DBMS (Database Management System)

Data is the raw material that can be processed for any computing machine. It lacks context and meaning on its own, for example, a number, image, or an employee's name.

Information is data that has been converted into a more useful or intelligible form. A report card is an example of information. We need information to gain knowledge, keep systems up to date, and understand rules and regulations.

Knowledge is the purposeful organization and evaluation of information by the human mind. The hard work required to get marks is an example of knowledge derived from information. Knowledge is categorized into two types:

- **Fact-Based Knowledge**: Gained from fundamental principles and through experiments.
- **Heuristic-Based Knowledge**: The knowledge of good practice and good judgment, like a hypothesis.

A **database** is a related information placed in an organized form. It is an organized collection of related information. Examples include a dictionary, telephone directory, or mobile contacts.

Operations performed on a database include:

- Insertion
- Updation
- Deletion
- Retrieval
- Sorting

The following table differentiates between data and information:

Data	Information
It is the raw fact[18].	It is the processed form of data[19].
It is not significant to a business[20].	It is significant to a business[21].
Data are atomic-level pieces of information[22].	It is a collection of data[23].
Data does not help in decision-making[24].	It helps in decision-making[25].
Example: Product name[26].	Example: Report card sheet[27].

A **traditional file system** is a method of storing and organizing computer files and the data they contain to make them easy to find and access.

Characteristics of a file system:

- It is a group of files for storing the data of an organization.

- Each file is independent from one another.
- Each file is also called a flat file.
- Files are designed by using programs written in programming languages such as C, or C++.

Limitations/Disadvantages of a file processing system:

- **Separated and isolated data**: A user might need data from two separate files to make a decision.
- **Duplication of data**: This costs time and money, takes up additional storage space, and can lead to a loss of data integrity.
- **Data dependencies**: Files and records are described by specific physical formats that were coded in the application program by the programmers.
- **Difficulty in representing the data** from the user's point of view.
- **Low data security**: The security of data is low in the file-based system because the data is maintained in a flat file and is easy to access.

Limitations/Disadvantages of a file processing system continued:

- **Transactional problems**: This system does not satisfy transactional properties, which are Atomicity, Consistency, Isolation, and Durability (ACID).
- **Concurrency problems**: This occurs when multiple users access the same piece of data at the same time[39]. A problem may result when users try to update the file simultaneously.

Building blocks of a database:

- Columns / Fields
- Rows / Tuples / Records
- Tables

A **DBMS (Database Management System)** is a software system that allows users to define, create, and maintain the database and provide controlled access to the data.

Applications of a database include:

- Library systems
- Banking systems
- ATMs

Examples of databases include MySQL, Oracle, SQL Server, and Microsoft Access.

Components of a DBMS:

- **Hardware**: The actual computer system used for keeping and accessing the database. Conventional DBMS hardware consists of secondary storage devices such as hard disks. Databases can run on a range of machines from microcomputers to mainframes.
- **Software**: The actual DBMS that sits between the physical database and the users. It handles all requests from users for accessing the database.

- **Data**: The most important component from the end user's point of view. The actual data is separated from the programs that use it. A database should always be designed, built, and populated for a particular audience and purpose.
- **Users**: There are many users who can access or retrieve data on demand using applications and interfaces provided by the DBMS.

The users can be classified into the following groups:

- Naive Users
- Online Users
- Sophisticated Users
- Specialized Users
- Application Programmers
- DBA - Database Administrator

Naive Users are those who do not need to be aware of the presence of the database system. They are the end users of the database.

- **Naive Users** work through a menu-driven application program, where the range of input and response is always indicated to the users.
- **Online Users** may communicate with the database directly through an online terminal or indirectly through a user interface and application program.
- **Sophisticated Users** are those who interact with the system without writing a program. Instead, they form their requests in a database query language.
- **Specialized Users** write specialized database applications that do not fit into the traditional database processing framework.
- **Application Programmers** are responsible for developing the application programs or user interfaces. The application programs could be written in a high-level language.
- **DBA - Database Administrator** is a person or group in charge of implementing the database system within the organization. The DBA has all the privileges allowed by the DBMS and can assign privileges to the users. The DBA job requires a high degree of technical expertise and may consist of a team of people.

The components of a DBMS also include

Procedures, which refer to the instructions and rules that govern the design and use of the database. Users and staff who manage the database require documented procedures on how to use or run the system.

Disadvantages of a DBMS:

- **Complexity**: Providing the functionality expected of a good DBMS makes it an extremely complex piece of software. Database designers, developers, and users must understand this functionality to take full advantage of it. A failure to understand the system can lead to bad design decisions.
- **Size**: The complexity and breadth of functionality make the DBMS an extremely large piece of software, occupying megabytes of disk space and requiring substantial amounts of memory to run efficiently.

- **Performance**: A file-based system written for a specific application generally has very good performance. However, a DBMS is written to be more general to cater to many applications rather than just one.
- **Higher impact of failure**: The centralization of resources increases the vulnerability of the system. Since all users rely on the DBMS, the failure of any component can bring operations to a halt.
- **Cost of DBMS**: The cost of a DBMS varies significantly depending on the environment and functionality provided. There is also a recurrent annual maintenance cost.

The **architecture of DBMS** has three levels:

- External level
- Conceptual Level
- Internal Level

The objective of this three-level architecture is to separate each user's view of the data from the way the database is physically represented.

Reasons for this architecture include:

- The internal structure of the database should be unaffected by changes to the physical aspects of storage.
- The DBA should be able to change the conceptual structure of the database without affecting all other users.

External level/view level: This level describes the part of the database that is relevant to each user. It insulates the users from the details of the conceptual and internal levels.

Conceptual level/logic level: This level describes what data is stored in the database and the relationships among the data.

It represents:

- All entities, attributes, and their relationships.
- The constraints on the data.
- Security and integrity information.

Internal level/storage level: It is the physical representation of the database on the computer. This level describes how the data is stored in the database and covers the data structure and file organization used to store the data on storage devices.

Schemas:

- **External Schema**: The external view is described by a schema called an external schema that corresponds to different views of the data.
- **Conceptual Schema**: The conceptual view is defined by a conceptual schema, which describes all the entities, attributes, and their relationships with the integrity constraints.
- **Internal Schema**: The internal level is defined by an internal schema, which is a complete description of the internal model.

There is only one conceptual schema and one internal schema per database, but there can be more than one external schema. A schema is also known as an **Intension**.

Mapping between the levels:

- **External-Conceptual Mapping**: Each external schema is related to the conceptual schema by this mapping. This mapping gives the correspondence among the entities and the relationships of the external and conceptual views.
- **Conceptual-Internal Mapping**: The conceptual schema is related to the internal schema by this mapping. This mapping specifies the method of deriving the conceptual record from the physical database.

Data Independence:

- **Logical data independence**: This indicates that the conceptual schema can be changed without affecting the existing external schema. The changes would be absorbed by the mapping between the external and conceptual levels.
- **Physical data independence**: This indicates that the physical storage structure or devices can be changed without affecting the conceptual schema. The change would be absorbed by the conceptual-internal mapping.

Logical data independence is the ability to change the Conceptual Schema without affecting the existing External Schemas. The changes would be absorbed by the mapping between the external and conceptual levels. This type of data independence is much more difficult to achieve than physical data independence. It requires flexibility in the database's design and foresight from the programmer regarding future modifications.

Physical data independence indicates that the physical storage structure or devices can be changed without affecting the conceptual schema. The change would be absorbed by the Conceptual-Internal mapping.

Limitations of a file processing system include **separated and isolated data**. A user might need data from two separate files to make a decision. The files would have to be evaluated by analysts and programmers to determine the required data and the relationships between them, and then an application would be written to process and extract the data.

Another limitation of a file processing system is the **difficulty in representing data from the user's point of view**. It was often difficult to create useful applications for users, as data from various files had to be combined. It was challenging to determine the relationships between isolated data to meet user application requirements.

Components of a DBMS include **data**, which is the most important component from the end user's point of view. A major feature of a database is that the actual data is separated from the programs that use it. A database should always be designed and built for a particular audience and purpose.

Procedures refer to the instructions and rules that govern the design and use of the database. The users of the system and the staff who manage the database require documented procedures on how to use or run it.

Disadvantages of a DBMS include

complexity. The functionality of a good DBMS makes it a very complex piece of software. Database designers, developers, and administrators must understand this functionality to take full advantage of it.

Failure to understand the system can lead to bad design decisions with serious consequences for an organization.

Other disadvantages of a DBMS include :

size. The complexity makes the DBMS an extremely large piece of software, occupying megabytes of disk space and requiring substantial amounts of memory to run efficiently.

Performance is another disadvantage. A file-based system is written for a specific application, resulting in very good performance. However, a DBMS is designed to be more general and to cater to many applications rather than just one.

A **higher impact of a failure** is also a disadvantage. The centralization of resources increases the vulnerability of the system. Since all users and applications rely on the availability of the DBMS, the failure of any component can bring operations to a halt.

The **cost of a DBMS** can be a disadvantage. The cost varies significantly depending on the environment and functionality provided. There is also a recurrent annual maintenance cost.

This page contains a table comparing a file management system and a database management system.

File Management System (e.g., C++, COBOL Program)	Database Management (e.g., Oracle or Sybase)
1. Cheap	1. Relatively expensive
2. Small System	2. Large System
3. Relatively few files	3. Many files
4. Files not necessarily related	4. Files are related
5. Simple structure	5. Complex structure
6. Little preliminary design	6. Vast preliminary design
7. Integrity left to application programmer	7. Rigorous inbuilt integrity checking
8. No security	8. Rigorous security
9. Simple, primitive backup/recovery	9. Complex & sophisticated backup/recovery
10. User is often single	10. Multiple user

The diagram illustrates the three-level architecture of a DBMS, showing how different users have different external schemas (views) that are mapped to a single conceptual schema, which in turn is mapped to a single internal schema.

The

Role of a DBA (Database Administrator) is to be the person or group in charge of implementing the database system within an organization. The DBA job requires a high degree of technical expertise. A DBA may consist of a team of people rather than just one person.

The **Responsibilities of a DBA** include:

- Making decisions concerning the content of the database.
- Planning the storage structure and access strategy.
- Providing support to users.
- Defining the security and integrity checks.
- Interpreting backup and recovery strategies.
- Monitoring performance and responding to changes in requirements.

A **Data Dictionary** or **Meta Data** is data about data. It is a self-describing nature of the database. It holds information about each data element in the database, such as names, values, and access authorization. It can also indicate which application program uses the data. Meta Data is used by developers to create programs and queries to manage and manipulate the data.

There are two types of data dictionaries:

- **Active Data Dictionary**: This type is automatically managed by the data management software. It is always consistent with the current structure of the database.
- **Passive Data Dictionary**: This is used for documentation purposes and is managed by the users of the system. It is modified manually by the users.

Database Languages include:

- **Data Definition Language (DDL)**: This is a language that allows the user to define the data and their relationship to other types of data.

Commands in DDL are:

- Create
- Alter
- Rename
- Drop

Data Manipulation Language (DML) is a language that provides a set of operations to support the basic data manipulation on the data held in the database. Commands used are:

- Insert
- Delete
- Select
- Update

Data Control Language (DCL) commands are:

- GRANT
- REVOKE

Data Models are of three types:

- Object-based
- Record-based Logical models

- Physical-based

A data model can be defined as an integrated collection of concepts for describing and manipulating the data, the relationship between the data, and constraints on the data in an organization. It comprises three components:

- **Structural part**: Rules that help in designing the model.
- **Manipulative part**: Defines the type of operation that can be applied to the model.
- **Integrity Rules**.

Data models are divided into three categories:

- **Object-based**: This model uses concepts such as entities, attributes, and their relationships. It can be used to describe the data at the Conceptual and External levels. An example is an E-R model.
- **Physical-based**: These models describe how the data is stored in the computer. This model is used to describe the data at the Internal level.
- **Record-based Logical models**: These models are used in describing the data at the logical and view levels. These models are used to specify the overall logical structure of the database. An example is a Hierarchical model.

Example of a Hierarchical model:

This model is based on a tree structure. It consists of a collection of records that are connected to each other by links. The tree structure in a Hierarchical model is known as a routed tree, and the root node of the tree is an empty node. The relationship that exists in the Hierarchical model is one to many.

The **Hierarchical model** is a collection of routed trees.

Advantages:

- It is easy to understand.
- It is more efficient than an ER model.

Disadvantages:

- **Data inconsistency** can occur when the parent node is deleted, which results in the deletion of the child node.
- **Wastage of storage space**.
- **Complex design**.
- **Absence of structural independency**.

A **Network model** consists of a collection of records that are connected to others by links.

Advantages:

- It is easy to design compared to the Hierarchical model.
- Data access is easy in the network model.

Disadvantages:

- It is complex to design compared to a relational model.

- Its efficiency is less than that of the relational model.
- It has an absence of structural independency.

A **Relational model** stores data in the form of tables.

The components of a relational model are:

- **Table/Relation**: The container for data.
- **Tuple/Rows**: Each row of data.
- **Attributes/Column**: Each column in the tuple.
- **Domain**: A set of permitted values.
- **Tuple variable**: Any value of a tuple.
- **Degree**: The number of columns in a relation.
- **Cardinality**: The number of rows (tuples) in a relation.

In the **Relational model**, many-to-many relationships can be easily implemented. This model is useful for representing most real-world objects and the relationships between them. The Relational model does not maintain a physical connection among records. Instead, data is organized logically in the form of rows and columns and is stored in a table.

A **Data Manager** handles user requests by interacting with the File Manager and Disk Manager to access the database.

Unit-2: Data Modeling Using ER Models

An **Enterprise** is an object about which information needs to be collected. An **Entity type** is a collection of entities that share the same attributes. An **Attribute** is a property or characteristic of an entity.

Types of Attributes:

- **Single-Value Attribute**: An attribute that contains a single value, such as age or salary.
- **Multivalued Attribute**: An attribute that can contain more than one value for a single entity, for example, a phone number.
- **Composite Attribute**: An attribute that can be further divided, such as a name (into first and last name) or a date of birth.
- **Simple or Atomic Attribute**: An attribute that cannot be further divided, such as age.
- **Stored Attribute**: An attribute that can be derived from another attribute.
- **Null value**: An empty attribute value.

An **Entity Set** is a collection of entities of a particular type, for example, a group of employees.

Representations in an ER diagram:

- **Composite attribute**: Represented by an oval containing sub-ovals for its components.
- **Multivalued attribute**: Represented by a double oval.
- **Derived attribute**: Represented by a dashed oval.

- **Key attribute**: Represented by an oval with the attribute name underlined. A key attribute uniquely identifies a record.

A **Weak Entity type** is one in which no key attribute is present.

A **Strong Entity type** is one that consists of a key attribute.

The **Domain of the attributes** is the set of possible values. A **Relationship type** represents an association between two or more entities. The **Degree of a relationship type** refers to how many entities are involved in a relationship. A **Binary relationship type** involves two entities , while a **Ternary relationship type** involves three entities.

Relationship Constraints:

- **Participation Constraints**:
    - **Total Participation**: Every entity in the entity set must be dependent on another entity. It is also known as existence dependency and is represented by a double line connecting the entity type to the relationship in an E-R diagram.
    - **Partial Participation**: Only some entities in the entity set are dependent upon another entity.

Cardinality ratio for a binary relationship specifies the number of relationship instances that an entity can participate in a relation set.

Relationship types based on cardinality are:

- One-to-one (1:1)
- One-to-many (1:N)
- Many-to-one (M:1)
- Many-to-many (M:N)

For example, an Employee works for a Department, which can be an M:1 relationship. An Employee works on Projects, which can be an M:N relationship.

Identifying Relationship: A weak entity type does not have a key attribute. To relate such an entity type, we use a combination of some of its attributes with the key of another entity type. This other entity type is called the **identifying** or **owner** entity type. The relationship type that links the weak entity to its owner is called the identifying relationship.

The **Discriminator** or **Partial Key** is an attribute of a weak entity that helps to uniquely identify it in conjunction with the owner entity's key.

This diagram shows an Employee (strong entity) having a Dependent (weak entity), with the identifying relationship represented by a double diamond and the partial key underlined with a dashed line.

The page also describes an exercise to create an E-R diagram for a company database based on a set of provided descriptions.

This page contains the problem description for the company database E-R diagram exercise. The company is organized into departments, each with a unique name and number. A department can have multiple locations. A department controls a number of projects, each with a unique name, number, and a single location. The company stores information about each employee, including name, social security number,

address, and salary. An employee is assigned to one department but may work on multiple projects. The company also needs to track the dependencies of each employee for insurance purposes.

The page also includes a diagram for a

Teacher-Student database , showing a many-to-many relationship (

M:N) between a Teacher and a Student, where a teacher can teach many students and a student can be taught by many teachers. It illustrates various attribute types, such as a

composite attribute for Name and a **multivalued attribute** for Phone number.

The **Database** is a collection of related information stored in an organized form.

The **Operations** that can be performed on a database include Insertion, Updation, Deletion, Retrieval, and Sorting.

A **Traditional File System** is a method of storing and organizing computer files and the data they contain to make them easy to find and access.

The **Characteristics of a file system** include:

- It is a group of files for storing the data of an organization.
- Each file is independent from one another.
- Each file is also called a flat file.
- Files are designed using programs written in programming languages such as C and C++.

The **limitations and disadvantages of a File Processing System** include:

- **Separated and isolated data**.
- **Duplication of data**, which costs time and money, takes up additional storage space, and can lead to a loss of data integrity.
- **Data dependencies**, where files and records are described by specific physical formats that were coded in the application program.
- **Difficulty in representing the data** from the user's point of view.
- **Low data security** because data is maintained in a flat file and is easily accessible.

Disadvantages of a File Processing System (cont'd):

- **Transactional problems**: The system does not satisfy transactional properties, such as Atomicity, Consistency, Isolation, and Durability (ACID).
- **Concurrency problems**: When multiple users access the same data at the same time, it can lead to a problem if they try to update the file simultaneously.

The **Building Blocks of a Database** are columns (fields), rows (tuples/records), and tables.

A **DBMS (Data Base Management System)** is a software system that allows users to define, create, and maintain a database and provides controlled access to the data.

Applications of a Database include library systems, banking systems, and ATMs.

Components of a DBMS:

- **Hardware**: The actual computer system used for keeping and accessing the database. Conventional DBMS hardware consists of secondary storage devices such as hard disks.
- **Software**: The actual DBMS that is between the physical database and the users of the system. All requests from the user for accessing the database are handled by the DBMS.
- **Data**.
- **Users**: There are a number of users who can access and retrieve data on demand using the application and interfaces provided by the DBMS. The users can be classified into different groups, including Naive Users, Online Users, Sophisticated Users, Specialized Users, Application Programmers, and the DBA (Database Administrator).

Types of DBMS Users (cont'd):

- **Naive Users**: Those users who do not need to be aware of the presence of the database system. They are the end users who work through a menu-driven application program.
- **Online Users**: Those users who may communicate with the database directly through an online terminal or indirectly through a user interface and application program.
- **Sophisticated Users**: Those users who interact with the system without writing a program. Instead, they form their requests in a database query language.
- **Specialized Users**: Those users who write specialized database applications that do not fit into the traditional database processing framework.
- **Application Programmers**: Those users who are responsible for developing the application programs or user interfaces.
- **DBA (Database Administrator)**: A person or group in charge of implementing the database system within the organization.

Disadvantages of a DBMS:

- **Complexity**: A DBMS is an extremely complex piece of software.
- **Size**: A DBMS is a large piece of software, occupying megabytes of disk space and requiring a substantial amount of memory.
- **Performance**: A file-based system, written for a specific application, generally has very good performance, whereas a DBMS is written to be more general.
- **Higher impact of failure**: The centralization of resources increases the vulnerability of the system, as a failure can bring operations to a halt.
- **Cost of DBMS**: The cost varies significantly, and there is also a recurrent annual maintenance cost.

A **Master file** is a file that remains static, with no changes made to it.

A **Transaction file** is a file that is dynamic in nature, and changes can be made to it.

An **Instance** is the state of the data in the database at a particular moment in time.

A **Schema** is the overall design of the database, or the description of the database.

A **Subschema** is a subset of the schema and gives users a window through which they can view only the part of the database that is of interest to them.

Architecture of DBMS: There is a three-level architecture.

- **External level**.
- **Conceptual level**.
- **Internal level**.

The objective of this three-level architecture is to separate each user's view of the data from the way the database is physically represented.

External level/view level: This level describes the part of the database that is relevant to each user. It insulates the users from the details of the conceptual and internal levels.

Conceptual level/logic level: This level describes what data is stored in the database and the relationship among the data. It represents all the entities, attributes, their relationships, and the constraints on the data.

Conceptual level also represents security and integrity information.

Internal level: This level is the physical representation of the database on the computer. It describes how the data is stored in the database and covers the data structures and file organization used to store the data on storage devices.

Schemas:

- **External Schema**: Describes the external view and corresponds to different views of the data.
- **Conceptual Schema**: Defines the conceptual view and describes all the entities, attributes, and their relationships with the integrity constraints.
- **Internal Schema**: Defines the internal level and is a complete description of the internal model.

There is only one Conceptual Schema and one Internal Schema per database, but more than one External Schema can exist. A Schema is also known as an **Intension**.

Mapping between the levels:

- **External-Conceptual Mapping**: This mapping relates each External Schema to the Conceptual Schema. It establishes the correspondence among the entities and relationships of the external and conceptual views.
- **Conceptual-Internal Mapping**: This mapping relates the Conceptual Schema to the Internal Schema. It specifies the method of deriving the conceptual record from the physical database.

Data Independence:

- **Logical Data Independence**: This indicates that the Conceptual Schema can be changed without affecting the existing External Schemas. The changes are absorbed by the mapping between the external and conceptual levels.
- **Physical Data Independence**: This indicates that the physical storage structure or devices can be changed without affecting the Conceptual Schema. The change is absorbed by the Conceptual-Internal mapping.

Logical data independence is much more difficult to achieve than physical data independence, as it requires flexibility in the database design and foresight for future modifications.

Limitations of the File Processing System:

- **Separated and Isolated Data**: To make a decision, a user may need data from two separate files. This requires analysts and programmers to evaluate the files, determine specific data and relationships, and then write an application to process and extract the needed data.
- **Difficulty in representing data from the user's view**: To create useful applications for the user, it was difficult to determine relationships between isolated data from various files that had to be combined.

Components of DBMS:

- **Data**: It is the most important component of the DBMS environment from the end user's point of view. A major feature of a database is that the actual data is separated from the programs that use it.
- **Procedures**: These refer to the instructions and rules that govern the design and use of the database. Both the users and the staff who manage the system require documented procedures.

Disadvantages of a DBMS:

- **Complexity**: The functionality of a good DBMS makes it an extremely complex piece of software. Database designers, developers, administrators, and end-users must understand this functionality, or it can lead to bad design decisions.

Disadvantages of a DBMS (cont'd):

- **Size**: The complexity makes the DBMS a very large piece of software, occupying megabytes of disk space and requiring a substantial amount of memory to run efficiently.
- **Performance**: A file-based system is written for a specific application, and as a result, its performance is generally very good. However, a DBMS is designed to be more general to cater to many applications.
- **Higher impact of a failure**: The centralization of resources increases the vulnerability of the system. Since all users rely on the DBMS, the failure of any component can bring operations to a halt.
- **Cost of DBMS**: The cost varies significantly depending on the environment and functionality. There is also a recurrent annual maintenance cost.

This page contains a table comparing the **File Management System** and **Database Management**.

File Management System	Database Management
1) Cheap	**1) Relatively expensive**
2) Small System	**2) Large System**
3) Relatively few files	**3) Many files**
4) Files not necessarily related	**4) Files are related**
5) Simple Structure	**5) Complex Structure**
6) Little preliminary design	**6) Vast preliminary design**

File Management System	Database Management
7) Integrity left to application programmer	**7) Rigorous inbuilt integrity checking**
8) No security	**8) Rigorous security**
9) Simple, primitive backup/recovery	**9) Complex & sophisticated backup/recovery**
10) User often single	**10) Multiple user**

The diagram illustrates the three-level architecture of a DBMS, showing how multiple **External Schemas** (views for end users) are mapped to a single **Conceptual Schema**. The Conceptual Schema is then mapped to a single **Internal Schema** (internal view).

The **Role of a DBA (Database Administrator)** is to be the person or group in charge of implementing the database system within an organization. The DBA job requires a high degree of technical expertise and may be performed by a team rather than one person.

The **Responsibilities of a DBA** include:

- Making decisions concerning the content of the database.
- Planning the storage structure and access strategy.
- Providing support to users.
- Interpreting backup and recovery strategies.
- Monitoring performance and responding to changes in requirements.

A **Data Dictionary** or **Meta Data** is data about the data. It holds information about each data element, such as its name, values, and access authorization. It also indicates which application program uses the data. Meta Data is used by developers to develop programs and queries to manage and manipulate the data.

There are two types of data dictionaries:

- **Active Data Dictionary**: This type is automatically managed by the data management software. It is always consistent with the current structure of the database.
- **Passive Data Dictionary**: This is used for documentation purposes. It is managed by the user and modified manually.

Database Languages:

- **Data Definition Language (DDL)**: This language allows the user to define the data and its relationships.

Commands in DDL are:

- CREATE
- ALTER
- RENAME

- DROP

Data Manipulation Language (DML) is a language that provides a set of operations to support basic data manipulation on the data held in the database. Commands used are:

- INSERT
- DELETE
- SELECT
- UPDATE

Data Control Language (DCL) commands are:

- GRANT
- REVOKE

Data Models: There are three types of data models:

- Object-based
- Record-based Logical models
- Physical-based: A data model can be defined as an integrated collection of concepts for describing and manipulating the data, relationships, and constraints on the data in an organization. It comprises three components:
- **Structural part**: Rules that help in designing the model.

Data Models (cont'd):

- **Manipulative part**: Defines the type of operation that can be applied to the model.
- **Integrity Rules**.

Data models are divided into three categories:

- **Object-based**: Uses concepts such as entities, attributes, and their relationships. It can be used to describe data at the Conceptual and External levels. An example is the E-R model.
- **Physical-based**: Describes how data is stored in the computer. It is used at the internal level.
- **Record-based Logical models**: Used for describing data at the logical and view levels. These models specify the overall logical structure of the database. An example is the Hierarchical model.

Data Modeling Using E-R Models :

An **Enterprise** is an object or thing about which information is to be collected. An **Entity type** is a collection of entities that share the same attributes. An **Attribute** is a property or characteristic of an entity, also known as a data element or data field.

Types of Attributes:

- **Single-Value Attribute**: An attribute that contains a single value, such as age or salary.
- **Multivalued Attribute**: An attribute that can contain more than one value for a single entity, for example, a phone number.

- **Composite Attribute**: An attribute that can be further divided, such as a name (into first and last name) or a date of birth.
- **Simple or Atomic Attribute**: An attribute that cannot be further divided, such as age.
- **Stored Attribute**: An attribute that can be derived from another attribute.
- **Null value**: An empty attribute value.

An **Entity Set** is a collection of entities of a particular type, for example, a group of employees.

Representations in an ER diagram:

- **Composite attribute**: Represented by an oval containing sub-ovals for its components.
- **Multivalued attribute**: Represented by a double oval.
- **Derived attribute**: Represented by a dashed oval.
- **Key attribute**: Represented by an oval with the attribute name underlined. A key attribute uniquely identifies a record.

A **Weak Entity type** is one in which no key attribute is present.

A **Strong Entity type** is one that consists of a key attribute.

The **Domain of the attributes** is the set of possible values.

A **Relationship type** represents an association between two or more entities.

The **Degree of a relationship type** refers to how many entities are involved in a relationship. A **Binary relationship type** involves two entities , while a **Ternary relationship type** involves three entities.

Relationship Constraints:

- **Participation Constraints**:
    - **Total Participation**: Every entity in the entity set must be dependent on another entity. It is also known as existence dependency and is represented by a double line connecting the entity type to the relationship in an E-R diagram.
    - **Partial Participation**: Only some entities in the entity set are dependent upon another entity.

Cardinality ratio for a binary relationship specifies the number of relationship instances that an entity can participate in a relation set.

Relationship types based on cardinality are:

- One-to-one (1:1)
- One-to-many (1:N)
- Many-to-one (M:1)
- Many-to-many (M:N)

For example, an Employee works for a Department, which can be an M:1 relationship. An Employee works on Projects, which can be an M:N relationship.

Identifying Relationship: A weak entity type does not have a key attribute. To relate such an entity type, we use a combination of some of its attributes with the key of another entity type. This other entity type is called the **identifying** or **owner** entity type. The relationship type that links the weak entity to its owner is called the identifying relationship.

The **Discriminator** or **Partial Key** is an attribute of a weak entity that helps to uniquely identify it in conjunction with the owner entity's key.

This diagram shows an Employee (strong entity) having a Dependent (weak entity), with the identifying relationship represented by a double diamond and the partial key underlined with a dashed line.

This page contains the problem description for a company database E-R diagram exercise. The company is organized into departments. Each department has a unique name and unique number. A department may have several locations. A department controls a number of projects, each of which has a unique name, unique number, and a single location. The company stores each employee's name, Social Security number, address, and salary. An employee is assigned to one department but may work on several projects, which are not necessarily controlled by the same department. The company also needs to keep track of the dependencies of each employee for insurance purposes.

This page includes a diagram for a **Teacher-Student database** to illustrate E-R concepts.

The diagram shows a many-to-many relationship (M:N) between a Teacher and a Student, where a teacher can teach many students and a student can be taught by many teachers. It illustrates various attribute types, such as a **composite attribute** for Name and a **multivalued attribute** for Phone number.

Data Manager: The data manager is a component of a DBMS that is responsible for handling user requests and interacting with the file manager and disk manager to access data from the database.

Relational Model Properties:

- The relational model is useful for representing most real-world objects and the relationships among them.
- Many-to-many relationships can be easily implemented in this model.
- It does not maintain physical connections among records. The data is organized logically in rows and columns and stored in tables.

Hierarchical Model:

- **Description**: It is based on a tree structure and consists of a collection of records connected by links. The tree is known as a "routed tree" with an empty root node. The relationships are one-to-many.
- **Advantages**: It is easy to understand and more efficient than an ER model.
- **Disadvantages**: Data inconsistency can occur when a parent node is deleted, leading to the deletion of child nodes. Other disadvantages include wastage of storage space, complex design, and a lack of structural independency.

Network Model:

- **Description**: This model is based on a collection of records connected by links.
- **Advantages**: It is easier to design than the Hierarchical model, and data access is easier.
- **Disadvantages**: It is more complex to design than a relational model. The efficiency is also less than that of the relational model, and it lacks structural independency.