Chapter 1 Introduction:

1.1 Concept and Applications:

Data: Collection of numbers, characters/ Data are values of qualitative or quantitative variables, belonging to a set of items.

Database: A database is an organized collection of data. The data are typically organized to model relevant aspects of reality in a way that supports processing requiring this information. For example, modeling the availability of rooms in hotels in a way that supports finding a hotel with vacancies.

Database Management System: A Database Management System is a collection of interrelated data and a set of programs to access those data. The main purpose of DBMS is to provide a way to store and retrieve database information that is both convenient and efficient. Management of data involves both defining structures for storage of information and providing mechanisms for the manipulation of information. In addition, the database system must ensure the safety of the information stored, despite system crashes or attempts at unauthorized access.

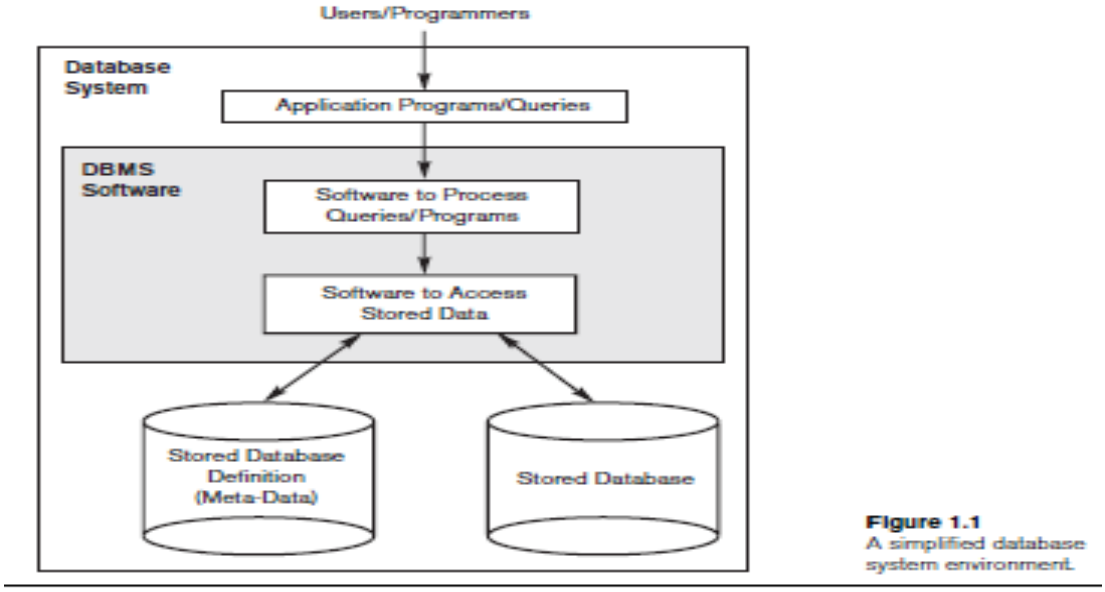

Figure 1.1
A simplified database system environment.

Applications of Database System:

Databases are widely used. Here are some representative applications:
• *Banking*: For customer information, accounts, and loans, and banking transactions.
• *Airlines*: For reservations and schedule information. Airlines were among the first to use databases in a geographically distributed manner—terminals situated around the world accessed the central database system through phone lines and other data networks.
• *Universities*: For student information, course registrations, and grades.
• *Credit card transactions*: For purchases on credit cards and generation of monthly statements.

• *Telecommunication*: For keeping records of calls made, generating monthly bills, maintaining balances on prepaid calling cards, and storing information about the communication networks.
• *Finance*: For storing information about holdings, sales, and purchases of financial instruments such as stocks and bonds.
• *Sales*: For customer, product, and purchase information.
• *Manufacturing*: For management of supply chain and for tracking production of items in factories, inventories of items in warehouses/stores, and orders for items.
• *Human resources*: For information about employees, salaries, payroll taxes and benefits, and for generation of paychecks.

1.2 Objectives of DBMS:

The objectives that the management should keep in mind when they design and organize their data base management systems are:
(i) Provide for mass storage of relevant data,
(ii) Make access to the data easy for the user,
(iii) Provide prompt response to user requests for data,
(iv) Make the latest modifications to the database available immediately,
(v) Eliminate redundant data,
(vi) Allow for multiple users to be active at one time,
(vii) Allow for growth in the database system,
(viii) Protect the data from physical harm and unauthorised access.

Evolution of DBMS:

- Over the course of the last four decades of the twentieth century, use of databases grew in all enterprises. In the early days, very few people interacted directly with database systems, although without realizing it they interacted with databases indirectly— through printed reports such as credit card statements, or through agents such as bank tellers and airline reservation agents.
- Then automated teller machines came along and let users interact directly with databases. Phone interfaces to computers (interactive voice response systems) also allowed users to deal directly with databases—a caller could dial a number, and press phone keys to enter information or to select alternative options, to find flight arrival/departure times, for example, or to register for courses in a university.
- The internet revolution of the late 1990s sharply increased direct user access to databases. Organizations converted many of their phone interfaces to databases into Web interfaces, and made a variety of services and information available online. For instance, when you access an online bookstore and browse a book or music collection, you are accessing data stored in a database.
- Moreover, now there Multimedia databases which can store pictures, video chips and sound messages. Geographic Information Systems (GIS) to store and analyze maps, weather data and satellite image. Data Ware house and Online Analytic Processing (OLAP) systems are used in many companies to extract and analyze useful information for decision making.

1.3 Needs of DBMS:

a) The ability to update and retrieve data :
This is a fundamental component of a DBMS and essential to database management. Without the ability to view or manipulate data, there would be no point to using a database system.

Updating data in a database includes adding new records, deleting existing records and changing information within a record. The user does not need to be aware of how DBMS structures this data, all the user needs to be aware of is the availability of updating and/or pulling up information, the DBMS handles the processes and the structure of the data on a disk.

b) Support Concurrent Updates :
Concurrent updates occur when multiple users make updates to the database simultaneously. Supporting concurrent updates is also crucial to database management as this component ensures that updates are made correctly and the end result is accurate. Without DBMS intervention, important data could be lost and/or inaccurate data stored.
DBMS uses features to support concurrent updates such as batch processing, locking, two-phase locking, and time stamping to help make certain that updates are done accurately. Again, the user is not aware all this is happening as it is the database management system's responsibility to make sure all updates are stored properly.

c) Recovery of Data :
In the event a catastrophe occurs, DBMS must provide ways to recover a database so that data is not permanently lost. There are times computers may crash, a fire or other natural disaster may occur, or a user may enter incorrect information invalidating or making records inconsistent.
If the database is destroyed or damaged in any way, the DBMS must be able to recover the correct state of the database, and this process is called Recovery. The easiest way to do this is to make regular backups of information. This can be done at a set structured time so in the event a disaster occurs, the database can be restored to the state that it was last at prior to backup

1.4 Data Abstraction:

For the system to be usable, it must retrieve data efficiently. The need for efficiency has led designers to use complex data structures to represent data in the database. Since many database-systems users are not computer trained, developers hide the complexity from users through several levels of abstraction, to simplify users' interactions with the system:
• **Physical level**. The lowest level of abstraction describes *how* the data are actually stored. The physical level describes complex low-level data structures in detail.
• **Logical level**. The next-higher level of abstraction describes *what* data are stored in the database, and what relationships exist among those data. The logical level thus describes the entire database in terms of a small number of relatively simple structures. Although implementation of the simple structures at the logical level may involve complex physical-level structures, the user of the logical level does not need to be aware of this complexity. Database administrators, who must decide what information to keep in the database, use the logical level of abstraction.
• **View level**. The highest level of abstraction describes only part of the entire database. Even though the logical level uses simpler structures, complexity remains because of the variety of information stored in a large database. Many users of the database system do not need all this information; instead, they need to access only a part of the database. The view level of abstraction exists to simplify their interaction with the system. The system may provide many views for the same database.
Figure 1.1 shows the relationship among the three levels of abstraction.

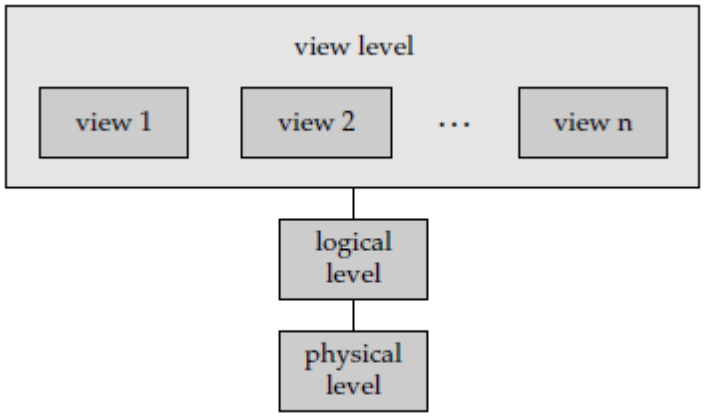

Figure 1.1 The three levels of data abstraction.

1.5 Data Independence:

IT can be defined as the capacity to change the schema at one level of a database system without having to change schema at the next higher level. There are mainly two types of Data Independence

1. Logical data independence: The ability to change the logical (conceptual) schema without changing the External schema (User View) is called logical data independence. For example, the addition or removal of new entities, attributes, or relationships to the conceptual schema should be possible without having to change existing external schemas or having to rewrite existing application programs.
2. Physical data independence: The ability to change the physical schema without changing the logical schema is called physical data independence. For example, a change to the internal schema, such as using different file organization or storage structures, storage devices, or indexing strategy, should be possible without having to change the conceptual or external schemas.

1.6 Instances and Schemas:

Databases change over time as information is inserted and deleted. The collection of information stored in the database at a particular moment is called an **instance** of the database. The overall design of the database is called the database **schema**. Schemas are changed infrequently, if at all. The concept of database schemas and instances can be understood by analogy to a program written in a programming language. A database schema corresponds to the variable declarations (along with associated type definitions) in a program. Each variable has a particular value at a given instant. The values of the variables in a program at a point in time correspond to an *instance* of a database schema. Database systems have several schemas, partitioned according to the levels of abstraction. The **physical schema** describes the database design at the physical level, while the **logical schema** describes the database design at the logical level. A database may also have several schemas at the view level, sometimes called **subschemas**, that describe different views of the database. Of these, the logical schema is by far the most important, in terms of its effect on application programs, since programmers construct applications by using the logical schema. The physical schema is hidden beneath the logical schema, and can usually be changed easily without affecting application programs. Application programs are said to exhibit **physical data independence** if they do not depend on the physical schema, and thus need not be rewritten if the physical schema changes.

1.7 Database Languages(Concept of DDL, DML, DCL)

A database system provides a **data definition language** to specify the database schema and a **data manipulation language** to express database queries and updates.

1.7.1 Data-Definition Language

We specify a database schema by a set of definitions expressed by a special language called a **data-definition language** (**DDL**).

For instance, the following statement in the SQL language defines the *account* table:

create table *account* (*account-number* **char**(10), *balance* **integer**)

Execution of the above DDL statement creates the *account* table. In addition, it updates a special set of tables called the **data dictionary** or **data directory**.

A data dictionary contains **metadata**—that is, data about data. The schema of a table is an example of metadata. A database system consults the data dictionary before reading or modifying actual data.

1.7.2 Data-Manipulation Language

Data manipulation is

- The retrieval of information stored in the database
- The insertion of new information into the database
- The deletion of information from the database
- The modification of information stored in the database

A **data-manipulation language (DML)** is a language that enables users to access or manipulate data as organized by the appropriate data model. There are basically two types:

- **Procedural DMLs** require a user to specify *what* data are needed and *how* to get those data.
- **Declarative DMLs** (also referred to as **nonprocedural** DMLs) require a user to specify *what* data are needed *without* specifying how to get those data.

The DML component of the SQL language is nonprocedural.

A **query** is a statement requesting the retrieval of information. The portion of a DML that involves information retrieval is called a **query language**. Although technically incorrect, it is common practice to use the terms *query language* and *data-manipulation language* synonymously.

This query in the SQL language finds the name of the customer whose customer-id is 192-83-7465:

select *customer.customer-name*
from *customer*
where *customer.customer-id* = 192-83-7465

1.7.3 Data Control Language(DCL): DCL languages are used to control the user access to the database, tables, views, procedures, functions and packages. They give different levels of access to the objects in the database. Some Examples:

Grant- gives user's access privileges to database.
Revoke- Withdraw access privileges given with GRANT command.

1.8 Database Users and Administrators

People who work with a database can be categorized as database users or database administrators.

1.8.1 Database Users and User Interfaces

There are four different types of database-system users, differentiated by the way they expect to interact with the system. Different types of user interfaces have been designed for the different types of users.

• **Naive users** are unsophisticated users who interact with the system by invoking one of the application programs that have been written previously. For example, a bank teller who needs to transfer $50 from account A to account B invokes a program called transfer. This program asks the teller for the amount of

money to be transferred, the account from which the money is to be transferred, and the account to which the money is to be transferred. Naive users may also simply read *reports* generated from the database.

• **Application programmers** are computer professionals who write application programs. Application programmers can choose from many tools to develop user interfaces. **Rapid application development (RAD)** tools are tools that enable an application programmer to construct forms and reports without writing a program.

• **Sophisticated users** interact with the system without writing programs. Instead, they form their requests in a database query language. They submit each such query to a **query processor**, whose function is to break down DML statements into instructions that the storage manager understands. Analysts who submit queries to explore data in the database fall in this category. **Online analytical processing (OLAP)** tools simplify analysts' tasks by letting them view summaries of data in different ways.

• **Specialized users** are sophisticated users who write specialized database applications that do not fit into the traditional data-processing framework. Among these applications are computer-aided design systems, knowledge-base and expert systems, systems that store data with complex data types (for example, graphics data and audio data), and environment-modeling systems.

1.8.2 Database Administrator(Manager)

A person having the central control over the system is called a **database administrator** (**DBA**) or **database manager**. The functions of a DBA include:

- **Schema definition**. The DBA creates the original database schema by executing a set of data definition statements in the DDL.
- **Storage structure and access-method definition**.
- **Schema and physical-organization modification**. The DBA carries out changes to the schema and physical organization to reflect the changing needs of the organization, or to alter the physical organization to improve performance.
- **Granting of authorization for data access**. By granting different types of authorization, the database administrator can regulate which parts of the database various users can access. The authorization information is kept in a special system structure that the database system consults whenever someone attempts to access the data in the system.
- **Routine maintenance**. Examples of the database administrator's routine maintenance activities are:
  - Periodically backing up the database, either onto tapes or onto remote servers, to prevent loss of data in case of disasters such as flooding.
  - Ensuring that enough free disk space is available for normal operations, and upgrading disk Space as required.
  - Monitoring jobs running on the database and ensuring that performance is not degraded by very expensive tasks submitted by some users.