Database:

A database is a collection of related data. By data, we mean known facts that can be recorded and have implicit meaning. For example, **University Database**: Data about students, faculty, courses, research-laboratories, course registration/enrolment etc. **Reflects** the state of affairs of the academic aspects of the university. **Purpose**: To keep an accurate track of the academic activities of the university.

- Representing/capturing the information about a real-world enterprise or part of an enterprise.
- Collected and maintained to serve specific data management needs of the enterprise.
- Activities of the enterprise are supported by the database and continually update the database.

A database may be generated and maintained **manually** or it may be **computerized**. For example, a library card catalogue a database that may be created and maintained manually.

**Database Management System (DBMS):**

It is a collection of programs that enables user to create and maintain a database. The DBMS is hence a general-purpose software system the facilitates the purpose of defining, constructing, manipulating, and the sharing databases among various users and applications. **Defining** a database involves specifying the data types, structures, and the constraints for the data to be stored in the database.

**Construction** the database is the process of storing the data itself on some storage medium that is controlled by the DBMS.

**Manipulating** a database includes such functions as querying the database to retrieve specific data, updating the database to reflect changes in the miniworld, and generating reports from the data.

**Sharing** a database allows multiple users and programs to access the database concurrently.
Characteristics of the Database Approach:

- Self-describing nature of a database system
- Insulation between programs and data, and data abstraction
- Support of multiple views of the data
- Sharing of data and multiuser transaction processing

1) Self-describing Nature of a Database System:

Database system contains not only the database itself but also a complete definition of description of database structure and constraints. This definition is stored in the DBMS catalog, which contains information such as the structure of each file, the type and storage format of each data item, and various constraints on the data. The information
stored in the catalog is called **meta-data**, and it describes the structure of the primary database.

2) **Insulation between programs and data, and data abstraction:**

The structure of the data files is stored in the DBMS catalog separately from the access programs. This property is called program-data independence. For example, a file access system program may be written in such a way that it can access only student records. If we want to add another piece of data to each student record, say the BirthDate, such a program will no longer work and must be changed. In a DBMS environment, we just need to change the description of student records in the catalog to reflect the inclusion of new data item BirthDate; no programs are changed. The next time a DBMS program refer to the catalog, the new structure of student records will be accessed and used.

**Data Abstraction:** The characteristic that allows program-data independence and program-operation independence is called **data abstraction**.

3) **Support of multiple views of the data:**

A database typically has many users, each of whom may require a different perspective or **view** of the database. A view may be a subset of the database or it may contain **virtual data** that is derived from the database files but is not explicitly stored. A multiuser DBMS whose users have a variety of distinct applications must provide facilities for defining multiple views. For example, one user of the database may be interested only in accessing and printing the transcript of each student. A second user, who is interested only in checking that students have taken all the prerequisites of each course for which they register.

4) **Sharing of data and multiuser transaction processing:**

A multiuser DBMS, as its name implies, must allow multiple users to access the database at the same time. This is essential if data for multiple applications is to be integrated and maintained in a single database. The DBMS must include **concurrency control** software to ensure that several users trying to update the same data do so in a controlled manner so that the result of the updates is correct.

For example, when several reservation agents try to assign a seat on an airline flight, the DBMS should ensure that each seat can be accessed by only one agent at a time for assignment to a passenger. These types of applications are generally called **online transaction processing (OLTP)** applications. A fundamental role of multiuser DBMS software is to ensure that concurrent transactions operate correctly and efficiently.
Database System Applications:
Databases are widely used. Some applications of databases are as follows:

- **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 registration, and grades.
- **Credit card transactions**: For purchases on credit cards and generation of monthly statements.
- **Telecommunications**: 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 holding, sales, and purchases of financial instruments such as stocks and bonds.
- **Sales**: For customers, 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 pay checks.

Advantages of using the DBMS approach:

1) **Controlling Redundancy**:
Since different programmers create the files and application programs over a long period, the various files are likely to have different formats and the programs may be written in several programming languages. Moreover, the same information may be duplicated in several places. For example, the address and telephone number of a particular customer may appear in a file that consists of savings-account records and in a file that consists of checking-account records. This redundancy leads to higher storage and access cost. This redundancy in storing the same data multiple times leads to several problems. First, there is the need to perform a single logical update. Second, *storage space is wasted* when the same data is stored repeatedly, and this problem may be serious for large databases. Third, files that represent the same data may become *inconsistent*. 
2) **Restricting Unauthorized Access:**
When multiple users share a large database, it is likely that most users will not be authorized to access all information in the database. For example, financial data is often considered confidential, and only authorized persons are allowed to access such data. In addition, some users may only be permitted to retrieve data, whereas others are allowed to retrieve and update. Hence, the type of access operation - retrieval or update - must also be controlled. A DBMS should provide a **security and authorization subsystem**, which the DBA uses to create accounts and to specify account restrictions.

3) **Providing Storage Structures for Efficient Query Processing:**
Database systems must provide capabilities for *efficiently executing queries and Updates*. The DBMS must provide specialized data structures and search techniques to speed up disk search for the desired records. Auxiliary files called **indexes** are used for this purpose. Indexes are typically based on tree data structures or hash data structures that are suitably modified for disk search. In order to process the database records needed by a particular query, those records must be copied from disk to main memory. Therefore, the DBMS often has a **buffering** or **caching** module that maintains parts of the database in main memory buffers.

4) **Providing Backup and Recovery:**
DBMS must provide facilities for recovering from hardware or software failures. The **backup and recovery subsystem** of the DBMS is responsible for recovery. For example, if the computer system fails in the middle of a complex update transaction, the recovery subsystem is responsible for making sure that the database is restored to the state it was in before the transaction started executing. Alternatively, the recovery subsystem could ensure that the transaction is resumed from the point at which it was interrupted so that its full effect is recorded in the database.

5) **Providing Multiple User Interfaces:**
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 query languages for casual users, programming language interfaces for application programmers, forms and command codes for parametric users, and menu-driven interfaces and natural language interfaces for standalone users.

6) **Representing Complex Relationship among Data:**
A database may include numerous varieties of data that are interrelated in many ways. The record for ‘Mazid’ in the STUDENT file is related to four records in the GRADE_REPORT
A 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.

7) **Enforcing Integrity Constraints:**
Most database applications have certain **integrity constraints** that must hold for the data. A DBMS should provide capabilities for defining and enforcing these constraints. The simplest type of integrity constraint involves specifying a data type for each data item. For example, the value of the Class data item within each STUDENT record must be a one-digit integer and that the value of Name must be a string of no more than 30 alphabetic characters. To restrict the value of Class between 1 and 5 would be an additional constraint.

**Database Schema:**
The *description* of a database. Includes descriptions of the database structure and the constraints that should hold on the database.

**Three-Schema Architecture:**
The three-schema architecture was proposed to help achieve and visualize the above said characteristics of the DBMS such as (a) Self-describing nature of a database system b) Insulation between programs and data, and data abstraction. c) Support of multiple views of the data). The goal of the three-schema architecture, is to separate the user applications from the physical database. In this architecture, schemas can be defined at the following three levels:

![Diagram of the three-schema architecture](image-url)

**Figure: The three-schema architecture**
1) The **external** or **view level** includes a number of **external schemas** or **user views**. Each external schema describes the part of the database that a particular user group is interested in and hides the rest of the database from that user group. As in the previous level, each external schema is typically implemented using a representational data model, possibly based on an external schema design in a high-level data model.

2) The **conceptual level** has a **conceptual schema**, which describes the structure of the whole database for a community of users. The conceptual schema hides the details of physical storage structures and concentrates on describing entities, data types, relationships, user operations, and constraints. Usually, a representational data model is used to describe the conceptual schema when a database system is implemented.

3) The **internal level** has an **internal schema**, which describes the physical storage structure of the database. It describes how data is stored in the database to achieve optimal performance and storage space utilization.
   - Storage allocation for data and indexes
   - Record description for storage
   - Record placement
   - Data compression & encryption techniques

**Notice** that the three schemas are only descriptions of data; the stored data that actually exists is at the physical level only.

**Mappings:**
The processes of transforming requests and results between levels are called **mappings**.