibility When an application cannot adapt to changing requirements, we say that the application is inflexible. By the very nature of sequential files, a file-oriented system can process transactions only in batch mode. That is, transactions are accumulated and processed as batches. You cannot print an invoice for a single customer for a single sale. Sequential files allow retrieval of data records, one at a time, in sequence. Such files do not possess the flexibility to meet new and changing information requirements.

Suppose you are interested in finding and listing all the purchases made by customers in Japan for the past three months. Or you want a list of all customers in a certain zip code range. It is very difficult to satisfy such ad hoc queries in a file-oriented system without reprogramming.

Limited Data Sharing Consider two typical applications, namely, order processing and inventory control. Each of these applications needs data on products. But data on products are repeated in each of these two applications when the business depends on file-oriented data systems. There is no sharing of data on products between the two applications. If product descriptions of certain products are changed, these changes must be made in both applications.

Difficult Data Integration Let us get back to the bank example with separate customer files in the checking, savings, and loan applications. If you wanted to combine data from these applications and send consolidated statements to customers showing the transactions in all three accounts, it would be nearly impossible with file-oriented data systems. You would have to run special programs to extract banking transactions from each application. Then you would have to come up with methods for matching customer accounts from each application and consolidate the transactions in a single statement, which is not an easy task. The proliferation of files and duplication of data continue as each new application is implemented with its own set of files.

Poor Enforcement of Standards and Controls Standards relate to data names, formats, value restrictions, and access controls. When duplicated data are spread across many applications, it is extremely difficult to enforce standards. For example, if the standard in your company is for the customer name field to be 35 bytes, then you will have to impose this standard, not in one place, but in many applications that store customer names. Suppose you have to include a business rule that the employee daily wage rate must be between 0 and 100; you may have to stipulate this rule in at least two different applications, namely, payroll and human resources.

Problems with the resolution of homonyms and synonyms deserve special attention. File-oriented data systems are likely to have problems with these.

Homonyms. If a single field name represents two or more objects in different applications, it is called a homonym. For example, the field name *balance* may represent

the checking balance in the checking accounts application. The same field name *balance* may also represent the savings balance in the savings accounts application. In this case, the term *balance* is a homonym. In file-oriented data systems, it is hard to control homonyms.

Synonyms. If different field names in different applications represent the same object, these names are known as synonyms. As an example, consider how a student may be referred to in a college data system. In the student registration system, the student may be referred to by the field name *student-number*. On the other hand, in the majors and graduation system, the student may be referred to by the field name *candidate-number*. The terms *student-number* and *candidate-number* are synonyms. In file-oriented data systems, controlling synonyms is difficult.

Excessive Program Maintenance How is program maintenance a problem in file-oriented systems? Consider the coding of computer programs in file-oriented applications. These programs are usually written in third-generation languages like COBOL. A program in languages like COBOL contains two distinct sections. One section of the program, namely, FD or file definition section, has the structures of the files used in the program embedded within the program itself. The other section has the processing logic. Data structures and processing logic are interconnected and combined together in the program. This means that reference to data is not independent of the processing logic. There is no data independence.

What is the effect of the lack of data independence in file-oriented systems? Consider all the computer programs that use the customer file for processing. In each of these programs, the structure of the customer file is embedded within the program itself. What happens when a new field such as cellular phone number is added to the customer record? Every one of the programs using the customer file need to be changed and recompiled, irrespective of whether the program uses the cellular phone number field or not. Lack of data independence results in excessive program maintenance.

Productivity Losses Two main factors cause reduction in productivity while using file-oriented systems:

- The same data need to be maintained in multiple places.
- Because of lack of data independence, multiple programs need to be changed frequently.

Database Systems Meet the Challenges

As the demand for information escalated, it became urgent to overcome the limitations of file-oriented data systems. With these limitations, companies could not meet the requirements of increased demand for information. They needed a different approach to storing, retrieving, and managing data. They could not afford the productivity losses. They could not waste space because of data duplication in file-oriented systems.

Specialists at Rockwell and General Electric began to work on better methods for managing data. These methods attempted to overcome the limitations of

file-oriented systems. Data and processing logic had to be separated so as to improve programmer productivity. The new approach of using databases instead of conventional flat files addressed the challenges for meeting the increased demand for information. The database approach overcame the limitations of the earlier data systems and produced enormous benefits. Let us review the specific benefits and understand in what way the database approach is superior to the earlier data systems.

Minimal Data Redundancy Unlike file-oriented data systems where data are duplicated among various applications, database systems integrate all the data into one logical structure. Duplication of data is minimized. Wastage of storage space is eliminated. Going back to the bank example, with a database, customer data is not duplicated in the checking account, savings account, and loan account applications. Customer data is entered and maintained in only one place in the database.

Sometimes, in a database, a few data elements may have to be duplicated. Let us say that product data consist of product number, description, price, and the corresponding product line number. All the fields relating to product line data are kept separately. Whenever the details of products and product lines are needed in applications, both data structures are retrieved from the database. Suppose a heavily used product forecast application needs all the details of the product from product data and just the product line description from the product line data. In that case, it will be efficient for the product data to duplicate the product line description from the product line data. Thus, in some instances, data duplication is permitted in a database for the purpose of access efficiency and performance improvement. However, such data duplications are kept to a minimum.

Data Integrity Data integrity in a database means reduction of data inconsistency. Because of the elimination or control of data redundancy, a database is less prone to errors creeping in through data duplication. Field sizes and field formats are the same for all applications. Each application uses the same data from one place in the database. In a bank, names and addresses will be the same for checking account, savings account, and loan applications.

Data Integration In a database, data objects are organized into single logical data structures. For example, in file-oriented data systems, data about employees are scattered among the various applications. The payroll application contains employee name and address, social security number, salary rate, deductions, and so on. The pension plan application contains pension data about each employee, whereas the human resources application contains employee qualifications, skills, training, and education. However, all data about each employee are integrated and stored together in a database.

So, in a database, data about each business object are integrated and stored separately as customer, order, product, invoice, manufacturer, sale, and so on. Data integration enables users to understand the data and the relationships among data structures easily. Programmers needing data about a business object can go to one place to get the details. For example, data about orders are consolidated in one place as order data.

Data Sharing This benefit of database systems follows from data integration. The various departments in any enterprise need to share the company's data for proper functioning. The sales department needs to share the data generated by the accounting department through the billing application. Consider the customer service department. It needs to share the data generated by several applications. The customer service application needs information about customers, their orders, billings, payments, and credit ratings. With data integration in a database, the application can get data from distinct and consolidated data structures relating to customer, orders, invoices, payments, and credit status.

Data sharing is a major benefit of database systems. Each department shares the data in the database that are most pertinent to it. Departments may be interested in data structures as follows:

Sales department—*Customer/Order*

Accounting department—*Customer/Order/Invoice/Payment*

Order processing department—*Customer/Product/Order*

Inventory control department—*Product/Order/Stock Quantity/Back Order Quantity*

Database technology lets each application use the portion of the database that is needed for that application. User views of the database are defined and controlled. We will have more to say about user views in later chapters.

Uniform Standards We have seen that, because of the spread of duplicate data across applications in file-oriented data systems, standards cannot be enforced easily and completely. Database systems remove this difficulty. As data duplication is controlled in database systems and as data is consolidated and integrated, standards can be implemented more easily. Restrictions and business rules for a single data element need to be applied in only one place. In database systems, it is possible to eliminate problems from homonyms and synonyms.

Security Controls Information is a corporate asset and, therefore, must be protected through proper security controls. In file-oriented systems, security controls cannot be established easily. Imagine the data administrator wanting to restrict and control the use of data relating to employees. In file-oriented systems, control has to be exercised in all applications having separate employee files. However, in a database system, all data about employees are consolidated, integrated, and kept in one place. Security controls on employee data need to be applied in only one place in the database. Database systems make centralized security controls possible. It is also easy to apply data access authorizations at various levels of data.

Data Independence Remember the lack of data independence in file-oriented systems where computer programs have data structure definitions embedded within the programs themselves. In database systems, file or data definitions are separated out of the programs and kept within the database itself. Program logic and data structure definitions are not intricately bound together. In a client/server environment, data and descriptions of data structures reside on the database server, whereas

the code for application logic executes on the client machine or on a separate application server.

Reduced Program Maintenance This benefit of database systems results primarily from data independence in applications. If the customer data structure changes by the addition of a field for cellular phone numbers, then this change is made in only one place within the database itself. Only those programs that need the new field need to be modified and recompiled to make use of the added piece of data. Within limits, you can change programs or data independently.

Simpler Backup and Recovery In a database system, generally all data are in one place. Therefore, it becomes easy to establish procedures to back up data. All the relationships among the data structures are also in one place. The arrangement of data in database systems makes it easier not only for backing up the data but also for initiating procedures for recovery of data lost because of malfunctions.

THE DATABASE APPROACH

What are the implications when an organization makes the transition from file-oriented systems to database systems? When an organization changes its approach to management of data and adopts database technology, what are the significant effects in the way business is conducted? What happens when an organization embraces the database approach?

Let us find answers to such questions. We will discuss the way applications are designed and implemented with database systems. We will explore the basic concepts of the database approach. We will also review some of the types of databases and how these are used.

We have reviewed the benefits of database systems and established how they are superior to the earlier file-oriented data systems. We caught a glimpse of the features of database systems that produce several benefits. Database systems reduce data redundancy, integrate corporate data, and enable information sharing among the various groups in the organization. Now you are ready for an initial, formal definition of a database.

Database: A Formal Definition

Let us examine the following definition:

> A database is an *ordered collection* of *related data elements* intended to meet *the information needs* of an organization and designed to be *shared* by multiple users.

Note the key terms in the definition:

Ordered collection. A database is a collection of data elements. Not just a random assembly of data structures, but a collection of data elements put together deliberately with proper order. The various data elements are linked together in the most logical manner.

Related data elements. The data elements in a database are not disjointed structures without any relationships among them. These are related among themselves and also pertinent to the particular organization.

Information needs. The collection of data elements in a database is there for a specific purpose. That purpose is to satisfy and meet the information needs of the organization. In a database for a bank, you will find data elements that are pertinent to the bank's business. You will find customer's bank balances and ATM transactions. You will not find data elements relating to a student's major and examination grades that belong in a database for a university. You will not find a patient's medical history that really belongs in a database for a medical center.

Shared. All authorized users in an organization can share the information stored in its database. Integrated information is kept in the database for the purpose of sharing so that all user groups may collaborate and accomplish the organization's objectives.

Data-Driven, Not Process-Driven

When an organization adopts a database approach to managing corporate data, the very method of designing and implementing applications changes. Traditionally, when you design and implement an application with file-oriented data systems, you use a process-driven approach. That method changes with database systems. You shift your design and implementation method to a data-driven approach. Figure 1-6 shows the difference between the two methods.

In both methods, you begin with the definition of requirements. However, there is an essential difference between the two methods. In the process-driven method,

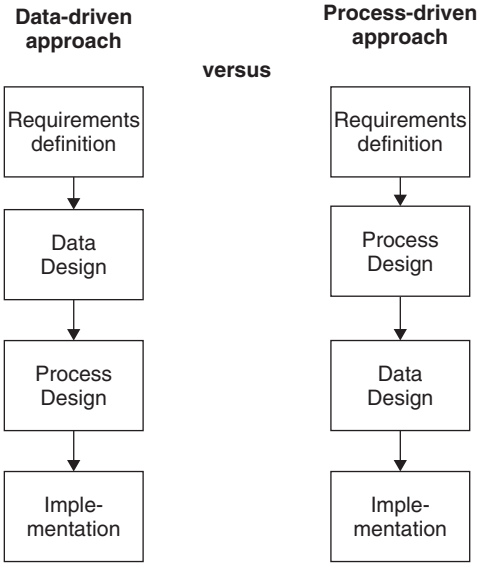

Figure 1-6 Data-driven, not process-driven.

you collect requirements for the outputs and the processes and then determine the inputs. Determination of the inputs leads to the design of the data files.

In the data-driven method, you gather requirements about the business objects that are pertinent to the business. You collect the data required to be included about these objects. Then you design the database to support the business. After this, the design of the initial processes follows.

Right away, you can see the advantage of the data-driven approach. Although you are interested in supporting the initial processes, because you have designed the database to contain all data relevant to the business, you will find it easy to add further processes later on. Data-driven approach provides enormous flexibility to provide for new and additional processing requirements at a later time. In practice, companies adopting the database approach use a combination of both methods, although the data-driven method dominates the application development phases.

Basic Concepts

You are now beginning to appreciate the significance of the database approach. You are discerning the major benefits of developing and using applications in a database environment. Before proceeding further, let us review a few fundamental concepts and become familiar with some key terminology.

Data Repository All data in the database reside in a data repository. This is the data storage unit where physical data files are kept. The data repository contains the physical data. Mostly, it is a central place of storage for the data content.

Data Dictionary The data repository contains the actual data. Let us say that you want to keep data about the customers of your company in your database. The structure of a customer's data could include fields such as customer name, customer address, city, state, zip code, phone number, and so on. Data about a particular customer could be as follows in the respective fields: Jane Smith/1234 Main Street/Piscataway/NJ/08820. There are two aspects of the data about customers. One aspect is the structure of the data consisting of the field names, field sizes, data types, and so on. This part is the structure of the data for customers. The other part is the actual data for each customer consisting of the actual data values in the various fields.

The first part relating to the structure resides separately in storage, and this is called the data dictionary or data catalog. A data dictionary contains the structures of the various data elements in the database. It also contains the relationships among data elements. The other part relating to the actual data about individual customers resides in the data repository. The data dictionary and the data repository work together to provide information to users.

Database Software Are Oracle and Informix databases? Oracle and Informix are really the software that manages data. These are database software or database management systems. Database software supports the storing, retrieving, and updating of data in a database. Database software is not the database itself. The software helps you store, manage, and protect the data in a database.

Data Abstraction Consider the example of customer data again. Data about each customer consist of several fields such as customer name, street address, city, state, zip code, credit status, and so on. We can look at customer data at three levels. The customer service representative can look at the customer from his or her point of view as consisting of only the fields that are of interest to the representative. This may be just customer name, phone number, and credit status. This is one level. The next level is the structure of the complete set of fields in customer data. This level is of interest to the database designer and application programmer. Another level is of interest to the database administrator, who is responsible for designing the physical layout for storing the data in files on disk storage.

Now go through the three levels. The customer service representative is just interested in what he or she needs from customer data, not the entire set of fields or how the data is physically stored on disk storage. The complexities of the other two levels may be hidden from the customer service representative. Similarly, the physical level of how the data is stored on disk storage may be hidden from the application programmer. Only the database administrator is interested in all three levels. This concept is the abstraction of data—the ability to hide the complexities of data design at the levels where they are not required. The database approach provides for data abstraction.

Data Access The database approach includes the fundamental operations that can be applied to data. Every database management system provides for the following basic operations:

- READ data contained in the database
- ADD data to the database
- UPDATE individual parts of the data in the database
- DELETE portions of the data in the database

Database practitioners refer to these operations by the acronym CRUD:

- C—Create or add data
- R—Read data
- U—Update data
- D—Delete data

Transaction Support Imagine the business function of entering an order from a customer into the computer system. The order entry clerk types in the customer number, the product code, and the quantity ordered. The order entry program reads the customer data and allows the clerk to sight verify the customer data, reads product data and displays the product description, reads inventory data, and finally updates inventory or creates a back order if inventory is insufficient. All these tasks performed by the order entry program to enter a single order comprise a single order entry transaction.

When a transaction is initiated it should complete all the tasks and leave the data in the database in a consistent state. That is, if the initial stock is 1000 units and the order is for 25 units, the stock value stored in the database after the transaction is

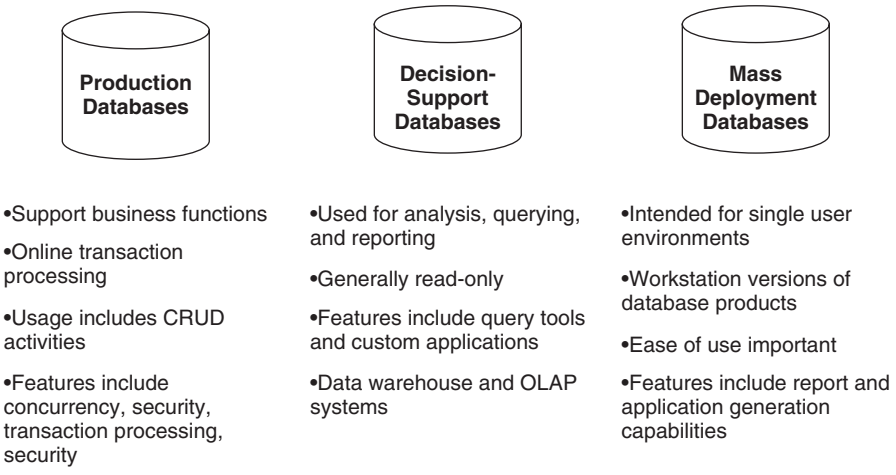

Figure 1-7 How databases are used.

completed must be 975 units. How can this be a problem? See what can happen in the execution of the transaction. First, the transaction may not be able to perform all its tasks because of some malfunction preventing its completion. Second, numerous transactions from different order entry clerks may be simultaneously looking for inventory of the same product. Database technology enables a transaction to complete a task in its entirety or back out intermediary data updates in case of malfunctions preventing completion.

How Databases Are Used

You now realize the use of databases for supporting the core business of an organization and enabling day-to-day operations. These are production databases that support the operational systems of an enterprise. More recently, with increasing demand for information, databases fulfill another important function. Databases provide support for strategic decision making in an organization. Such decision-support databases are designed and implemented separately and differently. Production databases and decision-support databases are large-scale databases for the several users within organizations.

Individuals and single departments may also use private databases. For example, a specialty department may want to send targeted mailings to specific customers and to keep these customers in a separate database. Individual business analysts may keep data and research results in a separate database just for their use. These are mass deployment individual databases.

Figure 1-7 shows the separation of databases by their uses and describes some of the features.

OVERVIEW OF DATA MODELS

A data model represents the data requirements of an organization. You can diagrammatically show a data model with symbols and figures. Data for an

organization reside in a database. Therefore, when designing a database, you first create a data model. The model would represent the real-world data requirements. It would show the arrangement of the data structures.

Database software has evolved to support different types of data models. As we try to represent real-world data requirements as close as possible in a data model, we come up with a replica of the real-world information requirements. It turns out that we can look at data requirements and create data models in a few different ways. At this stage, let us survey a few leading data models. Over time, different vendors have developed commercial database management systems to support each of these common data models.

Hierarchical

Let us examine the data requirements for a typical manufacturing company. Typically in manufacturing, you have major assemblies, with each major assembly consisting of subassemblies, each subassembly consisting of parts, each part consisting of subparts, and so on. In your database for the manufacturing company, you need to keep data for the assemblies, subassemblies, parts, and subparts. And the data model for manufacturing operations must represent these data requirements.

Think about this data model. This model should show that an assembly contains subassemblies, a subassembly contains parts, and a part contains subparts. Immediately you can observe that this data model must be hierarchical in nature, diagramming the assembly at the top with subassembly, part, and subpart at successive lower levels.

In the business world, many data structures are hierarchical in nature. You can notice a hierarchy in department, product category, product subcategory, product line, and product. You can trace a hierarchy in division, subdivision, department, and employee. Figure 1-8 illustrates one such model showing the hierarchy of customer, order, and order line item. A customer may have one or more orders, and an order may have one or more line items, perhaps one line item for each product ordered.

Let us review the key features of the hierarchical model by referring to Figure 1-8.

Levels. Each data structure representing a business object is at one of the hierarchical levels.

Parent-Child Relationships. The relationship between each pair of data structures at levels next to each other is a parent-child relationship. CUSTOMER is a parent data segment whose child is the ORDER data segment. In this arrangement, a child segment can have only one parent segment but one parent segment may have multiple child segments. You may want to separate orders into phone orders and mail orders. In that case, CUSTOMER may have PHONE ORDER and MAIL ORDER as two child segments.

Root Segment. The data segment at the top level of the hierarchy is known as the root data segment (as in an inverted tree).

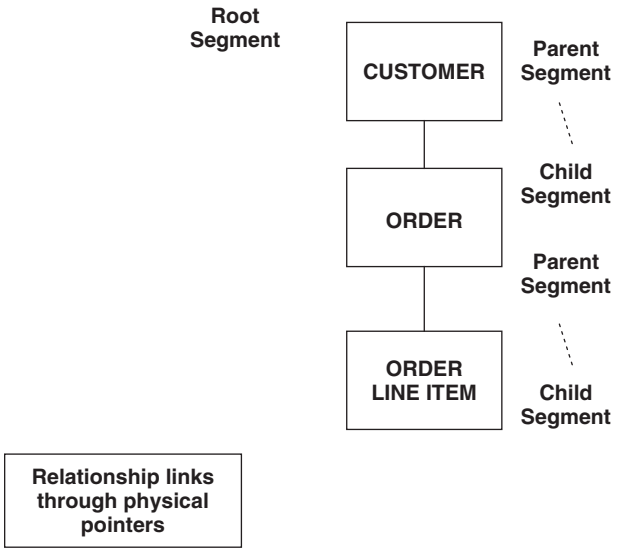

Figure 1-8 Hierarchical data model.

Physical Pointers. How are the orders of a particular customer linked in the implementation of the hierarchical data model? These linkages are by means of physical pointers or physical storage addresses embedded within physical records in the database. Physical pointers link records of the parent segments to those of the child segments by means of parent-child forward or backward pointers. Similarly, forward and backward physical pointers link records of the same segment type.

Network

The hierarchical data model represents well any business data that inherently contains levels one below the other. We have just discussed how the manufacturing application deals with hierarchical levels of plant inventory with assemblies broken down into lower-level components. The hierarchical data model suits this application well. However, in the real world, most data structures do not conform to a hierarchical arrangement. The levels of data structures do not fall into nice dependencies one below another as in a hierarchy. In the hierarchical data model, you have noticed that each data segment at any level can have only one parent at the next higher level. In practice, many sets of related elements may not be subjected to such restrictions.

Let us consider a common set of related data elements in a typical business. The data elements pertain to customers placing orders and making payments, salespersons being assigned, and salespersons being part of sales territories. All of these data elements cannot be arranged in a hierarchy. The relationships cross over among the data elements as though they form a network. Refer to Figure 1-9 and note how it represents a network arrangement and not a hierarchical arrangement. Observe the six data elements of sales territory, salesperson, customer, order, order line item, and payment as nodes in a network arrangement.

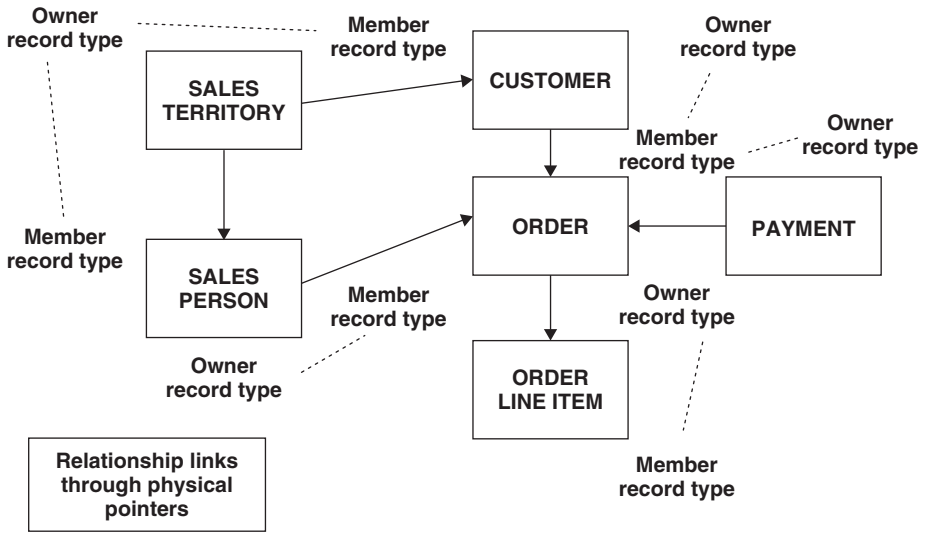

Figure 1-9 Network data model.

The network data model overcomes some of the limitations of the hierarchical data model. The network data model is more representative of real-world information requirements than the hierarchical model. The network data model can represent most business information.

Let us go over the key features of the network model by referring to Figure 1-9.

Levels. As in most real-world situations, no hierarchical levels exist in the network model. The lines in a network data model simply connect the appropriate data structures wherever necessary without the restriction of connecting only successive levels as in the hierarchical model. Note the lines connecting the various data structures with no restrictions.

Record Types. In the network data model, each data structure is known as a record type. For example, the CUSTOMER record type represents the data content of all customers. The ORDER record type represents the data content of all orders.

Relationships. The network data model expresses relationships between two record types by designating one as the owner record type and the other as the member record type. For each occurrence of an owner record type, there are one or more occurrences of the member record type. The owner record type may be reckoned as the parent and the member record type as the child. In a sense, the owner record type "owns" the corresponding member record type. Each member type with its corresponding owner record type is known as a set. A set represents the relationship between an owner and a member record type.

Multiple Parents. Look at the ORDER member record type. For ORDER there are two parents or owner records, namely, CUSTOMER and PAYMENT. In other

words, for one occurrence of CUSTOMER, one or more occurrences of ORDER exist. Similarly, for one occurrence of PAYMENT there are one or more occurrences of ORDER. By definition, a hierarchical data model cannot represent this kind of data arrangement with two parents for one child data structure.

Physical Pointers. Just as in the case of the hierarchical data model, related occurrences of two different record types in a network model are connected by physical pointers or physical storage addresses embedded within physical records in the database. Physical pointers link occurrences of an owner record type with the corresponding occurrences of the member record type. Within each record type itself the individual occurrences may be linked to one another by means of forward and backward pointers.

Relational

This book will provide you with in-depth discussions about the relational model. This data model is superior to the earlier models. Dr. E. F. Codd, the celebrated father of the relational model, stipulated the rules and put this model on a solid mathematical foundation. At this stage, however, we want to introduce the relational model as a superior data model that addresses the limitations of the earlier data models.

The earlier hierarchical data model is suitable for data structures that are naturally hierarchical, with each data structure placed at a certain level in the hierarchy. However, in the business arena, many of the data structures and their relationships cannot be readily placed in a hierarchical arrangement. The network data model evolved to dispense with the arbitrary restriction of the hierarchical model. Nevertheless, in both of these models, you need physical pointers to connect related data occurrences. This is a serious drawback because you have rewrite the physical addresses in the data records every time you reorganize the data, move the data to a different storage area, or change over to another storage medium. The relational model establishes the connections between related data occurrences by means of logical links implemented through foreign keys. Figure 1-10 illustrates the relational data model.

Let us note the key features of the relational data model by referring to Figure 1-10.

Levels. Just like the network data model, no hierarchical levels are present in the relational model. The lines in a relational data model simply indicate the relationships between the appropriate data structures wherever necessary without the restriction of connecting only successive levels as in the hierarchical model. As in the network model, note the lines connecting the various data structures with no restrictions.

Relations or Tables. The relational model consists of relations. A relation is a two-dimensional table of data observing relational rules. For example, the CUSTOMER relation represents the data content of all customers. The ORDER relation represents the data content of all orders.

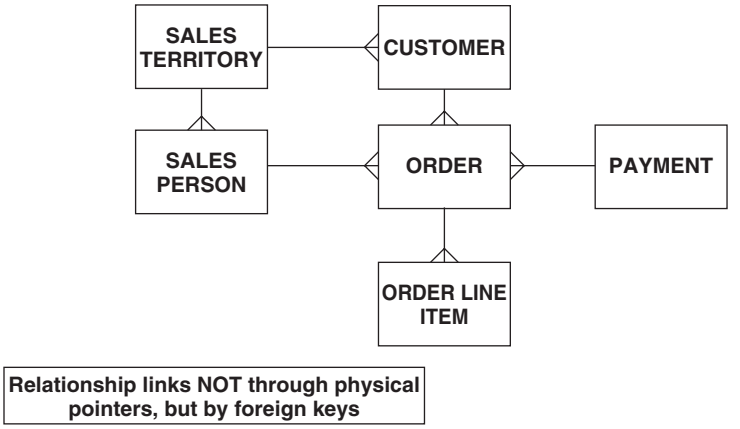

Figure 1-10 Relational data model.

Relationships. Consider the relationship between CUSTOMER and ORDER. For each customer one or more orders may exist. So this customer occurrence must be connected to all the related order occurrences. In the relational model, physical pointers do not establish these connections. Instead, a foreign key field is included in the ORDER data structure. In each of the order occurrences relating to a certain customer, the foreign key contains the identification of that customer. When you look for all the orders for a particular customer, you search through the foreign key field of ORDER and find those order occurrences with identification of that customer in the foreign key field. We will deal with this topic in more detail in later chapters.

No Physical Pointers. Unlike the hierarchical or the network data models, the relational model establishes relationships between data structures by means of foreign keys and not by physical pointers.

Object-Relational

Take the case of the State of California Department of Water Resources (DWR), which manages the waterways, canals, and water projects in that state. DWR maintains a library of more than half a million pictures. Users access this library several times a day. A user requests a picture by content: "Show me Lake Cachuma" (a Santa Barbara County reservoir with a low water level). Despite an index system of captions and keywords, retrieval of the right picture within a reasonable time is virtually impossible with the current relational database systems. Nor are purely object-oriented data systems totally adequate to handle the challenge. As the demand for information continues to grow, organizations need database systems that allow representation of complex data types, user-defined sophisticated functions, and user-defined operators for data access.

Object-relational database management systems (ORDBMS) present viable solutions for handling complex data types. The object-relational model combines the ability of object technology to handle advanced types of relationships with fea-

tures of data integrity, reliability, and recovery found in the relational realm. We will cover object-relational database systems in greater detail in Chapter 20.

TYPES OF DATABASES

By now you are convinced of the significance of information for an organization. You know that information is a key corporate asset and that it has to be managed, protected, and used like any other major asset. The corporate database that holds an organization's data is the underlying foundation for corporate information. Organizations are also faced with questions regarding how and where to hold the corporate data.

Where should an enterprise hold its data? Should all the corporate data be kept centrally in one place? If so, what are the advantages and disadvantages? Or should the corporate data be divided into suitable fragments and the pieces kept at different locations? What are the implications of this arrangement?

Organizations primarily adopt one of two approaches. If the entire database is kept in one centralized location, this type of database is a centralized database. On the other hand, if fragments of the database are physically placed at various locations, this type of database is a distributed database. Each type has its own benefits and shortcomings. Again, whether an enterprise adopts a centralized or a distributed approach depends on the organizational setup and the information requirements. Let us review the two types.

Centralized

Figure 1-11 illustrates a centralized database.

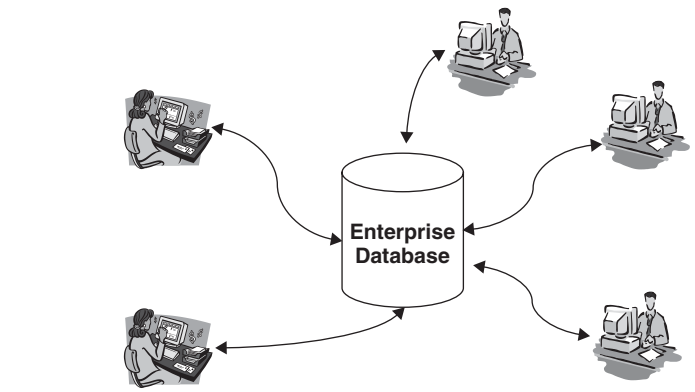

Figure 1-11 Centralized database.

Personalized databases are always centralized in one location. If your company has a centralized computer system, then the database must reside in that central location. In the client/server architecture, the database resides on a server machine. The entire database may be kept on a single server machine and placed in a central location.

When all corporate data is in one place in a centralized database, companies find it easier to manage and administer the database. You can control concurrent accesses to the same data in the database easily in a centralized database. You can maintain security controls easily. However, if your company's operations are spread across remote locations, these locations must access the centralized database through communication links. Here, data availability depends on the capacity and dependability of the communication links.

Distributed

Figure 1-12 shows how fragments of a corporate database are spread across remote locations.

Global organizations or enterprises with widespread domestic operations can benefit from distributed databases. In such organizations computer processing is also distributed, with processing done locally at each location. A distributed database gets fragmented into smaller data sets. Normally, you would divide the

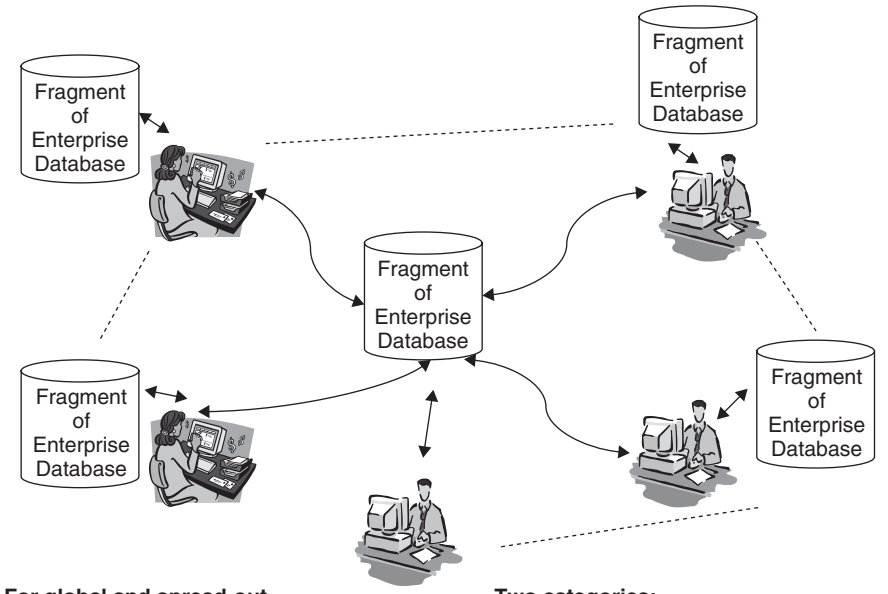

Figure 1-12 Distributed database.

database into data sets on the basis of usage. If a fragment contains data that are most relevant to one location, then that data set is kept at that location. At each location, a fragment of the enterprise data is placed based on the usage.

Each fragment of data at every location may be managed with the same type of database management system. For example, you may run Oracle DBMS at every location. In that case, you run your distributed database as a homogenous database. On the other hand, if you elect to manage the data fragments at different locations with different DBMSs, then you run your distributed database as a collection of heterogeneous database systems. Heterogeneous arrangement provides extra flexibility. However, heterogeneous distribution is difficult to coordinate and administer.

SURVEY OF THE DATABASE INDUSTRY

So far we have reviewed the basic concepts of the database approach. You have understood the forces that drove the transition from file-oriented systems to database systems. We discussed in detail how the escalating demand by organizations for information and the explosive growth of computing technology mutually contributed to the rise of database systems. We also looked into the types of databases.

We should now conclude with a brief survey of the database industry. Where is the industry now? How did it get here? You will find such a survey useful because it will provide you with the context and the foundation for further study. You have been introduced to the leading data models. As you walk through a brief history, you will grasp the significance of one data model following the earlier one. A list of the leading commercial database management systems will give you an appreciation of the spread of database systems. You will also get a quick glimpse of the future direction of database technology. We will be returning to these topics in later chapters. At this stage, let us just do a brief survey.

Brief History

Although the initial movement toward database systems began in the 1960s, software sophistication and widespread use of database systems began in the mid-1970s. More and more organizations began to adopt database technology to manage their corporate data. Figure 1-13 provides you with a historical summary of the database industry. The figure highlights the major events and developments in the various decades.

Generalized Update Access Method (GUAM) contained the first trace of the forerunner to the hierarchical database systems. In early 1960s, Rockwell developed this software to manage the data usually associated with manufacturing operations. IBM picked this up and introduced Information Management System (IMS) as a hierarchical database management system.

Integrated Data Store (IDS), developed at General Electric, formed the basis for database systems on the network data model. Database Task Group (DBTG) of the Conference on Data Systems and Languages (CODASYL) began to produce standards for the network model. CODASYL, a consortium of vendors and

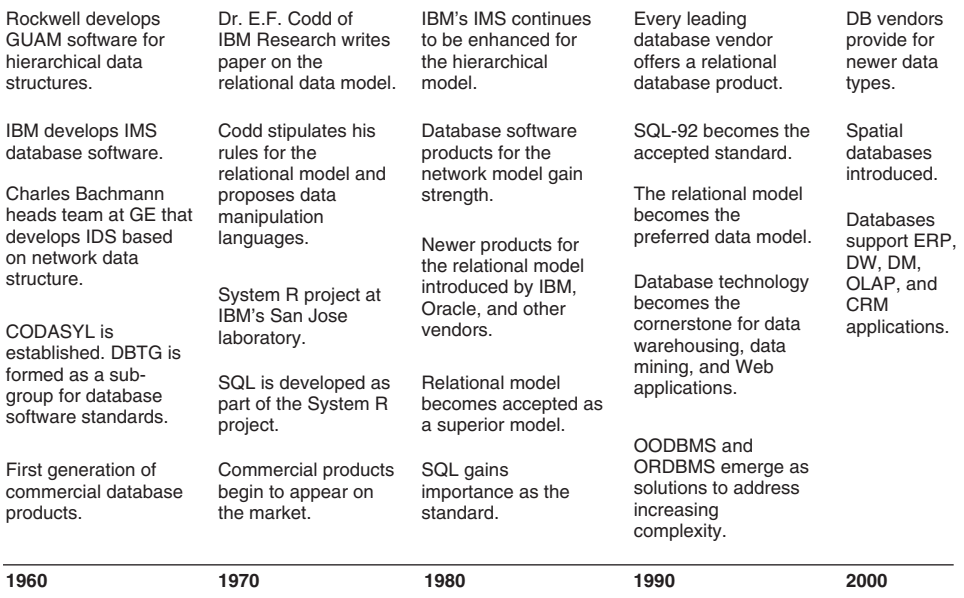

Figure 1-13 Database industry: historical summary.

leading businesses, serves as a group to establish standards. Among the many standards established, a major contribution by CODASYL is the set of standards for COBOL, a leading programming language. In the late 1960s, vendors started to release the first generation of commercial database systems supporting the network data model. Cincom's TOTAL database management system is a primary example.

The 1970s ushered in the era of relational database technology. Dr. Codd's foundational paper on the relational model revolutionized the thinking on data systems. The industry quickly realized the superiority of the relational model, and more and more vendors began to adapt their products to that model.

During the 1980s, the use of database systems gained a lot of ground and a large percentage of businesses made the transition from file-oriented data systems to database systems. All the three leading data models—hierarchical, network, and relational—were quite popular, although the relational model was steadily gaining ground.

Essentially the 1990s may be considered as a period of maturity of the relational model and the emergence of that data model as the leading one. Companies considered moving their data to relational databases from hierarchical and network databases. Also, vendors started to incorporate the features of both relational and object technologies in their products. Object-relational database management systems (ORDBMSs) hit the market.

Now in the new millennium, the usage of database technology is spreading into newer areas. Properly designed databases serve as a chief component in data warehousing (DW), enterprise resource planning (ERP), data mining (DM), on-line analytical processing (OLAP), and customer relationship management (CRM) applications.

Leading Commercial Databases

In Appendix E, you will find broad reviews of major commercial database systems. At this point, let us list some of the leading systems in alphabetic sequence of the product names:

ACCESS
DATACOM/DB
DB2
FOXPRO
IDMS
INFORMIX
INGRES
ORACLE
PARADOX
SQL SERVER
SYBASE
SYSTEM 2000

Peek into the Future

Chapter 20 covers the trends in database technology in extensive detail. The following list indicates some of the specific developments.

Very large databases. Modern commercial database management systems can support very large volumes of data in the terabyte range. Data access and manipulation do not pose problems. As databases store data for corporate data warehouses and for on-line analytical processing applications, the sizes of databases will continue grow rapidly.

User-defined data types. Newer applications for global organizations need special data types. One example is the spatial data type where data values consist of areas and volumes. In addition to standard data types, DBMSs have begun to enable database users to define their own data types with special characteristics.

Complex data manipulation. The current set of data manipulation functions such as sum, difference, average, and the like are no longer adequate when your users want to utilize the database for complex calculations. While performing analysis, they might need to calculate moving averages. Modern database software products provide the ability to perform complex data manipulations.

Intelligent database servers. The trend already includes automatic execution of programs based on exception conditions detected from the data in the database. Future database management systems will contain a substantial extent of logic for intelligent triggers to initiate automatic operations.

Wider use for decision support. Improved database technology spurs the spread of decision- support systems such as data warehousing, data mining, and on-line analytical processing.

Merging with Web technology. The database has become the centerpiece of Internet and intranet applications. The merging of database technology with Web technology will become even stronger and serve newer purposes.

CHAPTER SUMMARY

- Two remarkable phenomena together have caused the evolution of database systems: (1) escalating demand for information at every organization, and (2) explosive growth of computing technology.
- For nearly three decades, there has been tremendous growth in every sector of computing technology, especially in data storage devices and in the software to manage enterprise data.
- Organizations began to need not only more information but also different types of information for newer purposes than merely running day-to-day operations.
- Major forces driving the evolution of database systems: information considered as key corporate asset, explosive growth of computer technology, escalating demand for information, and inadequacy of earlier data systems.
- Benefits of database systems compared to file-oriented systems: minimal data redundancy, integration of corporate data, preservation of data integrity and consistency, data sharing, ease with which uniform standards can be established, better enforcement of security controls, reduced program maintenance through data independence, and simpler backup and recovery of data.
- Common data models: hierarchical, network, and relational.
- In an organization, data may be kept in a centralized location or suitable fragments of data may be spread across several physical locations. Organizations may opt for centralized or distributed databases.

REVIEW QUESTIONS

1. What is your understanding of the term *database*? Describe it in two or three sentences.
2. List three areas of technology growth that had direct positive effects on data systems. Briefly explain the positive impact.
3. What are ISAM and VSAM files? Why are they better than sequential files?
4. Name and briefly describe the major driving forces for database systems.
5. List and explain any three problems with file-oriented systems.
6. What are homonyms and synonyms? Explain why file-oriented systems may have problems with these.
7. List and describe any three benefits of database systems in comparison with file-oriented systems.

8. What is a data dictionary? Describe its purpose.
9. Briefly describe the features of a network data model. How are relationships represented in a network data model?
10. Is the relational data model better than the earlier data models? Give your reasons.

EXERCISES

1. Match the columns:

1. data redundancy	A. foreign keys for relationships
2. inconsistent data	B. data storage
3. data independence	C. large data volumes
4. data integration	D. used for storing data
5. data repository	E. structure definitions
6. data dictionary	F. easy to administer
7. decision-support database	G. wasted storage space
8. relational data model	H. unified view of corporate data
9. centralized database	I. data separated from program logic
10. punched card	J. errors on customer documents

2. "Escalating demand for information and explosive growth of computing technology together enabled the evolution of database systems." Discuss this statement, giving your reasons for agreeing or disagreeing.
3. You are hired as the project leader for a new database project at a local bank. The bank wants to convert the existing file-oriented systems to a database system. Your department head wants you to write a note to the Executive Vice President listing the expected benefits of the proposed database system in comparison with file-oriented systems. Write a one-page note.
4. Compare the features of the hierarchical, network, and relational data models. Explain how related data elements are linked in each model.
5. A manufacturer of electronic components has two factories in two different states. The company has a central warehouse at a location where the head office is situated. There are five sales offices in five different regions of the country. Discuss whether the company should have a centralized database or distributed databases. List the advantages and disadvantages of each approach.