**UNIT-1**

**Introduction to Database**

**CHAPTER 1**

# 1.1 Introduction

Database is a collection of related data. Database management system is software designed to assist the maintenance and utilization of large scale collection of data. DBMS came into existence in 1960 by Charles. Integrated data store which is also called as the first general purpose DBMS. Again in 1960 IBM brought IMS-Information management system. In 1970 EdgorCodd at IBM came with new database called RDBMS. In 1980 then came SQL Architecture- Structure Query Language. In 1980 to 1990 there were advances in DBMS e.g.

DB2, ORACLE.

# Data

* Data is raw fact or figures or entity.
* When activities in the organization takes place, the effect of these activities need to be recorded which is known as Data.

# Information

* Processed data is called information
* The purpose of data processing is to generate the information required for carrying out the business activities.

# In general data management consists of following tasks

* Data capture: Which is the task associated with gathering the data as and when they originate.
* Data classification: Captured data has to be classified based on the nature and intended usage.
* Data storage: The segregated data has to be stored properly.
* Data arranging: It is very important to arrange the data properly
* Data retrieval: Data will be required frequently for further processing,

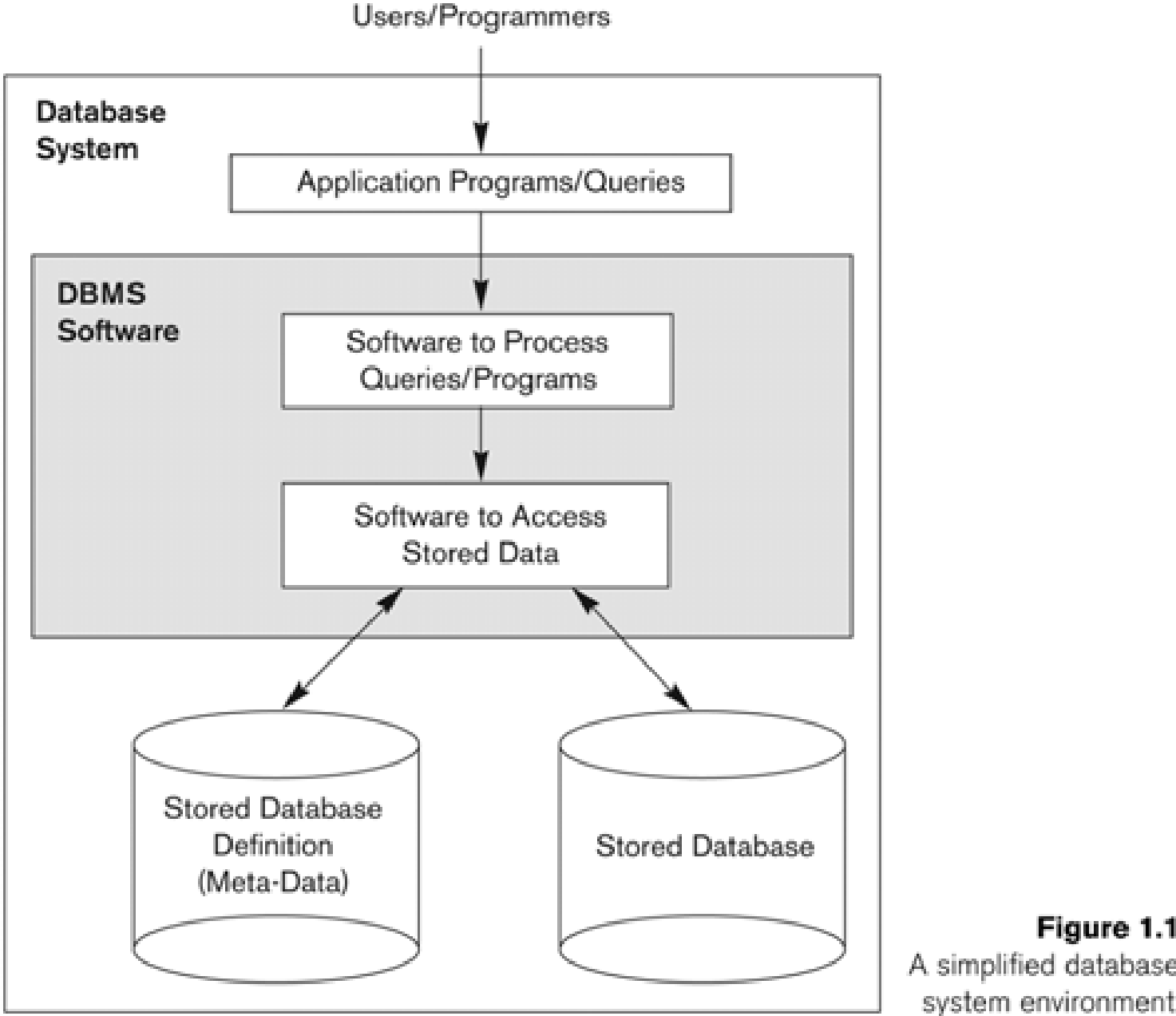
Hence it is very important to create some indexes so that data can be retrieved easily.

* Data maintenance: Maintenance is the task concerned with keeping the data upto- date.
* Data Verification: Before storing the data it must be verified for any error.
* Data Coding: Data will be coded for easy reference.
* Data Editing: Editing means re-arranging the data or modifying the data for presentation.
* Data transcription: This is the activity where the data is converted from one form into another.
* Data transmission: This is a function where data is forwarded to the place where it would be used further.

**Basic Definitions**

* **Database**: A collection of related data.
* **Data**: Known facts that can be recorded and have an implicit meaning
* **Metadata** (**meta data**, or sometimes **meta information**) is "data about data", of any sort in any media. An item of metadata may describe a collection of data including multiple content items and hierarchical levels, for example a database schema.
* **Mini-world or Universe of discourse(UoD)**: Some part of the real world about which data is stored in a database. For example, student grades and transcripts at a university.
* **Database Management System (DBMS)**: A collection of programs that enables users to create and maintain database. DBMS is a general purpose software system that facilitates the processes of defining, constructing, manipulating and sharing databases among various users and applications.
* **Database System**: The DBMS software together with the database. Sometimes, the applications are also included.

## Simplified Database System Environment



A database management system (DBMS) is a collection of programs that enables users to create and maintain database. The DBMS is a general purpose software system that facilitates the process of defining, constructing, manipulating and sharing databases among various users and applications.

Defining a database specifying the database involves specifying the data types, constraints and structures of the data to be stored in the database. The descriptive information is also stored in the database in the form database catalog or dictionary; it is called meta-data. Manipulating the data includes the querrying the database to retrieve the specific data. An application program accesses the database by sending the qurries or requests for data to DBMS.

The important function provided by the DBMS includes protecting the database and maintain the database.

## Example of a Database (with a Conceptual Data Model)

* **Mini-world for the example:**

Part of a UNIVERSITY environment.

* **Some mini-world *entities:***

STUDENTs

COURSEs

SECTIONs (of COURSEs)

(academic) DEPARTMENTs

INSTRUCTORs

# Example of a Database (with a Conceptual Data Model)

• **Some mini-world *relationships:***

SECTIONs *are of specific COURSEs*

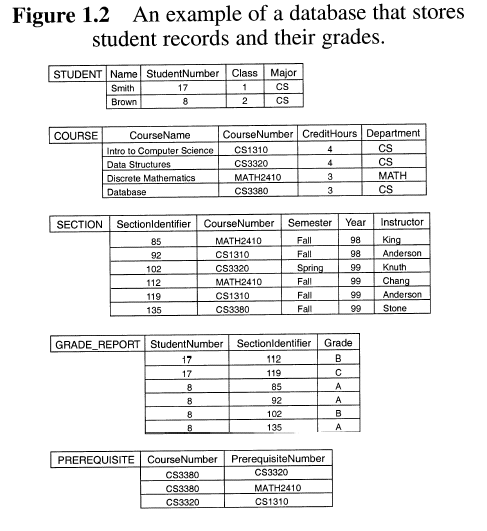
STUDENTs *take SECTIONs*

COURSEs *have prerequisite COURSEs*

INSTRUCTORs *teach SECTIONs*

COURSEs *are offered by DEPARTMENTs*

STUDENTs *major in DEPARTMENTs*



## 

## 1.2 Characteristics of Database approach

* **Self-describing nature of a database system:**
  + A DBMS **catalog** stores the *description* of the database. The description is called **meta-data**. This allows the DBMS software to work with different databases.
* **Insulation between programs and data:**
  + **Program – data independence**
    - Allows changing data storage structures and operations without having to change the DBMS access programs.
  + **Program – operation independence**
    - User application programs can operate on the data by invoking operations through their names and parameters regardless how operations are implemented.
* **Data Abstraction:** 
  + The characteristics that allows program – data independence and program – operation independence is called data abstraction.
  + A **data model** is used to hide storage details and present the users with a *conceptual view* of the database.
* **Support of multiple views of the data:** 
  + Each user may see a different view of the database, which describes *only* the data of interest to that user.
* **Sharing of data and multiuser transaction processing:**
  + Allowing a set of concurrent users to retrieve and to update the database. Concurrency control within the DBMS guarantees that each **transaction** is correctly executed or completely aborted.
  + OLTP (Online Transaction Processing) is a major part of database applications.

**1.3 Advantages of using the DBMS approach**

* **Controlling redundancy in data storage and in development and maintenance efforts.**

Redundancy in storing the same data multiple times leads to several problems like duplications of effort, storage space is wasted, data inconsistency. Therefore, it is necessary to use controlled redundancy to improve the performance of queries.

* **Restricting unauthorized access to data.**

When multiple users share a large database, it is likely that most users will not be authorized to access all information in the database. DBMS should provide a security and authorization subsystem, which the DBA uses to create accounts and to specify account restrictions.

* **Providing persistent storage for program Objects**

Databases can be used to provide persistent storage for program objects and data structures.A complex object in C++ can be stored permanently in an object-oriented DBMS. Such an object is said to be persistent, since it survives the termination of program execution and can later be directly retrieved by another program.

* **Providing Storage Structures for efficient Query Processing**

DBMS must provide specialized data structures and search techniques to speed up disk search for the desired records. Auxiliary files called indexes are often used for this purpose.

* **Providing backup and recovery services.**

A DBMS must provide facilities for recovering from hardware or software failures. The backup and recovery subsystem of the DBMS is responsible for recovery.

* **Providing multiple interfaces to different classes of users.**

Because many types of users with varying levels of technical knowledge use a database, a DBMS should provide a variety of user interfaces. These include apps for mobile users, query languages for casual users etc.

* **Representing complex relationships among data.**

DBMS must have the capability to represent a variety of complex relationships among the data, to define new relationships as they arise, and to retrieve and update related data easily and efficiently.

* **Enforcing integrity constraints on the database.**

Most database applications have certain integrity constraints that must hold for the data such as referral integrity constraint, key or uniqueness constrains etc.

A DBMS should provide capabilities for defining and enforcing these constraints.

* **Permitting Inferencing and Actions using rules**

Some database systems provide capabilities for defining deduction rules for inferencing new information from the stored database facts. Such systems are called deductive database systems

* **Additional Implications of Using the Database Approach**
  + **Potential for enforcing standards:**

This is very crucial for the success of database applications in large organizations Standards refer to data item names, display formats, screens, report structures, meta-data (description of data) etc.

* + **Reduced application development time:**

Incremental time to add each new application is reduced.

* **Flexibility to change data structures:**

Database structure may evolve as new requirements are defined.

* **Availability of up-to-date information:**

Very important for on-line transaction systems such as airline, hotel, car reservations.

* **Economies of scale:**

By consolidating data and applications across departments wasteful overlap of resources and personnel can be avoided.

# 1.4 When not to use a DBMS

* **Main inhibitors (costs) of using a DBMS:**
* High initial investment and possible need for additional hardware.
* Overhead for providing generality, security, concurrency control, recovery, and integrity functions .
* **When a DBMS may be unnecessary:**
* If the database and applications are simple, well defined and not expected to change.
* If there are stringent real-time requirements that may not be met because of DBMS overhead.
* If access to data by multiple users is not required.
* **When no DBMS may suffice:**
* If the database system is not able to handle the complexity of data because of modeling limitations.
* If the database users need special operations not supported by the DBMS.
* **Database Users**

## Users may be divided into those who actually use and control the content (called “Actors on the Scene”) and those who enable the database to be developed and the DBMS software to be designed and implemented (called “Workers Behind the Scene”).

**1.5 Actors on the scene**

**Database administrators:** Responsible for authorizing access to the database, for co – ordinating and monitoring its use, acquiring software, and hardware resources, controlling its use and monitoring efficiency of operations.

**Database Designers:** Responsible to define the content, the structure, the constraints, and functions or transactions against the database. They must communicate with the end-users and understand their needs.

**End-users:** They use the data for queries, reports and some of them actually update the database content.

**Categories of End-users:**

* **Casual** : Access database occasionally when needed
* **Naive or Parametric** : They make up a large section of the end-user population. They use previously well-defined functions in the form of “canned transactions” against the database. Examples are bank-tellers or reservation clerks who do this activity for an entire shift of operations.
* **Sophisticated** : These include business analysts, scientists, engineers, others thoroughly familiar with the system capabilities. Many use tools in the form of software packages that work closely with the stored database.
* **Stand-alone** : Mostly maintain personal databases using ready-to-use packaged applications. An example is a tax program user that creates his or her own internal database.

**System Analysts and Application Programmers(Software Engineers):**

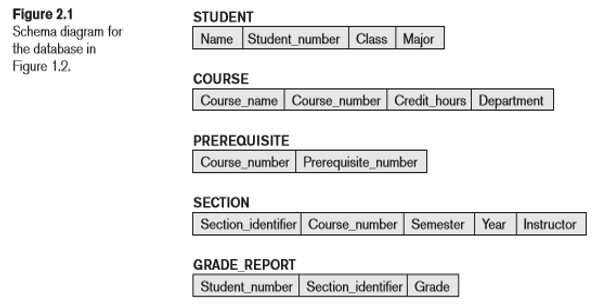
* System Analyst determine the requirements of end users, especially Naïve and parametric end users and develop specifications for canned transactions that meet these equirements.
* Application Programmers implement these specifications as programs, then they test, debug, document and maintain these canned transactions.

**1.6 Workers behind the scene**

* **DBMS System designer and implementers:** Design and implement the DBMS modules and interfaces as a software package. It includes: catlogs, query processing, interfaces, accessing data, data recovery and security.
* **Tool developers:** Designs and implements tools. They include packages for database design, performance monitoring, simulations, test data generation.
* **Operators and maintenance personnel:** Are responsible for actually running and maintenance of the hardware and software environment of the database system.

## NOTE : Schemas, Instances, and Database State

One must distinguish between the *description* of a database and the database itself. The former is called the **database schema**, which is specified during design and is not expected to change often.



The actual data stored in the database probably changes often. The data in the database at a particular time is called the **state** of the database, or a **snapshot**.

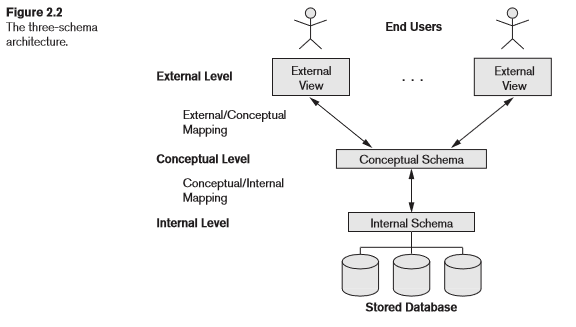
Application requirements change occasionally, which is one of the reasons why software maintenance is important. On such occasions, a change to a database's schema may be called for. An example would be to add a Date\_of\_Birth field/attribute to the STUDENT table. Making changes to a database schema is known as **schema evolution**. Most modern DBMS's support schema evolution operations that can be applied while a database is operational.

The DBMS is partly responsible for ensuring that every state of the database is a **valid state**—that is, a state that satisfies the structure and constraints specified in the schema.

## 1.7 Three-Schema Architecture and Data Independence

**Three-Schema Architecture**: This idea was first described by the ANSI/SPARC committee in late 1970's. The goal is to separate (i.e., insert layers of "insulation" between) user applications and the physical database. C.J. Date points out that it is an ideal that few, if any, real-life DBMS's achieve fully.

* **Internal Level**: has an internal/physical schema that describes the physical storage structure of the database using a low-level data model)
* **Conceptual Level**: has a conceptual schema describing the (logical) structure of the whole database for a community of users. It hides physical storage details, concentrating upon describing entities, data types, relationships, user operations, and constraints. Can be described using either high-level or implementational data model.
* **External/view Level**: includes a number of external schemas (or user views), each of which describes part of the database that a particular category of users is interested in, hiding rest of database. Can be described using either high-level or implementational data model. (In practice, usually described using same model as is the conceptual schema.)



Users (including application programs) submit queries that are expressed with respect to the external level. It is the responsibility of the DBMS to **transform** such a query into one that is expressed with respect to the internal level (and to transform the result, which is at the internal level, into its equivalent at the external level).

Example: Select students with GPA > 1.15.

Q: How is this accomplished?

A: By virtue of **mappings** between the levels:

* **external/conceptual** mapping (providing **logical** data independence)
* **conceptual/internal** mapping (providing **physical** data independence)

### DataIndependence

Data independence can be defined as the capacity to change the schema at one level without changing the schema at next higher level. There are two types of data Independence. They are

1. Logical data independence. 2. Physical data independence.

1. **Logical data independence** is the capacity to change the conceptual schema without having to change the external schema.
2. **Physical data independence** is the capacity to change the internal schema without changing the conceptual schema.

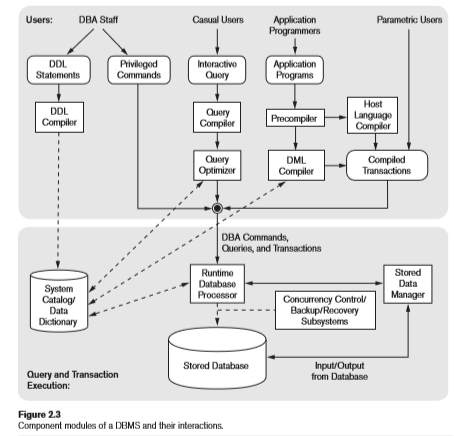
For an **example of physical data independence**, suppose that the internal schema is modified (because we decide to add a new index, or change the encoding scheme used in representing some field's value, or stipulate that some previously unordered file must be ordered by a particular field). Then we can change the mapping between the conceptual and internal schemas in order to avoid changing the conceptual schema itself. Not surprisingly, the process of transforming data via mappings can be costly (performance wise), which is probably one reason that real-life DBMS's don't fully implement this 3-schema architecture.

**2.4 The Database System Environment- DBMS ARCHITECTURE**

**2.4.1 DBMS Component Modules**

Figure 2.3 illustrates, in a simplified form, the typical DBMS components. The figure is divided into two parts. The top part of the figure refers to the various users of the database environment and their interfaces. The lower part shows the internal modules of the DBMS responsible for storage of data and processing of transactions.

* The database and the DBMS catalog are usually stored on disk. Access to the disk is controlled primarily by the operating system (OS), which schedules disk read/write.
* A higher-level stored data manager module of the DBMS controls access to DBMS information that is stored on disk, whether it is part of the database or the catalog.
* The DBA staff works on defining the database and tuning it by making changes to its definition using the DDL and other privileged commands
* The DDL compiler processes schema definitions, specified in the DDL, and stores descriptions of the schemas (meta-data) in the DBMS catalog.



* Casual users and persons with occasional need for information from the database interact using the interactive query interface in Figure 2.3.
* These queries are parsed and validated for correctness of the query syntax, the names of files and data elements, and so on by a query compiler that compiles them into an internal form. This internal query is subjected to query optimization.
* The query optimizer is concerned with the rearrangement and possible reordering of operations, elimination of redundancies, and use of efficient search algorithms during execution.
* Application programmers write programs in host languages such as Java, C, or C++ that are submitted to a precompiler. The precompiler extracts DML commands from an application program written in a host programming language. These commands are sent to the DML compiler for compilation into object code for database access.
* In the lower part of Figure 2.3, the runtime database processor executes (1) the privileged commands, (2) the executable query plans, and (3) the canned transactions with runtime parameters. It works with the system catalog and may update it with statistics. It also works with the stored data manager, which in turn uses basic operating system services for carrying out low-level input/output (read/write) operations between the disk and main memory.
* Concurrency control and backup and recovery systems are integrated into the working of the runtime database processor for purposes of transaction management.
* client program that accesses the DBMS running on a separate computer or device from the computer on which the database resides. The former is called the client computer running DBMS client software and the latter is called the database server. In many cases, the client accesses a middle computer, called the application server, which in turn accesses the database server.

The DBMS interacts with the operating system when disk accesses— to the database or to the catalog—are needed. If the computer system is shared by many users, the OS will schedule DBMS disk access requests and DBMS processing along with other processes. On the other hand, if the computer system is mainly dedicated to running the database server, the DBMS will control main memory buffering of disk pages. The DBMS also interfaces with compilers for general- purpose host programming languages, and with application servers and client programs running on separate machines through the system network interface.