23CAT603 DATABASE MANAGEMENT SYSTEMS

UNIT-I INTRODUCTION

File systems versus Database systems – Data Models – DBMS Architecture – Data Independence – Data Modeling using Entity – Relationship Model –E-R Modeling.

16 Marks Questions and Answers

1. File System Vs DBMS: Explore What is the Difference between File System and DBMS

- What is a File System?
- Features of the File System
- What is DBMS?
- Features of DBMS
- File System: DBMS

What is a File System?

A method and data structure for managing how data is saved and accessed is the operating system's file system. Without a file system, data would be stored in a single large body with no ability to distinguish between one piece of data and the next. By dividing the data into bits and giving each one a name, the data may be easily recovered and identified. A "file" is the term used to describe each data collection.

Features of the File System:

1. Space Management- File systems allot storage on a device in blocks of varying sizes, typically several physical units. The file system manages files and directories and keeps track of which storage areas are associated with particular files and which are not.
2. Metadata Management – A file system stores additional data about each file. A file system keeps all associated metadata, such as the file name, the size of a file's contents, and the file's location in the folder hierarchy, separate from the file's contents.

What is DBMS?

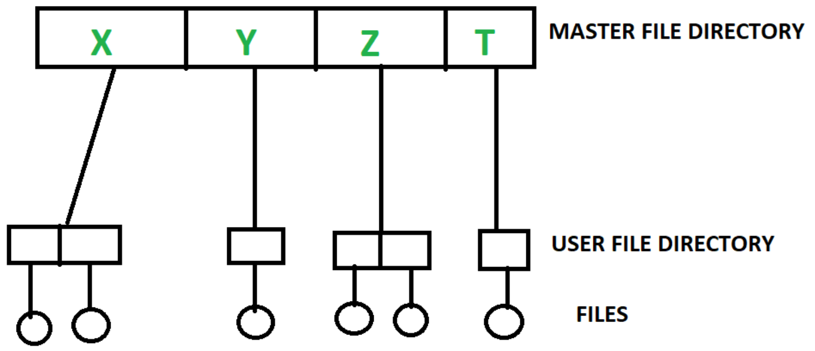
A software program that manages databases is called a database management system. (A database is a set of linked data that may be efficiently utilised for data retrieval, insertion, and deletion. Among other things, it organises data into tables, schemas, views, and reports. For instance, commercial databases like MySQL, Oracle, and others are frequently used in various applications. It offers an interface for operations including, but not limited to, creating a database, saving files, updating the database, and establishing a table in the database. It guarantees the security and safety of the database. Furthermore, it guarantees consistency of data when there are numerous users.

Features of DBMS:

1. Enhances File Consistency:By using database management systems, organisations may standardise their file handling procedures and guarantee that their data is consistent with that of other systems and applications.
2. Enhanced Security – As firms deal with huge amounts of data, security has become a top concern. A database management system is fully accessible to just the departmental head or the database administrator.

File System

The file system is basically a way of arranging the files in a storage medium like a hard disk. The file system organizes the files and helps in the retrieval of files when they are required. File systems consist of different files which are grouped into directories. The directories further contain other folders and files. The file system performs basic operations like management, file naming, giving access rules, etc. Example: NTFS(New Technology File System) , EXT(Extended File System).

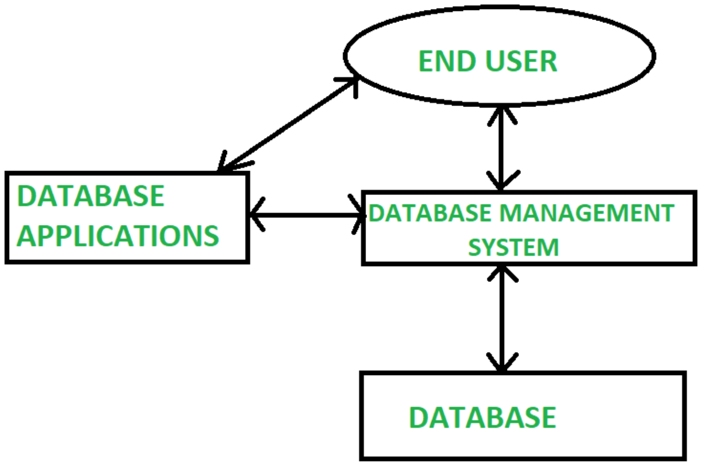


DBMS (Database Management System)

Database Management System is basically software that manages the collection of related data. It is used for storing data and retrieving the data effectively when it is needed. It also provides proper security measures for protecting the data from unauthorized access. In Database Management System the data can be fetched by SQL queries and relational algebra. It also provides mechanisms for data recovery and data backup.

Example:

Oracle, MySQL, MS SQL server.



Difference Between File System and DBMS

Basics	**File System**	**DBMS**
Structure	The file system is a way of arranging the files in a storage medium within a computer.	DBMS is software for managing the database.
Data Redundancy	Redundant data can be present in a file system.	In DBMS there is no redundant data.
Backup and Recovery	It doesn’t provide Inbuilt mechanism for backup and recovery of data if it is lost.	It provides in house tools for backup and recovery of data even if it is lost.
Query processing	There is no efficient query processing in the file system.	Efficient query processing is there in DBMS.
Consistency	There is less data consistency in the file system.	There is more data consistency because of the process of normalization .
Complexity	It is less complex as compared to DBMS.	It has more complexity in handling as compared to the file system.
Security Constraints	File systems provide less security in comparison to DBMS.	DBMS has more security mechanisms as compared to file systems.
Cost	It is less expensive than DBMS.	It has a comparatively higher cost than a file system.
Data Independence	There is no data independence.	In DBMS data independence exists, mainly of two types: 1) Logical Data Independence . 2)Physical Data Independence.
User Access	Only one user can access data at a time.	Multiple users can access data at a time.
Meaning	The users are not required to write procedures.	The user has to write procedures for managing databases
Sharing	Data is distributed in many files. So, it is not easy to share data.	Due to centralized nature data sharing is easy
Data Abstraction	It give details of storage and representation of data	It hides the internal details of Database
Integrity Constraints	Integrity Constraints are difficult to implement	Integrity constraints are easy to implement
Attribute *s*	To access data in a file , user requires attributes such as file name, file location.	No such attributes are required.
Example	Cobol , C++	Oracle , SQL Server

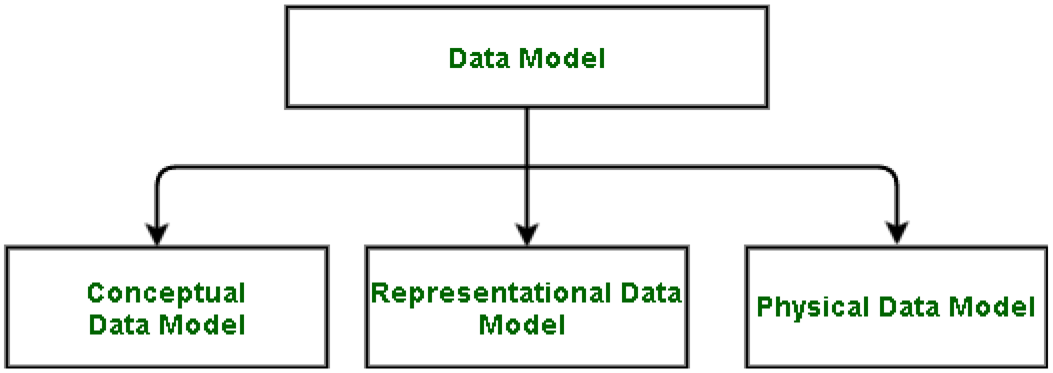
2. Data Models in DBMS

A Data Model in Database Management System (DBMS) is the concept of tools that are developed to summarize the description of the database. Data Models provide us with a transparent picture of data which helps us in creating an actual database. It shows us from the design of the data to its proper implementation of data.

Types of Relational Models

- Conceptual Data Model
- Representational Data Model
- Physical Data Model

It is basically classified into 3 types:-



1. Conceptual Data Model

The conceptual data model describes the database at a very high level and is useful to understand the needs or requirements of the database. It is this model, that is used in the requirement-gathering process i.e. before the Database Designers start making a particular database. One such popular model is the entity/relationship model (ER model). The E/R model specializes in entities, relationships, and even attributes that are used by database designers. In terms of this concept, a discussion can be made even with non-computer science(non-technical) users and stakeholders, and their requirements can be understood.

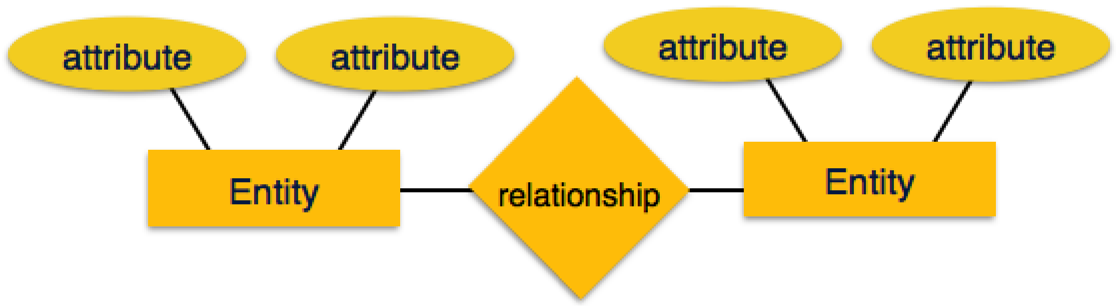
Entity-Relationship Model(ER Model): It is a high-level data model which is used to define the data and the relationships between them. It is basically a conceptual design of any database which is easy to design the view of data.

<u>Components of ER Model:</u>

Entity: An entity is referred to as a real-world object. It can be a name, place, object, class, etc. These are represented by a rectangle in an ER Diagram.

Attributes: An attribute can be defined as the description of the entity. These are represented by Ellipse in an ER Diagram. It can be Age, Roll Number, or Marks for a Student.

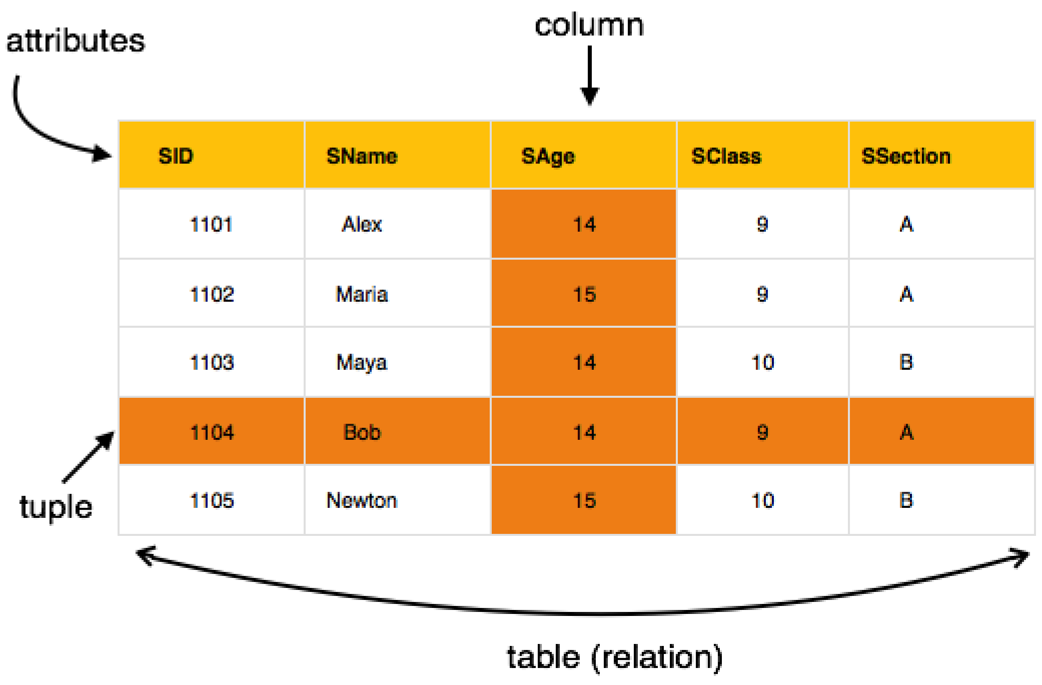
Relationship: Relationships are used to define relations among different entities. Diamonds and Rhombus are used to show Relationships.



2. Representational Data Model

This type of data model is used to represent only the logical part of the database and does not represent the physical structure of the database. The representational data model allows us to focus primarily, on the design part of the database. A popular representational model is a Relational model. The relational Model consists of Relational Algebra and Relational Calculus. In the Relational Model, we basically use tables to represent our data and the relationships between them. It is a theoretical concept whose practical implementation is done in Physical Data Model.

The advantage of using a Representational data model is to provide a foundation to form the base for the Physical model

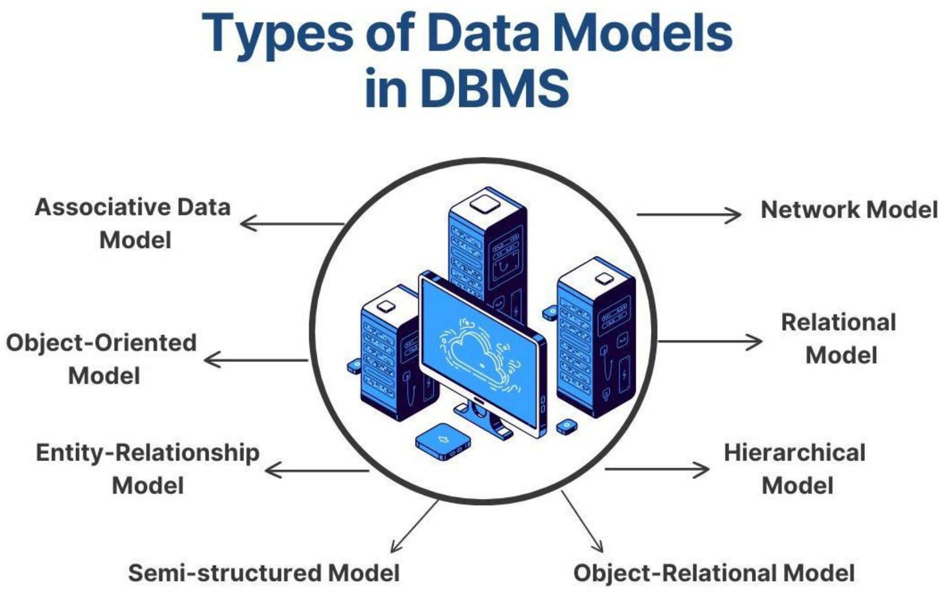

attributes
column
tuple
table (relation)

SID	SName	SAge	SClass	SSection
1101	Alex	14	9	A
1102	Maria	15	9	A
1103	Maya	14	10	B
1104	Bob	14	9	A
1105	Newton	15	10	B

3. Physical Data Model

The physical Data Model is used to practically implement Relational Data Model. Ultimately, all data in a database is stored physically on a secondary storage device such as discs and tapes. This is stored in the form of files, records, and certain other data structures. It has all the information on the format in which the files are present and the structure of the databases, the presence of external data structures, and their relation to each other. Here, we basically save tables in memory so they can be accessed efficiently. In order to come up with a good physical model, we have to work on the relational model in a better way. Structured Query Language (SQL) is used to practically implement Relational Algebra.

This Data Model describes HOW the system will be implemented using a specific DBMS system. This model is typically created by DBA and developers. The purpose is actual implementation of the database.



Types of Data Model in DBMS

There are a variety of models for data. Some of the most well-known models include:

- A Hierarchical model
- Network model
- Model of Entity-Relationship
- Relational Model
- Object-Oriented Data Model
- Object-Relational Data Model
- Semi-Structured Data Model
- Associative Data Model
- Flat Data Model
- Context Data Model

You can describe your database using any of these based on different variables. **The most important thing to consider is the degree to which the database management software that you're using is compatible with specific models.** The majority of database management systems are designed with a specific database model and will require users to follow that model; however, some support multiple models.

Furthermore, different models can be applied to various stages of the design process for databases. High-level conceptual data models are the best way to map out relationships between data in a way that people can see the data. Logical models based on records, however, are more closely reflected in how the data is stored at the data server.

When choosing a data model, it's equally about setting your priorities in relation to the database and the strengths of a specific model, regardless of whether your goals are speed, cost savings, usability, or something different.

Let's take a glance at the top popular database models.

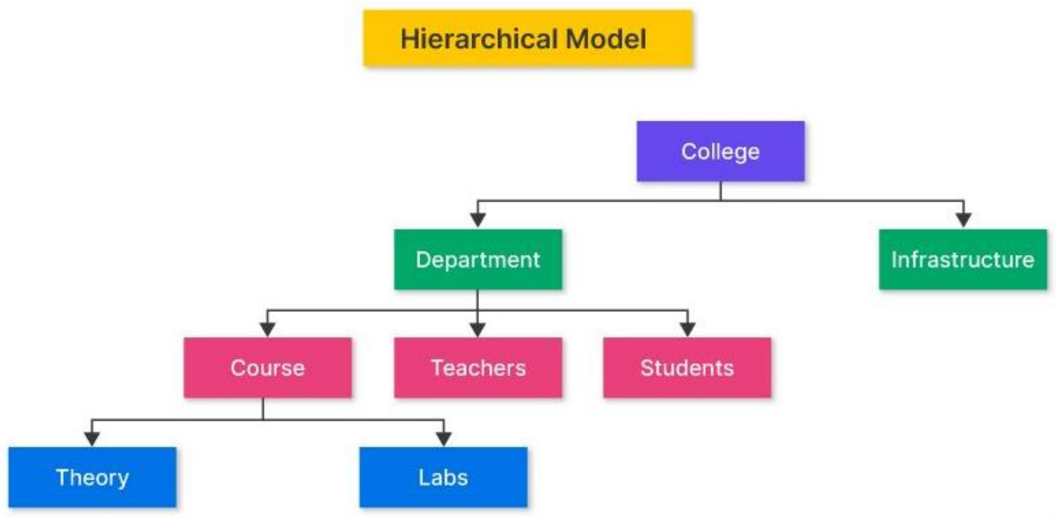
1. Hierarchical Data Model

The hierarchical data model was the first DBMS-based model. **This model organizes data in a hierarchical tree structure. The hierarchy begins at the root, which contains root data. It then expands into a tree, adding child nodes to the parent node.** This model can easily represent real-world relationships, such as sitemaps, food recipes, or website navigation.

Characteristics of Hierarchical Model:

1. ***One-to-many relationships:*** Here, the data is organized in a tree-like manner with a one-to-many relationship between data types. There can only be one path from any parent to any node.
2. ***Parent-Child Relationship*** Every child node has its parent node, but a parent can have more than one child. Multiple parents are prohibited.
3. ***Deletion Issue:*** When a parent node has been deleted, the child node will also be deleted.
4. ***Pointers:*** Pointers link the parent and child nodes. They are used for navigation between stored data.

Example: The visual representation of the basic concepts of the database schema is shown below, where the college node points to two other nodes, the *department* and the *infrastructure*. This example is explained by taking real-world entities.



The Advantages of the Hierarchical Model

- It's very easy and quick to navigate through a tree-like structure.
- Changes in the parent node are automatically reflected in child nodes**,** so data integrity is maintained.
- Easy to find the difference.

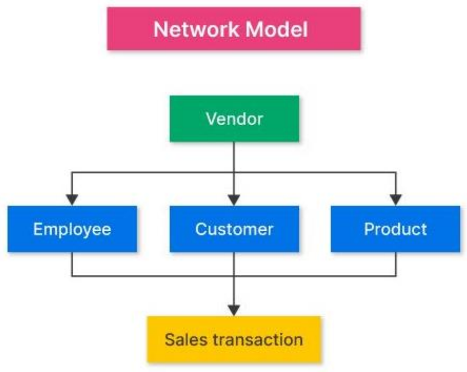
The Disadvantages of the Hierarchical Model

- Complex relationships cannot be supported.
- It does not support more than one parent of the child node. If there is a complex relationship where a child node requires two parent nodes, it cannot be represented with this model.
- The child node will be deleted automatically if a parent node has been deleted.

2. Network Model

This model is an extension of the hierarchical one - allowing for many-to-many relationships. Thus a record can have more than one parent in this model. This model was the most well-known before the relational one. This model replaces the hierarchical trees with a graph.

Example: In this example, we can see that all three child nodes (employee, customer and product) are connected to Vendor and Sales Transaction. In the hierarchical structure, this was not possible.



Characteristics of Network Model

1. ***Ability To Merge More Relationships:*** In this model, data is more closely related to more relationships. This model can manage both one-to-one and many-to-many relationships.
2. ***Many routes:*** Because there are many relationships, multiple paths to the same record may exist. This allows data access to be quick and easy.
3. ***Circular Link List:*** The operations on the network model can be done using the circular linked list. A program maintains the current position and navigates through the records based on the relationship.

The Advantages of the Network Model

- Data can be accessed more quickly than the hierarchical model. Because the data in the network model is more closely related, it is possible to take more than one route to reach a particular point. Thus, the data can be accessed faster and in many different ways.
- Data integrity is guaranteed because there is a parent/child relationship. Any changes in the parent record are reflected in a child record.

The Disadvantages of the Network Model

- The system can become more complicated as more relationships are needed to be managed. To use the model, the user must have a good understanding of it.
- Any change, such as updating, deletion, or insertion, is extremely complex.

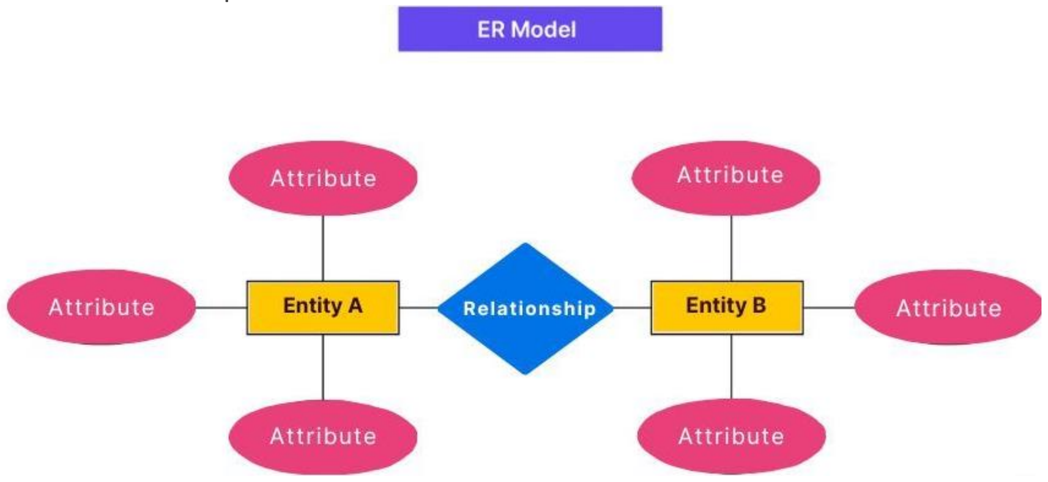
3. Entity-Relationship (ER) Data Mode

Entity-Relationship Data Model is a high-level diagram of a data model. **This model shows the real-world problem in a pictorial format to make it easier for stakeholders to understand.**

DBMS - ER Model diagram makes it easy for developers to see the system. The E-R model Diagram is used to show an ER Model or domain model. The following components make up the ER Diagram:

- **Entities:** A real-world thing is an entity - it could be a person, a place, or even an idea. ***Examples:*** Students, Teachers, Courses, Buildings, Departments, etc., are all examples of entities that make up a School Management system.
- ***Attributes***: An entity can contain a real-world property called an attribute. ***Example:*** An entity student owns the property, student ID and age, etc. In the ER Model diagram, an ellipse shape is used to represent attributes.

- ***Relationship:*** Relationship shows how two attributes are related. ***Example:*** A student working in a department. In the ER Model diagram, a diamond or rhombus shape represents the relationship.



Characteristics of the ER Model

1. ***Graphical representation for better understanding:*** It's very simple and easy to understand, so developers can use it to communicate with stakeholders.
2. ***ERD Diagram:*** The ERD diagram is used to represent the model visually.
3. ***Design of a Database:*** Database designers use this model to create a database.

Advantages of the ER Model

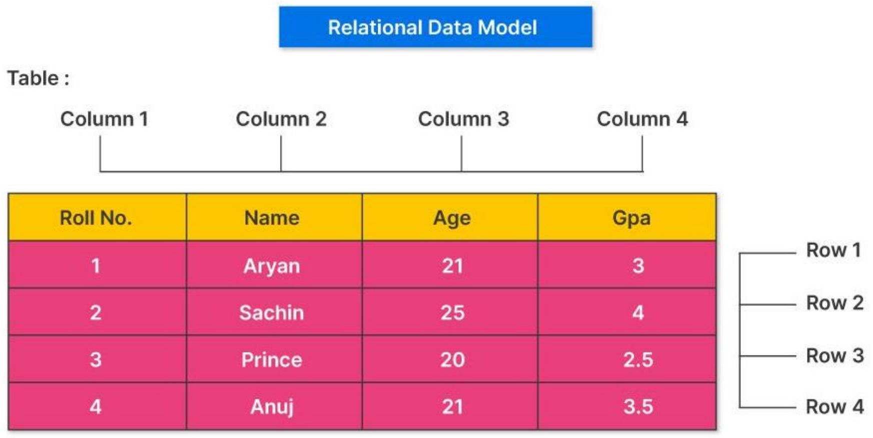
- **Easy:** Conceptually, an ER Model can be very simple to create. The ER Diagram can be easily created if we know the relationships between the entities and attributes.
- ***Effective Communication Tools***: Database designers use this model to communicate their ideas.
- ***Conversion to Any Model:*** This model is compatible with the relational model. It can easily be converted to a relational model by converting ER to the table. In fact, this model can be easily converted to any model, such as a hierarchical or network model.

Disadvantages of the ER Model

- ***There is no industry standard in notation:*** Developing an ER model according to industry standards is impossible. One developer may use notations that aren't understood by others.
- ***Hidden Information:*** Some information might be lost or hidden within the ER model. Some information may be lost or hidden because it is a high-level view.

4. Relational Model

The relational model is the most popular model. **It is a model where the information is stored as two-dimensional tables.** All information is stored in rows and columns. The foundation of a relational model is tables. Therefore, the tables are known as *relations* within the model.



Roll No.	Name	Age	Gpa
1	Aryan	21	3
2	Sachin	25	4
3	Prince	20	2.5
4	Anuj	21	3.5

Characteristics of the Relational Model

- ***Tuples:*** Each row within the table is known as the tuple. Each row is comprised of all the details regarding any particular specific instance. In the following example, each row is filled with all the information regarding a specific person - the first row contains information regarding Neha.
- ***Field or attribute:*** Attributes are the property that defines the table or relationship. The attributes' values must be within that same domain. In the following example, attributes are the aspects of a *student* such as Roll_no, Age, Address, etc.

Names	Roll_No	Phone_No	Address	AGE
Neha	98098	7305758990	Varanasi	20
Narayan	12830	8026288936	Delhi	10
Amit	330989	8583289090	Bihar	16
Khushi	2785790	708687878	Patna	14
Ganesh	17282	902893988	Bangalore	23

Advantages of the Relational Model

- ***Basic:*** This database model is less complex than the network and hierarchical models.
- ***Scalable:*** This model can scale easily to add as many columns and rows as a user may like.

- ***Structural Independence:*** It is possible to alter the structure of the database without altering the method for accessing the information. If we can modify the database structure without changing the ability of DBMS to access data, we can claim that structural independence is attained.

Disadvantages of the Relational Model

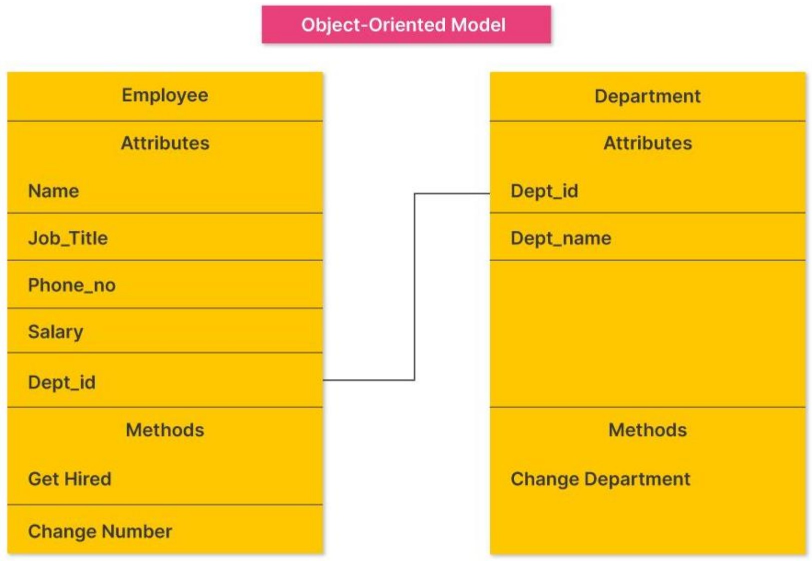
- ***Hardware Overheads:*** This model calls for more powerful hardware computers and devices for storing data to conceal the complexity and make it easier for the user.
- ***Poor Design:*** Because the model is extremely easy to create and utilize, users don't have to understand how data is stored for them to gain access to it. This simplicity of design could cause the development of a bad database, which could slow down as the database expands.

However, these issues are insignificant compared to the benefits of the relationship model. The problems could be prevented with the proper implementation and management.

5. Object-Oriented Database Model

Object-oriented database models combine both data and the relationship in a single structure referred to as an object. It is possible to store images, audio, videos, and more in databases that were not possible with the relational model (although you could save video and audio in a relational database, it is advised not to store them in relational databases).

In this case, the model has additional objects linked by hyperlinks. This link is used to connect one object to another object. Let's study the following example to understand the concept better.



In the example above, we have two objects (Employee and Department). All the data is contained in a single unit (object), including their attributes like name, job_title, etc. for employees and dept_ID, dept_Name, for department. These two objects are linked because of the one common attribute they share - the Department_Id. The communication between the two objects is done through this common identification. This is the representation of the physical model of the Object-oriented data model.

Advantages of Object-Oriented Database Models (OODBs)

1. **Natural Modeling:** OODBs excel at representing real-world entities and their relationships.
2. **Complex Object Handling:** OODBs can handle complex data types beyond simple scalar values. They can store multimedia content, documents, and other intricate data structures within objects.

Disadvantages of Object-Oriented Database Models (OODBs)

1. **Limited Adoption:** OODBs haven't achieved the same level of widespread adoption as relational models. This can lead to a smaller pool of developers experienced with OODB management systems and potentially limit the choice of tools and resources available.

6. Object-Relational Model

It is a mix of the relational model and object-oriented model. **This model was designed to bridge the gap between object-oriented and relational models**.

It has several advanced features, such as creating complicated data types based on our needs using available data types. **The issue with this particular model is that it is often complicated and challenging to manage.** Therefore, a thorough understanding of the model is necessary.

7. Semi-Structured Model

A semi-structured model is a modified version of the relational model, allowing for greater flexibility in representing data. It is impossible to distinguish between schema and data in this model because this model is based on the assumption that certain entities might have attributes that are missing. In contrast, other entities may include an additional attribute.

Example: An online news website stores information about news articles. The fixed attributes can be in the Authors Table (Article ID, Title, Author ID, Publication Date) and Articles Table (Author ID, Name, Bio). However, the news website might also want to store additional information about each article that doesn't fit neatly into predefined columns. This could include:

- A list of relevant keywords (tags) for better searchability.
- Links to related articles.
- Social media reactions (likes, shares) associated with the article (if applicable).
- Geolocation data if the article is about a specific location.

This additional information can be represented in a semi-structured format.

Remember! Semi-structured models **are not a replacement for relational models**, but rather a complementary approach for handling data that doesn't fit neatly into predefined tables and columns.

8. Associative Data Model

An associative model or Associative Data Model (ADM) is a model with the data divided into two components - entity and association. Everything with an independent existence is termed an *entity,* and the relation between the entities is called a *connection or association*.
The data is further divided into items and links:

- ***Items***: An item contains a name and an identifier (specific numerical value).
- ***Link:*** The link comprises the source, verb, subject, and unique identifier.

It's a more intuitive way of representing entities with an inherent relationship. Think of it as a way to model data that captures the "who connects to whom" directly within the entities themselves.
Example: A library keeps track of books and authors.
Items:

- **Item Name:** Book
    - Identifier: book_id (unique identifier for each book, could be a number or code)
    - Name: title (title of the book)
- **Item Name:** Author
    - Identifier: author_id (unique identifier for each author)
    - Name: full_name (author's full name)

Links:

- **Link Identifier:** link_id (unique identifier for each link)
    - Source: book_id (references the book identifier)
    - Verb: written_by (describes the relationship)
    - Subject: author_id (references the author identifier)

Explanation:

- We have two items: "Book" and "Author," each with an identifier and a name attribute.
- Links connect these items. A link has its own identifier, specifies the source item (book), the verb describing the relationship ("written_by"), and the subject item (author).

Example Data:

- Book:
    - book_id: 123
    - title: "The Lord of the Rings"
- Author:
    - author_id: 456
    - full_name: "J.R.R. Tolkien"
- Link:
    - link_id: 789
    - source: 123 (book_id)
    - verb: written_by
    - subject: 456 (author_id)

This example shows how the associative model captures the relationship between books and authors using items and links. By following the links, you can easily see which author wrote a particular book.

The associative data model focuses on representing real-world entities and their direct relationships, often in a single structure. Unlike relational models with separate tables, it emphasizes the connections between things.

9. Flat Data Model

A flat data model is a very basic way of storing data in a database management system (DBMS). It's like a simple spreadsheet with rows and columns. Imagine a single table with columns representing different data points (like name, age, etc.) and rows representing individual records (like entries for each customer).

Think of it as a flat list with no hierarchy or connections between entries. It's a good starting point for simple data storage, but for anything more complex, relational models (with tables linked through relationships) are more efficient.

10. Context Data Model

The context data model is a collection of a variety of models. It is comprised of models such as network models, object-oriented data models, relational models, etc. With this model, we can accomplish various tasks that aren't achievable using just one model.

By understanding the strengths and weaknesses of different data models, you can make informed decisions when designing and implementing your database, ensuring optimal data storage, retrieval, and manipulation. As the data landscape continues to evolve, the development of new data models and advancements in existing ones will be crucial for effectively managing and unlocking the potential of information in the digital age.

3. DBMS Architecture

A Database stores a lot of critical information to access data quickly and securely. Hence it is important to select the correct architecture for efficient data management. DBMS Architecture helps users to get their requests done while connecting to the database. We choose database architecture depending on several factors like the size of the database, number of users, and relationships between the users. There are two types of database models that we generally use, logical model and physical model.

Types of DBMS Architecture

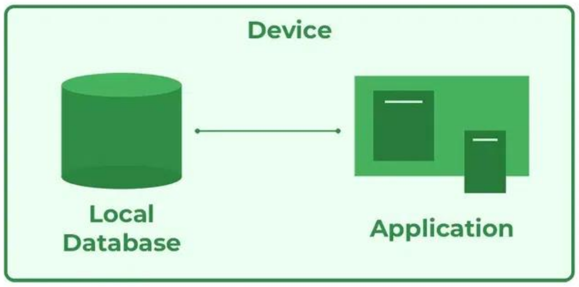
There are several types of DBMS Architecture that we use according to the usage requirements. Types of DBMS Architecture are discussed here.

- 1-Tier Architecture
- 2-Tier Architecture
- 3-Tier Architecture

The 3-level DBMS architecture provides logical and physical data independence.

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. The industry won’t use this architecture they logically go for 2-tier and 3-tier Architecture.



DBMS 1-Tier Architecture

Advantages of 1-Tier Architecture

Below mentioned are the advantages of 1-Tier Architecture.

Simple Architecture: 1-Tier Architecture is the most simple architecture to set up, as only a single machine is required to maintain it.

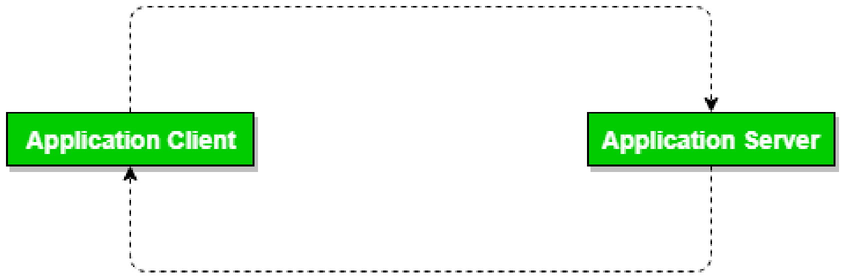
Cost-Effective: No additional hardware is required for implementing 1-Tier Architecture, which makes it cost-effective.

Easy to Implement: 1-Tier Architecture can be easily deployed, and hence it is mostly used in small projects.

2-Tier Architecture

The 2-tier architecture is similar to a basic 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.

An advantage of this type is that maintenance and understanding are easier, and compatible with existing systems. However, this model gives poor performance when there are a large number of users.



DBMS 2-Tier Architecture

Advantages of 2-Tier Architecture

Easy to Access: 2-Tier Architecture makes easy access to the database, which makes fast retrieval.

Scalable: We can scale the database easily, by adding clients or upgrading hardware.

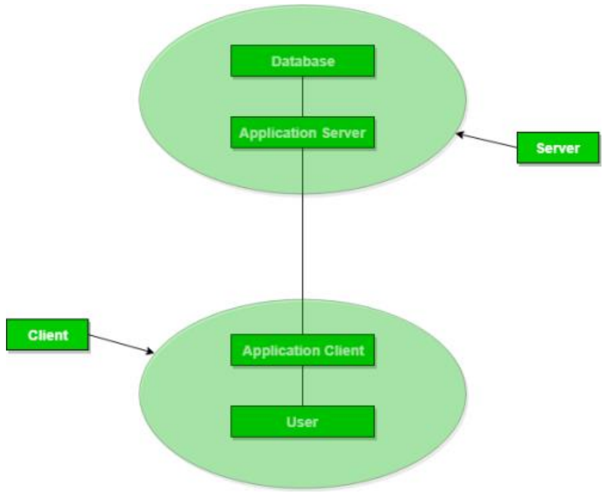
Low Cost: 2-Tier Architecture is cheaper than 3-Tier Architecture and Multi-Tier Architecture .

Easy Deployment: 2-Tier Architecture is easier to deploy than 3-Tier Architecture.

Simple: 2-Tier Architecture is easily understandable as well as simple because of only two components.

3-Tier Architecture

In 3-Tier Architecture , 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. This type of architecture is used in the case of large web applications.



Advantages of 3-Tier Architecture

Enhanced scalability: Scalability is enhanced due to the distributed deployment of application servers. Now, individual connections need not be made between the client and server.

Data Integrity: 3-Tier Architecture maintains Data Integrity. Since there is a middle layer between the client and the server, data corruption can be avoided/removed.

Security: 3-Tier Architecture Improves Security. This type of model prevents direct interaction of the client with the server thereby reducing access to unauthorized data.

Disadvantages of 3-Tier Architecture

More Complex: 3-Tier Architecture is more complex in comparison to 2-Tier Architecture. Communication Points are also doubled in 3-Tier Architecture.

Difficult to Interact: It becomes difficult for this sort of interaction to take place due to the presence of middle layers.

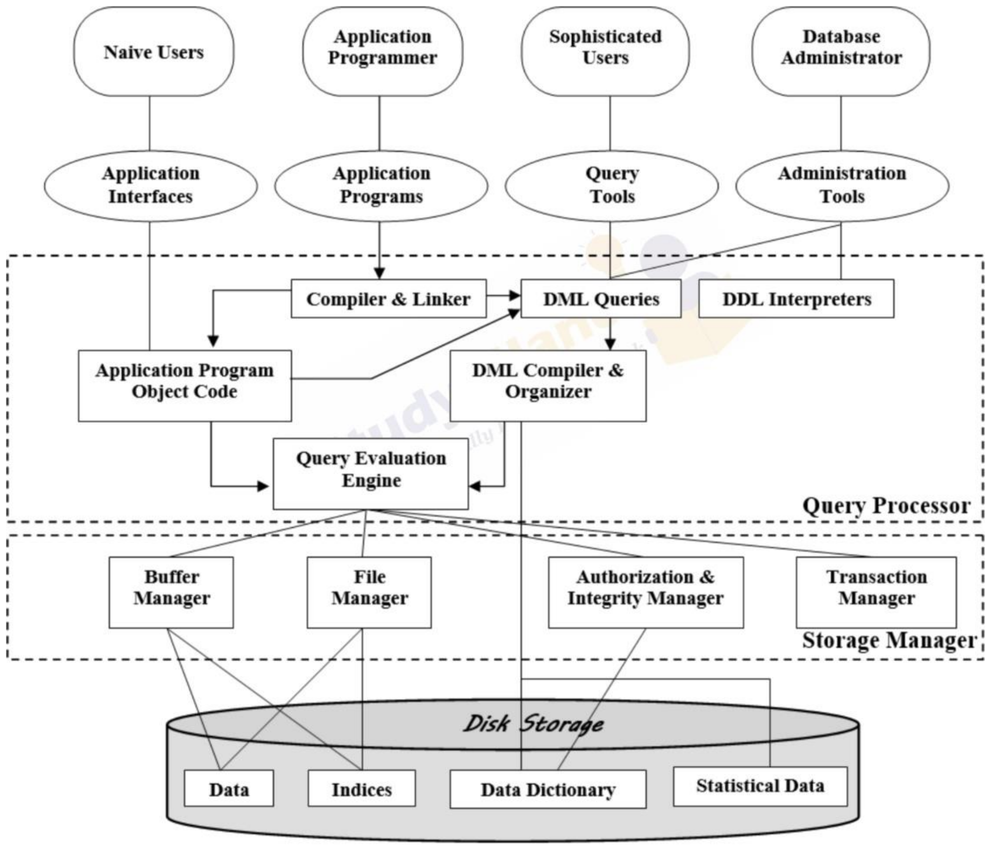
Database System Architecture

The typical structure of DBMS is based on Relational data model.

The top part of the architecture shows application interfaces used by naive users, application programs created by application programmers, query tools used by sophisticated users and administration tools used by database administrator.

The lowest part of the architecture is for disk storage.

The Middle two parts(**Query processor and storage manager**) are important components of database architecture.



Query processor:

The interactive query processor helps the database system to simplify and facilitate access to data. It consists of DDL(**Data Definition Language**) interpreter, DML(**Data Manipulation Language**) compiler and query evaluation engine.

The following are various functionalities and components of query processor

- **DDL interpreter**: This is basically a translator which interprets the DDL statements in data dictionaries.
- **DML compiler**: It translates DML statements query language into an evaluation plan. This plan consists of the instructions which query evaluation engine understands.
- **Query evaluation engine**: It executes the low-level instructions generated by the DML compiler.

When a user issues a query, the parsed query is presented to a query optimizer, which uses information about how the data is stored to produce an efficient execution plan for evaluating the query. An execution plan is a blueprint for evaluating a query. It is evaluated by query evaluation engine.

Storage manager:

Storage manager is the component of database system that provides interface between the low level data stored in the database and the application programs and queries submitted to the system.

The storage manager is responsible for storing, retrieving, and updating data in the database. The storage manager components include

- **Authorization and integrity manager**: Validates the users who want to access the data and tests for integrity constraints.
- **Transaction manager**: Ensures that the database remains in consistent despite of system failures and concurrent transaction execution proceeds without conflicting.
- **File manager**: Manages allocation of space on disk storage and representation of the information on disk.
- **Buffer manager**: Manages the fetching of data from disk storage into main memory. The buffer manager also decides what data to cache in main memory. Buffer manager is a crucial part of database system.

Storage manager implements several data structures such as

- **Data files**: Used for storing database itself.
- **Data dictionary**: Used for storing metadata, particularly schema of database.
- **Indices**: Indices are used to provide fast access to data items present in the database

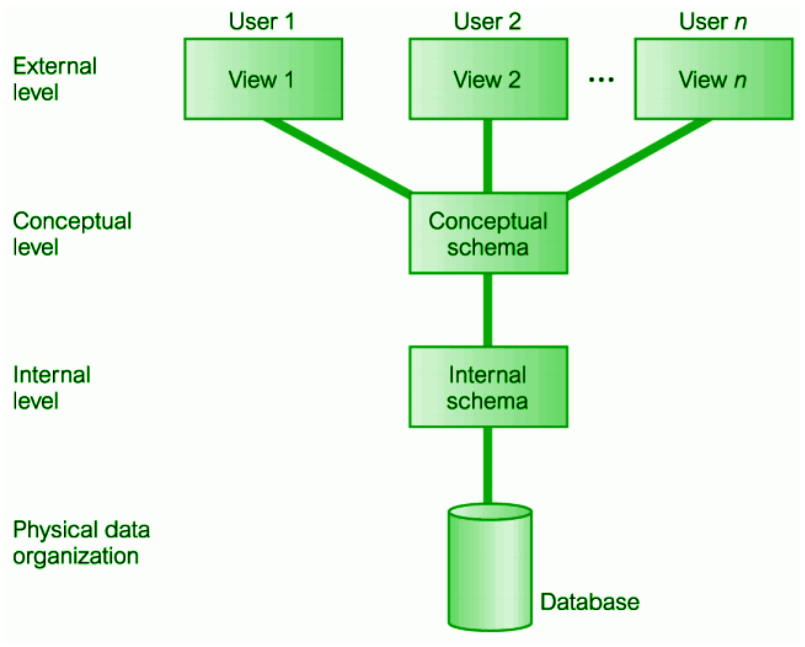
4. What is Data Independence in DBMS?

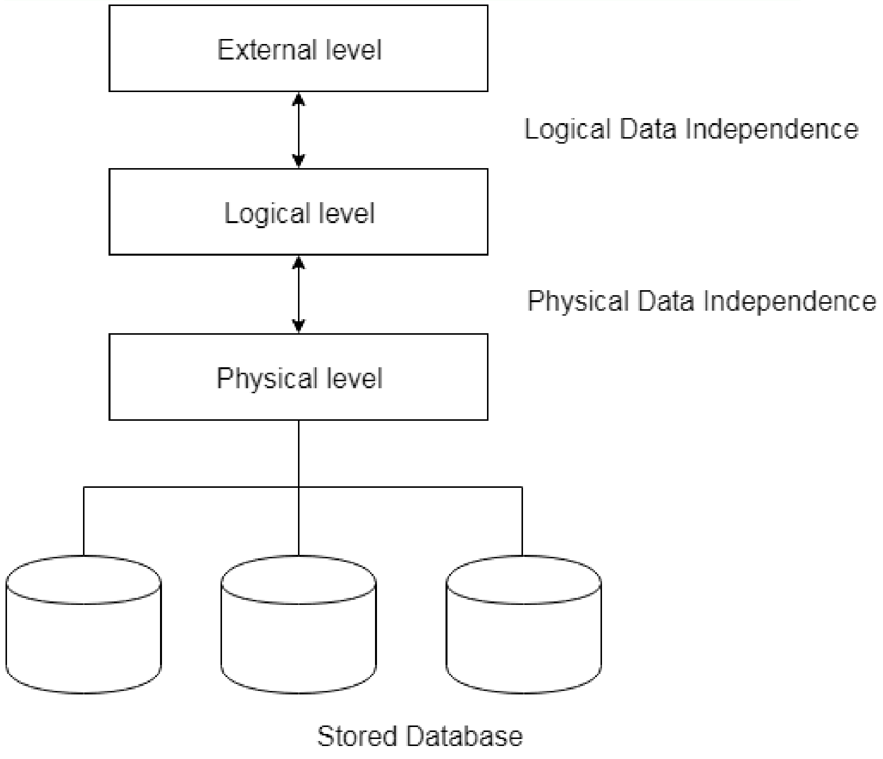
Data independence is a property of a database management system by which we can change the database schema at one level of the database system without changing the database schema at the next higher level.

What is Data Independence in DBMS?

In the context of a database management system, data independence is the feature that allows the schema of one layer of the database system to be changed without any impact on the schema of the next higher level of the database system. ” Through data independence, we can build an environment in which data is independent of all programs, and through the three schema architectures, data independence will be more understandable. Data via two card stencils along with centralized DBMS data is a form of transparency that has value for someone.

It can be summed up as a sort of immunity of user applications that adjusts correctly and does not change addresses, imparting the class of data and their order. I want the separate applications not to be forced to deal with data representation and storage specifics because this decreases quality and flexibility. DBMS permits you to see data with such a generalized sight. It actually means that the ability to change the structure of the lower-level schema without presenting the upper-level schema is called data independence.





Types of Data Independence

There are two types of data independence.

- logical data independence
- Physical data independence

Logical Data Independence

- Changing the logical schema (conceptual level) without changing the external schema (view level) is called logical data independence.
- It is used to keep the external schema separate from the logical schema.
- If we make any changes at the conceptual level of data, it does not affect the view level.
- This happens at the user interface level.
- For example, it is possible to add or delete new entities, attributes to the conceptual schema without making any changes to the external schema.

Physical Data Independence

- Making changes to the physical schema without changing the logical schema is called physical data independence.
- If we change the storage size of the database system server, it will not affect the conceptual structure of the database.
- It is used to keep the conceptual level separate from the internal level.
- This happens at the logical interface level.
- Example – Changing the location of the database from C drive to D drive.

Difference Between Physical and Logical Data Independence

Physical Data Independence	Logical Data Independence
It mainly concerns how the data is stored in the system.	It mainly concerns about changes to the structure or data definition.
It is easier to achieve than logical independence.	It is difficult to achieve compared to physical independence.
To make changes at the physical level we generally do not require changes at the application program level.	To make changes at the logical level, we need to make changes at the application level.
It tells about the internal schema.	It tells about the conceptual schema.
There may or may not be a need for changes to be made at the internal level to improve the structure.	Whenever the logical structure of the database has to be changed, the changes made at the logical level are important.
Example- change in compression technology, hashing algorithm, storage device etc.	Example – adding/modifying or deleting a new attribute.

5. Data Modeling using Entity

What Is Data Modeling?

Data modeling, at its core, is the process of organizing data into a structured format that makes it more accessible and useful for various applications and analyses. It involves creating visual representations of data objects, their relationships, and the rules that govern them. Data modeling is essential not just for technical reasons, but also for enforcing business rules, regulatory compliances, and ensuring data quality. Data modeling has a wide range of applications, from the functional design of software products and applications to the representations and models we use for analyzing business performance.

The Key Concepts Behind Data Modeling

Before delving into the types of data models and their applications, it is essential to understand the foundational concepts that data modeling is built upon. The basics behind virtually any data model are based upon three core concepts: entities, attributes, and relationships.

What Are Entities?

Entities are the objects or concepts for which data is collected. In database terms, entities usually become tables. They represent real-world objects or concepts that are of interest to a business or organization, such as customers, products, or orders. Entities are the building blocks of a data model and are used to define the structure of data.

Examples of entities:

1. **Customer**: Represents an individual or organization that purchases products or services from a business.
2. **Product**: Represents an item or service offered by a business for sale.
3. **Order**: Represents a transaction between a customer and a business, where the customer purchases one or more products.

What Are Attributes?

Attributes are the properties that define an entity. They describe the characteristics or properties of an entity, such as a customer's name, address, or age. Attributes are the most atomic parts of a data model and cannot be decomposed into lower-level components. In a relational data model, an attribute cannot exist independently from an entity type, and all attributes are always identified and shown as part of entity types. Attributes in a dataset are used to group, slice, filter, and reorder facts.

Examples of attributes:

1. **Customer attributes**: Name, address, email, phone number, date of birth.
2. **Product attributes:** Product ID, name, description, price, category.
3. **Order attributes**: Order ID, order date, customer ID, total amount, shipping address.

What Are Relationships?

Relationships describe the associations or connections between entities in a data model. They are essential for representing complex data structures and are used to connect related information between entities. There are three types of relationships in data modeling: one-to-many, many-to-many, and one-to-one.

Examples of relationships:

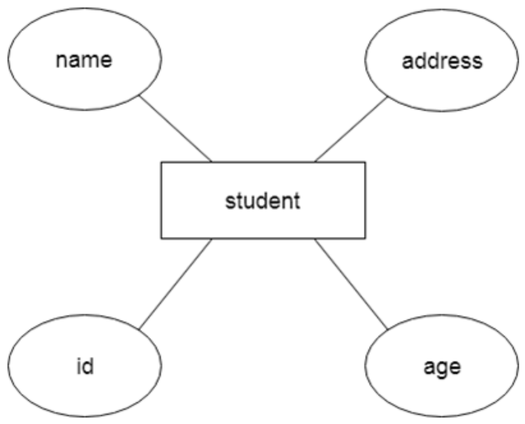
1. **One-to-many (1:M) relationships:** A customer can place multiple orders, but each order is associated with only one customer. In this case, the relationship between the Customer entity and the Order entity is one-to-many.
2. **Many-to-many (M: N) relationships:** A product can be part of multiple orders, and an order can contain multiple products. This relationship between the Product entity and the Order entity is many-to-many. In a physical database, it is implemented using a 1:M relationship with a junction or join table, such as an OrderDetails table that contains both Product ID and Order ID.
3. **One-to-one (1:1) relationships:** A user account is associated with only one customer profile, and a customer profile is associated with only one user account. In this case, the relationship between the UserAccount and CustomerProfile entities is one-to-one.

Understanding entities, attributes, and relationships is crucial for designing and implementing effective data models that meet the needs of organizations and support data-driven decision making. In addition, it's essential to understand that these structures shape every data model without fail and can be parsed and reconstructed from these elements to create highly complex and comprehensive models.

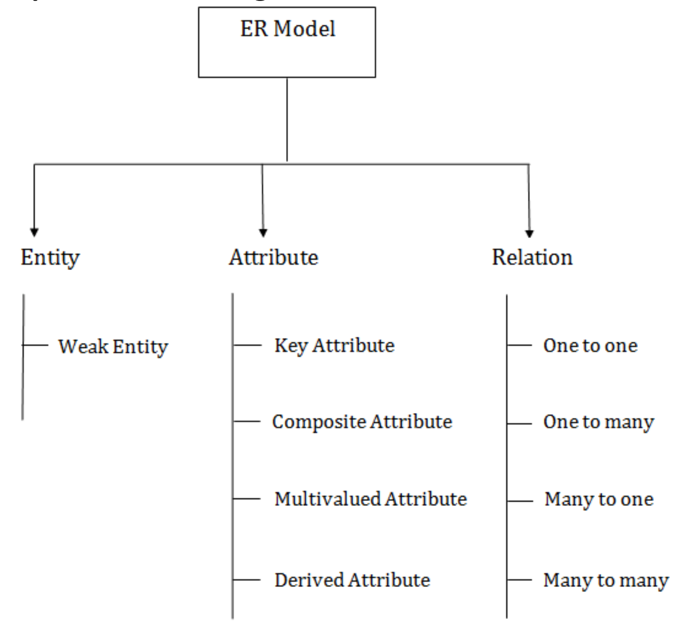
ER (Entity Relationship) Diagram in DBMS

- ER model stands for an Entity-Relationship model. It is a high-level data model. This model is used to define the data elements and relationship for a specified system.
- It develops a conceptual design for the database. It also develops a very simple and easy to design view of data.
- In ER modeling, the database structure is portrayed as a diagram called an entity-relationship diagram.

For example, Suppose we design a school database. In this database, the student will be an entity with attributes like address, name, id, age, etc. The address can be another entity with attributes like city, street name, pin code, etc and there will be a relationship between them.



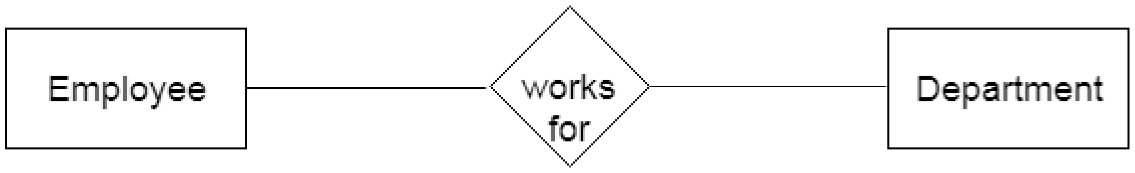
Component of ER Diagram



1. Entity:

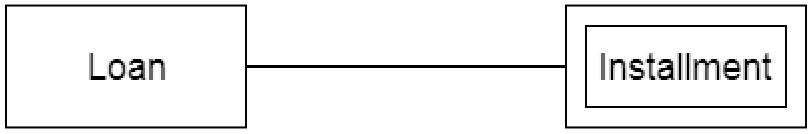
An entity may be any object, class, person or place. In the ER diagram, an entity can be represented as rectangles.

Consider an organization as an example- manager, product, employee, department etc. can be taken as an entity.



a. Weak Entity

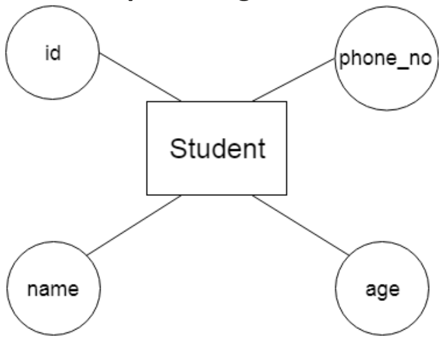
An entity that depends on another entity called a weak entity. The weak entity doesn't contain any key attribute of its own. The weak entity is represented by a double rectangle.



2. Attribute

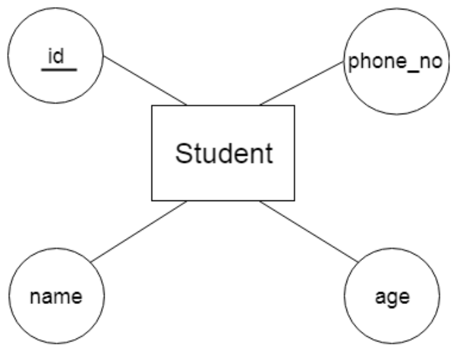
The attribute is used to describe the property of an entity. Eclipse is used to represent an attribute.

For example, id, age, contact number, name, etc. can be attributes of a student.



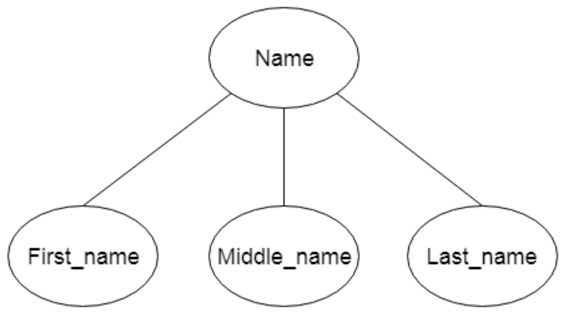
a. Key Attribute

The key attribute is used to represent the main characteristics of an entity. It represents a primary key. The key attribute is represented by an ellipse with the text underlined.



b. Composite Attribute

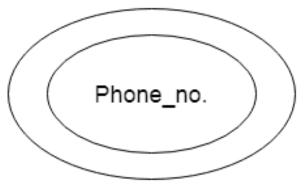
An attribute that composed of many other attributes is known as a composite attribute. The composite attribute is represented by an ellipse, and those ellipses are connected with an ellipse.



c. Multivalued Attribute

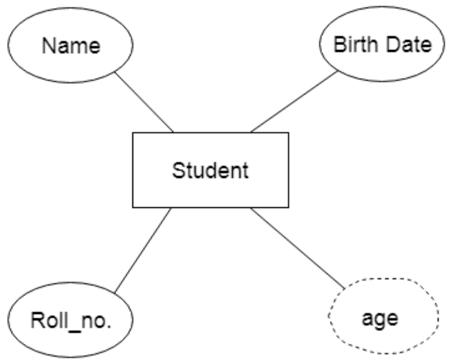
An attribute can have more than one value. These attributes are known as a multivalued attribute. The double oval is used to represent multivalued attribute.

For example, a student can have more than one phone number.



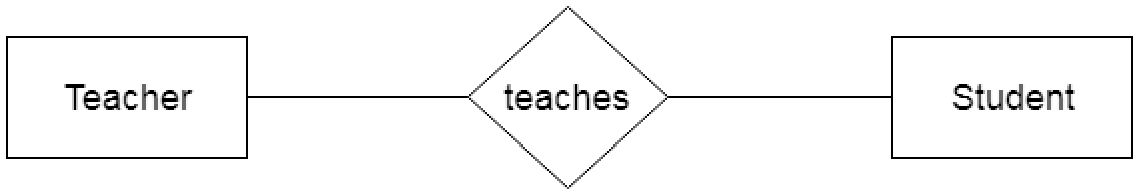
d. Derived Attribute
An attribute that can be derived from other attribute is known as a derived attribute. It can be represented by a dashed ellipse.

For example, A person's age changes over time and can be derived from another attribute like Date of birth.



3. Relationship

A relationship is used to describe the relation between entities. Diamond or rhombus is used to represent the relationship.

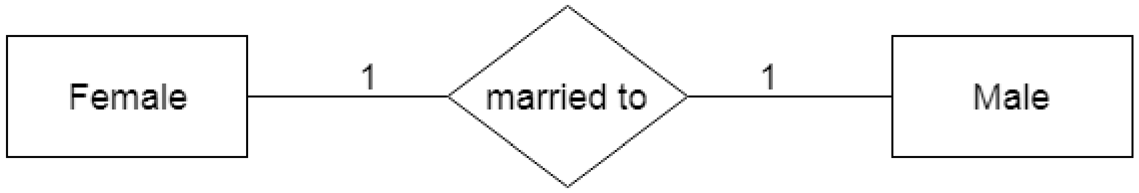


Types of relationship are as follows:

a. One-to-One Relationship

When only one instance of an entity is associated with the relationship, then it is known as one to one relationship.

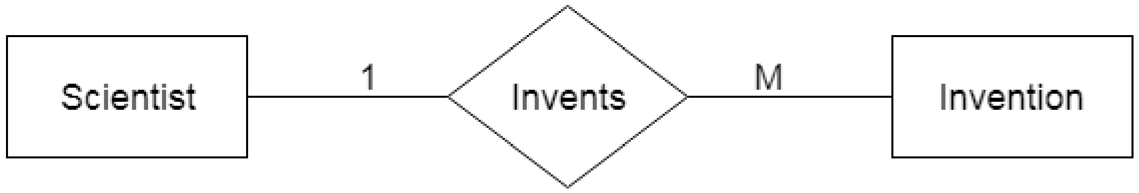
For example, A female can marry to one male, and a male can marry to one female.



b. One-to-many relationship

When only one instance of the entity on the left, and more than one instance of an entity on the right associates with the relationship then this is known as a one-to-many relationship.

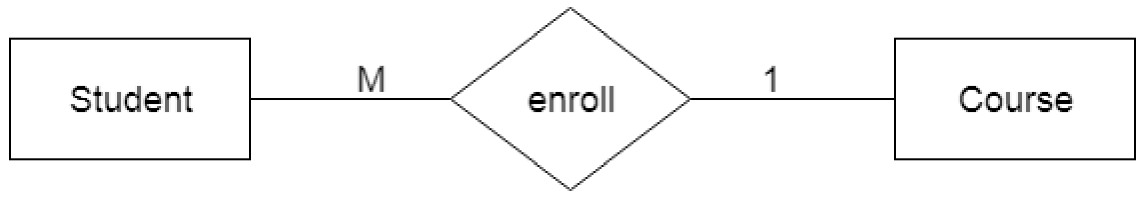
For example, Scientist can invent many inventions, but the invention is done by the only specific scientist.



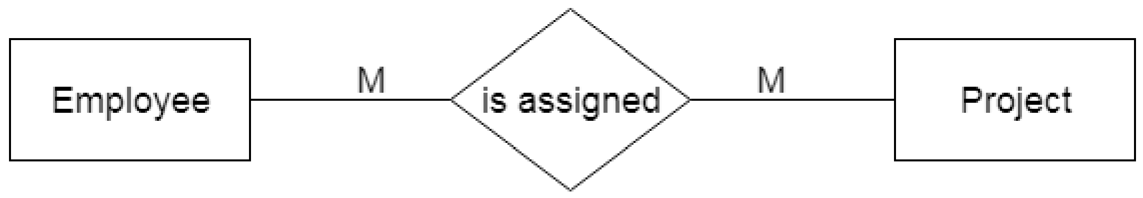
c. Many-to-one relationship

When more than one instance of the entity on the left, and only one instance of an entity on the right associates with the relationship then it is known as a many-to-one relationship.

For example, Student enrolls for only one course, but a course can have many students



d. Many-to-many relationship
When more than one instance of the entity on the left, and more than one instance of an entity on the right associates with the relationship then it is known as a many-to-many relationship.
For example, Employee can assign by many projects and project can have many employees.



Introduction of ER Model

Peter Chen developed the ER diagram in 1976. The ER model was created to provide a simple and understandable model for representing the structure and logic of databases. It has since evolved into variations such as the Enhanced ER Model and the Object Relationship 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.

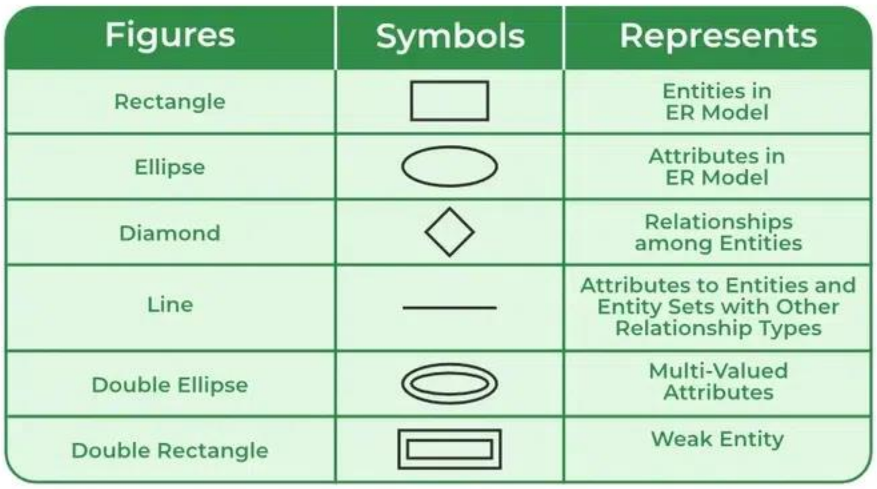
The Entity Relationship Diagram explains the relationship among the entities present in the database. ER models are used to model real-world objects like a person, a car, or a company and the relation between these real-world objects. In short, the ER Diagram is the structural format of the database.

Why Use ER Diagrams In DBMS?
ER diagrams represent the E-R model in a database, making them easy to convert into relations (tables).
ER diagrams provide the purpose of real-world modeling of objects which makes them intently useful.
ER diagrams require no technical knowledge and no hardware support.
These diagrams are very easy to understand and easy to create even for a naive user.
It gives a standard solution for visualizing the data logically.
A solid grasp of the ER Model is crucial for excelling in exams like GATE, where database management is a key topic.

Symbols Used in ER Model

ER Model is used to model the logical view of the system from a data perspective which consists of these symbols:

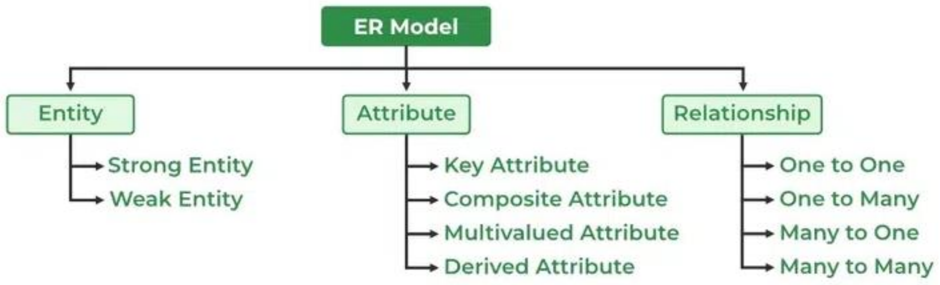
- Rectangles: Rectangles represent Entities in the ER Model.
- Ellipses: Ellipses represent Attributes in the ER Model.
- Diamond: Diamonds represent Relationships among Entities.
- Lines: Lines represent attributes to entities and entity sets with other relationship types.
- Double Ellipse: Double Ellipses represent Multi-Valued Attributes.
- Double Rectangle: Double Rectangle represents a Weak Entity.



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

Components of ER Diagram

ER Model consists of Entities, Attributes, and Relationships among Entities in a Database System.

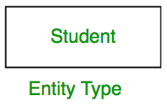


What is Entity?

An Entity may be an object with a physical existence – a particular person, car, house, or employee – or it may be an object with a conceptual existence – a company, a job, or a university course.

What is Entity Set?

An Entity is an object of Entity Type and a set of all entities is called an entity set. For Example, E1 is an entity having Entity Type Student and the set of all students is called Entity Set. In ER diagram, Entity Type is represented as:



E1
E2
E3
Entity Set

Types of Entity

There are two types of entity:

1. Strong Entity

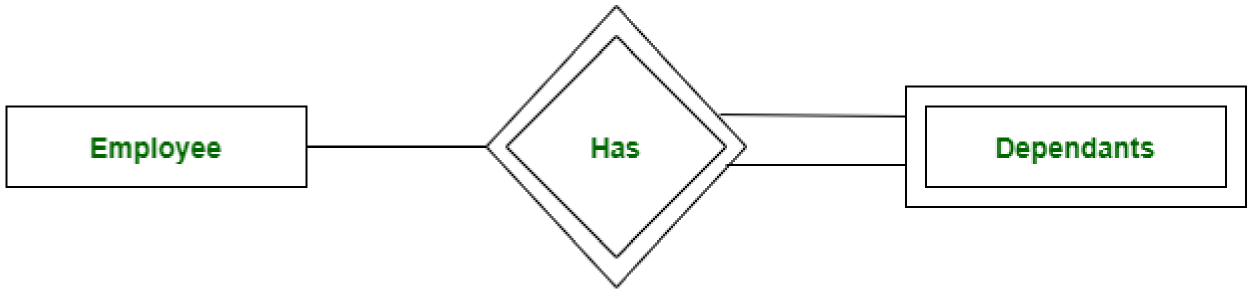
A Strong Entity is a type of entity that has a key Attribute. Strong Entity does not depend on other Entity in the Schema. It has a primary key, that helps in identifying it uniquely, and it is represented by a rectangle. These are called Strong Entity Types.

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.



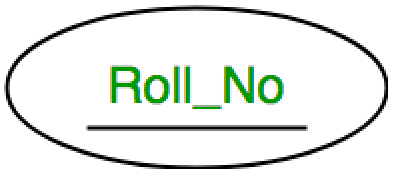
What is Attributes?

Attributes are the properties that define the entity type. For example, Roll_No, Name, DOB, Age, Address, and Mobile_No are the attributes that define entity type Student. In ER diagram, the attribute is represented by an oval.

Types of Attributes

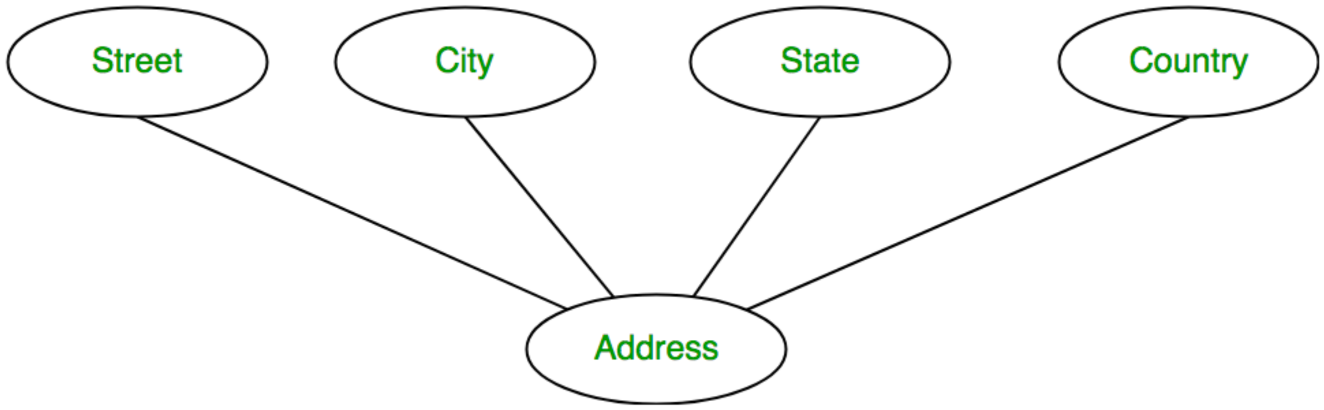
1. Key Attribute

The attribute which uniquely identifies each entity in the entity set is called the key attribute. For example, Roll_No will be unique for each student. In ER diagram, the key attribute is represented by an oval with underlying lines.



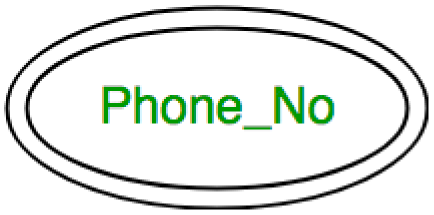
2. Composite Attribute

An attribute composed of many other attributes is called a composite attribute. For example, the Address attribute of the student Entity type consists of Street, City, State, and Country. In ER diagram, the composite attribute is represented by an oval comprising of ovals.



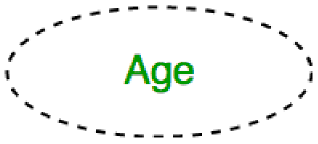
3. Multivalued Attribute

An attribute consisting of more than one value for a given entity. For example, Phone_No (can be more than one for a given student). In ER diagram, a multivalued attribute is represented by a double oval.

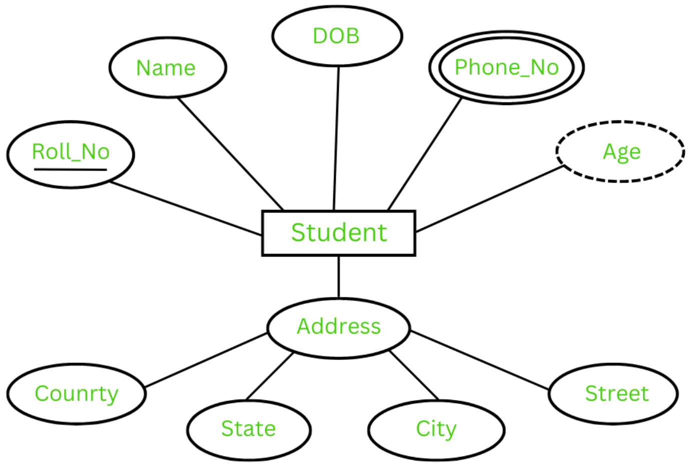


4. Derived Attribute

An attribute that can be derived from other attributes of the entity type is known as a derived attribute. e.g.; Age (can be derived from DOB). In ER diagram, the derived attribute is represented by a dashed oval.

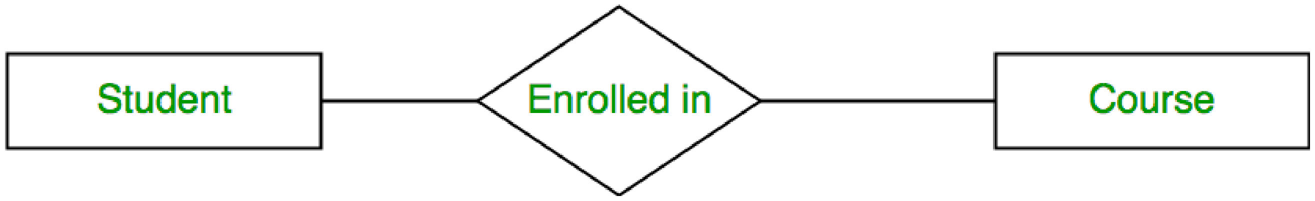


The Complete Entity Type Student with its Attributes can be represented as:

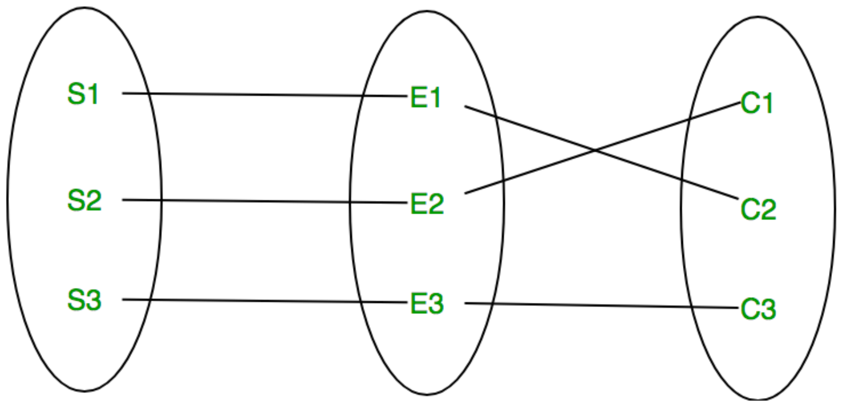


Relationship Type and Relationship Set

A Relationship Type represents the association between entity types. For example, 'Enrolled in' is a relationship type that exists between entity type Student and Course. In ER diagram, the relationship type is represented by a diamond and connecting the entities with lines.



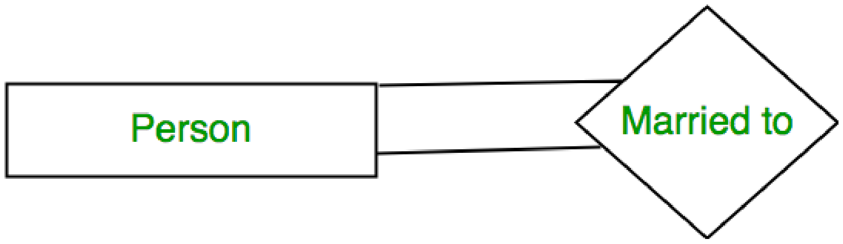
A set of relationships of the same type is known as a relationship set. The following relationship set depicts S1 as enrolled in C2, S2 as enrolled in C1, and S3 as registered in C3.



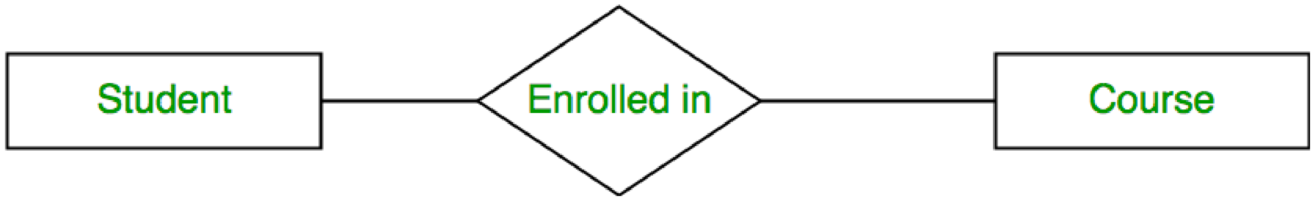
Degree of a Relationship Set

The number of different entity sets participating in a relationship set is called the degree of a relationship set.

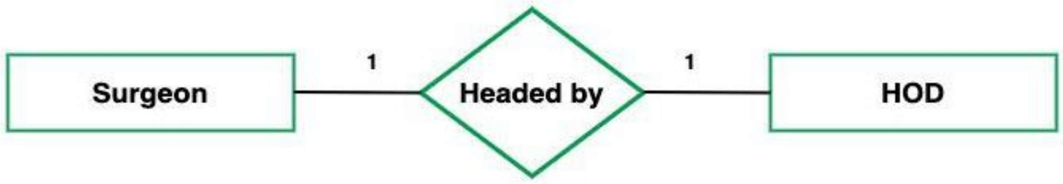
1. Unary Relationship: When there is only ONE entity set participating in a relation, the relationship is called a unary relationship. For example, one person is married to only one person.



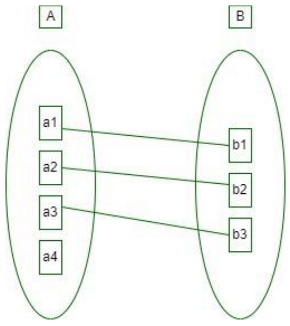
2. Binary Relationship: When there are TWO entities set participating in a relationship, the relationship is called a binary relationship. For example, a Student is enrolled in a Course.



3. Ternary Relationship: When there are three entity sets participating in a relationship, the relationship is called a ternary relationship.
4. N-ary Relationship: When there are n entities set participating in a relationship, the relationship is called an n-ary relationship.
What is Cardinality?
The number of times an entity of an entity set participates in a relationship set is known as cardinality . Cardinality can be of different types:
1. One-to-One: 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.



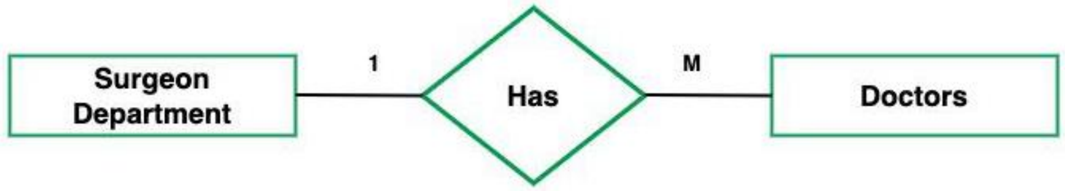
Using Sets, it can be represented as:



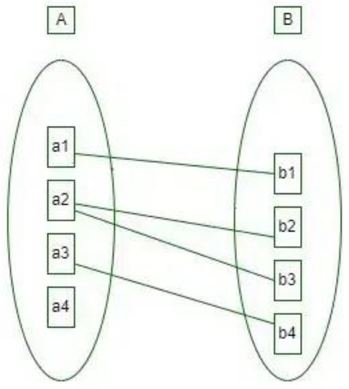
2. One-to-Many: In one-to-many mapping as well where each entity can be related to more than one entity and the total number of tables that can be used in this is 2. Let us assume that one surgeon department

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.

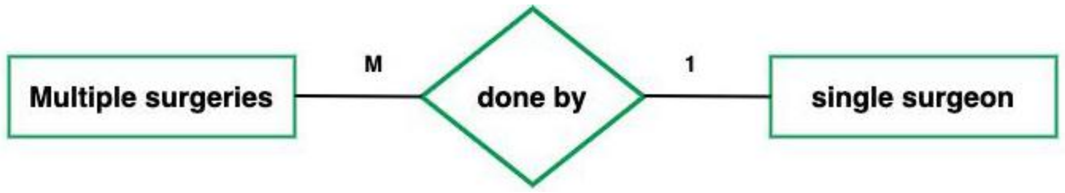


Using sets, one-to-many cardinality can be represented as:

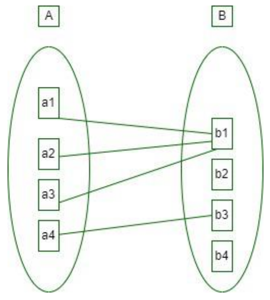


3. Many-to-One: When entities in one entity set can take part only once in the relationship set and entities in other entity sets can take part more than once in the relationship set, cardinality is many to one. Let us assume that a student can take only one course but one course can be taken by many 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.

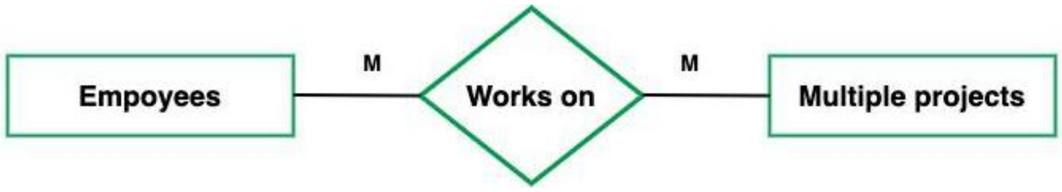


Using Sets, it can be represented as:

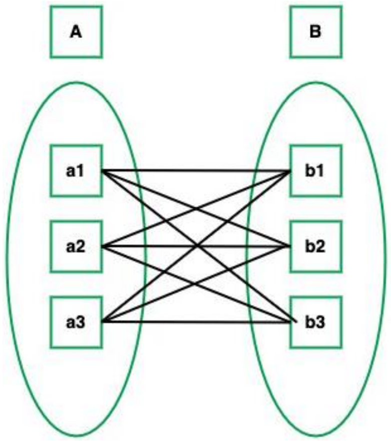


In this case, each student is taking only 1 course but 1 course has been taken by many students.

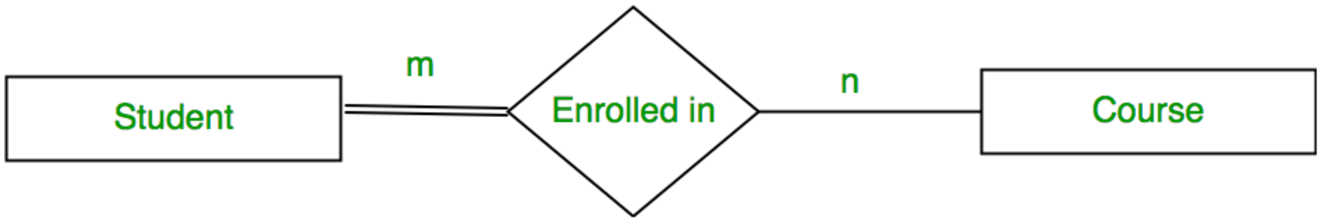
4. Many-to-Many: When entities in all entity sets can take part more than once in the relationship cardinality is many to many. Let us assume that a student can take more than one course and one course can be taken by many students. So the relationship will be many to many.
the total number of tables that can be used in this is 3.



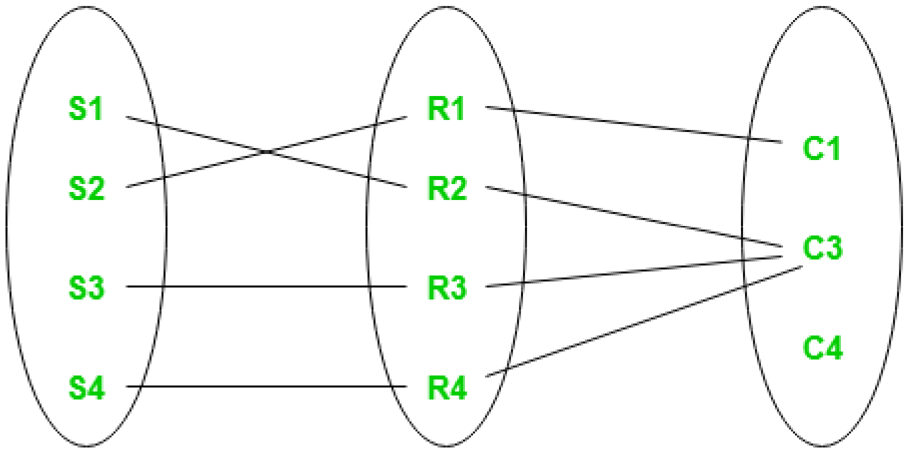
Using Sets, it can be represented as:



In this example, student S1 is enrolled in C1 and C3 and Course C3 is enrolled by S1, S3, and S4. So it is many-to-many relationships.
Participation Constraint
Participation Constraint 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.
The diagram depicts the 'Enrolled in' relationship set with Student Entity set having total participation and Course Entity set having partial participation.



Using Set, it can be represented as,



Every student in the Student Entity set participates in a relationship but there exists a course C4 that is not taking part in the relationship.

How to Draw ER Diagram?

- The very first step is Identifying all the Entities, and place them in a Rectangle, and labeling them accordingly.
- The next step is to identify the relationship between them and place them accordingly using the Diamond, and make sure that, Relationships are not connected to each other.
- Attach attributes to the entities properly.
- Remove redundant entities and relationships.
- Add proper colors to highlight the data present in the database.