CHAPTER 1

The Database Environment

Abstract

This chapter introduces the concept of database systems. Topics include the types of information systems that use databases, the ownership of data within the organization, database software, network architectures for database systems, the importance of security and privacy issues, and integration with legacy databases. The chapter also includes a list of open source database management systems (DBMSs).

Keywords

databases
database management systems
centralized databases
distributed databases
client/server databases
cloud storage

Can you think of a business that doesn't have a database that is stored on a computer? It's hard. I do know of one, however. It is a small used paperback bookstore. A customer brings in used paperbacks and receives credit for them based on the condition, and in some cases, the subject matter, of the books. That credit can be applied to purchasing books from the store at approximately twice what the store pays to acquire the books. The books are shelved by general type (for example, mystery, romance, and non-fiction), but otherwise not organized. The store doesn't have a precise inventory of what is on its shelves.

To keep track of customer credits, the store has a 4 × 6 card for each customer on which employees write a date and an amount of credit. The credit amount is incremented or decremented based on a customer's transactions. The cards themselves are stored in two long steel drawers that sit on a counter. (The cabinet from which the drawers were taken is nowhere in evidence.) Sales slips are written by hand and cash is kept in a drawer. (Credit card transactions are processed by a stand-alone terminal that uses a phone line to dial up the processing bank for card approval.) The business is small, and its system seems to work, but it certainly is an exception.

Although the bookstore just described doesn't have a computer or a database, it does have data. In fact, like a majority of businesses today, it relies on data as the foundation of what it does. The bookstore's operations require the customer credit data; it couldn't function without it.

Data form the basis of just about everything an organization that deals with money does. (It is possible to operate a business using bartering and not keep any data, but that certainly is a rarity.) Even a Girl Scout troop selling cookies must store and manipulate data. The troop needs to keep track of how many boxes of each type of cookie have been ordered, and by whom. They also need to manage data about money: payments received, payments owed, amount kept by the troop, amount sent to the national organization. The data may be kept on paper, but they still exist and manipulation of those data is central to the group's functioning.

In fact, just about the only "business" that doesn't deal with data is a lemonade stand that gets its supplies from Mom's kitchen and never has to pay Mom back. The kids take the entire gross income of the lemonade stand without worrying about how much is profit.

Data have always been part of businesses.[1] Until the mid-twentieth century, those data were processed manually. Because they were stored on paper, retrieving data was difficult, especially if the volume of data was large. In addition, paper documents tended to deteriorate with age, go up in smoke, or become water-logged. Computers changed that picture significantly, making it possible to store data in much less space than before, to retrieve data more easily, and usually to store it more permanently.

The downside to the change to automated data storage and retrieval was the need for at some specialized knowledge on the part of those who set up the computer systems. In addition, it costs more to purchase the equipment needed for electronic data manipulation than it does to purchase some file folders and file cabinets. Nonetheless, the ease of data access and manipulation that computing has brought to businesses has outweighed most other considerations.

Defining a Database

Nearly 35 years ago, when I first started working with databases, I would begin a college course I was teaching in database management with the following sentence: "There is no term more misunderstood and misused in all of business computing than 'database.' " Unfortunately, that is still true to some extent, and we can still lay much of the blame on commercial software developers. In this section, we will explore why that is so, and provide a complete definition for a database.

Lists and Files

A portion of the data used in a business is represented by lists of things. For example, most of us have a contact list that contains names, addresses, and phone numbers. Business people also commonly work with planners that list appointments. In our daily lives, we have shopping lists of all kinds as well as "to do" lists. For many years, we handled these lists manually, using paper, day planners, and a pen. It made sense to many people to migrate these lists from paper to their PCs.

to migrate these lists from paper to their PCs.

Software that helps us maintain simple lists stores those lists in files, generally one list per physical file. The software that manages the list typically lets you create a form for data entry, provides a method of querying the data based on logical criteria, and lets you design output formats. List management software can be found not only on desktop and laptop computers, but also on our handheld computing devices.

Unfortunately, list management software has been marketed under the name "database" since the advent of PCs. People have therefore come to think of anything that stores and manipulates data as database software. Nonetheless, a list handled by list-management software is not a database.

Databases

There is a fundamental concept behind all databases: There are things in a business environment about which we need to store data, and those things are related to one another in a variety of ways. In fact, to be considered a *database*, the place where data are stored must contain not only the data but also information about the relationships between those data. We might, for example, need to relate our customers to the orders they place with us and our inventory items to orders for those items.

The idea behind a database is that the user—either a person working interactively or an application program—has no need to worry about the way in which data are physically stored on disk. The user phrases data manipulation requests in terms of data relationships. A piece of software known as a *database management system* (DBMS) then translates between the user's request for data and the physical data storage.

Why, then, don't the simple "database" software packages (the list managers) produce true databases? Because they can't represent relationships between data, much less use such relationships to retrieve data. The problem is that list management software has been marketed for years as "database" software and many purchasers do not understand exactly what they are purchasing. Making the problem worse is that a rectangular area of a spreadsheet is also called a "database." Although you can use spreadsheet functions to reference data stored outside a given rectangular area, this is not the same as relationships in a real database. In a database, the relationships are between those things mentioned earlier (the customers, orders, inventory items, and so on) rather than between individual pieces of data. Because this problem of terminology remains, confusion about exactly what a database happens to be remains as well.

Note: A generic term that is commonly used to mean any place where data are stored, regardless of how those data are organized, is "data store."

Systems that Use Databases

Databases do not exist in a vacuum in any organization. Although they form the backbone for most organizational data processing, they are surrounded by information systems that include application software and users.

There are two major types of systems that use databases in medium to large organizations:

- Transaction processing: Transaction processing systems (*online transaction processing*, or OLTP) handle the day-to-day operations of an organization. Sales, accounting, manufacturing, human resources—all use OLTP systems. OLTP systems form the basis of information processing in most organizations of any size. (In fact, typically OLTP is the only type of information system used by a small business.) The data are dynamic, changing frequently as the organization sells, manufactures, and administers.
- Analytical processing: Analytic processing systems (*online analytical processing*, or OLAP) are used in support of the analysis of organizational performance, making high-level operational decisions, and strategic planning. Most data are extracted from operational systems, reformatted as necessary, and loaded into the OLAP system. However, once part of that system, the data values are not modified frequently.

Relational databases, which form the bulk of databases in use today (as well as the bulk of this book), were developed for transaction processing. They handle the data that organizations need to stay in business. The data needed by the organization are well known and are usually structured in a predictable and stable manner. In other words, we know generally what we need and how the data interact with one another. Our needs may change over time, but the changes are relatively gradual and, in most cases, changes to the data processing system do not have to be made in a hurry.

Some OLAP systems also use relational databases, including some large data warehouses (see Chapter 24). However, in recent years, volumes of unstructured data (data without a predictable and stable structure) have become important for corporate decision making. A new category of databases has arisen to handle these data. You will read about them in Chapter 28.

Data "Ownership"

Who "owns" the data in your organization? Departments? IT? How many databases are there? Are there departmental databases or is there a centralized, integrated database that serves the entire organization? The answers to these questions can determine the effectiveness of a company's database management.

The idea of data ownership has some important implications. To see them, we must consider the human side of owning data. People consider exclusive access to information a privilege and are often proud of their access: "I know something you don't know." In organizations where isolated databases have cropped up over the years, the data in a given database are often held in individual departments that are reluctant to share that data with other organizational units.

One problem with these isolated databases is that they may contain duplicated data that are inconsistent. A customer might be identified as "John J. Smith" in the marketing database but as "John Jacob Smith" in the sales database. It also can be technologically difficult to obtain data stored in multiple databases. For example, one database may store a customer number as text while another stores it as an integer. An application therefore may be unable to match customer numbers between the two databases. In addition, attempts to integrate the data into a single, shared data store may run into resistance from the data "owners," who are reluctant to give up control of their data.

In yet other organizations, data are held by the IT department, which carefully doles out access to those data as needed. IT requires supervisor signatures on requests for accounts and limits access to as little data as possible, often stating requirements for system security. Data users feel as if they are at the mercy of IT, even though the data are essential to corporate functioning.

The important psychological change that needs to occur in either of the above situations is that data belong to the organization and that they must be shared as needed throughout the organization without unnecessary roadblocks to access. This does not mean that an organization should ignore security concerns, but that where appropriate, data should be shared readily within the organization.

Service-Oriented Architecture (SOA)

One way to organize a company's entire information systems functions is *Service-Oriented Architecture* (SOA). In an SOA environment, all

One way to organize a company's entire information systems functions is *Service-Oriented Architecture* (SOA). In an SOA environment, all information systems components are viewed as services that are provided to the organization. The services are designed so that they interact smoothly, sharing data easily when needed.

An organization must make a commitment to implement SOA. Because services need to be able to integrate smoothly, information systems must be designed from the top down. (In contrast, organizations with many departmental databases and applications have grown from the bottom up.) In many cases, this may mean replacing most of an organization's existing information systems.

SOA certainly changes the role of a database in an organization: The database becomes a service provided to the organization. To serve that role, a database must be designed to integrate with a variety of departmental applications. The only way for this to happen is for the structure of the database to be well documented, usually in some form of *data dictionary*. For example, if a department needs an application program that uses a customer's telephone number, application programmers first consult the data dictionary to find out that a telephone number is stored with the area code separate from the rest of the phone number. Every application that accesses the database must use the same telephone number format. The result is services that can easily exchange data because all services are using the same data formats.

Shared data also place restrictions on how changes to the data dictionary are handled. Changes to a departmental database affect only that department's applications, but changes to a database service may affect many other services that use the data. An organization must therefore have procedures in place for notifying all users of data when changes are proposed, giving the users a chance to respond to the proposed change, and deciding whether the proposed change is warranted. As an example, consider the effect of a change from a five- to nine-digit zip code for a bank. The CFO believes that there will be a significant savings in postage if the change is implemented. (The post office charges discounted rates for pre-stamped bulk mail that is sorted by nine-digit zip codes.) However, the transparent windows in the envelopes used to mail paper account statements are too narrow to show the entire nine-digit zip code. Envelopes with wider windows are very expensive, so expensive that making the change will actually cost more than leaving the zip codes at five digits. The CFO was not aware of the cost of the envelopes; the cost was noticed by someone in the purchasing department.

SOA works best for large organizations. It is expensive to introduce because typically organizations have accumulated a significant number of independent programs and data stores that will need to be replaced. Just determining where all the data are stored, who controls the data, which data are stored, and how those data are formatted can be a daunting task. It is also a psychological change for those employees who are used to owning and controlling data.

Organizations undertake the change to SOA because in the long run it makes information systems easier to modify as corporate needs change.[2] It does not change the process for designing and maintaining a database, but does change how applications programs and users interact with it.

Database Software: DBMSs

There is a wide range of DBMS software available today. Some, such as Microsoft Access[3] (part of the Windows Microsoft Office suite) are designed for single users only.[4] The largest proportion of today's DBMSs, however, are multiuser, intended for concurrent use by many users. A few of those DBMSs are intended for small organizations, such as FileMaker Pro[5] (cross-platform, multiuser) and Helix[6] (Macintosh multiuser). Most, however, are intended for enterprise use. You may have heard of DB2[7] or Oracle,[8] both of which have versions for small businesses but are primarily intended for large installations using mainframes. As an alternative to these commercial products, many businesses have chosen to use open source products, a list of which can be found at the end of this chapter.

For the most part, enterprise-strength commercial DBMSs are large, expensive pieces of software. (This goes a long way when explaining interest in open source software.) They require significant training and expertise on the part of whoever will be implementing the database. It is not unusual for a large organization to employ one or more people to handle the physical implementation of the database along with a team (or teams) of people to develop the logical structure of the database. Yet more teams may be responsible for developing application programs that interact with the database and provide an interface for those who cannot, or should not, interact with the database directly.

Regardless of the database product you choose, there are some capabilities that you should expect to find:

- A DBMS must provide facilities for creating the structure of the database. Developers must be able to define the logical structure of the data to be stored, including the relationships among data.
- A DBMS must provide some way to enter, modify, and delete data. Small DBMSs typically focus on form-based interfaces; enterprise-level products begin with a command-line interface. The most commonly used language for interacting with a relational database (the type we are discussing in this book) is SQL (originally called Structured Query Language), which has been accepted throughout much of the world as a standard data manipulation language for relational databases.
- A DBMS must also provide a way to retrieve data. In particular, users must be able to formulate queries based on the logical relationships among the data. Smaller products support form-based querying while both small and enterprise-level products support SQL. A DBMS should support complex query statements using Boolean algebra (the AND, OR, and NOT operators) and should also be able to perform at least basic calculations (for example, computing totals and subtotals) on data retrieved by a query.

Note: You will find references to SQL throughout this book, even in chapters that don't discuss the language specifically. The emphasis on SQL isn't promotion of a particular company's product. In fact, SQL isn't a product; it's a set of standards, the most recent of which is SQL:2011. DBMS developers must add code to their software that implements what is described in the standards. There are a myriad of SQL implementations available that vary somewhat in which portions of the standard are supported. Nonetheless, if you are familiar with basic SQL, you will know at least 90% of what it takes to manipulate data in any DBMS that uses SQL.

- Although it is possible to interact with a DBMS either with basic forms (for a smaller product) or at the SQL command line (for enterprise-level products), doing so requires some measure of specialized training. A business usually has employees who need to manipulate data, but either don't have the necessary expertise, can't or don't want to gain the necessary expertise, or shouldn't have direct access to the database for security reasons. Application developers therefore create programs that simplify access to the database for such users. Most DBMSs designed for business use provide some way to develop such applications. The larger the DBMS, the more likely it is that application development requires traditional programming skills. Smaller products support graphic tools for "drawing" forms and report layouts.
- A DBMS should provide methods for restricting access to data. Such methods often include creating user names and passwords specific to the database, and tying access to data items to the user name. Security provided by the DBMS is in addition to security in place to protect an organization's network.

What DBMSs are Companies Really Using

There are a lot of DBMSs on the market, so what products are really being used? Recent surveys[9] have uncovered relatively consistent results, which are summarized in Table 1.1. Notice that the percentages add up to far more than 100% because many companies run multiple DBMSs

which are summarized in Table 1.1. Notice that the percentages add up to far more than 100% because many companies run multiple DBMSs.

Table 1.1

DBMS Use in Medium to Large Businesses

Relational DBMSs	Percent Companies Using	Non-Relational DBMSs	Percent Companies Using
Microsoft SQL Server	60–90%	MongoDB	10–15%
Oracle	40–80%	Hadoop	8%
MySQL	80%	Cassandra	4%
IBM DB2	15–30%	Riak	2%
PostgreSQL	15%	Couchbase	1%

Sources: King, Dr Elliott, Research Analyst, Unisphere Research. The Real World of the Database Administrator; Tesora. Database Usage in the Public and Private Cloud: Choices and Preferences.

Database Hardware Architecture

Because databases are almost always designed for concurrent access by multiple users, database access has always involved some type of computer network. The hardware architecture of these networks has matured along with more general computing networks.

Centralized

Originally, network architecture was centralized, with all processing done on a mainframe. Remote users—who were almost always located within the same building, or at least the same office park—worked with dumb terminals that could accept input and display output but had no processing power of their own. The terminals were hard-wired to the mainframe (usually through some type of specialized controller) using coaxial cable, as in Figure 1.1.

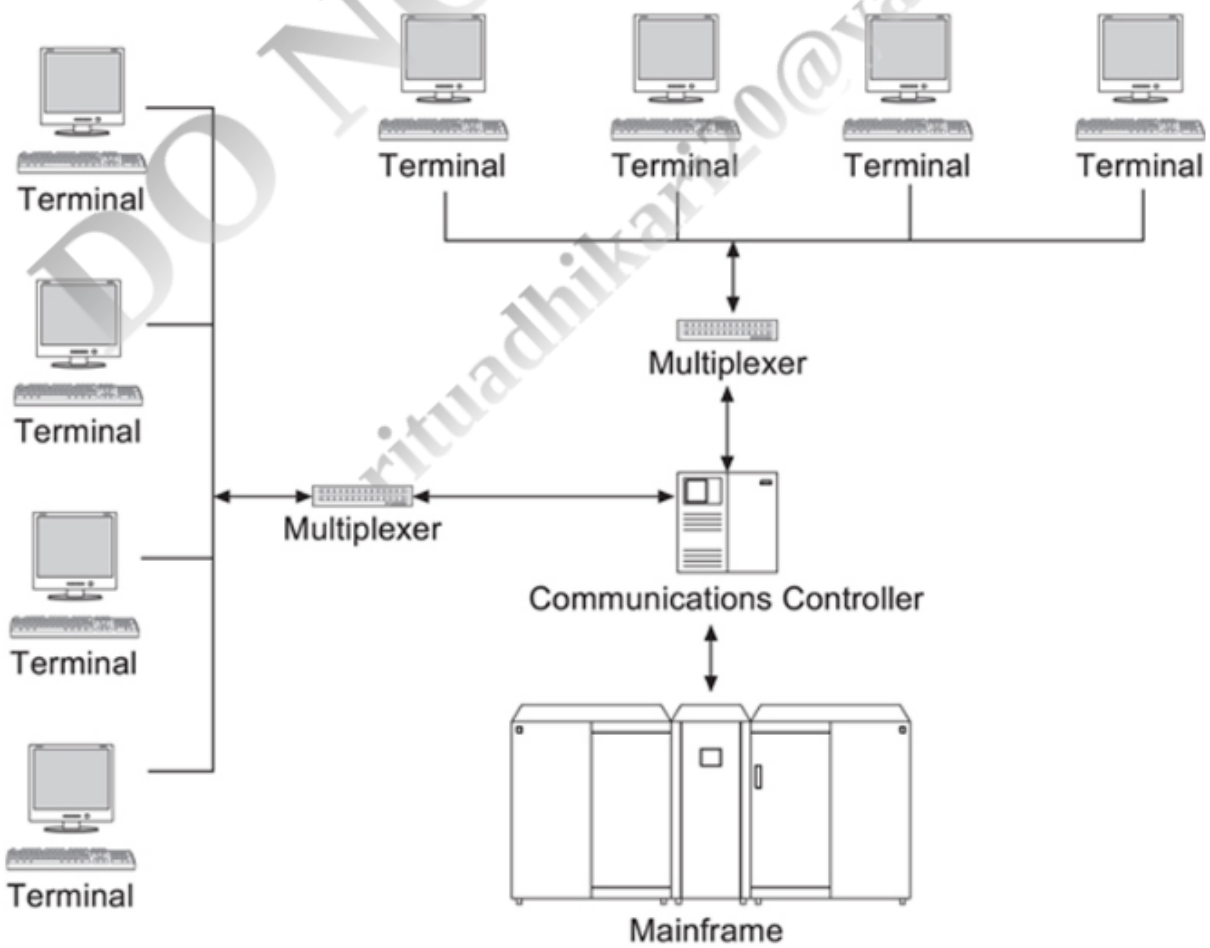


FIGURE 1.1 **Classic centralized database architecture.**

During the time that the classic centralized architecture was in wide use, network security also was not a major issue. The Internet was not publically available, there was no World Wide Web, and security threats were predominantly internal.

Centralized database architecture, in the sense we have been describing, is rarely found today. Instead, those organizations that maintain a centralized database typically have both local and remote users connecting using PCs, local area networks (LANs), and a wide area network (WAN) of some kind. As you look at Figure 1.2, keep in mind that although the terminals have been replaced with PCs, the PCs are not using their own processing power when interacting with the database. All processing is still done on the mainframe.

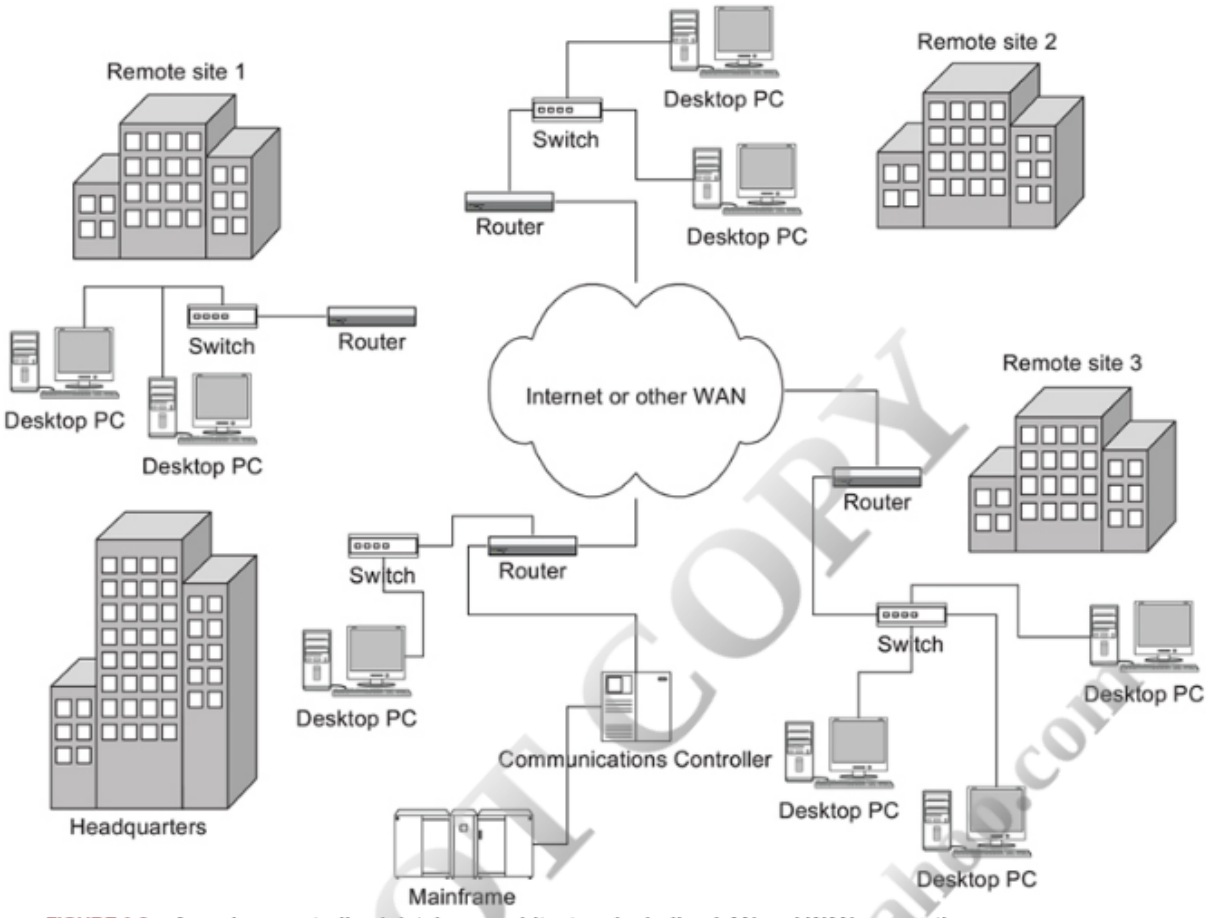


FIGURE 1.2 **A modern centralized database architecture including LAN and WAN connections.**

From the point of view of an IT department, there is one major advantage to the centralized architecture: control. All the computing is done on one computer to which only IT has direct access. Software management is easier because all software resides and executes on one machine. Security efforts can be concentrated on a single point of vulnerability. In addition, mainframes have the significant processing power to handle data-intensive operations as well as the capacity to handle large volumes of I/O.

One drawback to a centralized database architecture is network performance. Because the terminals (or PCs acting as terminals) do no processing power on their own, all processing must be done on the mainframe. The database needs to send formatted output to the terminals, which consumes more network bandwidth than would sending just the data.

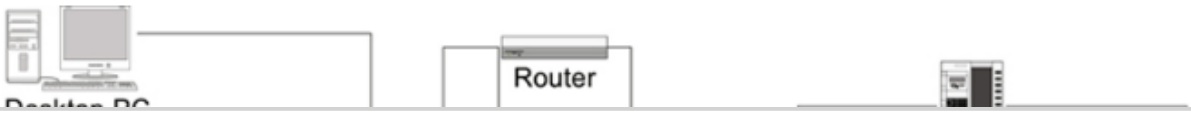
A second drawback to centralized architecture is reliability. If the database goes down, the entire organization is prevented from doing any data processing.

Mainframes are not gone, but their role has changed as client/server architecture has become popular.

Client/Server

Client/server architecture shares the data processing chores between a server—typically, a high-end workstation but quite possibly a mainframe—and clients, which are usually PCs. PCs have significant processing power and therefore are capable of taking raw data returned by the server and formatting the result for output. Application programs and query processors can be stored and executed on the PCs. Network traffic is reduced to data manipulation requests sent from the PC to the database server and raw data returned as a result of that request. The result is significantly less network traffic and theoretically better performance.

Today's client/server architectures exchange messages over LANs. Although a few older Token Ring LANs are still in use, most of today's LANs are based on Ethernet standards. As an example, take a look at the small network in Figure 1.3. The database runs on its own server (the database server), using additional disk space on the network attached storage device. Access to the database is controlled not only by the DBMS itself, but by the authentication server.



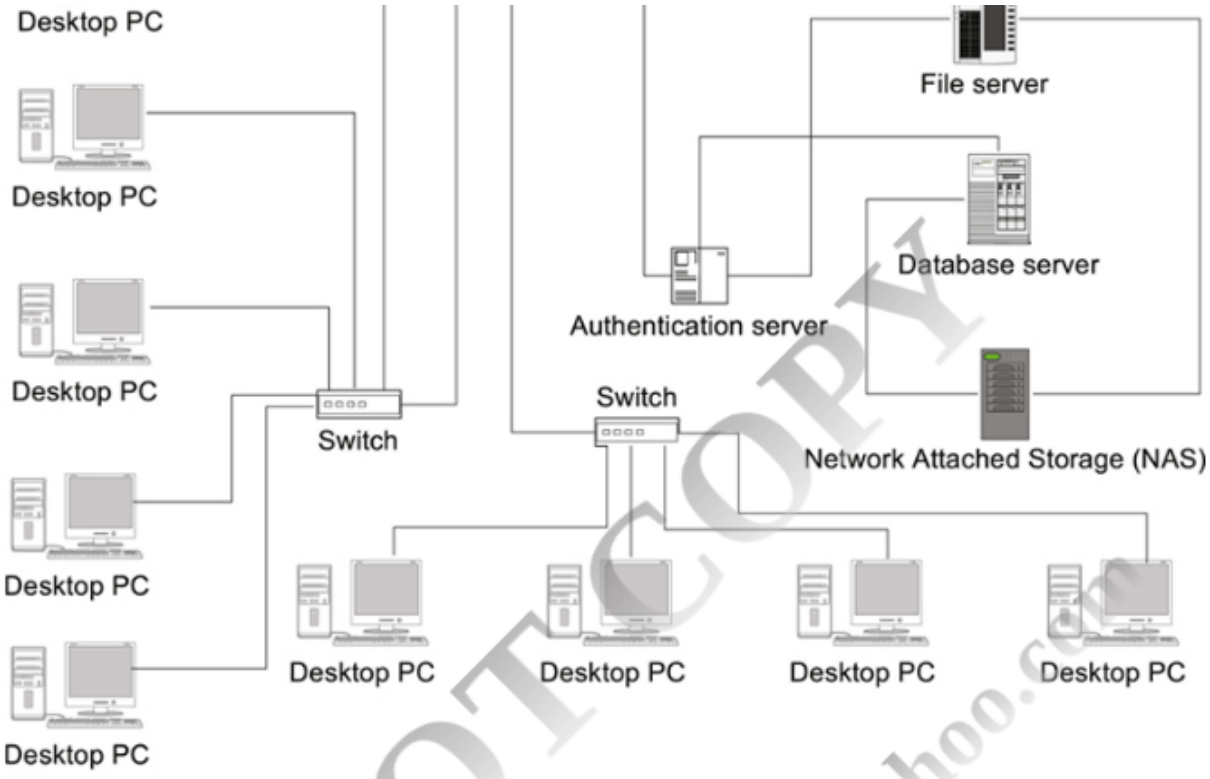


FIGURE 1.3 **Small LAN with network-accessible database server.**

A client/server architecture is similar to the traditional centralized architecture in that the DBMS resides on a single computer. In fact, many of today's mainframes actually function as large, fast servers. The need to handle large data sets still exists although the location of some of the processing has changed.

Because a client/server architecture uses a centralized database server, it suffers from the same reliability problems as the traditional centralized architecture: if the server goes down, data access is cut off. However, because the "terminals" are PCs, any data downloaded to a PC can be processed without access to the server.

Distributed

Not long after centralized databases became common—and before the introduction of client/server architecture—large organizations began experimenting with placing portions of their databases at different locations, each site running a DBMS against part of the entire data set. This architecture is known as a *distributed database*. (For example, see Figure 1.4.) It is different from the WAN-using centralized database in Figure 1.2 in that there is a DBMS and part of the database at each site as opposed to having one computer doing all of the processing and data storage.

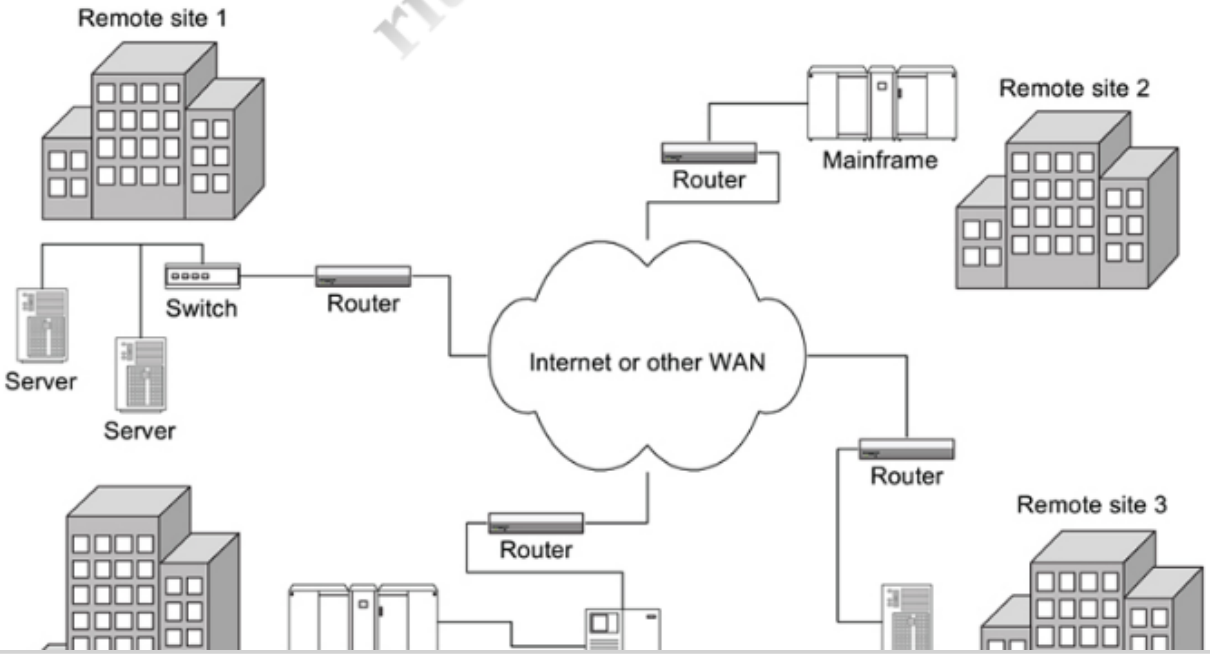


FIGURE 1.4 **Distributed database architecture.**

A distributed database architecture has several advantages:

- The hardware and software placed at each site can be tailored to the needs of the site. If a mainframe is warranted, then the organization uses a mainframe. If smaller servers will provide enough capacity, then the organization can save money by not needing to install excess hardware. Software, too, can be adapted to the needs of the specific site. Most current distributed DBMS software will accept data manipulation requests from mores than one DBMS that uses SQL. Therefore, the DBMSs at each site can be different.
- Each site keeps that portion of the database that contains the data that it uses most frequently. As a result, network traffic is reduced because most queries stay on a site's LAN rather than needing to use the organization's WAN.
- Performance for local queries is better because there is no time lag for travel over the WAN.
- Distributed databases are more reliable than centralized systems. If the WAN goes down, each site can continue processing using its own portion of the database. Only those data manipulation operations that require data not on site will be delayed. If one site goes down, the other sites can continue to process using their local data.

Despite the advantages, there are reasons why distributed databases are not widely implemented:

- Although performance of queries involving locally stored data is enhanced, queries that require data from another site are relatively slow.
- Maintenance of the data dictionary (the catalog of the structure of the database) becomes an issue: Should there be a single data dictionary or a copy of it at each site? If the organization keeps a single data dictionary, then any changes made to it will be available to the entire database. However, each time a remote site needs to access the data dictionary, it must send a query over the WAN, increasing network traffic, and slowing down performance. If a copy of the data dictionary is stored at each site, then changes to the data dictionary must be sent to each site. There is a significant chance that, at times, the copies of the data dictionary will be out of sync.
- Some of the data in the database will exist at more than one site, usually because more than one site includes the same data in the "used most often" category. This introduces a number of problems in terms of ensuring that the duplicated copies remain consistent, some of which may be serious enough to prevent an organization from using a distributed architecture. (You will read more about this problem in Chapter 25.)
- Because data are traveling over network media not owned by the company (the WAN), security risks are increased.

Web

The need for Web sites to interact with database data has introduced yet another alterative database architecture. A Web server needing data must query the database, accept the results, and format the result with HTML tags for transmission to the end user and display by the user's Web browser. Complicating the picture is the need to keep the database secure from Internet intruders.

Figure 1.5 provides an example of how a Web server affects the hardware on a network when the Web server must communicate with a database server. For most installations, an overriding concern is security. The Web server is isolated from the internal LAN and a special firewall is placed between the Web server and the database server. The only traffic allowed through that firewall is traffic to the database server from the Web server and from the database server to the Web server.

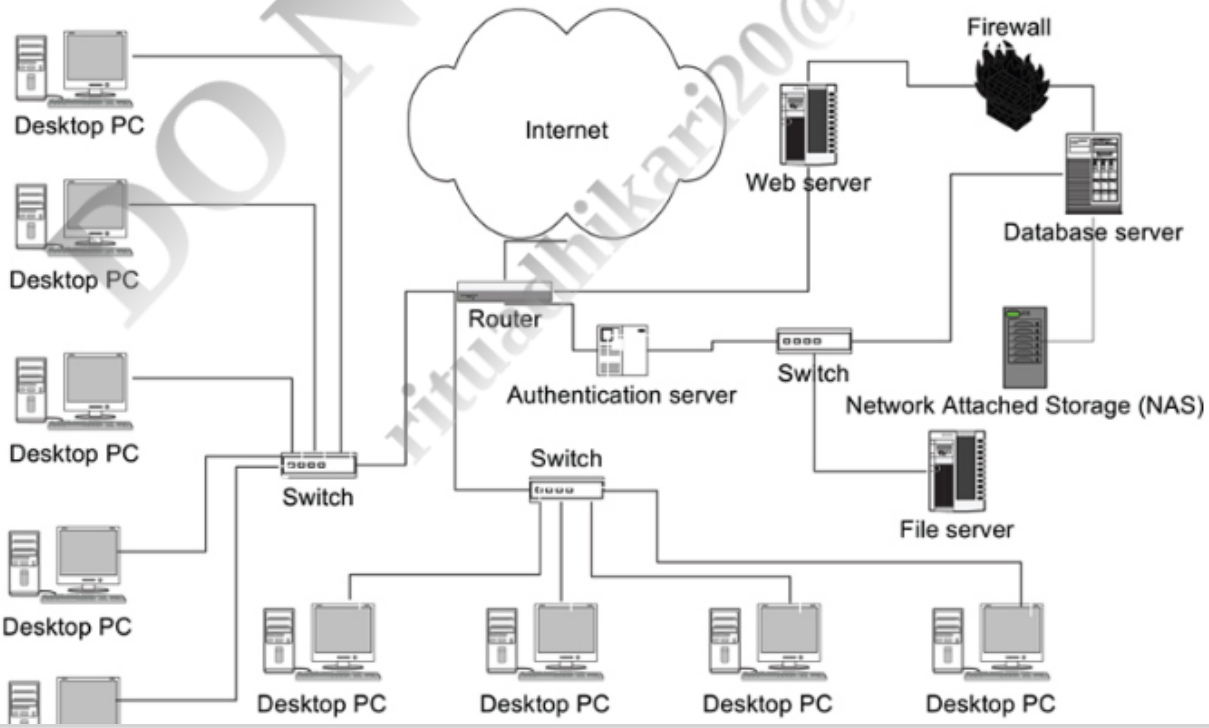


FIGURE 1.5 **The placement of a database server in a network when a Web server interacting with the database is present.**

Some organizations prefer to isolate an internal database server from a database server that interacts with a Web server. This usually means that there will be two database servers: The database server that interacts with the Web server is a copy of the internal database that is inaccessible from the internal LAN. Although more secure than the architecture in Figure 1.5, keeping two copies of the database means that those copies must be reconciled at regular intervals. The database server for Web use will become out-of-date as soon as changes are made to the internal database, and there is the chance that changes to the internal database will make portions of the Web-accessible database invalid or inaccurate. Retail organizations that need live, integrated inventory for both physical and Web sales cannot use the duplicated architecture. You will see an example of such as organization in Chapter 15.

Remote Access

In addition to the basic architecture we have chosen for our database hardware, we often have to accommodate remote users. Salespeople, agents in the field, telecommuters, executives on vacation—all may have the need to access a database that is usually available only over a LAN. Initially, remote access involved using a phone line and a modem to dial into the office network. Today, however, the Internet (usually with the help of a virtual private network (VPN)) provides cheaper and faster access, along with serious security concerns.

As you can see in Figure 1.6, the VPN creates a secure encrypted tunnel between the business's internal network and the remote user.[10] The remote user must also pass through an authentication server before being granted access to the internal LAN. Once authenticated, the remote user has the same access to the internal LAN—including the database server—as if he or she were present in the organization's offices.

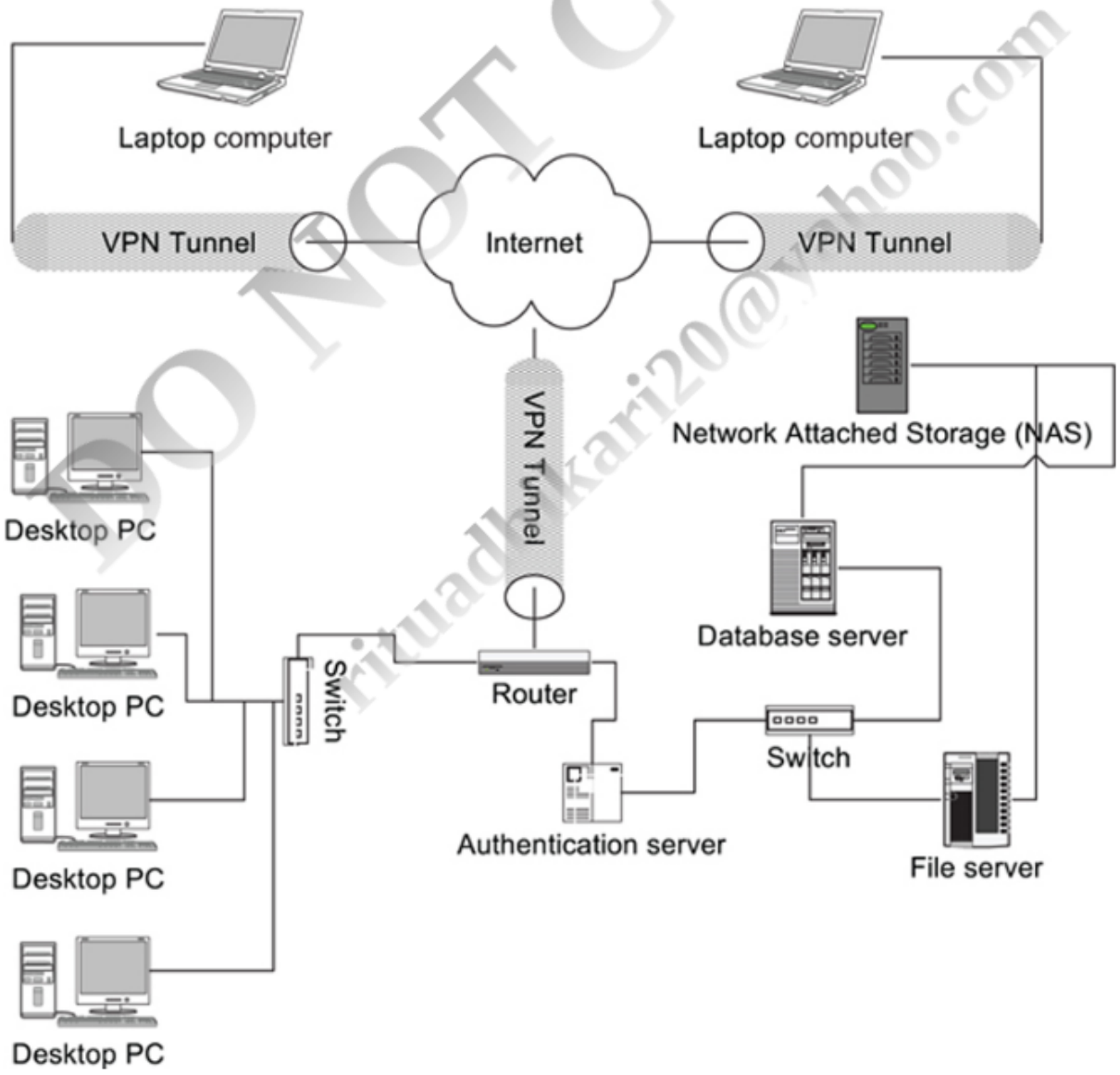


FIGURE 1.6 **Using a VPN to secure remote access to a database.**

Cloud Storage

All of the architectures you have seen to this point assume that the database is stored on hardware at one or more business-owned locations. However, the past few years have seen a migration to *cloud storage*. The term "cloud" has long been used as a generic term for the Internet. (Notice that the Internet is represented in Figures 1.5 and 1.6 by a picture of an amorphous cloud.) When databases are stored in the cloud, they are hosted on hardware not owned by the organization that owns the data. The data reside on servers maintained by another organization that is in the business of storing software and data for other organizations.[11]

You can find a sample architecture for a cloud-stored database in Figure 1.7. The most important characteristic of this architecture is that the DBMS and the database to which it provides access are not located on the company's premises: They are stored on hardware owned and maintained by the cloud service provider. Someone who needs to use the data in the database communicates with the database using the Internet as the communications pathway.

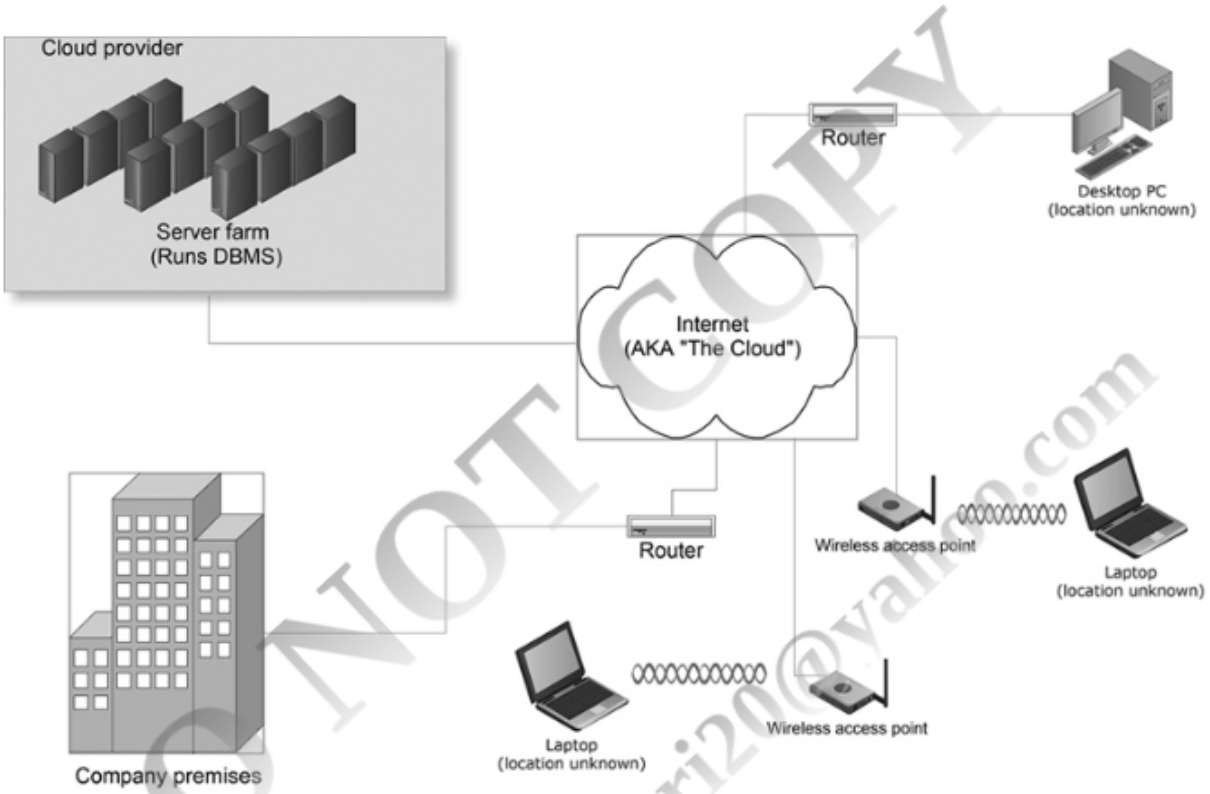


FIGURE 1.7 Storing a database in the cloud.

Advantages of Cloud Storage

There are some tangible benefits of moving a database to the cloud:

- The owner of the data does not need to maintain database hardware or a DBMS. That becomes the responsibility of the cloud service provider. This can significantly reduce the cost of supporting the database, not only because the data owner does not need to purchase hardware and software, but because it does not need to hire staff members or consultants who can maintain the database environment.
- The database can scale seamlessly. If new/larger/faster hardware is needed, the cloud service provider purchases and installs the replacement hardware. The cloud service provider may also be responsible for upgrading the DBMS. (Responsibility for application software that interacts with the DBMS may be the responsibility of the owner of the data or application development may be included in cloud service package.)
- The company using the cloud has fixed, predictable expenses for database maintenance that are negotiated up front.
- The database is accessible from anywhere the Internet is available.

The bottom line is that in most cases, cloud storage can save money and a lot of effort.

Problems with Cloud Storage

As ideal as cloud storage may sound initially, there are some serious issues that a company must consider:

- Because the database is not located on company premises, security becomes an enormous challenge. The company that owns the data no longer has complete control over security measures. It must rely on the cloud service provider to secure the database from unauthorized access; it must also implicitly trust the service provider's employees.
- Access to the database requires a live Internet connection. Unlike architectures where the database is located on the company's internal network, no processing can continue when the Internet is unavailable.
- The company that owns the data must rely on the cloud storage provider for consistent up-time. The responsibility for ensuring that the database is accessible is no longer with the company; it lies with a third party.

Overall, the owner of the data loses a great deal of control over the data when the data are stored in the cloud. The more important the security of the data, the riskier cloud storage becomes.