CHAPTER 2

OVERVIEW OF MAJOR COMPONENTS

CHAPTER OBJECTIVES

- Understand the various aspects of the database environment
- Examine the overall architecture of the database environment
- Identify the different components that make up the database environment
- Note the distinction between DB and DBMS
- Study the functions and features of each component

What is the database approach? You have gained an insight into how the database approach for storing, retrieving, and managing data is superior to the earlier file-oriented approaches. We have discussed the evolution of database systems in detail. Organizations could not tolerate the limitations of file-oriented systems because of escalating demand for information. Explosive growth of computing technology, especially in the area of data systems, acted as a strong catalyst for development of the database approach. Over the past decades, different types of data models—hierarchical, network, and relational—have evolved to offer tremendous benefits to organizations. More and more commercial databases emerged in the marketplace.

Most organizations have adopted the database approach. To manage corporate data and satisfy information needs, companies have created database environments within their organizations. The database environment exists to support the organization's business. Figure 2-1 traces the typical progression to a database environment in an organization. Note the differences between the database environment and the earlier environments.

If a database environment is established in an organization to support the information requirements, what must the environment be composed of? What is a

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

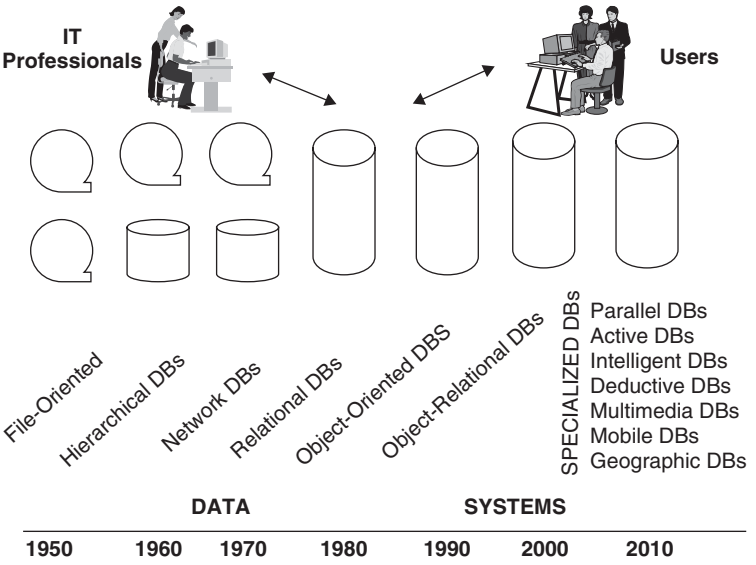

Figure 2-1 Progression to database environment.

database environment? What are the various pieces in the environment? Obviously, corporate data must be a key component. What else? What is the complete setup? What are the components? What are their features? What functions do they serve? Let us examine these questions and understand the composition and significance of a database environment.

WHAT MAKES UP A DATABASE ENVIRONMENT?

Consider an organization running file-oriented applications. The applications depend on the data content in sequential or flat files. These files primarily constitute the data system for the organization, and in such a system no special sophistication is required. No special storage mechanisms are necessary. The operating system itself mostly supports data storage and data access facilities. Basic hardware, data storage, and operating system alone establish the data environment. You do not need anything too special. Such an environment provides information potential on a moderate scale.

Contrast the file-oriented environment with the database environment. The newer database environment provides data access in ways totally unmatched in the file-oriented environment. On-line, real-time retrieval of substantially large result sets is fairly common. Random and indexed access to data replaces the slow and inefficient sequential access. The new environment clearly recognizes the value of information as a key corporate asset and therefore protects the data. For similar reasons, the environment also controls the access privileges and prevents unauthorized data retrieval. The new environment enables users to share information and collaborate smoothly to achieve corporate goals.

So, in this new environment providing database functions, you require robust data storage mechanisms. The operating system alone cannot support data retrieval and presentation. Simple methods of data backup and recovery are not adequate. Complex data structures have to be properly defined. Data delivery systems must be properly tuned to provide instant data access and retrieval. The environment must be open to hundreds or thousands of users. It should be able to handle concurrent transactions seeking to access the same pieces of data and still ensure the integrity of the data. Powerful processors, robust data repository, enhanced operating systems, improved networking capabilities, specialized database software—all of these become necessary components of the database environment.

We have mentioned just the software and hardware components necessary in the database environment. Software and hardware components, however important they may be, are not enough to make the database work. We need the people components. We need procedures for people to carry out the various functions. Who are these people? Users on one hand and database practitioners with special skills on the other hand. The database environment, therefore, also consists of people and procedures. Let us look at the overall architecture of the database environment and then discuss each component in sufficient detail.

Overall Architecture

How do you determine the architecture of a building? You look at the building blocks. You notice the arrangement of the components in the structure. These building blocks, together with their arrangement, make up the overall architecture of the building. Similarly, what is the overall architecture of a database environment? What are the pieces that make up the architecture? How are they put together and arranged to become a cohesive whole?

Essentially, a database environment includes the storage of data. The data repository, important as it is, constitutes just one piece in the overall architecture. You need specialized software to manage and make use of the data. You need hardware to store and process the data. You already know about the people component. You are also aware of the procedures necessary to guide the storage, management, and use of data. Figure 2-2 illustrates the overall architecture of a database environment in an organization. Note the individual components and observe how they are connected.

Before we discuss hardware, software, and people in detail, let us quickly go over the functions of the individual components as seen in Figure 2-2.

Data repository. Stores the organization's data in databases. This is a primary component, essential in every database environment.

Data dictionary. Contains the definition of structures of the data stored in the data repository. As mentioned previously, keeping the structure definitions apart from the actual data provides a level of independence in the database environment.

DBMS. Database Management System is specialized software needed to manage storage, use, and protection of the data stored in the database.

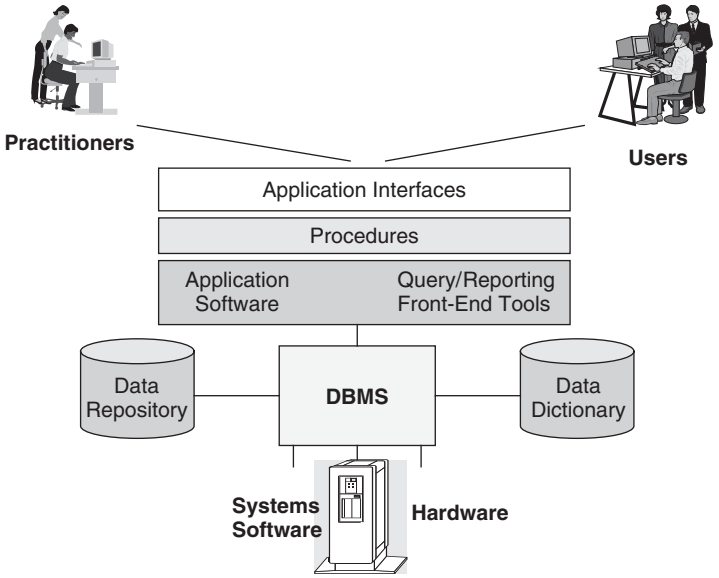

Figure 2-2 Database environment: overall architecture.

Systems software. Includes the operating system, network operating system, and utility programs that are needed to support and work with the DBMS.

Hardware. Comprises the processors, storage devices, and network connections. Hardware is part of the underlying infrastructure.

Application software. Suite of application programs to perform various business processes of the organization by making use of the database. The application programs provide users with interfaces to the database.

Front-end tools. These are query tools, report writers, and other front-end tools that supplement the application programs for data access and usage.

Procedures. Relate to instructions and guidelines for use by database practitioners and end users. These are needed to streamline the operations in the database environment.

Application interfaces. These include hardware and software components at the client workstation to connect to the database through application programs, query tools, or report writers.

Practitioners. Nothing happens in the database environment until database practitioners perform their various functions and responsibilities. Database practitioners include data administrators, database administrators, analysts, and programmers.

Users. The fundamental goal of a database environment is to satisfy the information needs of the end users. The user population extends to the entire organization in need of information for performing their day-to-day tasks.

When you review the components in the overall architecture, you will notice that these fall into three major groups: (1) hardware, (2) software, and (3) people and procedures. Let us now briefly examine the general features of these three groups that make up the database environment. In later sections of this chapter, we will get into the details of the functions and features of individual components.

Hardware Infrastructure

Hardware components form the basic infrastructure over which the functions of the database environment take place. First of all, you need hardware storage to store the data and the structure definitions. Then you need the processors and the computer memory components to retrieve, process, and update the data by using the structure definitions. You require communication links to connect the database to the user machines. Figure 2-3 shows parts of the hardware infrastructure.

Note the choice of hardware platforms. Database environments may exist in a mainframe, mini-computer, or client/server configuration. Modern database systems use storage devices that operate using the RAID (Redundant Array of Inexpensive Disks) technology. You will get a detailed description of RAID in Chapter 12. The client machines and the communication links are also part of the hardware infrastructure.

Supporting Software

If hardware forms the infrastructure for the database environment, software provides the supporting layer for the environment to function. Software controls and enables the storage and use of data. Figure 2-4 presents the software group of components.

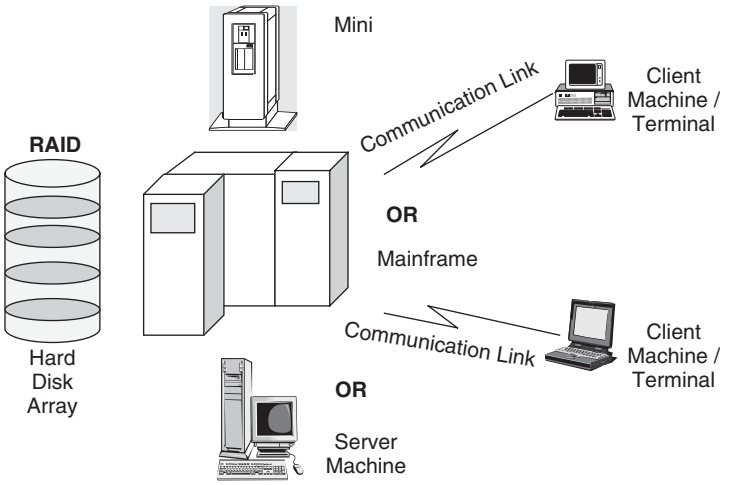

Figure 2-3 Hardware infrastructure.

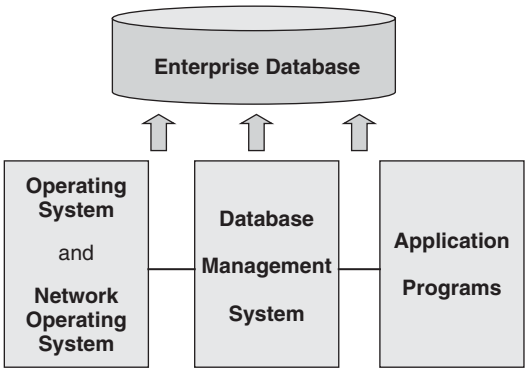

Figure 2-4 Supporting software.

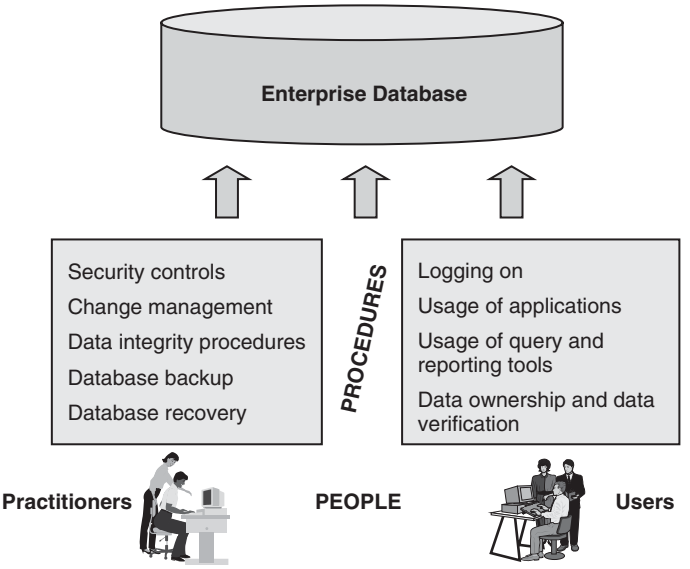

Figure 2-5 People and procedures.

Observe the major pieces of supporting software. The operating system piece includes the basic software such as DOS, Windows, or UNIX plus the network operating system. The database management system is the selected DBMS running the database environment. The suite of application programs forms a significant part of the software component.

People and Procedures

This group of components makes things happen in the database environment with the aid of the underlying hardware infrastructure and the supporting software. Standards are set through manual and automated procedures. Figure 2-5 shows the people and procedures component of the database environment.

Note the two major groups of people associated with the database environment. Each group needs a set of procedures. Observe the types of procedures applicable to each group.

DATABASE AND DBMS

Is Oracle a database? Is Informix a database? Is DB2 a database? Frequently, when we talk about such commercial products, we tend to imply that these products themselves are databases. Someone asks you what database your company uses. You probably respond by mentioning the name of the commercial product used in your company.

It is crucial to have a clear understanding of the term *database*. This book is about database design and development. What do we mean by *database*? When we discuss the term *database*, what is the scope of our discussion? Commercial products such as Oracle, Informix, and DB2 are not databases per se. They are software products that support the storage and management of databases. Let us explore the differences between databases and database software.

DB and DBMS—Not Synonymous

Database Management System (DBMS) is a collection of software components. It is a set of programs that enables us to create, access, and maintain a database. In other words, DBMS is a general-purpose software to define databases, build them, and manipulate the data content. Oracle, Informix, DB2, and similar products are examples of DBMSs. *Database*, on the other hand, refers to the data itself. Let us come up with broad definitions.

Database. Relates to the data and the data structure definitions stored according to the designed data model with the following features: shared, persistent, secure, valid, consistent, nonredundant, and independent of application logic.

DBMS. Software that provides services to access and use the database, incorporating the following features: control of concurrent transactions, recovery from malfunctions, security against unauthorized data access, reliability in data access, efficient storage management, and effective language interface for data structure definition and data access.

Why Use a DBMS?

Consider data management in a file-oriented system. Here, the sequential or indexed files provide the data content. The applications manipulate the data contained in these files. When you want to list information about a customer in your organization, you read the particular customer record from the customer sequential file. Simple file access software and an operating system—just these are all the software needed for the data access. The same set of software also enables you to rewrite the updated record or mark outdated records for deletion. Nothing more is expected from the set of software components for data management.

Contrast this simplicity with the sophistication required in database software. Realization of the significance of information as a key corporate asset drove the evolution to database systems. If information is a significant asset, then it has to be managed well and safeguarded with utmost care. Database software, therefore, has to be much more than a mere data access mechanism; it has to perform many other complex functions. This type of complexity warrants specialized software, and that specialized software is the set of sophisticated programs collectively known as the database management system (DBMS). The DBMS enables us to represent the data structures, relationships, and constraints.

In a later section of this chapter, we will look inside the DBMS and examine the individual components. At this point, you need to convince yourself of the need for the specialized set of software called DBMS by reviewing the major functions.

Language Interfaces You must already have guessed the need for two types of languages in the database environment. For defining data structures, you need a data definition language (DDL). For performing data access operations of retrieving, updating, and deleting data, you need another type of language. Data manipulation language (DML) serves the requirements of data access.

DBMS provides support for DDL and DML. Each particular DBMS works with a specific DDL, and using that data definition language you can define the data structures in the data dictionary. The DBMS interprets the syntax and semantics of the language and records the data structure definitions in the data dictionary. Again, each particular DBMS works with a specific DML to enable access to data. The DBMS translates the DML commands into data access instructions. In later chapters, we will discuss DDL and DML for relational DBMSs in detail.

Storage Management Storage management involves support for sophisticated data retrieval and update requests. Among the thousands of customers in your database, you may want to get a list of all customers in the state of Wisconsin, who have bought for more than $10,000 during the past six months, and who have no past due balances. The storage management component of the DBMS must provide for complex data requests.

Specifically, the storage manager performs the following functions:

- Simplifies and facilitates access to data
- Provides interface with the operating system to access data
- Translates the data manipulation language (DML) commands to low-level file access instructions for storing, retrieving, and updating data
- Works with the structure definitions in the data dictionary to store and find data
- Manages space allocations on permanent secondary storage

Transaction Management What is a transaction in a database environment? Assume you are an order entry operator and you are in the process of entering an order from one of your customers for 100 units of Product A. Usually, the order entry application program works in the following manner. You enter the customer number, date, order location, shipping details, and 100 units for Product A. The

application program reads the quantity in stock for Product A from the database and displays 10,000 units. You indicate that you want to complete the order. The program changes the stock quantity to 9,900 units and updates the database accordingly. As far as the database is concerned, your order entry transaction consists of a read of the record for Product A containing 10,000 units and an update of the record to store the revised number of 9,900 units.

In a database environment, many data entry operators like you will be entering orders simultaneously. Some of these operators may be entering orders for Product A at the same time you are entering your order. The DBMS must be able to handle such concurrent transactions and ensure that each transaction leaves the database in a consistent state. The following components in the DBMS perform transaction management functions.

Transaction manager. Institutes proper controls for transactions and coordinates their execution within a database. Enables each transaction to start, perform database accesses, commit the updates permanently in the database, and complete the execution properly.

Integrity manager. Ensures correctness, accuracy, and validity of the data in the database through edit and data validity constraints.

Concurrency control manager. Enables concurrent transactions to update the same database record correctly by using appropriate locking mechanisms. DBMS has special lock, unlock, and deadlock control modules to assist in concurrency control.

Security Management Protecting the information resource is a major function of DBMS. All the applications supporting the various business processes in the organization depend on the data resources in the corporate database. The database acts as the foundation for running the day-to-day business. The major asset of corporate information must be safeguarded.

Security management involves preventing unauthorized access to the organization's database. Outsiders must be kept away from the database. Several security methods are available for preventing unauthorized access. The second aspect of security management deals with providing proper access privileges to those who should be allowed to use the database. Some parts of the database should be available for access to selected groups only. For example, corporate payroll data should be available only to the payroll department and a few other selected personnel.

Recovery Management What happens to the database when hardware or software malfunctions happen? Suppose you lose data because of a disk head crash. Maybe an application program or system software corrupts the data content and affects the consistency of the database. How do you deal with database failures? How do you recover lost data or bring the database back to a consistent state?

Recovery of a database returns it to a consistent state after any kind of failure. Two modules of the DBMS—log manager and recovery manager—enable recovery after failures. Log manager saves database updates on another data set, called the

log file, while updates take place in the database. Recovery manager uses the log file and brings the database to a consistent state.

Multilevel Data Independence Remember that one of the significant advantages of the database approach relates to data independence. In the earlier file-oriented systems, data definitions and application logic were intricately interwoven within application programs. Every time data structures changed programs had to be modified, causing serious productivity problems. This is not the case in a database environment. DBMS provides data independence. Let us see how.

DBMS supports a three-schema architecture that separates the level of details about the data as indicated below.

Internal level. Deals with physical storage; describes details of storage and access mechanisms. At this level, DBMS keeps track of the files, records, blocks, and physical locations.

Conceptual level. At this level, DBMS keeps the structure of the whole database for the entire user community. This level concentrates on entities, relationships, constraints, and user data access operations. Details of physical storage are hidden from this level.

External level. User views of individual user groups form this level. From this level, only the data covered by specific user views are available for individual user groups. For example, the user view of the accounting group may consist of just parts of customer and billing data.

The DBMS separates out the three level schemas. Requests for data go through a mapping between the external level to the conceptual level and then between the conceptual level and the internal level. So, what do these levels have to do with data independence? Very simply, a change of the schema at one level need not result in a change at another level.

You may rearrange or modify the conceptual level schema. For example, you may add some new fields to customer data. This change will affect only those user views where the new fields are required. All other user views need not be modified. The ability to change the conceptual schema without changing the external schema or the application programs refers to logical data independence. Also, you may shift the physical files and move them to new storage media without having to change the conceptual or external schemas. This is physical data independence. With a three-schema architecture, the DBMS provides for logical and physical data independence.

DBMS Classifications

In any database environment, DBMS forms the most significant software component. We have briefly looked at the major functions of DBMS. We have reviewed the reasons why you need a specialized set of software modules in a database environment. If every DBMS serves the same purposes, are all DBMSs the same? Will any DBMS work in every environment?

Although all DBMSs have similar functions and features, they are not all the same. There are subtle but significant differences. DBMSs may be broadly classified based on the following criteria:

- Meant for single user or multiple users
- Physically linked (hierarchical, network) or logically linked (relational)
- Whether based on inverted file structure with interfile connections
- Whether restricted to specific data models (hierarchical, network, relational, object-relational)
- Intended for centralized or distributed databases
- Based on the type of data distribution (homogeneous, heterogeneous, or federated)
- General purpose or special purpose

Languages and Interfaces

Let us now elaborate on the means by which database practitioners and users interact with the database. They require interface mechanisms to access the database. The database administrators need languages to manage the database; users need languages to access the database. Figure 2-6 indicates the interface mechanisms for interacting with the database.

Let us examine the different types of languages and interfaces and study their functions. Each type serves a distinct function.

Data Definition Language (DDL) Database administrators use DDL to define the data structures and store the definitions in the data dictionary also known as the data catalog. DDL provides language syntax for defining the conceptual and external schemas. Specifically, DDL includes features to define the following:

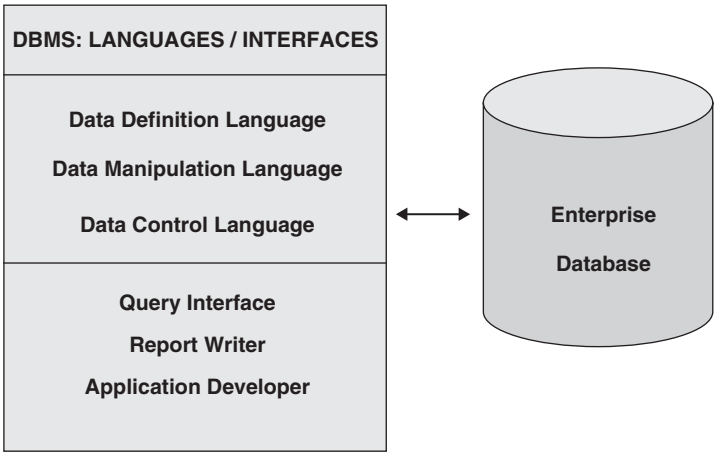

Figure 2-6 Languages and interfaces.

- Tables, segments, or files
- Each data item and groups of data items
- Data item properties such as type, encoding, and length
- Range of values for data items, wherever necessary
- Edit rules and error checking rules
- Key fields to identify structures independently
- Relationships between tables, segments, or files
- Unique names for data components
- Privacy locks

Data Manipulation Language (DML) Programmers and analysts use DML to provide data access to users. When you want to write programs to retrieve data from the database, update data, or delete data, you have to use a data manipulation language. DMLs contain commands for performing individual data retrieval and update operations. In practice, DDL and DML usually get integrated into a single language interface. For example, Structure Query Language (SQL), which has become the standard for relational DBMSs, includes both DDL and DML components.

On the basis of the way the language enables programmers to state their data requests, DMLs may be placed in two distinct groups.

Procedural. When a programmer uses a procedural DML, he or she will have to write the specific commands and individual steps in proper sequence to perform the database operation. The programmer will have to give detailed instructions on how the data retrieval task must be done. You have to write the detailed procedure specifying the sequence of the explicit instructions. A procedural DML details how to perform the data access operation, step by step.

Nonprocedural. On the other hand, when a programmer uses a nonprocedural language, he or she need not list the individual instructions and the sequence in which they have to be executed to complete the data retrieval process. It is not necessary to write detailed procedures. The language is nonprocedural. A nonprocedural DML just mentions what data are to be retrieved. The implementation of the language takes care of how the data retrieval will be carried out. Nonprocedural languages are easier to learn and use.

Data Control Language (DCL) Database administrators have the responsibility for protecting databases from unauthorized access. They need a language to control access to the database. First, unauthorized personnel must be kept away from the database. Second, even for the authorized users, database access must be provided only for those parts of the database each group is entitled to. Even if a group is authorized to access certain parts of the database, access privileges may vary. Some in the group may only retrieve and view the data. Others might be allowed to add data, and yet a few others may be authorized to update or delete data.

Database administrators use DCL to grant or revoke access privileges to users. Granting access privileges could involve granting privileges to the entire

database or only to some parts. Access privileges also cover privileges to retrieve, add, update, or delete data. When circumstances change, database administrators may also need to revise the initial authorizations and revoke the privileges. DCL provides commands extending to various types of granting and revoking data access privileges.

Query Interface A database query is a request for information. A query may take the form: "Display all the customers in the country of Switzerland who bought products from Japanese manufacturers during the prior month." Or the query may just relate to one customer: "Display the last three orders from customer 1234." When a programmer or a user submits a query, the query interface interprets, optimizes, executes the query, and displays the result.

Report Writer This is another interface between the programmer or user and the database. Many types of report writers exist. At a minimum, a report writer allows a programmer to construct a report with proper headings, page breaks, and totals. It also includes a method for submitting the reports for execution and printing them.

Application Developer The application developer comprises software for creating applications. Mostly used by programmers, the application developer enables them to combine input forms, reports, menus, submenus, and application logic to create program modules and then to pull all of these together as specific applications that can be used by various groups of users.

FUNCTIONS AND FEATURES

We have looked at the individual components that make up the database environment in an organization. Briefly, we have seen the hardware, software, and people components. You have observed how the various pieces fit together and support the environment. You have gained a broad understanding of database management systems and the necessity for such sets of specialized software.

Each component in the environment serves a special purpose. Each component is required to perform certain functions. To serve its designated purpose, each component is equipped with specific features. For example, the modern storage devices have fault-tolerant features that help the continued functioning of the database despite hardware malfunctions. If you want to appreciate why the various components exist in the database environment, you have to study the functions and features. Let us go over the major components and examine their functions and features.

Hardware

Hardware, of course, forms the underlying basis of the infrastructure. The collection of physical devices for storing and processing of data in the database constitute the hardware in the environment. Remarkable progress in hardware components during the last few decades has promoted the use of database systems. Apart from the physical storage devices, other pieces of hardware support the processing and

transmission of data. Note how these pieces are especially important in a database environment.

Processor. Processors have become more powerful, faster, and cheaper. In a database environment, the processors in the client workstations and the application servers have to be powerful and fast. The processors in the client machines perform instructions to present data to the users in many new and sophisticated ways. The fast and powerful processors in the application servers must process complex requests for data from the database. In a three-tier client/server architecture, data storage and retrieval functions are separated out of application servers and the database resides on separate database server machines.

Main Memory. When data are fetched from a database for processing, they are initially brought into memory buffers and then moved through the processors. In a database environment, you deal with movement of large volumes of data through the memory buffers. Proper buffer management is essential. When a request is made for data, the memory buffers are searched first to see whether the requested data are already in the memory buffers as a result of prior data retrievals. If the data are already found in the memory buffers, then the data could be accessed and used much faster. Main memory that can accommodate many large buffers is conducive to the performance of a database environment.

Input/Output Control. Computer systems use I/O (input/output) mechanisms to move data into and out of the computer's main memory from secondary storage. A typical sequence followed by I/O control for movement of data runs like this: select the storage device, wait until the device is ready, transfer a piece of data from the device I/O buffer to the processor's accumulator, transfer the data from the accumulator into a memory location, reckon the next memory location, go back, and repeat the steps until all requested data is moved into memory. When you deal with high volumes of data in a database environment, you need sophisticated I/O control to improve performance.

Tape Storage Devices. Content of databases is backed up regularly and used for recovery if and when malfunctions destroy databases, fully or partially. Also, periodically, parts of old data are archived and saved, sometimes at remote locations. In small and medium-sized database environments, magnetic tapes serve as the medium for backing up and saving the data. Although data can be stored and retrieved sequentially, this restriction is not a handicap for content backup and storage. In the modern database environment, compact tape cartridges are widely used.

Communication Links. In earlier database environments of organizations, most of the processing was done locally around a centralized database. Communication links transferred data between the centralized database and the PCs or computer terminals—all mostly at a single site. Today's database environments are different. Distributed databases have become common. Client/server systems have spread rapidly. Data communications move high volumes of data from distributed databases to different locations, from the database server to the client workstations, and

from the central computer to remote sites. Communication links with sufficient capacity and speed are essential to handle all the data movement.

Storage of Data

Without the storing of data on secondary storage devices, there is no database. How you use secondary storage and how you are able to retrieve data from it affects the performance of the database environment. The storage component contains all data related to the environment, even including screen formats and report formats.

Data Repository. The repository includes all the data values that are part of the database. If your database contains information about customers, orders, invoices, products, and employees, then you store data values of individual customers, orders, invoices, products, and employees in secondary storage. Two special features are important for data repository. The secondary storage must provide for fast access to data so that data requests may be processed speedily. Modern storage media used for databases are robust and fast. Secondly, data storage must be fault tolerant. Data storage mechanisms should ensure that the database operations continue even when some malfunction affects parts of the storage. RAID technology applied to today's data storage allows you to store copies of data redundantly so that even if one part gets corrupted, the database operations continue from the duplicate set of data.

Storage of Structure Definitions. The data storage component also relates to the storage of the structure definitions in the data dictionary. These definitions are also stored on physical storage as files and records. Structure definitions specify the composition of each data element as well as the mappings between external, conceptual, and internal schemas. Again, speed is of the essence for the storage devices holding the data dictionary. This is more so because the data dictionary has to be accessed first for every data access.

Operating System Software

As we have seen, the collective software component constitutes the specialized database management system (DBMS), operating system software, and application programs. Operating system software, among other things, acts as the layer between database management system and hardware. When a user requests some data from the database, the database management system processes the request. From the data dictionary, it determines what data are requested and where the data are stored in physical storage. Then DBMS requests the operating system software to fetch the data from physical storage. The operating system software, in turn, interacts with physical hardware to retrieve the data.

Review the following functions of the operating system software in the database environment.

Hardware Management. The operating system manages allocation and control of hardware resources for processing database requests. When multiple users initiate their requests for data, the operating system prioritizes and sequences the requests and allocates CPU and other hardware resources to the several database transac-

tions. To accomplish these functions, the operating system uses hardware timers and interrupts.

Process Management. The operating system executes database and other operations as processes. A process is the lowest executable level of software. Say that one of your users makes a request for data retrieval from the database. This is a process for the operating system. If another of your users makes a request for his or her data retrieval, that is another process. The operating system manages all such processes by allocating resources, initiating and completing the processes, and resolving deadlocks when processes compete simultaneously and wait for the same resources.

Memory Management. The operating system coordinates the movement of data in and out of the primary and cache memory of a computer. It maintains the free memory space. It manages the memory hierarchy. Main memory is divided into pages, and the memory manager of the operating system allocates memory pages for the execution of the individual processes.

File Management. An important function of the operating system, file management, controls and coordinates the structure and storing of data on secondary storage. Files stored on disk drives and tape devices are under the control of the file manager. Files are collections of data that are moved to and from main memory. The file manager enables applications to create, delete, and insert data into files or alter the contents of files. Suppose a file is mostly accessed in a sequential manner. Then the data blocks of the file must be kept sequentially in disk storage on adjacent cylinders. Ealier mainframe operating systems allowed the systems programmers to control the placement of a file on specific cylinders. However, this control placed a burden on the systems programmers to calculate and allocate the cylinders needed for each file. Also, volatile files had to be reorganized frequently. But, modern operating systems make disk organization transparent to the programmers and manage the organization internally.

Input/Output Management. This area covers the setting up and maintaining of logical channels or paths between processor tasks and workstations. The functions encompass channel allocation and deallocation, channel setup, channel coordination, and remote data movement. Error detection and correction over the channels are also part of I/O management. Physical links are initiated, and the data transfer is controlled.

Network Control. Network management software manages the flow of data over communications media. This is especially critical in distributed database environments. Major functions of network management include message routing, naming schemes, addressing of nodes, protection against potential deadlocks of competing data movements, media access control, error detection and correction, and communications setup.

Fault Detection and Recovery. The operating system monitors the overall system for malfunctions, faults, and failures. When you consider a busy database environ-

ment with thousands of users needing information, fault detection and recovery assumes crucial importance. The operating system relies on hardware agents to detect errors. Once the software monitors of the operating system detect errors, the errors are localized and resolved.

Database Software

By now, you are fully aware that everything in the database environment hangs together by the specialized set of software components known as the database management system (DBMS). Database software enables you to create, maintain, and manage the database environment. As we have noted, DBMS is an integrated collection of software modules with each component designated for special purposes. In the last section of this chapter, we will delve into a DBMS and examine each component in detail.

Users

Who are the users in a database environment? It is clear that the database in an organization exists solely to serve the needs of the users at various levels, in different locations, and performing distinct functions. We can classify the users into a few groups on the basis of their data requirements, levels of their computer skills, and the frequency of their data access.

Casual Users. This group uses the database occasionally from time to time. The types and content of the needed data differ each time. Usually, middle- and upper-level executives fall into this group. Casual users, being somewhat naïve of database features, need special considerations for providing data access. This group of users is comfortable with predefined queries and reports where they can enter simple parameters to make the data requests. You have to provide these users with simple menu-driven applications.

Regular Users. The database in a production environment serves as the information repository for the regular users to perform their day-to-day functions. This group of users—whether from the accounting department entering customer orders, from the marketing department setting monthly quotas for the salespersons, or from the inventory control department reconciling physical inventory—constantly retrieve data and update the database. These users are familiar with what data are available and how to make use of the data. They work with programs that retrieve data, manipulate data in many ways, and change data to current values. You have to provide these users with customary menu-driven online applications and standard reports on a daily basis.

Power Users. These users do not require well-structured menu-driven applications to access the database. They can write their own queries and format their reports. For the power users, you have to provide general guidance about the query and reporting tools and let them have a road map of the database contents. Some power users may even construct their own menu-driven specialized applications.

Specialized Users. Business analysts, scientists, researchers, and engineers who are normally part of research and development departments need the database for specialized purposes. These users know the power and features of database systems. They write their own applications. Users of computer-aided design (CAD) tools also fall into this group. These users need special database tools and utilities.

Individual Users. Some users in your organization may justify maintaining stand-alone, private databases for individual departments. Departments working with very highly sensitive and confidential data may qualify for personal databases. Software packages with easy graphics and menus meet the requirements of this group of users.

Practitioners

We refer to the entire group of IT personnel who create and maintain databases as database practitioners. These people keep the database environment functioning and enable the users to reap the benefits. Broadly, we can classify database practitioners into three categories: those who design, those who maintain, and those who create the applications. Let us review the individual roles.

Business analysts. Work with senior management, conduct special studies, and interpret management's direction for information requirements. Interact with other database practitioners.

Data modelers. On the basis of information requirements, create semantic data models with popular modeling techniques. Use computer-aided software engineering (CASE) tools to create data models.

Database designers. Develop user views of data, integrate the views, and create the logical design for the database to be implemented with the selected commercial DBMS. Use CASE tools to generate the outputs of the logical design process.

Systems analysts. Determine the process requirements of the various user groups and write application specifications. Work with the programmers in testing and implementing applications.

Programmer/analysts. Translate applications into program code. Test and implement applications. Also, write programs to convert and populate databases from prior file systems.

Database administrators. Exercise overall control and manage the database. Perform key functions in the database environment including the following:

- Assist in acquiring additional hardware and system software.
- Recommend and assist in obtaining the DBMS.
- Create the physical model from the logical model and enter data dictionary definitions.

- Perform the physical organization and implementation.
- Authorize access to the database.
- Monitor and coordinate database usage.
- Perform ongoing maintenance functions.
- Provide consultation to other practitioners.
- Define backup and recovery mechanisms.
- Improve performance.

Methods and Procedures

How do methods and procedures fit into the discussion of the components of the database environment? These are not tangible components. Can methods and procedures be really considered as components? Without proper methods and procedures, any database environment will slip into confusion and chaos. Some procedures are essential just for making the database stay up and running. Users and practitioners both need good methods and procedures.

Procedures may be manual and automated. Here is a brief list of their purpose and features.

Usage. Users need clear and concise procedures on how to make use of the database. They must have plain and understandable instructions on how to sign on, initiate the menus, navigate through submenus, and use the various functions provided by the applications.

Queries and Reports. Casual and regular users must have a list of available predefined queries and reports along with instructions on how to supply the parameters and run them. Power users require guidelines on how to format and print reports.

Routine Maintenance. Routine maintenance consists of monitoring usage of storage, allocation of additional space whenever necessary, keeping track of access patterns, and looking out for problem areas. The database administrators must have proper procedures to perform the routine maintenance functions.

Structure Change Management. Do not imagine that after the initial implementation of the database in your organization the data structures remain static. Business conditions keep changing continuously, and the data structures must change accordingly. Every organization must have detailed procedures on how to initiate structure changes, review and approve the changes, and implement the changes.

Backup and Recovery. Your users cannot afford to be without the database for long periods. In the event of a disaster, database administrators must be able to recover from the malfunctions and restore the functioning of the database. Clearly defined and tested procedures will resolve disruptions. Backup and recovery procedures stipulate the type and frequency of backups, the planning and testing of

recovery, and proper methods for recovering from malfunctions with the backed up data.

Database Tuning. As the volume of data increases the number of users multiplies, and as the usage rises database performance tends to degrade. Database administrators must constantly monitor performance and find ways to tune the database. Tuning procedures help them with instructions on how to locate areas that slow down and to apply techniques for improving performance.

How Databases Are Used

Figure 2-7 summarizes the functions and features of a database environment by illustrating how the users interface with the database for their data requirements and how database practitioners interact to perform their roles.

Note the Methods and Procedures shown on the right next to Users and Practitioners. As discussed just above, both groups of people refer to methods and procedures for interacting with the database. One the other side of the figure, note the DBMS component. All interaction with the database passes through the database management system. Look at the other blocks in the middle of the figure. These software components enable users and practitioners to achieve their objectives in the database environment.

Applications. Systems analysts and programmers develop and implement applications. Regular users interact with the database through applications.

CASE tools. Data modelers create models with CASE tools. Database designers complete the logical design through CASE tools. Database administrators may use CASE tools to create schemas for defining data structures in the data dictionary.

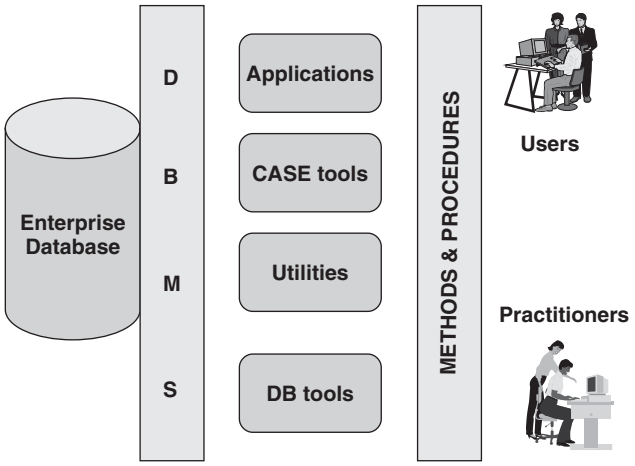

Figure 2-7 How databases are used.

Utilities. Database administrators use a comprehensive suite of utility programs to perform their maintenance and tuning functions.

DB tools. Programmer/analysts make use of query and reporting tools to create components of applications. Power users and specialized users create their own queries and reports with the aid of these tools.

INSIDE A DBMS

We have discussed the significance and capabilities of the database management system (DBMS). You know that it is a collection of specialized software modules. Database administrators cannot implement, control, and manage databases without the DBMS. Every request for data from the user groups must go through the DBMS. Power users need the facilities of the DBMS to create and run their database queries. DBMS stands in the center of an organization's database environment.

Let us now take a look inside the DBMS. Let us find out what components form the software collection and examine each component. Over the years, the features and functions of DBMSs have been broadened. Today's DBMSs do not just provide data access; they contain a powerful and versatile set of tools. Making the toolkit a part of DBMSs is a major step forward. Figure 2-8 shows how the various software modules inside a DBMS may be grouped as major components. We will study each of these components.

Database Engine

The kernel or heart of the DBMS is the database engine. From this central position, the database engine coordinates all the other components. When you review every other component, you will note their specific tasks. The engine has the special responsibility of coordinating the tasks performed by the other components. Because of this coordination, every database operation gets completed correctly and

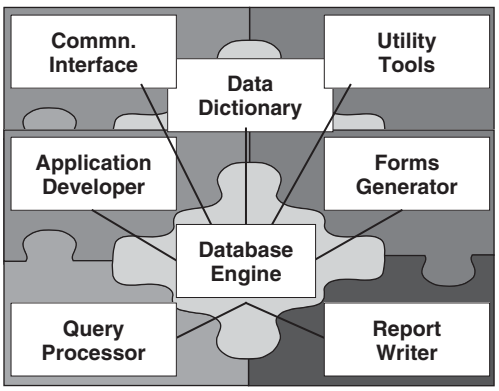

Figure 2-8 Inside a DBMS.

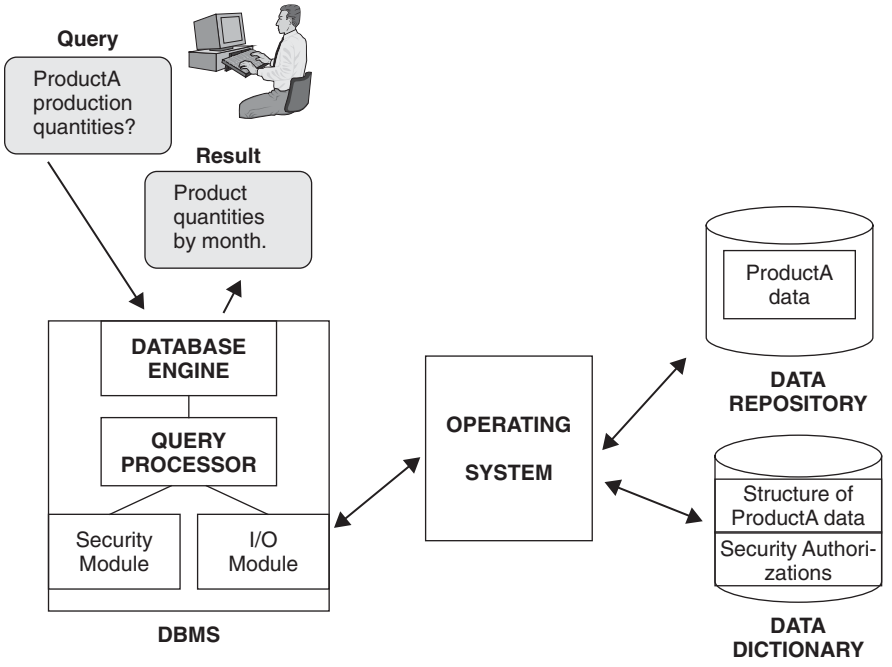

Figure 2-9 Database engine: data access.

completely. The other components depend on the database engine to store data and other system information so that their tasks may be carried out.

In the following subsections, review a few of the major features of the database engine component and note how this component interacts with other components.

Data Access Suppose you, as a programmer, create a query for one of your user groups that needs production data of products from the database of your organization. When your users run this query, the DBMS must coordinate the tasks to be carried out by its various components. Once the structures of the product data and production data are determined, the DBMS interacts with the operating system to retrieve the requested data. Database engine component is responsible for storing data, retrieving data, and updating data in the database.

Figure 2-9 demonstrates how the engine component processes the query request by coordinating the functions of other components.

Note the interaction with the data dictionary. When your query requests for the production quantities of Product A, the query need not specify that the quantity for the product is numeric, is 8 bytes long, and starts in position 24 of the product record. All of this information is stored in the data dictionary. The query just names the quantity field and specifies Product A by name. Data dictionary definitions enable the query to find the location of the required data.

Does the user executing the query have authorization to access the data specified in the query? The database engine component coordinates the services of the security module to verify the authorization. When the location of the requested data is determined from the data dictionary, the engine component coordinates the tasks

of the Input/Output module to work with the operating system to actually fetch the data from secondary storage.

To run a program for updating quantities for Product A, the database engine has to invoke the services of other modules for transaction management, concurrency control, and logging and recovery management. As we have seen, the database engine exercises control and coordination for every data storage, retrieval, and update operation.

Business Rules Business rules ensure data integrity and data validity. Can you have negative numbers in the field for product cost or product price? In the employee record, can the days worked in a year plus the vacation days exceed 366? In a bank loan system, can the minimum interest rate be more than the maximum rate for a variable loan? All such discrepancies are resolved by enforcing of business rules by the database engine component.

Edit and business rules are stored in the data dictionary along with the structure definitions. The database engine component enforces the rules while storing or updating data. Some rules other than the ones mentioned above may be more involved.

For example, consider a program used for assigning employees for special projects. When an employee is assigned to a project and an authorization report is printed by the program, the employee can start on the project. Only employees who have undergone extensive training programs may be assigned to special projects. When an employee is assigned to a special project before he or she completes the required training programs, the system must provide warnings. This is not just a matter of having a few edit rules. Here you require a more complex software module to check whether the employee has completed the training. Such modules are stored in the database itself and are triggered when someone tries to print an authorization report for an employee lacking training.

Database Performance The database engine component is also responsible for the speed of database operations. When a database request arrives, the database engine component is able to determine the needs of each database request, to figure out which modules must be brought into action, to specify the efficient order of the actions, and to coordinate the actions rapidly. The efficiency and the pace of database operations rest on the makeup of the database engine.

Now turn your attention to the volumes of data accessed by individual transactions. A request for the name and address of a customer produces a small result set. If you want the names of all customers in San Francisco, you are looking for more data in the result set. If you require the names of all customers in the whole of the western region, now the data volume of the result set is even larger. The database engine must be able to scale up to accommodate data requests producing large result sets.

Data Dictionary

In an earlier section, we considered the data dictionary as one of the components in the overall architecture of a database environment. In that discussion, *data*

dictionary refers to the storage of structure definitions—the storage repository of structure definitions. Apart from this storage repository itself, within a DBMS you have software components to link the data dictionary repository with other software components. That data dictionary interface software is considered a distinct component within a DBMS.

Let us now examine all aspects of data dictionary—both the storage of structure definitions and the software interface to the data dictionary storage.

Consider a distinct difference between database systems and file-oriented systems. The difference lies in the way data is accessed. Database systems enable you to access physical data in a more abstract manner. You do not have to specify the physical layout of the data structure of how data is stored, nor do you need to indicate where exactly the data is to be found on secondary storage media. This provides you with enormous flexibility and ease in writing programs with database operations. The data dictionary is a fundamental component providing this capability. In most relational databases, the data dictionary comprises a set of system tables. It is a like a directory system. The data dictionary is also known as metadata—data about data.

Particularly, the data dictionary performs the following tasks:

- Keeps the definitions of all data structures and relationships
- Describes data types of all stored data items
- Specifies field sizes and formats of data items
- Holds information on space allocations and storage locations for all data
- Enables the database engine to keep track of the data and store it in the assigned places
- Maintains indexes
- Stores report and screen definitions

Query Processor

Power users, who know more about computer capabilities and usage, write queries to retrieve information from the enterprise database. You, as a programmer or analyst working within the Information Technology department, formulate and write queries to be used by other user groups. Specialized users like business analysts, engineers, and scientists also write their own queries for their information requirements.

To write queries, you need a language to communicate your request to the computer system. Each commercial vendor adopts a standard query language or develops its own proprietary language for creating queries. SQL (Structured Query Language) has become the standard for relational DBMSs. Each relational database vendor enhances the standard SQL and adds a few features here and there, but the essential features of SQL remain in every commercial DBMS. You write queries with SQL and run them in your database environment.

Let us say you write the following query in SQL for the marketing VP in your company, who wants to plan a marketing campaign for promotional mailings to a certain segment of the customer base:

```
SELECT *
        FROM CUSTOMER
                WHERE CUSTOMER_ZONE = 'North' and
                        CUST_BALANCE > 10000
```

You submit the query for execution and printing of the customer list for the VP. What happens in the DBMS when your query is presented to it for execution? The database engine coordinates the services of the query processor to go through different steps and produce the results for the query. Figure 2-10 shows the steps of the query execution.

Note the processing of the query through the following key steps:

- Database engine initiates the execution.
- Data dictionary determines the data elements needed.
- Security module checks data access authorizations.
- Query processor verifies language syntax.
- Query optimizer subcomponent examines the query and creates an optimal execution plan.
- Query processor works with the database engine to execute the optimal plan.
- Database engine interfaces with the operating system to retrieve the requested data.
- Query processor presents the data.

When you write and submit a query for execution, the query component takes over and produces the result set. Apart from using direct queries, you also request information through other means. You may use a form or screen to initiate the data request. Data retrieval may also be through applications. The query processor component is involved when you make data requests through other such methods.

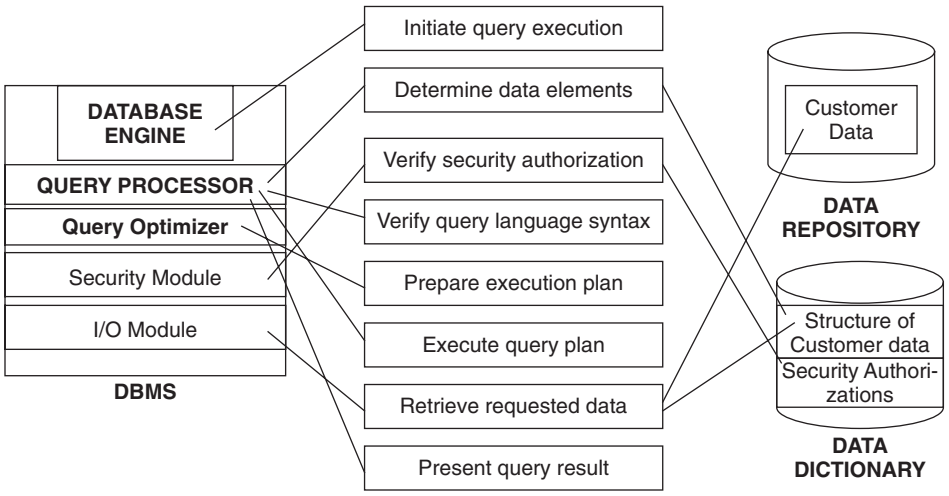

Figure 2-10 Query processor: query execution.

Forms Generator

The forms generator component in the DBMS enables you to create screen layouts or forms for data input and data display. Data input and data display are the two equally important purposes served by forms. You want to display the details of a customer order to your user. You need a layout to display the various data elements such as the order number, date, method of shipment, products ordered, price, and so on. Forms may also include graphics and images. You may want to display the picture of the employee in a form containing employee information.

Figure 2-11 indicates the modules used for forms design and execution. Leading commercial DBMSs offer facilities within the forms generator to create self-contained small applications.

The following elements are present in the forms generator component of many DBMSs, serving several purposes:

Navigator. Presents all the elements that may be combined to create a small application with forms.

Trigger. Function executed when some condition or activity occurs on a form.

Alert. Used to provide warnings or messages on the form to the user for a response.

Block and item. In a form, a block corresponds to an individual structure in the database, for example, customer data structure. An item refers to an individual piece

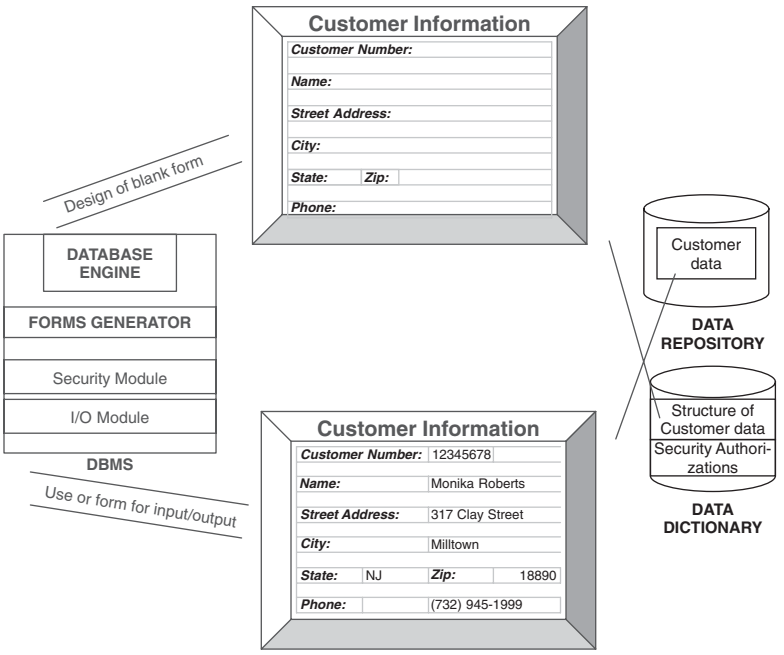

Figure 2-11 Forms generator: design and execution.

of data or data field. Blocks and items on the form are the places where data is input or displayed.

Relationship. Represents the associations between separate blocks and items. Presents all the elements that may be combined to create a small application with forms.

Editor. Window for viewing and maintaining large data structures.

Lists of values. List of permissible values for individual fields.

Parameters. Used to pass values from form to form.

Property sheets. Contain the characteristics of individual data elements on the form.

Layout editor. Enables form objects to be laid out on a canvas to create the format of a form.

Report Writer

In earlier file-oriented data systems, reports were the only means of providing information to users. Also, all batch processing and printing of reports happened overnight. In today's database environment, most data retrieval and data manipulation occur on-line in an interactive manner. As we have seen, the query processor and forms generator components provide online facilities for data access. So, what is the need for report writers?

Although most information access is on-line, a good deal of information delivery still remains in the medium of printed reports. Reports may first be viewed online before selecting and printing only the ones needed on hard copy. Nevertheless, these are reports, and the use of reports is not likely to be eliminated altogether. Power users and specialized users have needs to format and generate their own reports. All applications generally include a substantial suite of daily, weekly, and monthly reports. You can create reports to show detailed and summary data. Reports can show results of complex calculations; they can print multilevel totals.

Figure 2-12 illustrates design and execution of reports. Note how the report writer interfaces with the query processor to retrieve the necessary data. After the query processor delivers the requested data, the report writer formats and prints the report.

Here are a few significant capabilities of report writers in modern DBMSs:

- Easy interface with query processor to retrieve data based on complicated retrieval conditions
- Sophisticated layout editors to format intricate reports
- Methods for storing individual report preferences
- Quick construction of tabular reports
- Creation of summary and detailed reports

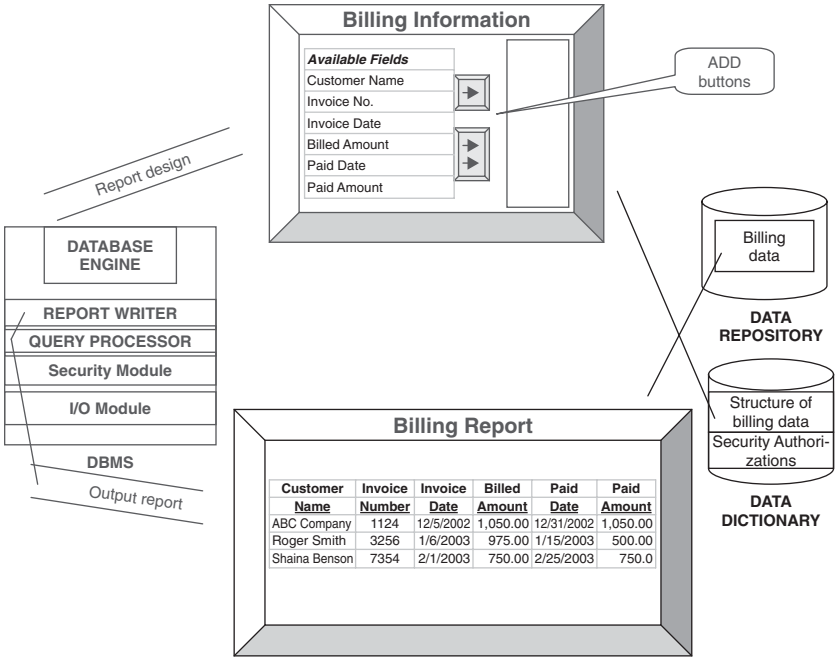

Figure 2-12 Report writer: design and execution.

- Incorporation of report break levels and subtotals
- Usage of subqueries during execution of reports
- Customizing reports with boilerplate graphics and text
- Generation of specialized formats: mailing label, form letter, matrix
- Dynamic reports using dynamic query parameters

Applications Developer

We have discussed the components of query processor, forms generator, and report writer. The outputs of these components form the basis for developing applications for the users. An application consists of forms, reports, and underlying queries. The set of forms and reports provides the users with the capabilities to input, retrieve, and manipulate data in the database. In a sense, an application is a complete package to serve the needs of the users for performing their specific day-to-day tasks. Sales entry, order processing, inventory control, payroll, employee project assignment, and general ledger are examples of applications built with forms, reports, and queries.

The applications developer or applications generator component assembles the outputs of other components to develop coherent systems to support specific functions in the organization. The applications developer components enable the use of menus, submenus, and navigation among various execution options. Small applications consist of just a handful of forms and reports. With the powerful applications developer in modern DBMSs, you can put together a small application easily and

quickly. Application developers contain several tools to develop even the most complex of applications within a reasonably short time.

Communications Interface

Some database management systems have separate modules for establishing and managing communications among databases running on different computer systems and under multiple operating systems. The databases may be distributed among various locations.

Communications interface modules interact with the operating system and network software to initiate and manage the connections. They manage the data flow across databases; they promote true global data sharing.

Another aspect of the communications interface is the link with traditional programming languages. Programs written in traditional languages like COBOL or C++ are able to establish links with the database through the communication interface, which in turn interacts with the query processor.

Utilities

No modern database management system is complete without a comprehensive set of utility programs and interface modules. Most commercial vendors have enhanced their DBMSs and equipped them with rich sets of utility tools. This set of tools and interface modules is commonly known as the toolkit portion of database software. The toolkit is primarily intended for database administrators to perform their various functions. Database environments are much more broad and complex than file-oriented systems. Data sharing also implies proper authorization for data retrieval and update. Organizations cannot afford to be without their databases even for a short time. Quick recovery from disasters and malfunctions is mandatory.

Figure 2-13 presents the types of utility tools and interface modules.

Observe how the set of utilities supports the following broad areas:

- Access control
- Preservation of data integrity
- Performance improvement
- Recovery from malfunctions
- Interfaces to languages, programs, and other database systems

CHAPTER SUMMARY

- Most of today's organizations have adopted database environments for their information needs.
- The database environment encompasses several components, not just the data and the software to manage it. The environment includes the following: data repository, data dictionary, DBMS, systems software, hardware, application

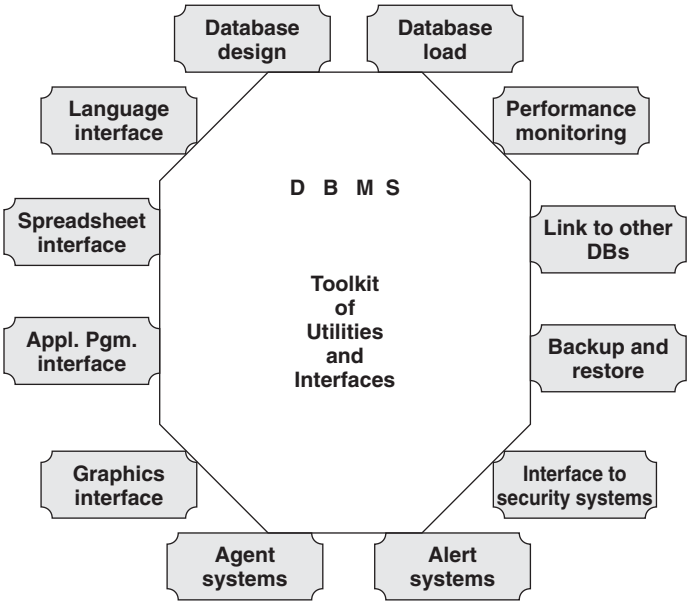

Figure 2-13 DBMS: toolkit of utilities and interfaces.

programs, front-end tools, procedures, people interface, database practitioners, and, importantly, the users.

- Database and DBMS do not mean the same thing. Database refers to the stored data and the definition of its structures. DBMS is the collection of software modules to create, access, maintain, and manage a database.
- Unlike a file-oriented environment, specialized software as a DBMS package is needed because of sophistication in the areas of language interface, storage management, transaction management, and so on.
- Languages needed for database practitioners and users to interact with the database: data definition language (DDL), data manipulation language (DML), and data control language (DCL).
- Each component such as hardware, operating system software, DBMS, and the others possesses specific features and performs definite functions.
- Looking inside a DBMS you will find distinct software modules that perform precise functions and provide specific services in a database environment.

REVIEW QUESTIONS

1. List any six major components of a database environment. Why do you think components other than data storage and database software are also included as part of the environment?
2. Explain the distinction between database and DBMS.

3. Unlike in a file-oriented environment, specialized software (DBMS) is absolutely necessary in a database environment. Give any four reasons why this is true.
4. What is multilevel data independence? How does DBMS provide this independence?
5. What are the major hardware components that support a database environment?
6. Describe any three functions of operating system software in a database environment.
7. How can you classify the users in a database environment? List the requirements of any two of the user groups.
8. "The database engine is the heart of the DBMS." Is this so? Explain briefly.
9. List any four types of services provided by the data dictionary.
10. What are forms? What is the role of the forms generator in the DBMS?

EXERCISES

1. Indicate whether true or false:
   A. People and procedures are part of a database environment.
   B. DML contains features for defining data structures.
   C. The external level schema defines files, records, blocks, and physical locations.
   D. In a database environment, large volumes of data pass through memory buffers.
   E. Tape storage devices are not useful for database backup.
   F. CASE tools may be used for data modeling.
   G. Sometimes the data dictionary is also used for storing the actual data.
   H. The applications developer component in the DBMS is independent and does not use the outputs from other components.
   I. Fault-tolerant data storage generally keeps duplicate sets of data.
   J. The database engine component is responsible for the speed of database operations.
2. Manual and automated procedures are part of the overall database environment in an organization. Discuss the types of procedures needed by database practitioners and end users.
3. What is transaction management in a database environment? Describe the DBMS components that perform transaction management.
4. Discuss the role of operating system software in a database environment. What functions must this software provide?
5. You are one of the power users in your company's database environment. How would you plan to use the database and what services do you expect to be provided for you?