Chapter 1. Database System

CHAPTER 1

DATABASE SYSTEM

After reading this chapter, the reader will understand:

- *The basic concept of database and database management system*
- *The need of database system*
- *Limitations of traditional file processing system that led to the development of database management system*
- *Advantages of database system*
- *Various developments in the field of database system*
- *Areas where the database system is used*
- *Cost and risk involved in database system*
- *Different types of individuals that interact with the database system*
- *The concept of database schema and database instance*
- *Three-level DBMS architecture, which provides the abstract view of the database*
- *The concept of data independence, which helps the user to change the schema at one level of the database system without having to change the schema at the other levels*
- *Database models, which provide the necessary means to achieve data abstraction Different types of DBMS languages*
- *Component modules of DBMS*
- *Difference between centralized and client/server database system*
- *Various types of database management systems*
- *The steps involved in designing the database system*

Storing data and retrieving information has been a necessity in all ages of business. The term **data** can be defined as a set of isolated and unrelated raw facts with an implicit meaning. Data can be anything such as, name of a person, a number, images, sound, etc. For example, ‘Monica,’ ‘25,’ ‘student,’ etc., is a data. When the data is processed and

converted into a meaningful and useful form, it is known as **information**. For example, 'Monica is 25 years old and she is a student.' is an information. For a business to be successful, a fast access to information is vital as important decisions are based on the information available at any point of time. Traditionally, the data was stored in voluminous repositories such as files, books, and ledgers. However, storing data and retrieving information from these repositories was a time-consuming task. With the development of computers, the problem of information storage and retrieval was resolved. Computers replaced tons of paper, file folders, and ledgers as the principal media for storing important information.

Since then numerous techniques and devices have been developed to manage and organize the data so that useful information can be accessed easily and efficiently. The eternal quest for data management has led to the development of the database technology. A **database** can be defined as a collection of related data from which users can efficiently retrieve the desired information. A database can be anything from a simple collection of roll numbers, names, addresses, and phone numbers of students to a complex collection of sound, images, and even video or film clippings. Though databases are generally computerized, instances of non-computerized databases from everyday life can be cited in abundance. A dictionary, a phone book, a collection of recipes, and a TV guide are all common examples of non-computerized databases. The examples of computerized databases include customer files, employee rosters, books catalogue, equipment inventories, and sales transactions.

In addition to the storage and retrieval of data, certain other operations can also be performed on a database. These operations include adding, updating, and deleting data. All these operations on a database are performed using a database management system. A **Database Management System (DBMS)** is an integrated set of programs used to create and maintain a database. The main objective of a DBMS is to provide a convenient and effective method of defining, storing, retrieving, and manipulating the data contained in the database. In addition, the

DBMS must ensure the security of the database from unauthorized access and recovery of the data during system failures. It must also provide techniques for data sharing among several users. The database and the DBMS software are collectively known as **database system**.

1.1 NEED OF DATABASE SYSTEM

The need of database systems arose in the early 1960s in response to the traditional file processing system. In the **file processing system**, the data is stored in the form of files, and a number of application programs are written by programmers to add, modify, delete, and retrieve data to and from appropriate files. New application programs are written as and when needed by the organization.

For example, consider a bookstore that uses a file processing system to keep track of all the available books. The system maintains a file named BOOK to store information related to books. This information include book title, ISBN, price (in dollars), year of publishing, copyright date, category (such as language books, novels, and textbooks), number of pages, name of the author, name and address of the publisher, etc. In addition, the system has many application programs that allow users to manipulate the information stored in BOOK file. For example, the system may contain programs to add information about a new book, modify any existing book information, print the details of books according to their categories, etc.

Learn More

The ISBN is a unique commercial book identifier barcode. It is assigned to each edition of a book. An ISBN consists of several parts such as publisher code, title code, etc. These parts are usually separated by hyphens.

Now, suppose a need arises to keep additional information about the publishers of the books that includes phone number and email id. To fulfil this need, the system creates a new file say PUBLISHER, which includes name, address, phone number, and email id of the publishers. New

application programs are written and added to the system to manipulate the information in the PUBLISHER file. In this way, as the time goes by, more files and application programs are added to the system.

This system of storing data in the form of files is supported by a conventional operating system. However, the file processing system has a number of disadvantages, which are given here.

- Same information may be duplicated in several files. For example, the name and the address of publisher are stored in BOOK file as well as in PUBLISHER file. This duplication of data is known as **data redundancy**, which leads to wastage of storage space. The other problem with file processing system is that the data may not be updated consistently. Suppose a publisher requests for a change in his address. Since the address of the publisher is stored in the BOOK as well as PUBLISHER file, both the files must be updated. If the address of the publisher is not modified in any of the two files, then the same publisher will have different address in two different files. This is known as **data inconsistency**.
- In any application, there are certain data integrity rules that need to be maintained. These rules could be in the form of certain conditions or constraints. In a file processing system, all these rules need to be explicitly programmed in all application programs, which are using that particular data item. For example, the integrity rule that each book should have a book title has to be implemented in all the application programs separately that are using the BOOK file. In addition, when new constraints are to be enforced, all the application programs should be changed accordingly.
- The file processing system lacks the insulation between program and data. This is because the file structure is embedded in the application program itself, thus, it is difficult to change the structure of a file as it requires changing all the application programs accessing it. For example, an application program in C++ defines the file structure using the keyword `struct` or `class`. Suppose the data type of the field ISBN is changed from string to number, changes

have to be made in all the application programs that are accessing the BOOK file.

- Handling new queries is difficult, since it requires change in the existing application programs or requires a new application program. For example, suppose a need arises to print details of all the textbooks published by a particular publisher. One way to handle this request is to use the existing application program that prints details of the books according to their categories, and then manually generate the list of textbooks published by a particular publisher. Obviously, it is unacceptable. Alternatively, the programmer is requested to write a new application program. Suppose such a new application program is written. Now suppose after some time, a request arises to filter the list of textbooks with price greater than *$50*. Then again we have to write a new application program.
- Since many users are involved in creating files and writing application programs, the various files in the system may have different file structures. Moreover, the programmers may choose different programming languages to write application programs.
- Application programs are added in an unplanned manner, due to which details of each file are easily available to every user. Thus, the file processing system lacks security feature.

NOTE In spite of its limitations, the file processing system is still in use mainly for database backup.

1.2 ADVANTAGES OF DATABASE SYSTEM

Database approach came into existence due to the bottlenecks of file processing system. In the database approach, the data is stored at a central location and is shared among multiple users. Thus, the main advantage of DBMS is **centralized data management**. The centralized nature of database system provides several advantages, which overcome the limitations of the conventional file processing system. These advantages are listed here.

- **Controlled data redundancy:** During database design, various files

are integrated and each logical data item is stored at central location. This eliminates replicating the data item in different files, and ensures consistency and saves the storage space. Note that the redundancy in the database systems cannot be eliminated completely as there could be some performance and technical reasons for having some amount of redundancy. However, the DBMS should be capable of controlling this redundancy in order to avoid data inconsistencies.

- **Enforcing data integrity:** In database approach, enforcing data integrity is much easier. Various integrity constraints are identified by database designer during database design. Some of these data integrity constraints can be enforced automatically by the DBMS, and others may have to be checked by the application programs.
- **Data sharing:** The data stored in the database can be shared among multiple users or application programs. Moreover, new applications can be developed to use the same stored data. Due to shared data, it is possible to satisfy the data requirements of the new applications without having to create any additional data or with minimal modification.
- **Ease of application development:** The application programmer needs to develop the application programs according to the users' needs. The other issues like concurrent access, security, data integrity, etc., are handled by the DBMS itself. This makes the application development an easier task.
- **Data security:** Since the data is stored centrally, enforcing security constraints is much easier. The DBMS ensures that the only means of access to the database is through an authorized channel. Hence, data security checks can be carried out whenever access is attempted to sensitive data. To ensure security, a DBMS provides security tools such as user codes and passwords. Different checks can be established for each type of access (addition, modification, deletion, etc.) to each piece of information in the database.
- **Multiple user interfaces:** In order to meet the needs of various users having different technical knowledge, DBMS provides different

types of interfaces such as query languages, application program interfaces, and graphical user interfaces (GUI) that include forms-style and menu-driven interfaces. A form-style interface displays a form to each user and user interacts using these forms. In menu-driven interface, the user interaction is through lists of options known as menus.

- **Backup and recovery:** The DBMS provides backup and recovery subsystem that is responsible for recovery from hardware and software failures. For example, if the failure occurs in between the transaction, the DBMS recovery subsystem either reverts back the database to the state which existed prior to the start of the transaction or resumes the transaction from the point it was interrupted so that its complete effect can be recorded in the database.

In addition to centralized data management, DBMS also has some other advantages, which are discussed here.

- **Program-data independence:** The independence between the programs and the data is known as program-data independence (or simply data independence). It is an important characteristic of DBMS as it allows changing the structure of the database without making any changes in the application programs that are using the database. To provide a high degree of data independence, the definition or the description of the database structure (structure of each file, the type and storage format of each data item) and various constraints on the data are stored separately in **DBMS catalog**. The information contained in the catalog is called the **metadata** (data about data). This independence is provided by a three-level DBMS architecture, which is discussed in Section 1.7.
- **Data abstraction:** The property of DBMS that allows program-data independence is known as data abstraction. Data abstraction allows the database system to provide an abstract view of the data to its users without giving the physical storage and implementation details.

- **Supports multiple views of the data:** A database can be accessed by many users and each of them may have a different perspective or view of the data. A database system provides a facility to define different views of the data for different users. A **view** is a subset of the database that contains virtual data derived from the database files but it does not exist in physical form. That is, no physical file is created for storing the data values of the view; rather, only the definition of the view is stored.

Due to these advantages, the traditional file processing system for the bookstore is replaced by an *Online Book* database to maintain the information about various books, publishers, and authors. The author details include the name, address, URL, and phone number of author. Note that some of the authors also review the books and give ratings and feedbacks about the books. The database also maintains the details about the ratings given to the books by different reviewers.

1.3 DEVELOPMENTS IN DATABASE SYSTEM

The database systems are divided into three generations. The first generation includes the database systems based on hierarchical and network data model. The second generation includes the database systems based on relational data model, and the third generation includes the database systems based on object-oriented and object-relational data model.

In the early 1960s, the first general purpose DBMS, called the Integrated Data Store was designed by Charles Bachman. The development of magnetic tapes led to the automation of various data processing tasks such as payroll and inventory. In the late 1960s, the Information Management System (IMS) DBMS was developed by IBM. Two data models, network data model and hierarchical data model were also developed. During this period, use of hard disks instead of magnetic tapes facilitated direct access to data. In 1970, Edger Frank 'Ted' Codd (E.F. Codd) proposed the relational data model and non-procedural way to query data that led to the development of relational databases. Codd

was felicitated with the prestigious Association of Computing Machinery Turing Award for his work.

In the 1980s, relational model became the dominant DBMS paradigm. SQL was standardized and the current standard known as SQL:1999 was adopted by ANSI/ISO. Research on parallel and distributed databases was carried out. During this period, several commercial relational databases such as IBM's DB2, Oracle, Informix UDS extended their systems to store new and complex data types such as images and text.

In the early 1990s, SQL was designed mainly for decision support applications. In this period, vendors introduced products for parallel database. They also began to add object-relational support to their databases. In the late 1990s, with the advent of World Wide Web, the use of DBMS to store data accessed through a Web browser became widespread. Database systems were designed to support very high transaction processing rates and 24 × 7 availability. Moreover, the database systems became more reliable during this period. In the early 2000s, XML databases and associated query language Xquery were developed to handle large amount of data and high transaction rates.

The emergence of several enterprise resource planning (ERP) and management resource planning (MRP) packages like PeopleSoft, SAP, Siebel, etc., has added a substantial layer of application-oriented features on top of DBMS. These packages provide a general application layer to carry out a common set of tasks such as financial analysis, inventory control, human resource planning, etc. Different companies can customize this layer according to their need, which leads to the reduction in the overall cost of building the application layer.

1.4 APPLICATION AREAS OF DATABASE SYSTEM

Database systems are widely used in different areas because of their numerous advantages. Some of the most common database applications are listed here.

- **Airlines and railways:** Airlines and railways use online databases for reservation, and for displaying the schedule information.
- **Banking:** Banks use databases for customer inquiry, accounts, loans, and other transactions.
- **Education:** Schools and colleges use databases for course registration, result, and other information.
- **Telecommunications:** Telecommunication departments use databases to store information about the communication network, telephone numbers, record of calls, for generating monthly bills, etc.
- **Credit card transactions:** Databases are used for keeping track of purchases on credit cards in order to generate monthly statements.
- **E-commerce:** Integration of heterogeneous information sources (for example, catalogs) for business activity such as online shopping, booking of holiday package, consulting a doctor, etc.
- **Health care information systems and electronic patient record:** Databases are used for maintaining the patient health care details.
- **Digital libraries and digital publishing:** Databases are used for management and delivery of large bodies of textual and multimedia data.
- **Finance:** Databases are used for storing information such as sales, purchases of stocks and bonds or data useful for online trading.
- **Sales:** Databases are used to store product, customer and transaction details.
- **Human resources:** Organizations use databases for storing information about their employees, salaries, benefits, taxes, and for generating salary checks.

1.5 COST AND RISK OF DATABASE SYSTEM

Database systems have many advantages; however, before implementing database systems over the traditional file processing system, the cost and risk factors of implementing database systems should also be considered. The various cost and risk factors are given here.

- **High cost:** Installing new database system may require investment

in hardware and software. The DBMS requires more main memory and disk storage. Moreover, DBMS is quite expensive. Therefore, a company needs to consider the overhead cost of implementing a new database system.

- **Training new personnel:** When an organization plans to adopt a database system, it may need to recruit or hire a specialized data administration group, which can coordinate with different user groups for designing views, establishing recovery procedures, fine tuning data structures to meet the organization requirements. Hiring such professionals is expensive.
- **Explicit backup and recovery:** A shared corporate database must be accurate and available at all times. Therefore, a system using online updation requires explicit backup and recovery procedures.
- **System failure:** When a computer system containing the database fails, all users have to wait until the system is functional again. Moreover, if DBMS or application program fails a permanent damage may occur to the database.

NOTE In spite of cost and risk involved, database systems are widely used.

1.6 PEOPLE WHO INTERACT WITH DATABASES

Database is an important asset of business. To design, use, and maintain such important asset many people are involved. The people who work with databases include *database users, system analysts, application programmers,* and *database administrator (DBA)*. **Database users** are those who interact with the database in order to query and update the database, and generate reports. Database users are further classified into the following categories:

- **Naive users:** The users who query and update the database by invoking some already written application programs. For example, the owner of the bookstore enters the details of various books in the database by invoking appropriate application program. The na i ve user interacts with the database using form interface.

- **Sophisticated users:** The users, such as business analyst, scientist, etc., who are familiar with the facilities provided by a DBMS interact with the system without writing any application programs. Such users use database query language to retrieve information from the database to meet their complicated requirements.

  Things to Remember

  Though DBMS provides several facilities to access a database, but all kind of users need not learn about all these facilities. Naive users need to learn only some simple facilities of DBMS, whereas sophisticated users have to know most of the DBMS facilities in order to fulfill their complex requirements.

- **Specialized users:** The users who write specialized database programs, which are different from traditional data processing applications, such as banking and payroll management which use simple data types. Specialized users write applications such as computer-aided design systems, knowledge-base and expert systems that store data having complex data types.

System analysts determine the requirements of the database users (especially na i ve users) to create a solution for their business need, and focus on non-technical and technical aspects. The nontechnical aspects involve defining system requirements, facilitating interaction between business users and technical staff, etc. Technical aspects involve developing the specification for user interface (application programs). **Application programmers** are the computer professionals who implement the specifications given by the system analysts, and develop application programs. They can choose tools, such as rapid application development (RAD) to develop the application program with minimal effort. The database application programmer develops application program to facilitate easy data access for the database users.

Database administrator (DBA) is a person who has central control over both data and application programs. The responsibilities of DBA vary

depending upon the job description and corporate and organization policies. Some of the responsibilities of DBA are given here.

- **Schema definition and modification:** The overall structure of the database is known as **database schema**. It is the responsibility of the DBA to create the database schema by executing a set of data definition statements in DDL. The DBA also carries out the changes to the schema according to the changing needs of the organization.
- **New software installation:** It is the responsibility of the DBA to install new DBMS software, application software, and other related software. After installation, the DBA must test the new software.
- **Security enforcement and administration:** DBA is responsible for establishing and monitoring the security of the database system. It involves adding and removing users, auditing, and checking for security problems.
- **Data analysis:** DBA is responsible for analyzing the data stored in the database, and studying its performance and efficiency in order to effectively use indexes, parallel query execution, etc.
- **Preliminary database design:** The DBA works along with the development team during the database design stage due to which many potential problems that can arise later (after installation) can be avoided.
- **Physical organization modification:** The DBA is responsible for carrying out the modifications in the physical organization of the database for better performance.
- **Routine maintenance checks:** The DBA is responsible for taking the database backup periodically in order to recover from any hardware or software failure (if occurs). Other routine maintenance checks that are carried out by the DBA are checking data storage and ensuring the availability of free disk space for normal operations, upgrading disk space as and when required, etc.

1.7 DBMS ARCHITECTURE AND DATA INDEPENDENCE

The DBMS architecture describes how data in the database is viewed by

the users. It is not concerned with how the data is handled and processed by the DBMS. The database users are provided with an abstract view of the data by hiding certain details of how data is physically stored. This enables the users to manipulate the data without worrying about where it is located or how it is actually stored.

In this architecture, the overall database description can be defined at three levels, namely, *internal, conceptual*, and *external* levels and thus, named **three-level DBMS architecture**. This architecture is proposed by ANSI/SPARC (American National Standards Institute/Standards Planning and Requirements Committee) and hence, is also known as **ANSI/SPARC architecture**. The three levels are discussed here.

- **Internal level**: It is the lowest level of data abstraction that deals with the physical representation of the database on the computer and thus, is also known as **physical level**. It describes *how* the data is physically stored and organized on the storage medium. At this level, various aspects are considered to achieve optimal runtime performance and storage space utilization. These aspects include storage space allocation techniques for data and indexes, access paths such as indexes, data compression and encryption techniques, and record placement.
- **Conceptual level:** This level of abstraction deals with the logical structure of the entire database and thus, is also known as **logical level**. It describes *what* data is stored in the database, the relationships among the data and complete view of the user's requirements without any concern for the physical implementation. That is, it hides the complexity of physical storage structures. The conceptual view is the overall view of the database and it includes all the information that is going to be represented in the database.
- **External level:** It is the highest level of abstraction that deals with the user's view of the database and thus, is also known as **view level**. In general, most of the users and application programs do not require the entire data stored in the database. The external level describes a part of the database for a particular group of users. It

permits users to access data in a way that is customized according to their needs, so that the same data can be seen by different users in different ways, at the same time. In this way, it provides a powerful and flexible security mechanism by hiding the parts of the database from certain users, as the user is not aware of existence of any attributes that are missing from the view.

These three levels are used to describe the schema of the database at various levels. Thus, the three-level architecture is also known as **three-schema architecture**. The internal level has an **internal schema**, which describes the physical storage structure of the database. The conceptual level has a **conceptual schema**, which describes the structure of entire database. The external level has **external schemas** or **user views**, which describe the part of the database according to a particular user's requirements, and hide the rest of the database from that user. The physical level is managed by the operating system under the direction of DBMS. The three-schema architecture is shown in Figure 1.1.

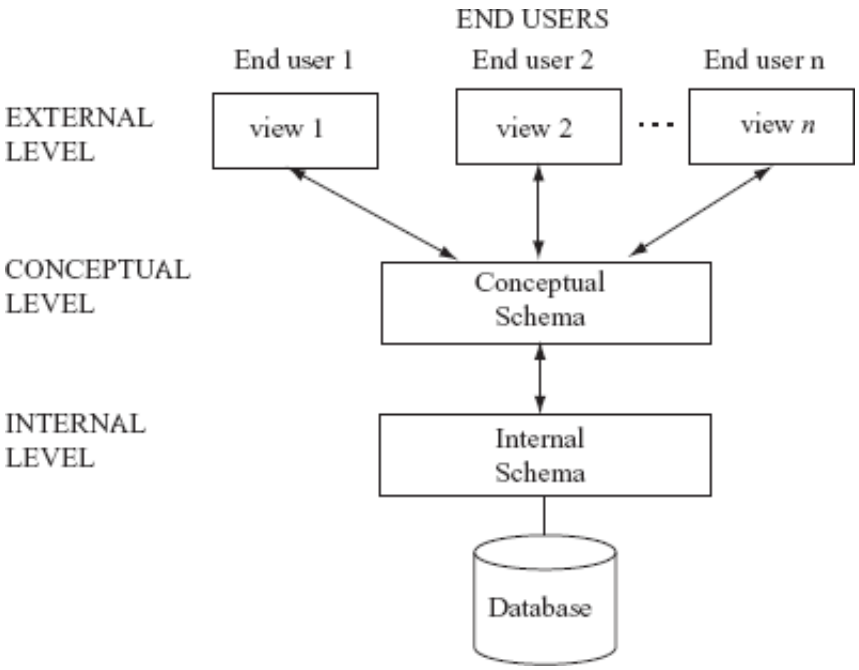

Fig. 1.1 *Three-schema architecture*

To understand the three-schema architecture, consider the three levels of the BOOK file in *Online Book* database as shown in Figure 1.2. In this figure, two views (*view 1* and *view 2*) of the BOOK file have been defined at the external level. Different database users can see these views. The

details of the data types are hidden from the users. At the conceptual level, the BOOK records are described by a type definition. The application programmers and the DBA generally work at this level of abstraction. At the internal level, the BOOK records are described as a block of consecutive storage locations such as words or bytes. The database users and the application programmers are not aware of these details; however, the DBA may be aware of certain details of the physical organization of the data.

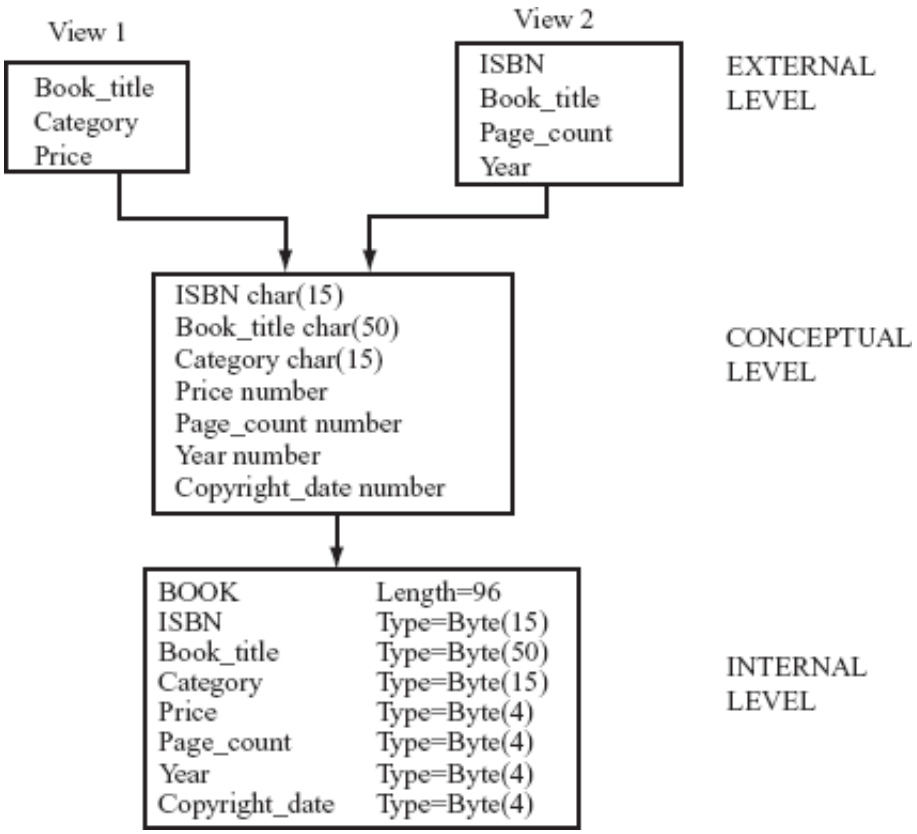

Fig. 1.2 *Three levels of Online Book database (BOOK file)*

In three-schema architecture, each user group refers only to its own external view. Whenever a user specifies a request to generate a new external view, the DBMS must transform the request specified at external level into a request at conceptual level, and then into a request at physical level. If the user requests for data retrieval, the data extracted from the database must be presented according to the need of the user. This process of transforming the requests and results between various levels of DBMS architecture is known as **mapping**.

The main advantage of three-schema architecture is that it provides data

independence. **Data independence** is the ability to change the schema at one level of the database system without having to change the schema at the other levels. Data independence is of two types, namely, *logical data independence* and *physical data independence*.

- **Logical data independence:** It is the ability to change the conceptual schema without affecting the external schemas or application programs. The conceptual schema may be changed due to change in constraints or addition of new data item or removal of existing data item, etc., from the database. The separation of the external level from the conceptual level enables the users to make changes at the conceptual level without affecting the external level or the application programs. For example, if a new data item, say `Edition` is added to the `BOOK` file, the two views (*view 1* and *view 2* shown in Figure 1.2) are not affected.
- **Physical data independence:** It is the ability to change the internal schema without affecting the conceptual or external schema. An internal schema may be changed due to several reasons such as for creating additional access structure, changing the storage structure, etc. The separation of internal schema from the conceptual schema facilitates physical data independence.

Logical data independence is more difficult to achieve than the physical data independence because the application programs are always dependent on the logical structure of the database. Therefore, the change in the logical structure of the database may require change in the application programs.

1.8 DATABASE MODELS

As discussed earlier, the main objective of database system is to highlight only the essential features and to hide the storage and data organization details from the user. This is known as **data abstraction**. A database model provides the necessary means to achieve data abstraction. A **database model** or simply a **data model** is an abstract model that describes how the data is represented and used. A data

model consists of a set of data structures and conceptual tools that is used to describe the structure (data types, relationships, and constraints) of a database.

A data model not only describes the structure of the data, it also defines a set of operations that can be performed on the data. A data model generally consists of **data model theory**, which is a formal description of how data may be structured and used, and **data model instance**, which is a practical data model designed for a particular application. The process of applying a data model theory to create a data model instance is known as **data modeling**.

Depending on the concept they use to model the structure of the database, the data models are categorized into three types, namely, *high-level* or *conceptual data models, representational* or *implementation data models* and *low-level* or *physical data models*.

1.8.1 Conceptual Data Model

Conceptual data model describes the information used by an organization in a way that is independent of any implementation-level issues and details. The main advantage of conceptual data model is that it is independent of implementation details and hence, can be understood even by the end users having non-technical background. The most popular conceptual data model, that is, entity-relationship (E-R) model is discussed in detail in Chapter 02.

1.8.2 Representational Data Model

The representational or implementation data models hide some data storage details from the users; however, can be implemented directly on a computer system. Representational data models are used most frequently in all traditional commercial DBMSs. The various representational data models are discussed here.

Hierarchical Data Model

The hierarchical data model is the oldest type of data model, developed by IBM in 1968. This data model organizes the data in a tree-like structure, in which each **child node** (also known as **dependents**) can have only one **parent node**. The database based on the hierarchical data model comprises a set of records connected to one another through links. The link is an association between two or more records. The top of the tree structure consists of a single node that does not have any parent and is called the **root node**.

The root may have any number of dependents; each of these dependents may have any number of lower level dependents. Each child node can have only one parent node and a parent node can have any number of (many) child nodes. It, therefore, represents only one-to-one and one-to-many relationships. The collection of same type of records is known as a **record type**. Figure 1.3 shows the hierarchical model of *Online Book* database. It consists of three record types, namely, `PUBLISHER`, `BOOK`, and `REVIEW`. For simplicity, only few fields of each record type are shown. One complete record of each record type represents a **node**.

The main advantage of the hierarchical data model is that the data access is quite predictable in the structure and, therefore, both the retrieval and updates can be highly optimized by the DBMS.

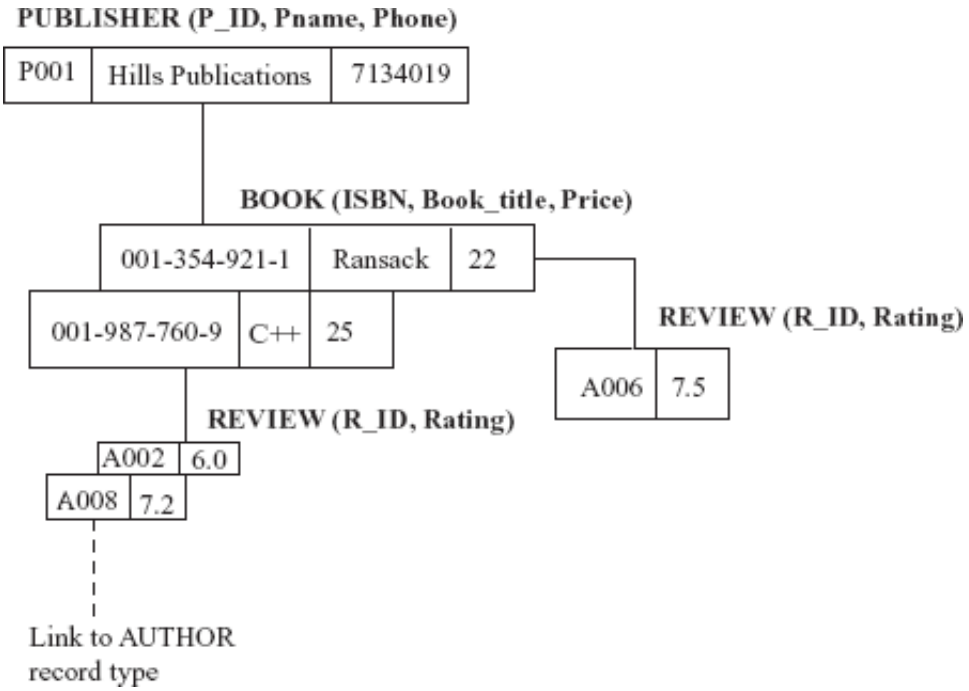

Fig. 1.3 *Hierarchical data model for Online Book database*

However, the main drawback of this model is that the links are 'hard coded' into the data structure, that is, the link is permanently established and cannot be modified. The hard coding makes the hierarchical model rigid. In addition, the physical links make it difficult to expand or modify the database and the changes require substantial redesigning efforts.

Network Data Model

The first specification of network data model was presented by Conference on Data Systems Languages (CODASYL) in 1969, followed by the second specification in 1971. It is powerful but complicated. In a network model the data is also represented by a collection of records, and relationships among data are represented by links. However, the link in a network data model represents an association between precisely two records. Like hierarchical data model, each record of a particular record type represents a node. However, unlike hierarchical data model, all the nodes are linked to each other without any hierarchy. The main difference between hierarchical and network data model is that in hierarchical data model, the data is organized in the form of trees and in network data model, the data is organized in the form of graphs.

The main advantage of network data model is that a parent node can have many child nodes and a child can also have many parent nodes. Thus, the network model permits the modeling of many-to-many relationships in data. The main limitation of the network data model is that it can be quite complicated to maintain all the links and a single broken link can lead to problems in the database. In addition, since there are no restrictions on the number of relationships, the database design can become complex. Figure 1.4 shows the network model of *Online Book* database.

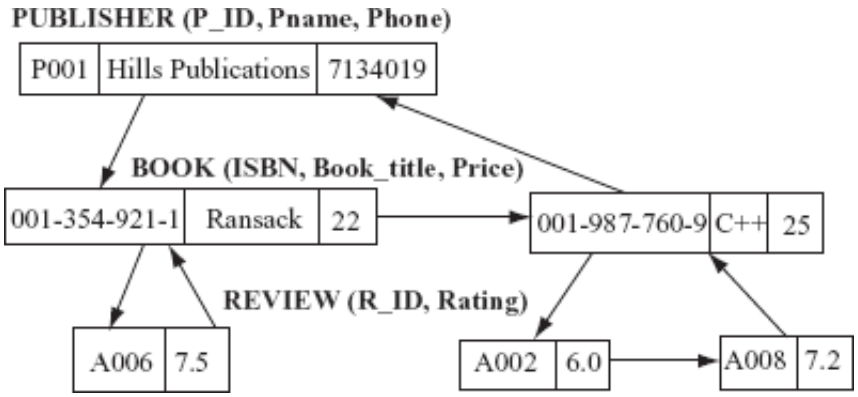

Fig. 1.4 *Network data model for Online Book database*

NOTE Links in hierarchical and network data models are generally implemented through pointers.

Relational Data Model

The relational data model was developed by E. F. Codd in 1970. In the relational data model, unlike the hierarchical and network models, there are no physical links. All data is maintained in the form of tables (generally, known as **relations**) consisting of rows and columns. Each row (record) represents an entity and a column (field) represents an attribute of the entity. The relationship between the two tables is implemented through a common attribute in the tables and not by physical links or pointers. This makes the querying much easier in a relational database system than in the hierarchical or network database systems. Thus, the relational model has become more programmer friendly and much more dominant and popular in both industrial and academic scenarios. Oracle, Sybase, DB2, Ingres, Informix, MS-SQL Server are few of the popular relational DBMSs. Figure 1.5 shows the relational model of *Online Book* database.

BOOK

ISBN	Book_title	Category	Price	Copyright_date	Year	Page_count	P_ID
001-354-921-1	Ransack	Novel	22	2005	2006	200	P001
001-987-760-9	C++	Textbook	25	2004	2005	800	P001

PUBLISHER

P_ID	Pname	Address	State	Phone	Email_ID
P001	Hills Publications	12, Park street, Atlanta	Georgia	7134019	H_pub@hills.com

REVIEW

R_ID	ISBN	Rating
A002	001-987-760-9	6.0
A006	001-354-921-1	7.5
A008	001-987-760-9	7.2

Fig. 1.5 *Relational data model for Online Book database*

NOTE The network, hierarchical, and relational data models are known as **record-based data models** as these models represent data in the form of records.

Object-Based Data Model

In the recent years, the object-oriented paradigm has been applied to database technology, creating two new data models known as object-oriented data model and object-relational data model. The **object-oriented data model** extends the concepts of object-oriented programming language with persistence, versioning, concurrency control, data recovery, security, and other database capabilities. On the other hand, the **object-relational data model** is an extension of relational data model. It combines the features of both the relational data model and object-oriented data model.

Semistructured Data Model

Unlike other data models, where every data item of a particular type must have the same set of attributes, the semistructured data model allows individual data items of the same type to have different set of attributes. In semistructured data model, the information about the description of the data (schema) is contained within the data itself, which is sometimes called *self-describing* data. In such databases there is no clear

separation between the data and the schema and thus, allowing data of any type. Semistructured data model has recently emerged as an important topic of study for different reasons given here.

- There are data sources such as the Web, which is to be treated as databases; however, they cannot be constrained by a schema.
- The need of flexible format for data exchange between heterogeneous databases.
- To facilitate browsing of data.

Semistructured data model facilitates data exchange among heterogeneous data sources. It helps to discover new data easily and store it. It also facilitates querying the database without knowing the data types. However, it loses the data type information.

Learn More

XML (Extensible Markup Language) is a means of representing semistructured data. XML enables to define schema using XML schema language, which allows varying levels of schema over data elements.

BOOK

ISBN	Book_title	Category	Price	Copyright_date	Year	Page_count	P_ID

PUBLISHER

P_ID	Pname	Address	State	Phone	Email_ID

AUTHOR

A_ID	Aname	City	State	Zip	Phone	URL

AUTHOR_BOOK

A_ID	ISBN

REVIEW

R_ID	ISBN	Rating

Fig. 1.6 *Database schema for Online Book database*

1.8.3 Physical Data Model

Physical data model describes the data in terms of a collection of files, indices, and other storage structures such as record formats, record ordering, and access paths. This model specifies how the database will be executed in a particular DBMS software such as Oracle, Sybase, etc., by taking into account the facilities and constraints of a given database management system. It also describes how the data is stored on disk and what access methods are available to it.

NOTE An access path (for example, an index) is a structure that helps in efficient searching and retrieval of data from a particular database.

1.9 DATABASE SCHEMA VERSUS DATABASE INSTANCE

While working with any data model, it is necessary to distinguish between the overall design or description of the database (database schema) and the database itself. As discussed earlier, the overall design or description of the database is known as **database schema** or simply **schema**. The database schema is also known as **intension** of the database, and is specified while designing the database. Figure 1.6 shows the database schema for *Online Book* database.

The data in the database at a particular point of time is known as **database instance** or **database state** or **snapshot**. The database state is also called an **extension** of the schema. The various states of the database are given here.

- **Empty state:** When a new database is defined, only its schema is specified. At this point, the database is said to be in empty state as it contains no data.
- **Initial state:** When the database is loaded with data for the first time, it is said to be in initial state.
- **Current state:** The data in the database is updated frequently. Thus, at any point of time, the database is said to be in the current state.

The DBMS is responsible to check whether the database state is **valid**

state. Thus, each time the database is updated, DBMS ensures that the database remains in the valid state. The DBMS refers to DBMS catalog where the metadata is stored in order to check whether the database state satisfies the structure and constraints specified in the schema. Figure 1.7 shows an example of an instance for PUBLISHER schema.

P_ID	Pname	Address	State	Phone	Email_id
P001	Hills Publications	12, Park street, Atlanta	Georgia	7134019	h_pub@hills.com
P002	Sunshine Publishers Ltd.	45, Second street, Newark	New Jersey	6548909	Null
P003	Bright Publications	123, Main street, Honolulu	Hawai	7678985	bright@bp.com
P004	Paramount Publishing House	789, Oak street, New York	New York	9254834	param_house@param.com
P005	Wesley Publications	456, First street, Las Vegas	Nevada	5683452	Null

Fig. 1.7 *An example of instance* - PUBLISHER *instance*

The schema and instance can be compared with a program written in a programming language. The database schema is similar to a variable declared along with the type description in a program. The variable contains a value at a given point of time. This value of a variable corresponds to an *instance* of a database schema.

NOTE The database schema changes rarely, whereas the data in the database may change frequently according to the requirements. The process of making any changes in the schema is known as **schema evolution**.

1.10 DBMS LANGUAGES

The main objective of a database management system is to allow its users to perform a number of operations on the database such as insert, delete, and retrieve data in abstract terms without knowing about the

physical representations of data. To provide the various facilities to different types of users, a DBMS normally provides one or more specialized programming languages called **Database** (or **DBMS**) **Languages**.

Learn More

In addition to DDL and DML, DBMS also provides two more languages, namely, *data control language (DCL)* and *transaction control language (TCL)*. Data control language is used to create user roles, grant permissions, and control access to database by securing it. Transaction control language is used to manage different transactions occurring within a database.

The DBMS mainly provides two database languages, namely, *data definition language* and *data manipulation language* to implement the databases. Data definition language (DDL) is used for defining the database schema. The DBMS comprises DDL compiler that identifies and stores the schema description in the DBMS catalog. Data manipulation language (DML) is used to manipulate the database.

1.10.1 Data Definition Language

In DBMSs where no strict separation between the levels of the database is maintained, the data definition language is used to define the conceptual and internal schemas for the database. On the other hand, in DBMSs, where a clear separation is maintained between the conceptual and internal levels, the DDL is used to specify the conceptual schema only. In such DBMSs, a separate language, namely, *storage definition language (SDL)* is used to define the internal schema. Some of the DBMSs that are based on true three-schema architecture use a third language, namely, *view definition language (VDL)* to define the external schema.

The DDL statements are also used to specify the integrity rules (constraints) in order to maintain the integrity of the database. The

various integrity constraints are domain constraints, referential integrity, assertions and authorization. These constraints are discussed in detail in subsequent chapters. Like any other programming language, DDL also accepts input in the form of instructions (statements) and generates the description of schema as output. The output is placed in the **data dictionary**, which is a special type of table containing metadata. The DBMS refers the data dictionary before reading or modifying the data. Note that the database users cannot update the data dictionary; instead it is only modified by database system itself.

1.10.2 Data Manipulation Language

Once the database schemas are defined and the initial data is loaded into the database, several operations such as retrieval, insertion, deletion, and modification can be applied to the database. The DBMS provides data manipulation language (DML) that enables users to retrieve and manipulate the data. The statement which is used to retrieve the information is called a **query**. The part of the DML used to retrieve the information is called a **query language.** However, query language and DML are used synonymously though technically incorrect. The DML are of two types, namely, *non-procedural DML* and *procedural DML*.

The **non-procedural** or **high-level** or **declarative DML** enables to specify the complex database operations concisely. It requires a user to specify *what* data is required without specifying *how* to retrieve the required data. For example, SQL (Structured Query Language) is a non-procedural query language as it enables user to easily define the structure or modify the data in the database without specifying the details of *how* to manipulate the database. The high-level DML statements can either be entered interactively or embedded in a general purpose programming language. On the other hand, the **procedural** or **low-level DML** requires user to specify *what* data is required and *how* to access that data by providing step-by-step procedure. For example, relational algebra is procedural query language, which consists of set of operations such as select, project, union, etc., to manipulate the data in

the database. Relational algebra and SQL are discussed in Chapters 04 and 05, respectively.

1.11 COMPONENT MODULES OF DBMS

A database system being a complex software system is partitioned into several software components that handle various tasks such as data definition and manipulation, security and data integrity, data recovery and concurrency control, and performance optimization (see Figure 1.8).

- **Data definition:** The DBMS provides functions to define the structure of the data. These functions include defining and modifying the record structure, the data type of fields, and the various constraints to be satisfied by the data in each field. It is the responsibility of database administrator to define the database, and make changes to its definition (if required) using the DDL and other privileged commands. The **DDL compiler** component of DBMS processes these schema definitions, and stores the schema descriptions in the DBMS catalog (data dictionary). Other DBMS components then refer to the catalog information as and when required.
- **Data manipulation:** Once the data structure is defined, data needs to be manipulated. The manipulation of data includes insertion, deletion, and modification of records. The functions that perform these operations are also part of the DBMS. These functions can handle planned as well as unplanned data manipulation needs.
    - The queries that are defined as a part of the application programs are known as **planned queries**. The application programs are submitted to a **precompiler**, which extracts DML commands from the application program and send them to **DML compiler** for compilation. The rest of the program is sent to the **host language compiler**. The object codes of both the DML commands and the rest of the program are linked and sent to the **query evaluation engine** for execution.
    - The sudden queries that are executed as and when the need

arises are known as **unplanned queries (interactive queries)**. These queries are compiled by the **query complier**, and then optimized by the **query optimizer**. The query optimizer consults the data dictionary for statistical and other physical information about the stored data (query optimization is discussed in detail in Chapter 08). The optimized query is finally passed to the query evaluation engine for execution. The na i ve users of the database can also query and update the database by using some already given application program interfaces. The object code of these queries is also passed to query evaluation engine for processing.

- **Data security and integrity:** The DBMS contains functions, which handle the security and integrity of data stored in the database. Since these functions can be easily invoked by the application, the application programmer need not code these functions in the programs.
- **Concurrency and data recovery:** The DBMS also contains some functions that deal with the concurrent access of records by multiple users and the recovery of data after a system failure. The concurrency control techniques and database recovery is discussed in Chapters 10 and 11, respectively.
- **Performance optimization:** The DBMS has a set of functions that optimize the performance of the queries by evaluating the different execution plans of a query and choosing the best among them.

In this way, the DBMS provides an environment that is both convenient and efficient to use when there is a large volume of data and many transactions need to be processed concurrently.

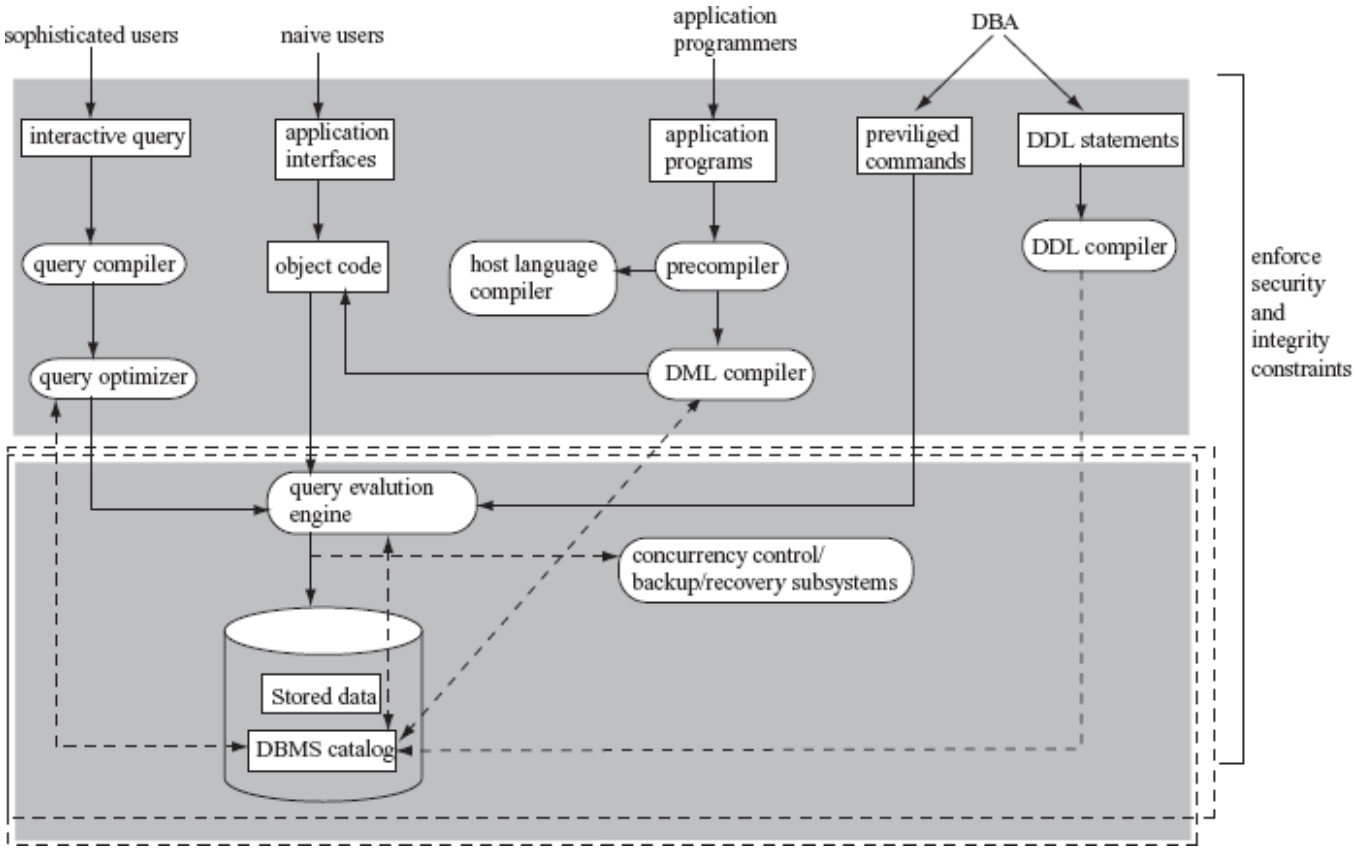

Fig. 1.8 *Component modules of DBMS*

1.12 CENTRALIZED AND CLIENT/SERVER DATABASE SYSTEMS

In **centralized database systems**, the database system, application programs, and user-interface all are executed on a single system and dummy terminals are connected to it. The processing power of single system is utilized and dummy terminals are used only to display the information. As the personal computers became faster, more powerful, and cheaper, the database system started to exploit the available processing power of the system at the user's side, which led to the development of client/server architecture. In **client/server architecture**, the processing power of the computer system at the user's end is utilized by processing the user-interface on that system.

A **client** is a computer system that sends request to the server connected to the network, and a **server** is a computer system that receives the request, processes it, and returns the requested information back to the client. Client and server are usually present at different sites. The end users (remote database users) work on client computer system and database system runs on the server. Servers can be of several types,

for example, file servers, printer servers, web servers, database servers, etc. The client machines have user interfaces that help users to utilize the servers. It also provides users the local processing power to run local applications on the client side.

There are two approaches to implement client/server architecture. In the first approach, the user interface and application programs are placed on the client side and the database system on the server side. This architecture is called **two-tier architecture**. The application programs that reside at the client side invoke the DBMS at the server side. The application program interface standards like Open Database Connectivity (ODBC) and Java Database Connectivity (JDBC) are used for interaction between client and server. Figure 1.9 shows two-tier architecture.

The second approach, that is, **three-tier architecture** is primarily used for web-based applications. It adds intermediate layer known as **application server** (or **web server**) between the client and the database server. The client communicates with the application server, which in turn communicates with the database server. The application server stores the business rules (procedures and constraints) used for accessing data from database server. It checks the client's credentials before forwarding a request to database server. Hence, it improves database security.

When a client requests for information, the application server accepts the request, processes it, and sends corresponding database commands to database server. The database server sends the result back to application server which is converted into GUI format and presented to the client. Figure 1.10 shows the three-tier architecture.

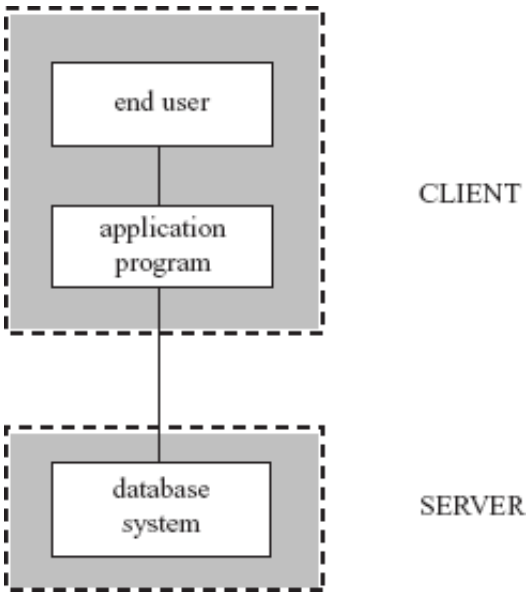

Fig. 1.9 *Two-tier architecture*

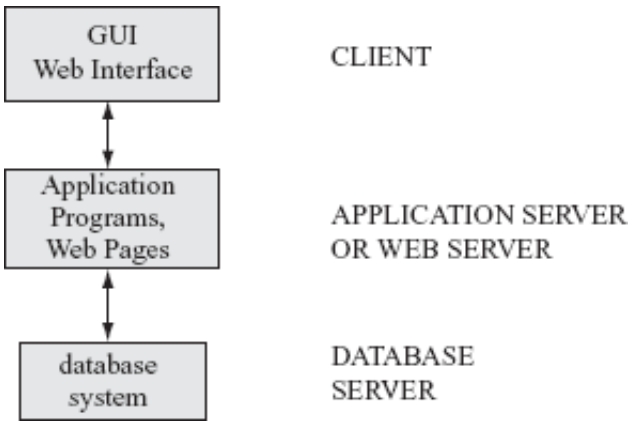

Fig. 1.10 *Three-tier architecture*

1.13 CLASSIFICATION OF DBMSS

The DBMSs can be classified into different categories on the basis of several criteria such as the data model they are using, number of users they support, number of sites over which the database is distributed, and the purpose they serve.

Based on Data Models

The various data models have already been discussed. Depending on the data model they use, the DBMSs can be classified as *hierarchical, network, relational, object-oriented*, and *object-relational*. Among these, the hierarchical and network data models are the older data models and now known as **legacy data models**. Some of the old applications still run

on the database systems based on these models. Most of the popular and current commercial DBMSs are based on relational data model. The object-based data models have been implemented in some DBMSs; however, have not become popular. Due to the popularity of relational databases, the object-oriented concepts have been introduced in these databases that led to the development of a new class of DBMSs called object-relational DBMSs.

Learn More

MYSQL and PostgreSQL are open source (free) DBMS products that are supported by third-party vendors with additional services.

Based on Number of Users

Depending on the number of users the DBMS supports, it is divided into two categories, namely, *single-user system* and *multi-user system*. In **single-user system** the database resides on one computer and is only accessed by one user at a time. The user may design, maintain, and write programs for accessing and manipulating the database according to the requirements, as well as perform all the user roles. The user may also hire database system designers to design a system. In such a case, the single user performs the role of end user only. However, in most enterprises the large amount of data is to be managed and accessed by multiple users and thus, requires **multi-user systems**. In multi-user system, multiple users can access the database simultaneously. In multi-user DBMS, the data is both integrated and shared. For example, the *Online Book* database is a multi-user database system in which the data of books, authors, and publishers are stored centrally and can be accessed by many users.

Based on Number of Sites

Depending on the number of sites over which the database is distributed, it is divided into two types, namely, *centralized* and *distributed database systems*. **Centralized database systems** run on a single computer

system. Both the database and DBMS software reside at a single computer site. The user interacts with the centralized system through a dummy terminal connected to it for information retrieval. In **distributed database systems**, the database and DBMS are distributed over several computers located at different sites. The computers communicate with each other through various communication media such as high-speed networks or telephone lines. Distributed databases can be classified as *homogeneous* and *heterogeneous*. In **homogeneous** distributed database system, all sites have identical database management system software, whereas in **heterogeneous** distributed database system, different sites use different database management system software.

Based on the Purpose

Depending on the purpose the DBMS serves, it can be classified as **general purpose** or **specific purpose**. DBMS is a general purpose software system. It can; however, be designed for specific purposes such as airline or railway reservation. Such systems cannot be used for other applications without major changes. These database systems fall under the category of **online transaction processing** (OLTP) systems. Online transaction processing system is specifically used for data entry and retrieval. It supports large number of concurrent transactions without excessive delays. An automatic teller machine (ATM) for a bank is an example of online commercial transaction processing application. The OLTP technology is used in various industries, such as banking, airlines, supermarkets, manufacturing, etc.

1.14 DATABASE DESIGN PROCESS

The overall database design process has to follow a series of steps. The systematic process of designing a database is known as **design methodology**. Database design involves understanding operational and business needs of an organization, modeling the specified requirements, and realizing the requirements using a database. The goal of designing a database is to produce efficient, high quality, and minimum cost database. In large organizations, database administrator (DBA) is

responsible for designing an efficient database system. He is responsible for controlling the database life-cycle process. The overall database design and implementation process consists of several phases.

1. **Requirement collection and analysis:** It is the process of knowing and analyzing the expectations of the users for the new database application in as much detail as possible. A team of analysts or requirement experts are responsible for carrying out the task of requirement analysis. They review the current file processing system or DBMS system, and interact with the users extensively to analyze the nature of business area to be supported and to justify the need for data and databases. The initial requirements may be informal, incomplete, inconsistent, and partially incorrect. The requirement specification techniques such as object-oriented analysis (OOA), data flow diagrams (DFDs), etc., are used to transform these requirements into better structured form. This phase can be quite time-consuming; however, it plays the most crucial and important role in the success of the database system. The result of this phase is the document containing the specification of user requirements.
2. **Conceptual database design:** In this phase, the database designer selects a suitable data model and translates the data requirements resulting from previous phase into a conceptual database schema by applying the concepts of chosen data model. The conceptual schema is independent of any specific DBMS. The main objective of conceptual schema is to provide a detailed overview of the organization. In this phase, a high-level description of the data and constraints are developed. The entity-relationship (E-R) diagram is generally used to represent the conceptual database design. The conceptual schema should be expressive, simple, understandable, minimal, and formal.
3. **Choice of a DBMS:** The choice of a DBMS depends on many factors such as cost, DBMS features and tools, underlying model, portability, and DBMS hardware requirements. The technical factors that affect the choice of a DBMS are the type of DBMS (relational, object, object-relational, etc.), storage structures and access paths that

DBMS supports, the interfaces available, the types of high-level query languages, and the architecture it supports (client/server, parallel or distributed). The various types of costs that must be considered while choosing a DBMS are software and hardware acquisition cost, maintenance cost, database creation and conversion cost, personnel cost, training cost, and operating cost.

4. **Logical database design:** Once an appropriate DBMS is chosen, the next step is to map the high-level conceptual schema onto the implementation data model of the selected DBMS. In this phase, the database designer moves from an abstract data model to the implementation of the database. In case of relational model, this phase generally consists of mapping the E-R model into a relational schema.
5. **Physical database design:** In this phase, the physical features such as storage structures, file organization, and access paths for the database files are specified to achieve good performance. The various options for file organization and access paths include various types of indexing, clustering of records, hashing techniques, etc.
6. **Database system implementation:** Once the logical and physical database designs are completed, the database system can be implemented. DDL statements of the selected DBMS are used and compiled to create the database schema and database files, and finally the database is loaded with the data.
7. **Testing and evaluation:** In this phase, the database is tested and fine-tuned for the performance, integrity, concurrent access, and security constraints. This phase is carried out in parallel with application programming. If the testing fails, various actions are taken such as modification of physical design, modification of logical design or upgrade or change DBMS software or hardware.

Note that once the application programs are developed, it is easier to change the physical database design. However, it is difficult to modify the logical database design as it may affect the queries (written using DML commands) embedded in the program code. Thus, it is necessary to carry out the design process effectively before developing the application

programs. While designing a database schema, it is necessary to avoid two major issues, namely, *redundancy* and *incompleteness*. These problems may lead to bad database design. The anomalies (an inconsistent, incomplete or contradictory state of the database) that can arise due to these problems are discussed in detail in Chapter 06.

NOTE The database design phases, namely, conceptual, logical, and physical are discussed in subsequent chapters.

SUMMARY

1. The term data can be defined as a set of isolated and unrelated raw facts with an implicit meaning.
2. When the data is processed and converted into a meaningful and useful form, it is known as information.
3. A database can be defined as a collection of related data from which users can efficiently retrieve the desired information.
4. A Database Management System (DBMS) is an integrated set of programs used to create and maintain a database. The main objective of a DBMS is to provide a convenient and effective method of defining, storing, retrieving, and manipulating the data contained in the database.
5. The database and the DBMS software are collectively known as database system.
6. Database approach came into existence due to the bottlenecks of file processing system. In the database approach, the data is stored at a central location and is shared among multiple users. Thus, the main advantage of DBMS is centralized data management. Other advantages of DBMS are program-data independence, data abstraction, and support for multiple views of data.
7. Centralized data management in DBMS offers many advantages such as controlled data redundancy, data consistency, enforcing data integrity, data sharing, ease of application development, data security, multiple user interfaces, and backup and recovery.
8. The database systems are divided into three generations. The first

generation includes the database systems based on hierarchical and network data model. The second generation includes the database systems based on relational data model and the third generation includes the database systems based on object-oriented and object-relational data model.

9. The most common database application areas are airlines and railways, banking, education, telecommunications, credit card transactions, e-commerce, health care information systems and electronic patient records, digital libraries and digital publishing, finance, sales, and human resources.
10. The people who work with databases include database users, system analysts, application programmers, and database administrator (DBA). Database administrator (DBA) is a person who has central control over both data and application programs.
11. The DBMS architecture describes how data in the database is viewed by the users. It is not concerned with how the data is handled and processed by the DBMS. In this architecture, the overall database description can be defined at three levels, namely, internal, conceptual, and external levels and thus, named three-level DBMS architecture. This architecture is proposed by ANSI/SPARC, and hence is also known ANSI/SPARC architecture.
12. Internal level is the lowest level of data abstraction that deals with the physical representation of the database on the computer. Conceptual level deals with the logical structure of the entire database and thus, is also known as logical level. External level is the highest level of abstraction that deals with the user's view of the database and thus is also known as view level.
13. The process of transforming the requests and results between various levels of DBMS architecture is known as mapping.
14. The main advantage of three-schema architecture is that it provides data independence. Data independence is the ability to change the schema at one level of the database system without having to change the schema at the other levels. Data independence is of two types, namely, logical data independence and physical data

independence.

15. The process of highlighting only the essential features of a database system and hiding the storage and data organization details from the user is known as data abstraction.
16. A database model or simply a data model is an abstract model that describes how the data is represented and used. It not only describes the structure of the data but also defines a set of operations that can be performed on the data.
17. Depending on the concept they use to model the structure of the database, the data models are categorized into three types, namely, high-level or conceptual data models, representational or implementation data models, and low-level or physical data models.
18. Conceptual data model describes the information used by an organization in a way that is independent of any implementation-level issues and details.
19. The representational or implementation data models hide some data storage details from the users; however, can be implemented directly on a computer system. The various representational data models are hierarchical, network, relational, object-based, and semistructured data model.
20. Physical data model describes the data in terms of a collection of files, indices, and other storage structures such as record formats, record ordering, and access paths.
21. The overall design or description of the database is known as database schema or simply schema. The database schema is also known as intension of the database, and is specified while designing the database. The data in the database at a particular point of time is known as database instance or database state or snapshot. The database state is also called an extension of the schema.
22. To provide various facilities to different types of users, a DBMS normally provides one or more specialized programming languages often called Database (or DBMS) languages.
23. The DBMS mainly provides two database languages, namely, data definition language and data manipulation language to implement

the databases.

24. Data definition language (DDL) is used for defining the database schema. The DBMS comprises DDL compiler that identifies and stores the schema description in the DBMS catalog. Data manipulation language (DML) is used to manipulate the database.
25. A database system being a complex software system is partitioned into several software components that handle various tasks such as data definition and manipulation, security and data integrity, data recovery and concurrency control, and performance optimization.
26. In client/server architecture, the client is the computer system that sends request to the server connected to the network and the server is a computer system that receives the request, processes it, and returns the requested information to the client.
27. There are two approaches of implementing client/server architectures, namely, two-tier architecture and three-tier architecture.
28. The DBMS can be classified into different categories on the basis of several criteria, namely, data models, number of users, number of sites, and purpose.

KEY TERMS

- Database
- Database management system (DBMS)
- File processing system
- Data redundancy
- Data inconsistency
- Centralized data management
- Program-data independence
- DBMS catalog
- Metadata
- Data abstraction
- View
- Database users
- Naive users

- Sophisticated users
- Specialized users
- System analysts
- Application programmers
- Database administrator (DBA)
- Database schema
- Three-level DBMS architecture (ANSI-SPARC architecture)
- Internal level
- Conceptual level
- External level
- Internal schema
- Conceptual schema
- External schema
- Mapping
- Data independence
- Logical data independence
- Physical data independence
- Data model
- Conceptual data model
- Representational data model
- Hierarchical data model
- Network data model
- Relational data model
- Object-based data model
- Semistructured data model
- Physical data model
- Database instance
- Schema evolution
- Data definition language (DDL)
- Storage definition language (SDL)
- View definition language (VDL)
- Data dictionary
- Data manipulation language (DML)
- Query

- Query language
- Client/server architecture
- Single-user system
- Multi-user system
- Distributed database system
- General purpose system
- Specific purpose system
- Online transaction processing (OLTP) system
- Design methodology

EXERCISES

A. Multiple Choice Questions

1. Which of these is not an advantage of database systems?
    1. Data redundancy
    2. Program-data independence
    3. Centralized data management
    4. Data abstraction
2. Which type of users query and update the database by invoking some already written application programs?
    1. Sophisticated users
    2. Naive users
    3. Special users
    4. System analysts
3. Which of these individuals plays an important role in defining and maintaining a database for an organization?
    1. System analyst
    2. Application programmer
    3. Database administrator
    4. All of these
4. Which of these levels deals with the physical representation of the database on the computer?
    1. External level
    2. Conceptual level

3. Internal level
4. None of these

5. The ability to change the conceptual schema without affecting the external schemas or application programs is known as _____.
    1. Program-data independence
    2. Logical data independence
    3. Physical data independence
    4. Data abstraction
6. Which of these is not a representational data model?
    1. Entity-relationship model
    2. Hierarchical data model
    3. Relational data model
    4. Network data model
7. Which of these is not a feature of Hierarchical model?
    1. Organizes the data in tree-like structure
    2. Parent node can have any number of child nodes
    3. Root node does not have any parent
    4. Child node can have any number of parent nodes
8. Which of these data models is an extension of relational data model?
    1. Object-oriented data model
    2. Object-relational data model
    3. Semistructured data model
    4. None of these
9. The information about data in a database is called _____.
    1. Metadata
    2. Hyper data
    3. Tera data
    4. None of these
10. Which of these DBMS languages is employed by end users and programmers to manipulate data in the database?
    1. Data definition language
    2. Data presentation language
    3. Data manipulation language

4. Data translation language

B. Fill in the Blanks

1. The overall structure of the database is known as _____.
2. A _____ is an integrated set of programs used to create and maintain a database.
3. A _____ is a subset of the database that contains virtual data derived from the database files but it does not exist in physical form.
4. The process of transforming the requests and results between various levels of DBMS architecture is known as _____.
5. Data independence is of two types, namely, _____ and _____.
6. The process of highlighting only the essential features and hiding the storage and data organization details from the user is known as _____.
7. The process of making any changes in the schema is known as_____.
8. A _____ is a computer system that sends request to the server connected to the network, and _____ is a computer system that receives the request, processes it, and returns the requested information back to the client.
9. Depending on the number of users the DBMS supports, it is divided into two categories, namely, _____ and _____.
10. In a _____ distributed database system, all sites have identical database management system software.

C. Answer the Questions

1. Define the following terms:
    1. Database
    2. Database management system
    3. Database system
    4. Database schema
    5. Database instance
    6. Database state

7. Database languages
8. DBA
9. Client/server architecture
10. View
11. Mapping
12. Data abstraction
13. Application server
14. Database server

2. Discuss the disadvantages of file processing system that led to the development of database system.
3. What are the advantages of a database system? What are the various cost and risk factors involved in implementing a database system?
4. What do you understand by the term data abstraction?
5. Discuss the various areas in which database system is used.
6. Who is DBA? List the various responsibilities of DBA.
7. What are the different types of database users who interact with the database system?
8. Explain the three-level architecture of DBMS with the help of an example. Mention its advantages also.
9. What is the difference between logical and physical data independence?
10. Illustrate the differences between hierarchical and network data models. Explain why relational model is preferred over the two.
11. What is the difference between object-based data models and record-based data models?
12. What are the roles of system analyst and application programmer in database system?
13. Write a short note on data definition language and data manipulation language.
14. Explain the various functional components of a DBMS with the help of a suitable diagram.
15. What is the difference between a centralized and a client/server database system?

16. How is a two-tier architecture different from a three-tier architecture?
17. Explain the different criteria on the basis of which DBMS is classified into different categories.
18. What is the aim of designing a database? Explain the overall database design and implementation process.