

# DBMS

PANA ACADEMY

# DBMS and advantages

A DBMS is a software system designed to manage databases. It provides an interface between users and the database, allowing users to store, retrieve, and manipulate data efficiently.

Components of DBMS

**Database:** Collection of related data.

**DBMS Engine:** Core component managing data storage, retrieval, and updates.

**Database Schema:** Structure that defines database organization.

**Query Processor:** Translates queries into instructions for the DBMS engine.

**Transaction Management:** Ensures data integrity during transactional operations.

**Security Management:** Controls access to data.

**Backup and Recovery:** Facilitates data backup and restoration.

# Advantages of DBMS compared to File Management

**Data Storage:** DBMS stores data in a centralized database, organized in tables (for relational DBMS) or other structures (for non-relational DBMS like NoSQL). This centralization reduces redundancy and improves data consistency.

**Data Independence:** DBMS provides both physical and logical data independence. Changes to the database schema (logical) can be managed independently from the application programs, reducing maintenance efforts and risks.

**Data Integrity:** DBMS enforces data integrity through constraints (e.g., uniqueness, referential integrity) and transactions. ACID properties (Atomicity, Consistency, Isolation, Durability) ensure reliable and secure transaction processing.

**Security:** DBMS offers robust security features, including access control mechanisms (roles and permissions), encryption, and auditing. Security policies can be centrally managed and enforced across the entire database.

**Concurrency Control:** DBMS manages concurrent access to data using sophisticated concurrency control mechanisms (e.g., locking, multiversion concurrency control). This ensures data consistency and prevents conflicts among concurrent transactions.

**Scalability and Flexibility:** DBMS systems are designed for scalability and flexibility. They can handle large volumes of data and support various data types and models (relational, document-oriented, key-value, etc.). Scaling can be achieved through replication, sharding, or clustering.

**Performance:** DBMS optimizes data access and retrieval through query optimization techniques, indexing, and caching mechanisms, resulting in improved performance compared to file-based systems.

# Data Abstraction and Data Independence

Data abstraction is a fundamental concept in computer science that refers to the process of exposing only the essential features of a complex system while hiding its internal details.

## Three Levels of Data Abstraction

There are three levels of data abstraction in a DBMS:

1. **Physical Level:** This is the lowest level of abstraction, where the physical storage and organization of data are defined. This level is concerned with how the data is stored on disk, how it is accessed, and how it is retrieved.
2. **Logical Level:** This level is concerned with the logical structure of the data, including the relationships between different data entities. This level is responsible for defining the schema of the database, including the tables, fields, and relationships between them.
3. **View Level:** This is the highest level of abstraction, where the data is presented to the user in a simplified and abstracted form. This level is concerned with how the data is presented to the user, including the format and structure of the data.

# Data Independence

Data independence is the ability to modify the physical storage and organization of data without affecting the logical structure of the data. There are two types of data independence.

1. Physical Data Independence: This refers to the ability to modify the physical storage and organization of data without affecting the logical structure of the data. For example, changing the storage device or the file organization of the data does not affect the logical structure of the data.
1. Logical Data Independence: This refers to the ability to modify the logical structure of the data without affecting the physical storage and organization of the data. For example, adding a new field to a table or changing the relationships between tables does not affect the physical storage and organization of the data.






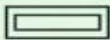
# Schema vs Instance

Schema	Instance
It is the overall description of the database.	It is the collection of information stored in a database at a particular moment.
The schema is same for the whole database.	Data in instances can be changed using addition, deletion, and updation.
Does not change Frequently.	Changes Frequently.
Defines the basic structure of the database i.e. how the data will be stored in the database.	It is the set of Information stored at a particular time.

# ER model

The Entity Relational Model is a model for identifying entities to be represented in the database and representation of how those entities are related.

The ER data model specifies enterprise schema that represents the overall logical structure of a database graphically.

Figures	Symbols	Represents
Rectangle		Entities in ER Model
Ellipse		Attributes in ER Model
Diamond		Relationships among Entities
Line		Attributes to Entities and Entity Sets with Other Relationship Types
Double Ellipse		Multi-Valued Attributes
Double Rectangle		Weak Entity

*Symbols used in ER Diagram*

# Strong and Weak entity sets

## 1. Strong Entity

A [Strong Entity](#) is a type of entity that has a key Attribute. Strong Entity does not depend on other Entity in the Schema. It has a primary key, that helps in identifying it uniquely, and it is represented by a rectangle. These are called Strong Entity Types.

## 2. Weak Entity

An Entity type has a key attribute that uniquely identifies each entity in the entity set. But some entity type exists for which key attributes can't be defined. These are called [Weak Entity types](#).

**For Example**, A company may store the information of dependents (Parents, Children, Spouse) of an Employee. But the dependents can't exist without the employee. So Dependent will be a **Weak Entity Type** and Employee will be Identifying Entity type for Dependent, which means it is **Strong Entity Type**.

A weak entity type is represented by a Double Rectangle. The participation of weak entity types is always total. The relationship between the weak entity type and its identifying strong entity type is called identifying relationship and it is represented by a double diamond.



# Attributes and Keys

A key refers to an attribute/a set of attributes that help us identify a row (or tuple) uniquely in a table (or relation).

Keys are of seven broad types in DBMS:

1. Candidate Key
2. Primary Key
3. Foreign Key
4. Super Key
5. Alternate Key
6. Composite Key
7. Unique Key

# Primary Key

The primary key refers to a column or a set of columns of a table that helps us identify all the records uniquely present in that table.

A table can consist of just one primary key.

The PK (PRIMARY KEY) constraint that we put on a column/set of columns won't allow these to have a null value or a duplicate

# Candidate Key

The candidate keys refer to those attributes that identify rows uniquely in a table. In a table, **we select the primary key from a candidate key**. Thus, a candidate key has similar properties as that of the primary keys that we have explained above. In a table, **there can be multiple candidate keys**.

# Foreign Key

We use a foreign key to establish relationships between two available tables. The foreign key would require every value present in a column/set of columns to match the referential table's primary key. A foreign key helps us to maintain data as well as referential integrity.

# Composite Key

The composite key refers to a set of multiple attributes that help us uniquely identify every tuple present in a table. The attributes present in a set may not be unique whenever we consider them separately. Thus, when we take them all together, it will ensure total uniqueness.

# Normalization

A large database defined as a single relation may result in data duplication. This repetition of data may result in:

- Making relations very large.
  - It isn't easy to maintain and update data as it would involve searching many records in relation.
- 
- Normalization is the process of organizing the data in the database.
  - Normalization is used to minimize the redundancy from a relation or set of relations. It is also used to eliminate undesirable characteristics like Insertion, Update, and Deletion Anomalies.
  - Normalization divides the larger table into smaller and links them using relationships.
  - The normal form is used to reduce redundancy from the database table.

# MCQ Practice

<https://www.sanfoundry.com/database-mcqs-entity-relationship-model/>

<https://www.sanfoundry.com/database-mcqs-enttiy-relationship-diagram/>