**What is Database?**

A **database** is a collection of interrelated data that helps in the efficient retrieval, insertion, and deletion of data from the database and organizes the data in the form of tables, views, schemas, reports,

## **What is DBMS?**

A Database Management System (DBMS) is a software system that is designed to manage and organize data in a structured manner.

**Key Feature of DBMS**

Data modeling,Data store and retrieval,Concurrency control(multiple user access),Data integrity and security,Backup and Recovery

**Data integrity** is a concept and process that ensures the accuracy, completeness, consistency, and validity of an organization’s data. By following the process, organizations not only ensure the integrity of the data but guarantee they have accurate and correct data in their database.

**Types of DBMS**

**1.RDBMS:-**Data is organized into tables (relations) with rows and columns, and the relationships between the data are managed through primary and foreign keys. SQL (Structured Query Language) is used to query and manipulate the data.

**2.NoSQL DBMS:-**Designed for high-performance scenarios and large-scale data, NoSQL databases store data in various non-relational formats such as key-value pairs, documents, graphs, or columns.

**3.Object-Oriented DBMS (OODBMS):-** Stores data as objects, similar to those used in object-oriented programming, allowing for complex data representations and relationships

## **Database Languages**

* Data Definition Language(DDL)
* Data Manipulation Language(DML)
* Data Control Language(DCL)
* Transactional Control Language(TCL)

**Data Definition Language:-**create,alter,drop,truncate,comment,rename

**Data Manipulation Language:-**select,insert,update,delete,merge,call,explain plan,lock table

**Data Control Language:-**Here we control operation of data manipulation language to give permission or revoke permission

**Transaction Control Language:-**Rollback,commit,save point

**Data query language(DQL)** is the subset of **“Data Manipulation Language”:-**Select

## **Paradigm Shift from File System to DBMS**

File System manages data using files on a hard disk. Users are allowed to create, delete, and update the files according to their requirements.

* **Redundancy of data:** Data is said to be redundant if the same data is copied at many places. If a student wants to change their Phone number, he or she has to get it updated in various sections. Similarly, old records must be deleted from all sections representing that student.
* **Inconsistency of Data:** Data is said to be inconsistent if multiple copies of the same data do not match each other. If the Phone number is different in Accounts Section and Academics Section, it will be inconsistent. Inconsistency may be because of typing errors or not updating all copies of the same data.
* **Difficult Data Access:** A user should know the exact location of the file to access data, so the process is very cumbersome and tedious. If the user wants to search the student hostel allotment number of a student from 10000 unsorted students’ records, how difficult it can be.
* **Unauthorized Access:** File Systems may lead to unauthorized access to data. If a student gets access to a file having his marks, he can change it in an unauthorized way.
* **No Concurrent Access:** The access of the same data by multiple users at the same time is known as concurrency. The file system does not allow concurrency as data can be accessed by only one user at a time.
* **No Backup and Recovery:** The file system does not incorporate any backup and recovery of data if a file is lost or corrupted.

**Advantages of DBMS**

Data Organization,Data integrity,Concurrent access,Data security,backup and recovery,Data sharing

**Disadvantages of DBMS**

Complexity,Performance overhead,scalability,cost,limited use case

**Application of DBMS**

Banking,airline,university,Telecommunications,Railway Reservation system,Library Management System

**Types of DBMS Architecture**

* **1-Tier Architecture**
* **2-Tier Architecture**
* [**3-Tier Architecture**](https://www.geeksforgeeks.org/introduction-of-3-tier-architecture-in-dbms-set-2/)

## **1-Tier Architecture**

In 1-Tier Architecture the database is directly available to the user, the user can directly sit on the DBMS and use it that is, the client, server, and Database are all present on the same machine. For Example: to learn SQL we set up an SQL server and the database on the local system. This enables us to directly interact with the relational database and execute operations.

**Advantages**Simple Architecture,Cost efficient,Easy to implement

## **2-Tier Architecture**

The 2-tier architecture is similar to a basic [client-server model](https://www.geeksforgeeks.org/client-server-model/). The application at the client end directly communicates with the database on the server side. APIs like ODBC and JDBC are used for this interaction. The server side is responsible for providing query processing and transaction management functionalities. On the client side, the user interfaces and application programs are run. The application on the client side establishes a connection with the server side to communicate with the DBMS.

**Advantages**

Easy to access,scalable,low cost ,Simple

## **3-Tier Architecture**

In [3-Tier Architecture](https://www.geeksforgeeks.org/introduction-of-3-tier-architecture-in-dbms-set-2/), there is another layer between the client and the server. The client does not directly communicate with the server. Instead, it interacts with an application server which further communicates with the database system and then the query processing and transaction management takes place. This intermediate layer acts as a medium for the exchange of partially processed data between the server and the client.

**Advantages**

Enhanced scalability,data integrity,Security

* **Physical Level:** At the physical level, the information about the location of database objects in the data store is kept. In simple terms,physical level of a database describes how the data is being stored in secondary storage devices like disks and tapes and also gives insights on additional storage details.
* **Conceptual Level:** At conceptual level, data is represented in the form of various database tables. For Example, STUDENT database may contain STUDENT and COURSE tables which will be visible to users but users are unaware of their storage.Also referred as logical schema,it describes what kind of data is to be stored in the database.
* **External Level:**  An external level specifies a view of the data in terms of conceptual level tables. Each external level view is used to cater to the needs of a particular category of users. For Example, FACULTY of a university is interested in looking course details of students, STUDENTS are interested in looking at all details related to academics, accounts, courses and hostel details as well. So, different views can be generated for different users. The main focus of external level is data abstraction.

## **Data Independence**

Data independence means a change of data at one level should not affect another level. Two types of data independence are present in this architecture:

* **Physical Data Independence:** Any change in the physical location of tables and indexes should not affect the conceptual level or external view of data.
* **Conceptual Data Independence:** The data at conceptual level schema and external level schema must be independent. This means a change in conceptual schema should not affect external schema. e.g.; Adding or deleting attributes of a table should not affect the user’s view of the table. But this type of independence is difficult to achieve as compared to physical data independence because the changes in conceptual schema are reflected in the user’s view.

**Phase of Database Design**

### **1. Conceptual Design**

* **Definition**: This is the high-level design that outlines what data will be stored and the relationships among the data without getting into the specifics of how it will be implemented.

### **2. Logical Design**

* **Definition**: This translates the conceptual design into a logical structure that defines the data model, specifying the data types, constraints, and relationships in more detail.

### **3. Physical Design**

* **Definition**: This is the actual implementation of the database on a specific platform, detailing how the data will be stored physically in the database system.

**The main difference between a file system and a DBMS (Database Management System) is the way they organize and manage data.**

1. File systems are used to manage files and directories, and provide basic operations for creating, deleting, renaming, and accessing files. They typically store data in a hierarchical structure, where files are organized in directories and subdirectories. File systems are simple and efficient, but they lack the ability to manage complex data relationships and ensure data consistency.
2. On the other hand, DBMS is a software system designed to manage large amounts of structured data, and provide advanced operations for storing, retrieving, and manipulating data. DBMS provides a centralized and organized way of storing data, which can be accessed and modified by multiple users or applications. DBMS offers advanced features like data validation, [**indexing**](https://www.geeksforgeeks.org/indexing-in-databases-set-1/), transactions, [concurrency control](https://www.geeksforgeeks.org/concurrency-control-in-dbms/), and backup and recovery mechanisms. DBMS ensures data consistency, accuracy, and integrity by enforcing data constraints, such as primary keys, foreign keys, and data types.

## **What is Cardinality?**

The number of times an entity of an entity set participates in a relationship set is known as [cardinality](https://www.geeksforgeeks.org/cardinality-in-dbms/) . Cardinality can be of different types:

**1. One-to-one (1:1)** When each entity in each entity set can take part only once in the relationship, the cardinality is one-to-one. Let us assume that a male can marry one female and a female can marry one male. So the relationship will be one-to-one.

the total number of tables that can be used in this is 2.

**2. One-to-many (1: N)** In one-to-many mapping as well where each entity can be related to more than one entity and the total number of tables that can be used in this is 2. Let us assume that one surgeon department can accommodate many doctors. So the Cardinality will be 1 to M. It means one department has many Doctors.

total number of tables that can used is 3.

**3. Many-to-one (N:1)** When entities in one entity set can take part only once in the relationship set and entities in other entity sets can take part more than once in the relationship set, cardinality is many to one. Let us assume that a student can take only one course but one course can be taken by many students. So the cardinality will be n to 1. It means that for one course there can be n students but for one student, there will be only one course.

The total number of tables that can be used in this is 3.

**4. Many-to-many (N: N)** When entities in all entity sets can take part more than once in the relationship cardinality is many to many. Let us assume that a student can take more than one course and one course can be taken by many students. So the relationship will be many to many.

the total number of tables that can be used in this is 3.

[**Participation Constraint**](https://www.geeksforgeeks.org/structural-constraints-of-relationships-in-er-model/) is applied to the entity participating in the relationship set.

**1. Total Participation –** Each entity in the entity set must participate in the relationship. If each student must enroll in a course, the participation of students will be total. Total participation is shown by a double line in the ER diagram.

**2. Partial Participation –** The entity in the entity set may or may NOT participate in the relationship. If some courses are not enrolled by any of the students, the participation in the course will be partial.

| **Entity** | **Entity Type** | **Entity Set** |
| --- | --- | --- |
| A thing in the real world with independent existence | A category of a particular entity | Set of all entities of a particular entity type. |
| Any particular row (a record) in a relation(table) is known as an entity. | The name of a relation (table) in RDBMS is an entity type | All rows of a relation (table) in RDBMS is entity set |

| **Strong Entity** | **Weak Entity** |
| --- | --- |
| Strong entity always has a [primary key](https://www.geeksforgeeks.org/primary-key-constraint-in-sql/). | While a weak entity has a partial discriminator key. |
| Strong entity is not dependent on any other entity. | Weak entity depends on strong entity. |
| Strong entity is represented by a single rectangle. | Weak entity is represented by a double rectangle. |
| Two strong entity’s relationship is represented by a single diamond. | While the relation between one strong and one weak entity is represented by a double diamond. |
| Strong entities have either total participation or partial participation. | A weak entity has a total participation constraint. |

Using the ER model for bigger data creates a lot of complexity while designing a database model, So in order to minimizethe complexityGeneralization, Specialization, and Aggregation were introduced in the ER model and these were used for data abstraction in which an abstraction mechanism is used to hide details of a set of objects.

* Generalization
* Specialization
* Aggregation

## **Generalization**

Generalization is the process of extracting common properties from a set of entities and creating a generalized entity from it. It is a bottom-up approach in which two or more entities can be generalized to a higher-level entity if they have some attributes in common. For Example, STUDENT and FACULTY can be generalized to a higher-level entity called PERSON

Generalization is also called as ‘ Bottom-up approach”.

## **Specialization**

In specialization, an entity is divided into sub-entities based on its characteristics. It is a top-down approach where the higher-level entity is specialized into two or more lower-level [entities](https://www.geeksforgeeks.org/difference-between-entity-entity-set-and-entity-type/). For Example, an EMPLOYEE entity in an Employee management system can be specialized into DEVELOPER, TESTER, etc.

Specialization is also called as ” Top-Down approach”.

## **Aggregation**

An ER diagram is not capable of representing the relationship between an entity and a relationship which may be required in some scenarios. In those cases, a relationship with its corresponding entities is aggregated into a higher-level entity. Aggregation is an abstraction through which we can represent relationships as higher-level entity sets.

# **Recursive Relationships in ER diagrams**

A relationship between two entities of a similar entity type is called a **recursive** relationship.

## **Terminologies**

**1.** [**Relational Schema:**](https://www.geeksforgeeks.org/relation-schema-in-dbms/) Schema represents the structure of a relation.

**Example:** Relational Schema of STUDENT relation can be represented as STUDENT(STUD\_NO, STUD\_NAME, STUD\_PHONE, STUD\_STATE, STUD\_COUNTRY, STUD\_AGE).

**2.** [**Relational Instance:**](https://www.geeksforgeeks.org/difference-between-schema-and-instance-in-dbms/) The set of values present in a relationship at a particular instance of time is known as a relational instance as shown in Table 1 and Table 2.

**3. Attribute:** Each relation is defined in terms of some properties, each of which is known as an attribute.

**Example:** STUD\_NO, STUD\_NAME, etc. are attributes of relation STUDENT.

**4. The domain of an attribute:** The possible values an attribute can take in a relation is called its domain.

**Example:** domain of STUD\_AGE can be from 18 to 40.

**5. Tuple:** Each row of a relation is known as a tuple.

# **Difference Between DBMS and RDBMS**

| **DBMS** | **RDBMS** |
| --- | --- |
| [DBMS](https://www.geeksforgeeks.org/introduction-of-dbms-database-management-system-set-1/) stores data as file. | [RDBMS](https://www.geeksforgeeks.org/rdbms-architecture/) stores data in tabular form. |
| Data elements need to access individually. | Multiple data elements can be accessed at the same time. |
| No relationship between data. | Data is stored in the form of tables which are related to each other. |
| Normalization is not present. | Normalization is present. |
| DBMS does not support distributed database. | RDBMS supports distributed database. |
| It stores data in either a navigational or hierarchical form. | It uses a tabular structure where the headers are the column names, and the rows contain corresponding values. |
| It deals with small quantity of data. | It deals with large amount of data. |
| Data redundancy is common in this model. | Keys and indexes do not allow Data redundancy. |
| It is used for small organization and deal with small data. | It is used to handle large amount of data. |
| Not all Codd rules are satisfied. | All 12 Codd rules are satisfied. |
| Security is less | More security measures provided. |
| It supports single user. | It supports multiple users. |
| Data fetching is slower for the large amount of data. | Data fetching is fast because of relational approach. |
| The data in a DBMS is subject to low security levels with regards to data manipulation. | There exists multiple levels of data security in a RDBMS. |
| Low software and hardware necessities. | Higher software and hardware necessities. |
| Examples: [XML](https://www.geeksforgeeks.org/xml-basics/), Window Registry, Forxpro, dbaseIIIplus etc. | Examples: [MySQL](https://www.geeksforgeeks.org/architecture-of-mysql/), [PostgreSQL](https://www.geeksforgeeks.org/what-is-postgresql-introduction/), [SQL](https://www.geeksforgeeks.org/what-is-sql/) Server, Oracle, Microsoft Access etc. |

### **Candidate Key**

The minimal set of attributes that can uniquely identify a tuple is known as a candidate key. For Example, STUD\_NO in STUDENT relation.

* It is a minimal super key.
* It is a super key with no repeated data is called a candidate key.
* The minimal set of attributes that can uniquely identify a record.
* It must contain unique values.
* It can contain NULL values.
* Every table must have at least a single candidate key.
* A table can have multiple candidate keys but only one primary key.
* The value of the Candidate Key is unique and may be null for a tuple.
* There can be more than one candidate key in a relationship.

### **Primary Key**

There can be more than one candidate key in relation out of which one can be chosen as the primary key. For Example, STUD\_NO, as well as STUD\_PHONE, are candidate keys for relation STUDENT but STUD\_NO can be chosen as the [primary key](https://www.geeksforgeeks.org/primary-key-in-dbms/) (only one out of many candidate keys).

* It is a unique key.
* It can identify only one tuple (a record) at a time.
* It has no duplicate values, it has unique values.
* It cannot be NULL.
* Primary keys are not necessarily to be a single column; more than one column can also be a primary key for a table.

### **Super Key**

The set of attributes that can uniquely identify a tuple is known as Super Key. For Example, STUD\_NO, (STUD\_NO, STUD\_NAME), etc. A super key is a group of single or multiple keys that identifies rows in a table. It supports NULL values.

* Adding zero or more attributes to the candidate key generates the super key.
* A candidate key is a super key but vice versa is not true.
* Super Key values may also be NULL.

### **Alternate Key**

The candidate key other than the primary key is called an [alternate key](https://www.geeksforgeeks.org/sql-alternate-key/) .

* All the keys which are not primary keys are called alternate keys.
* It is a secondary key.
* It contains two or more fields to identify two or more records.
* These values are repeated.
* Eg:- SNAME, and ADDRESS is Alternate keys

### **Foreign Key**

If an attribute can only take the values which are present as values of some other attribute, it will be a [foreign key](https://www.geeksforgeeks.org/foreign-key-constraint-in-sql/) to the attribute to which it refers. The relation which is being referenced is called referenced relation and the corresponding attribute is called referenced attribute. The referenced attribute of the referenced relation should be the primary key to it.

* It is a key it acts as a primary key in one table and it acts as  
  secondary key in another table.
* It combines two or more relations (tables) at a time.
* They act as a cross-reference between the tables.
* For example, DNO is a primary key in the DEPT table and a non-key in EMP

### **Composite Key**

Sometimes, a table might not have a single column/attribute that uniquely identifies all the records of a table. To uniquely identify rows of a table, a combination of two or more columns/attributes can be used. It still can give duplicate values in rare cases. So, we need to find the optimal set of attributes that can uniquely identify rows in a table.

* It acts as a primary key if there is no primary key in a table
* Two or more attributes are used together to make a [composite key](https://www.geeksforgeeks.org/composite-key-in-sql/) .
* Different combinations of attributes may give different accuracy in terms of identifying the rows uniquely.

## **How Are Anomalies Caused in DBMS?**

Database anomalies are the faults in the database caused due to poor management of storing everything in the flat database. It can be removed with the process of [Normalization](https://www.geeksforgeeks.org/normal-forms-in-dbms/), which generally splits the database which results in reducing the anomalies in the database.

* Insertion Anomalies
* Deletion Anomalies
* Update Anomalies.

**Insertion Anomaly:** If a tuple is inserted in referencing relation and referencing attribute value is not present in referenced attribute, it will not allow insertion in referencing relation.

**Example:** If we try to insert a record in STUDENT\_COURSE with STUD\_NO =7, it will not allow it.

**Deletion and Updation Anomaly:** If a tuple is deleted or updated from referenced relation and the referenced attribute value is used by referencing attribute in referencing relation, it will not allow deleting the tuple from referenced relation.

**Example:** If we want to update a record from STUDENT\_COURSE with STUD\_NO =1, We have to update it in both rows of the table. If we try to delete a record from STUDENT with STUD\_NO =1, it will not allow it.

To avoid this, the following can be used in query:

* **ON DELETE/UPDATE SET NULL:** If a tuple is deleted or updated from referenced relation and the referenced attribute value is used by referencing attribute in referencing relation, it will delete/update the tuple from referenced relation and set the value of referencing attribute to NULL.
* **ON DELETE/UPDATE CASCADE:** If a tuple is deleted or updated from referenced relation and the referenced attribute value is used by referencing attribute in referencing relation, it will delete/update the tuple from referenced relation and referencing relation as well.

### **How These Anomalies Occur?**

* **Insertion Anomalies:** These anomalies occur when it is not possible to insert data into a database because the required fields are missing or because the data is incomplete. For example, if a database requires that every record has a [primary key](https://www.geeksforgeeks.org/postgresql-primary-key/), but no value is provided for a particular record, it cannot be inserted into the [database](https://www.geeksforgeeks.org/introduction-of-dbms-database-management-system-set-1/).
* **Deletion anomalies:** These anomalies occur when deleting a record from a database and can result in the unintentional loss of data. For example, if a database contains information about customers and orders, deleting a customer record may also delete all the orders associated with that customer.
* **Update anomalies:**  These anomalies occur when modifying data in a database and can result in inconsistencies or errors. For example, if a database contains information about employees and their salaries, updating an employee’s salary in one record but not in all related records could lead to incorrect calculations and reporting.

## **Fundamental Operators**

1. [Selection(σ)](https://www.geeksforgeeks.org/select-operation-in-relational-algebra/)
2. [Projection(π)](https://www.geeksforgeeks.org/difference-between-selection-and-projection-in-dbms/)
3. [Union(U)](https://www.geeksforgeeks.org/sql-union-operator/)
4. [Set Difference(-)](https://www.geeksforgeeks.org/set-theory-operations-in-relational-algebra/)
5. [Set Intersection(∩)](https://www.geeksforgeeks.org/sql-intersect-clause/)
6. [Rename(ρ)](https://www.geeksforgeeks.org/rename-operation-in-relational-algebra/)
7. [Cartesian Product(X)](https://www.geeksforgeeks.org/cartesian-product-operation-in-relational-algebra/)

## **Normals Forms in DBMS**

| **Normal Forms** | **Description of Normal Forms** |
| --- | --- |
| **First Normal Form (1NF)** | A relation is in [first normal form](https://www.geeksforgeeks.org/first-normal-form-1nf/) if every attribute in that relation is single-valued attribute. |
| **Second Normal Form (2NF)** | A relation that is in First Normal Form and every non-primary-key attribute is fully functionally dependent on the primary key, then the relation is in [Second Normal Form (2NF).](https://www.geeksforgeeks.org/second-normal-form-2nf/) |
| **Third Normal Form (3NF)** | A relation is in the [third normal form](https://www.geeksforgeeks.org/third-normal-form-3nf/), if there is no transitive dependency for non-prime attributes as well as it is in the second normal form. A relation is in 3NF if at least one of the following conditions holds in every non-trivial function dependency X –> Y.   * X is a super key. * Y is a prime attribute (each element of Y is part of some candidate key). |
| **Boyce-Codd Normal Form (BCNF)** | For BCNF the relation should satisfy the below conditions   * The relation should be in the 3rd Normal Form. * X should be a superkey for every functional dependency (FD) X−>Y in a given relation. |
| **Fourth Normal Form (4NF)** | A relation R is in [4NF](https://www.geeksforgeeks.org/introduction-of-4th-and-5th-normal-form-in-dbms/) if and only if the following conditions are satisfied:   * It should be in the [Boyce-Codd Normal Form (BCNF)](https://www.geeksforgeeks.org/boyce-codd-normal-form-bcnf/). * The table should not have any Multi-valued Dependency. |
| **Fifth Normal Form (5NF)** | A relation R is in [5NF](https://www.geeksforgeeks.org/what-is-fifth-normal-form-5nf-in-dbms/) if and only if it satisfies the following conditions:   * R should be already in 4NF. * It cannot be further non loss decomposed (join dependency). |

## **Advantages of Normalization**

* Normalization eliminates data redundancy and ensures that each piece of data is stored in only one place, reducing the risk of data inconsistency and making it easier to maintain data accuracy.
* By breaking down data into smaller, more specific tables, normalization helps ensure that each table stores only relevant data, which improves the overall data integrity of the database.
* Normalization simplifies the process of updating data, as it only needs to be changed in one place rather than in multiple places throughout the database.
* Normalization enables users to query the database using a variety of different criteria, as the data is organized into smaller, more specific tables that can be joined together as needed.
* Normalization helps ensure that the database can scale to meet future needs by reducing data redundancy.
* Normalization can help ensure that data is consistent across different applications that use the same database, making it easier to integrate different applications and ensuring that all users have access to accurate and consistent data.

## **Disadvantages of Normalization**

* Normalization can result in increased performance overhead due to the need for additional [join operations](https://www.geeksforgeeks.org/joins-in-dbms/) and the potential for slower query execution times.
* Normalization can result in the loss of data context, as data may be split across multiple tables and require additional joins to retrieve.
* Proper implementation of normalization requires expert knowledge of database design and the normalization process.
* Normalization can increase the complexity of a database design, especially if the data model is not well understood or if the normalization process is not carried out correctly