Databases and Database Systems

9.1 INTRODUCTION: A database can be summarily described as a repository for data. This makes clear that building databases is really a continuation of a human activity that has existed since writing began; it can be applied to the result of any bookkeeping or recording activity that occurred long before the advent of the computer era. However, this description is too vague for some of our purposes, and we refine it as we go along.

The creation of a database is required by the operation of an enterprise. We use the term enterprise to designate a variety of endeavors that range from an airline, a bank, or a manufacturing company to a stamp collection or keeping track of people to whom you want to write New Year cards.

Database Management Systems

A database management system (DBMS) is an aggregate of data, hardware, software, and users that helps an enterprise manage its operational data. The main function of a DBMS is to provide efficient and reliable methods of data retrieval to many users. For example a college has 10,000 students each year and each student can have approximately 10 grade records per year, then over 10 years, the college will accumulate 1,000,000 grade records. It is not easy to extract records satisfying certain criteria from such a set, and by current standards, this set of records is quite small!

Given the current concern for “grade inflation”, a typical question that we may try to answer is determining the evolution of the grade averages in introductory programming courses over a 10-year period. Therefore, it is clear that efficient data retrieval is an essential function of database systems.

Most DBMSs deal with several users who try simultaneously to access several data items and, frequently, the same data item. For instance, suppose that we wish to introduce an automatic registration system for students. Students may register by using terminals or workstations. Of course, we assume that the database contains information that describes the capacity of the courses and the number of seats currently available. Suppose that several students wish to register for cs210 in the spring semester of 2003. Unfortunately, the capacity of the course is limited, and not all demands can be satisfied. If, say, only one seat remains available in that class, the database must handle these competing demands and allow only one registration to go through.

Database System Hardware
Database management systems are, in most cases, installed on general-purpose computers. Since the characteristics of the hardware have strongly influenced the development of DBMSs, we discuss some of the most important of these characteristics.

For our purposes, it is helpful to categorize computer memory into two classes: internal memory and external memory. Although some internal memory is permanent, such as ROM,1 we are interested here only in memory that can be changed by programs. This memory is often known as RAM.2 This memory is volatile, and any electrical interruption causes the loss of data. By contrast, magnetic disks and tapes are common forms of external memory. They are nonvolatile memory, and they retain their content for practically unlimited amounts of time. The physical characteristics of magnetic tapes force them to be accessed sequentially, making them useful for backup purposes, but not for quick access to specific data.

In examining the memory needs of a DBMS, we need to consider the following issues:
• Data of a DBMS must have a persistent character; in other words, data must remain available long after any program that is using it has completed its work. Also, data must remain intact even if the system breaks down.
• A DBMS must access data at a relatively high rate.
• Such a large quantity of data need to be stored that the storage medium must be low cost. These requirements are satisfied at the present stage of technological development only by magnetic disks.

Database System Software
Users interact with database systems through query languages. The query language of a DBMS has two broad tasks: to define the data structures that serve as receptacles for the data of the database, and to allow the speedy retrieval and modification of data. Accordingly, we distinguish between two components of a query language: the data definition component and the data manipulation component.

The main tasks of data manipulation are data retrieval and data update. Data retrieval entails obtaining data stored in the database that satisfies a certain specification formulated by the user in a query. Data updates include data modification, deletion and insertion.

Programming in query languages of DBMSs is done differently from programming in higher-level programming languages. The typical program written in C, Pascal, or PL/1 directly implements an algorithm for solving a problem. A query written in a database query language merely states what the problem is and leaves the construction of the code that solves the problem to a special component of the DBMS software. This approach to programming is called nonprocedural.

A central task of DBMSs is transaction management. A transaction is a sequence of database operations (that usually consists of updates, with possible retrievals) that must be executed in its entirety or not at all. This property of transactions is known as atomicity. A typical example includes the transfer of funds between two account records A and B in the database of a bank. Such a banking operation should not modify the total amount of funds that the bank has in its accounts, which is a clear consistency requirement for the database. The transaction consists of the following sequence of operations:
1. Decrease the balance of account A by d dollars;
2. Increase the balance of account B by d dollars.

9.2 Introduction to Database Concepts

If only the first operation is executed, then d dollars will disappear from the funds deposited with the bank. If only the second is executed, then the total funds will increase by d dollars. In either case, the consistency of the database will be compromised. Thus, a transaction transforms one consistent database state into another consistent database state, a property of transactions known as consistency.

Typically, at any given moment in time a large number of transactions coexist in the database system. The transaction management component ensures that the execution of one transaction is not influenced by the execution of any other transaction. This is the isolation property of transactions. Finally, the effect of a transaction to the state of the database must by durable, that is, it must persist in the database after the execution of the transaction is completed. This is the durability property of transactions. Collectively, the four fundamental properties of transactions outlined above are known as the ACID properties, the acronym of atomicity, consistency, isolation, and durability. DBMS software usually contains application development tools in addition to query languages. The role of these tools is to facilitate user interface development. They include forms systems, procedural and nonprocedural programming languages that integrate database querying with various user interfaces, etc.

The Users of a Database System

The community of users of a DBMS includes a variety of individuals and organizational entities. These users are classified based on their roles and interests in accessing and managing the databases.

Once a database is created, it is the job of the database administrator to make decisions about the nature of data to be stored in the database, the access policies to be enforced (who is going to access certain parts of the database), monitoring and tuning the performance of the database, etc. At the other extremity of the user range, we have the end users. These users have limited access rights, and they need to have only minimal technical knowledge of the database. For instance, the end users of the database of the reservation system of an airline are travel and sales agents. The end users of a DBMS of a bank are bank tellers, users of the ATM machines, etc. A particularly important category of users of DBMSs (on whom we focus in this book) consists of application programmers. Their role is to work within existing DBMS systems and, using a combination of the query languages and higher-level languages, to create various reports based on the data contained in the database. In some cases, they write more general programs that depend on these data.

The Architecture of Database Systems

The architecture of a DBMS can be examined from several angles: the functional architecture that identifies the main components of a DBMS, the application

The Architecture of Database Systems 5 architecture that focuses on application uses of DBMSs, and the logical architecture that describes various levels of data abstractions.

The query processor converts a user query into instructions the DBMS can process efficiently, taking into account the current structure of the database (also referred as metadata — which means data about data).

The memory manager obtains data from the database that satisfies queries compiled by the query processor and manages the structures that contain data, according to the DDL directives.

Finally, the transaction manager ensures that the execution of possibly many transactions on the DBMS satisfies the ACID properties mentioned above and, also, provides facilities for recovery from system and media failures. The standard application architecture of DBMSs is based on a client/server model. The client, which can be a user or an application, generates a query that is conveyed to the server. The server processes the query (a process that includes parsing, generation of optimized execution code, and execution) and returns an answer to the client. This architecture is known as two-tier architecture. In general, the number of clients may vary over time.

In large organizations, it is often necessary to create more layers of processing, with, say, a layer of software to concentrate the data activities of a branch office and organize the communication between the branch and the main data repository. This leads to what is called a multi-tier architecture. In this setting data are scattered among various data sources that could be DBMSs, file systems, etc. These constitute the lowest tier of the architecture, that is, the tier that is closest to the data. The highest tier consists of users that act through user interfaces and applications to obtain answers to queries. The intermediate tiers constitute the middleware, and their role is, in general, to serve as mediators between the highest and the lowest tiers. Middleware may be consist of web servers, data warehouses, and may be considerably complex. Multi-tier architecture is virtually a requirement for world wide web applications. The logical architecture, also known as the ANSI/SPARC architecture, was elaborated at the beginning of the 1970s. It distinguishes three layers of data abstraction:

1. The physical layer contains specific and detailed information that describes how data are stored: addresses of various data components, lengths in bytes, etc. DBMSs aim to achieve data independence, which means that the database organization at the physical level should be indifferent to application programs.

2. The logical layer describes data in a manner that is similar to, say, definitions of structures in C. This layer has a conceptual character; it shields the user from the tedium of details contained by the physical layer, but is essential in formulating queries for the DMBS.

3. The user layer contains each user's perspective of the content of the database.

A Historical Perspective of Database Systems

The history of DBMSs begins in the late 1960s, when an IBM product named IMS (Information Management System) was launched. Data was structured hierarchically, in forests of trees of records, providing very fast access. A few years after IMS appeared, in 1971, the CODASYL Database Task Group proposed a new type of database models known today as the network model. The original report considered DBMSs as extensions of the COBOL language, and structured data contained by databases as graphs of records, consisting essentially of circular linked lists. The origins of the relational model, that is the mainstay of contemporary databases are in E. F. Codd's work in the early and mid 1970s. The development of relational database began in the late 1970s and early 1980s with an experimental relational database sytem at IBM called System R, a precursor of commercial IBM DBMSs, SQL/DS and DB2. A multitude of DBMSs emerged in the 1980s, such as ORACLE, INGRES, Rdb, etc. Relational technology evolved further in the 1990s with the addition of ideas and techniques inspired by object-oriented programming.