Introduction to Database System Concepts and Architecture

Data:

- It is a collection of information.
- The facts that can be recorded and which have implicit meaning known as 'data'.
- Example: Customer ----- 1. cname.
  2. cno.
  3. ccity

Database:

- It is a collection of interrelated data.
- These can be stored in the form of tables.
- A database can be of any size and varying complexity.
- A database may be generated and manipulated manually or it may be computerized.
- Example: Customer database consists the fields as cname, cno, and ccity

Cname	Cno	Ccity

Database System:

- It is computerized system, whose overall purpose is to maintain the information and to make that the information is available on demand.

Advantages:

1. Redundancy can be reduced.
2. Inconsistency can be avoided.
3. Data can be shared.
4. Standards can be enforced.
5. Security restrictions can be applied.
6. Integrity can be maintained.
7. Data gathering can be possible.
8. Requirements can be balanced.

Database Management System (DBMS):

- It is a collection of programs that enables user to create and maintain a database.
- In other words, it is general-purpose software that provides the users with the processes of defining, constructing and manipulating the database for various applications.

  Disadvantages in File Processing

  - Data redundancy and inconsistency.
  - Difficult in accessing data. Data isolation.

- Data integrity.
- Concurrent access is not possible.
- Security Problems.

Advantages of DBMS:

- Data Independence.
- Efficient Data Access.
- Data Integrity and security.
- Data administration.
- Concurrent access and Crash recovery.
- Reduced Application Development Time.

Applications Database Applications:

- Banking: all transactions
- Airlines: reservations, schedules
- Universities: registration, grades
- Sales: customers, products, purchases Online retailers: order tracking, customized recommendations
- Manufacturing: production, inventory, orders, supply chain
- Human resources: employee records, salaries, tax deductions

Database Users:

Database users are categorized based up on their interaction with the data base.
These are seven types of data base users in DBMS.

1. **Database Administrator (DBA):**
   - Database Administrator (DBA) is a person/team who defines the schema and also controls the 3 levels of database.
   - The DBA will then create a new account id and password for the user if he/she need to access the data base.
   - DBA is also responsible for providing security to the data base and he allows only the authorized users to access/modify the data base.
     - DBA also monitors the recovery and back up and provide technical support.
     - The DBA has a DBA account in the DBMS which called a system or super-user account.
     - DBA repairs damage caused due to hardware and/or software failures.

2. **Naive / Parametric End Users:**
   - Parametric End Users are the unsophisticated who don't have any DBMS knowledge but they frequently use the data base applications in their daily life to get the desired results.
   - For examples, Railway's ticket booking users are naive users. Clerks in any bank is a naive user because they don't have any DBMS knowledge but they still use the database and perform their given task.

3. **System Analyst:**
   - System Analyst is a user who analyzes the requirements of parametric end users. They check whether all the requirements of end users are satisfied.

4. **Sophisticated Users:**
   - Sophisticated users can be engineers, scientists, business analyst, who are familiar with the database.
   - They can develop their own data base applications according to their requirement.

- They don’t write the program code but they interact the data base by writing SQL queries directly through the query processor.

5. **Data Base Designers:**
   - Data Base Designers are the users who design the structure of data base which includes tables, indexes, views, constraints, triggers, stored procedures. He/she controls what data must be stored and how the data items to be related.
6. **Application Program:**
   - Application Program are the back end programmers who writes the code for the application programs.
   - They are the computer professionals.
   - These programs could be written in Programming languages such as Visual Basic, Developer, C, FORTRAN, COBOL etc.
7. **Casual Users / Temporary Users:**
   - Casual Users are the users who occasionally use/access the data base but each time when they access the data base they require the new information.
   - For example, Middle or higher level manager.

Characteristics of Database Approach:

- A number of **characteristics** distinguish the database approach from the much older approach of programming with files.
- In traditional **file processing**, each user defines and implements the files needed for a specific software application as part of programming the application.
- The main **characteristics** of the database approach versus the file-processing approach are the following:
  - Self-describing nature of a database system.
  - Insulation between programs and data, and data abstraction.
  - Support of multiple views of the data.
  - Sharing of data and multiuser transaction processing.

The database approach has some very characteristic features which are discussed in detail below:

1. **Structured and Described Data:**
   - Fundamental feature of the database approach is that the database system does not only contain the data but also the complete definition and description of these data.
   - These descriptions are basically details about the extent, the structure, the type and the format of all data and, additionally, the relationship between the data. This kind of stored data is called metadata ("data about data").
2. **Separation of Data and Applications:**
   - Application software does not need any knowledge about the physical data storage like encoding, format, storage place, etc. It only communicates with the **management system of a database (DBMS)** via a standardized interface with the help of a standardized language like SQL.
3. **Data Integrity:**
   - Data integrity is a byword for the quality and the reliability of the data of a database system.
   - In a broader sense data integrity includes also the protection of the database from unauthorized access **(confidentiality)** and unauthorized changes. Data reflect facts of the real world.

4. **Transactions:**
   - A transaction is a bundle of actions which are done within a database to bring it from one consistent state to a new consistent state. In between the data are inevitable inconsistent.
5. **Data Persistence:**
   - Data persistence means that in a DBMS all data is maintained as long as it is not deleted explicitly.
   - The life span of data needs to be determined directly or indirectly be the user and must not be dependent on system features.
   - Additionally, data once stored in a database must not be lost. Changes of a database which are done by a transaction are persistent.
   - When a transaction is finished even a system crash cannot put the data in danger.

Actors on the Scene:

The people, whose jobs involve the day-to-day use of a database are called as 'Actors on the scene', listed as below.

1. **Database Administrators (DBA):**
   - The DBA is responsible for authorizing access to the database, for Coordinating and monitoring its use and for acquiring software and hardware resources as needed.
   - These are the people, who maintain and design the database daily. DBA is responsible for the following issues.
     - Design of the conceptual and physical schemas:
       - The DBA is responsible for interacting with the users of the system to understand what data is to be stored in the DBMS and how it is likely to be used.
       - The DBA creates the original schema by writing a set of definitions and is Permanently stored in the 'Data Dictionary'.
     - Security and Authorization:
       - The DBA is responsible for ensuring the unauthorized data access is not permitted.
       - The granting of different types of authorization allows the DBA to regulate which parts of the database various users can access.
     - Storage structure and Access method definition:
       - The DBA creates appropriate storage structures and access methods by writing a set of definitions, which are translated by the DDL compiler.
     - Data Availability and Recovery from Failures:
       - The DBA must take steps to ensure that if the system fails, users can continue to access as much of the uncorrupted data as possible. The DBA also work to restore the data to consistent state.
     - Database Tuning:
       - The DBA is responsible for modifying the database to ensure adequate Performance as requirements change.
     - Integrity Constraint Specification:
       - The integrity constraints are kept in a special system structure that is consulted by the DBA whenever an update takes place in the system.
2. **Database Designers:**
   - Database designers are responsible for identifying the data to be stored in the database and for choosing appropriate structures to represent and store this data.

3. **End Users:**
   - People who wish to store and use data in a database. End users are the people whose jobs require access to the database for querying, updating and generating reports, listed as below.
     - Casual End users:
       - These people occasionally access the database, but they may need different information each time.
     - Naive or Parametric End Users
       - Their job function revolves around constantly querying and updating the database using standard types of queries and updates.
     - Sophisticated End Users:
       - These include Engineers, Scientists, Business analyst and others familiarize to implement their applications to meet their complex requirements.
     - Standalone End users:
       - These people maintain personal databases by using ready-made program packages that provide easy to use menu based interfaces.

Advantages of Using a DBMS

Some of them are given as following below.

1. **Better Data Transferring:**
   - Database management creates a place where users have an advantage of more and better managed data.
   - Thus making it possible for end-users to have a quick look and to respond fast to any changes made in their environment.

2. **Better Data Security:**
   - As number of users increases data transferring or data sharing rate also increases thus increasing the risk of data security.
   - It is widely used in corporation world where companies invest money, time and effort in large amount to ensure data is secure and is used properly.
   - A Database Management System (DBMS) provide a better platform for data privacy and security policies thus, helping companies to improve Data Security.

3. **Better data integration:**
   - Due to Database Management System we have an access to well managed and synchronized form of data thus it makes data handling very easy and gives integrated view of how a particular organization is working and also helps to keep a track on how one segment of the company affects other segment.

4. **Minimized Data Inconsistency:**
   - Data inconsistency occurs between files when different versions of the same data appear in different places.
   - For Example, data inconsistency occurs when a student name is saved as “John Wayne” on a main computer of school but on teacher registered system same student name is “William J.
   - Wayne”, or when the price of a product is $86.95 in local system of company and its National sales office system shows the same product price as $84.95.
   - So if a database is properly designed then Data inconsistency can be greatly reduced hence minimizing data inconsistency.

5. **Faster data Access:**
   - The Data base management system (DBMS) helps to produce quick answers to database queries thus making data accessing faster and more accurate.
   - For example, to read or update the data.
   - For example, end users, when dealing with large amounts of sale data, will have enhanced access to the data, enabling faster sales cycle.
   - Some queries may be like:
     - What is the increase of the sale in last three months?
     - What is the bonus given to each of the salespeople in last five months?
     - How many customers have credit score of 850 or more?
6. **Better decision making:**
   - Due to DBMS now we have Better managed data and Improved data accessing because of which we can generate better quality information hence on this basis better decisions can be made.
   - Better Data quality improves accuracy, validity and time it takes to read data. DBMS does not guarantee data quality, it provides a framework to make it is easy to improve data quality .

7. **Increased end-user productivity:**
   - The data which is available with the help of combination of tools which transform data into useful information, helps end user to make quick, informative and better decisions that can make difference between success and failure in the global economy.

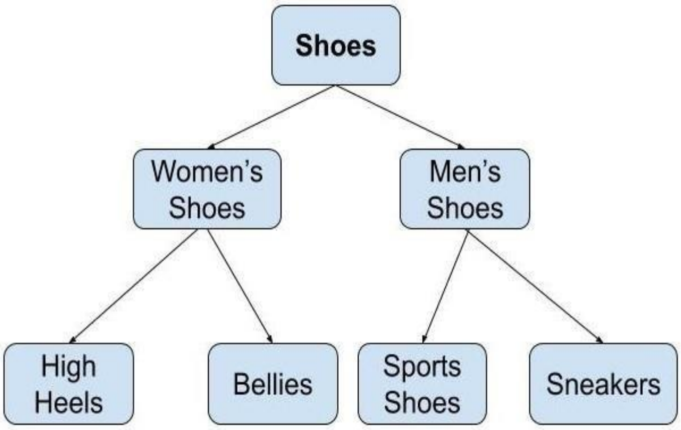
8. **Simple:**
   - Data base management system (DBMS) gives simple and clear logical view of data. Many operations like insertion, deletion or creation of file or data are easy to implement.

Data Models:

- Data Model gives us an idea that how the final system will look like after its complete implementation.
- It defines the data elements and the relationships between the data elements. Data Models are used to show how data is stored, connected, accessed and updated in the database management system.
- Here, we use a set of symbols and text to represent the information so that members of the organization can communicate and understand it.
- Though there are many data models being used nowadays but the Relational model is the most widely used model.
- Apart from the Relational model, there are many other types of data models about which we will study in details in this blog.
- Some of the Data Models in DBMS are:
  1. Hierarchical Model
  2. Network Model
  3. Entity-Relationship Model
  4. Relational Model
  5. Object-Oriented Data Model
  6. Object-Relational Data Model
  7. Flat Data Model
  8. Semi-Structured Data Model

9. Associative Data Model
10. Context Data Model

1. Hierarchical Model
    - Hierarchical Model was the first DBMS model. This model organizes the data in the hierarchical tree structure.
    - The hierarchy starts from the root which has root data and then it expands in the form of a tree adding child node to the parent node.
    - This model easily represents some of the real-world relationships like food recipes, sitemap of a website etc.
    - Example: We can represent the relationship between the shoes present on a shopping website in the following way:



Hierarchical Model

Features of a Hierarchical Model

1. **One-to-many relationship:** The data here is organized in a tree-like structure where the one-to-many relationship is between the datatypes. Also, there can be only one path from parent to any node. **Example:** In the above example, if we want to go to the node sneakers we only have one path to reach there i.e through men's shoes node.
2. **Parent-Child Relationship:** Each child node has a parent node but a parent node can have more than one child node. Multiple parents are not allowed.
3. **Deletion Problem:** If a parent node is deleted then the child node is automatically deleted.
4. **Pointers:** Pointers are used to link the parent node with the child node and are used to navigate between the stored data. Example: In the above example the 'shoes' node points to the two other nodes 'women shoes' node and 'men's shoes' node.

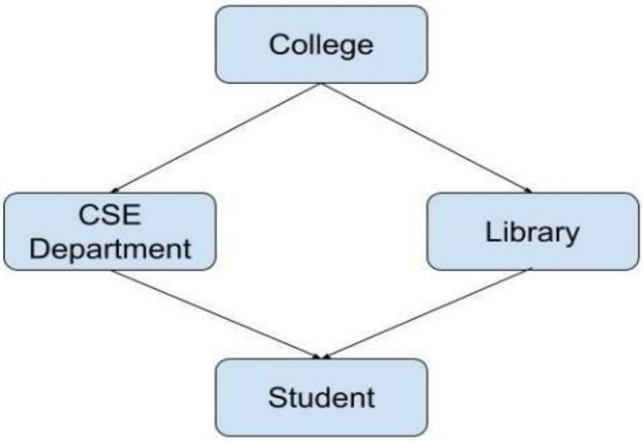
Advantages of Hierarchical Model

- It is very simple and fast to traverse through a tree-like structure.
- Any change in the parent node is automatically reflected in the child node so, the integrity of data is maintained.

Disadvantages of Hierarchical Model

- Complex relationships are not supported.
- As it does not support more than one parent of the child node so if we have some complex relationship where a child node needs to have two parent node then that can't be represented using this model.
- If a parent node is deleted, then the child node is automatically deleted

2. **Network Model**
    - This model is an extension of the hierarchical model. It was the most popular model before the relational model.
    - This model is the same as the hierarchical model, the only difference is that a record can have more than one parent. It replaces the hierarchical tree with a graph.
    - **Example:** In the example below we can see that node student has two parents i.e. CSE Department and Library. This was earlier not possible in the hierarchical model.



Network Model

Features of a Network Model

1. **Ability to Merge more Relationships:** In this model, as there are more relationships so data is more related. This model has the ability to manage one-to-one relationships as well as many-to-many relationships.
2. **Many paths:** As there are more relationships so there can be more than one path to the same record. This makes data access fast and simple.
3. **Circular Linked List:** The operations on the network model are done with the help of the circular linked list. The current position is maintained with the help of a program and this position navigates through the records according to the relationship.

Advantages of Network Model

- The data can be accessed faster as compared to the hierarchical model. This is because the data is more related in the network model and there can be more than one path to reach a particular node. So the data can be accessed in many ways.
- As there is a parent-child relationship so data integrity is present. Any change in parent record is reflected in the child record.

Disadvantages of Network Model

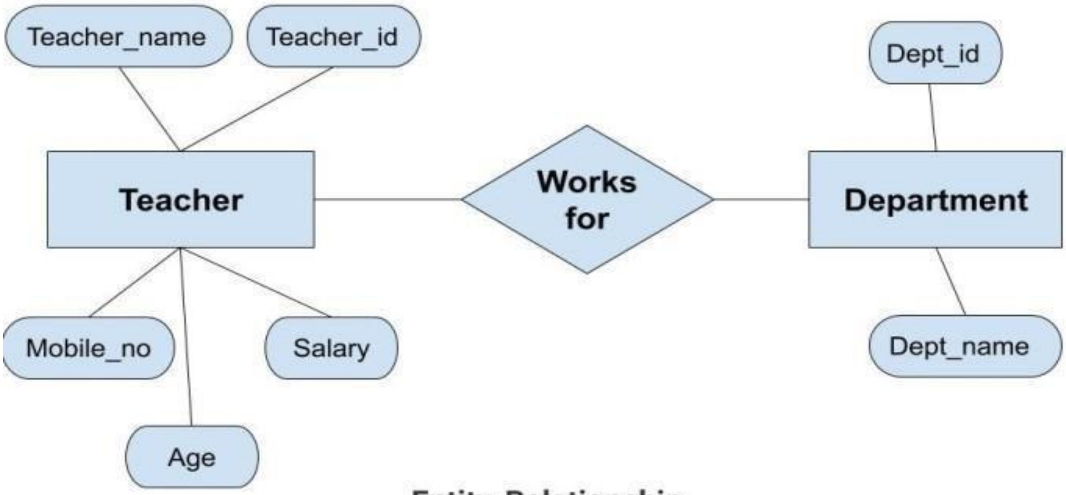
- As more and more relationships need to be handled the system might get complex. So, a user must be having detailed knowledge of the model to work with the model.
- Any change like update, deletion, insertion is very complex.

3. **Entity-Relationship Model**

- Entity-Relationship Model or simply ER Model is a high-level data model diagram.
- In this model, we represent the real-world problem in the pictorial form to make it easy for the stakeholders to understand.
- It is also very easy for the developers to understand the system by just looking at the ER diagram. We use the ER diagram as a visual tool to represent an ER Model.

- ER diagram has the following three components:
- **Entities:** Entity is a real-world thing. It can be a person, place, or even a concept. Example: Teachers, Students, Course, Building, Department, etc are some of the entities of a School Management System.
- **Attributes:** An entity contains a real-world property called attribute. This is the characteristics of that attribute. Example: The entity teacher has the property like teacher id, salary, age, etc.
- **Relationship:** Relationship tells how two attributes are related. Example: Teacher works for a department.

Example:



Entity-Relationship Model

- In the above diagram, the entities are Teacher and Department. The attributes of **Teacher** entity are Teacher_Name, Teacher_id, Age, Salary, Mobile_Number. The attributes of entity **Department** entity are Dept_id, Dept_name. The two entities are connected using the relationship. Here, each teacher works for a department.

Features of ER Model

- **Graphical Representation for Better Understanding:** It is very easy and simple to understand so it can be used by the developers to communicate with the stakeholders.
- **ER Diagram:** ER diagram is used as a visual tool for representing the model.
- **Database Design:** This model helps the database designers to build the database and is widely used in database design.

Advantages of ER Model

- **Simple:** Conceptually ER Model is very easy to build. If we know the relationship between the attributes and the entities we can easily build the ER Diagram for the model.
- **Effective Communication Tool**: This model is used widely by the database designers for communicating their ideas.
- **Easy Conversion to any Model**: This model maps well to the relational model and can be easily converted relational model by converting the ER model to the table. This model can also be converted to any other model like network model, hierarchical model etc.

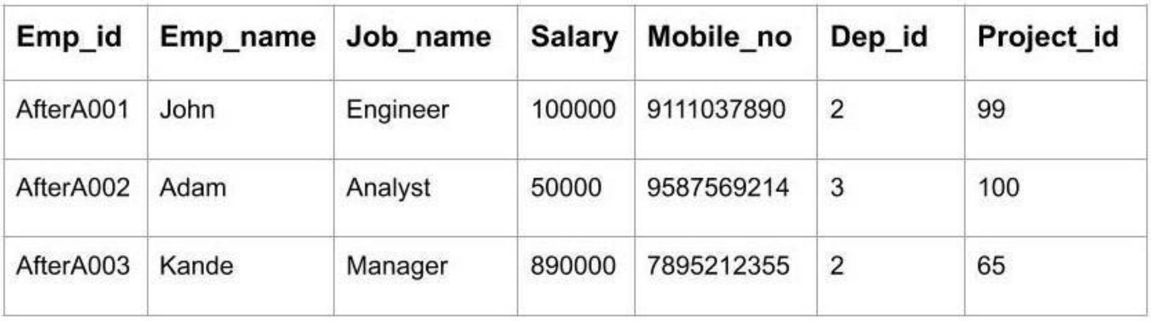
Disadvantages of ER Model

- **No industry standard for notation:** There is no industry standard for developing an ER model. So one developer might use notations which are not understood by other developers.

- **Hidden information:** Some information might be lost or hidden in the ER model. As it is a high-level view so there are chances that some details of information might be hidden.

4. Relational Model

- Relational Model is the most widely used model. In this model, the data is maintained in the form of a two-dimensional table. All the information is stored in the form of row and columns.
- The basic structure of a relational model is tables. So, the tables are also called relations in the relational model. **Example:** In this example, we have an Employee table.



Emp_id	Emp_name	Job_name	Salary	Mobile_no	Dep_id	Project_id
AfterA001	John	Engineer	100000	9111037890	2	99
AfterA002	Adam	Analyst	50000	9587569214	3	100
AfterA003	Kande	Manager	890000	7895212355	2	65

EMPLOYEE TABLE

Features of Relational Model

- **Tuples**: Each row in the table is called tuple. A row contains all the information about any instance of the object. In the above example, each row has all the information about any specific individual like the first row has information about John.
- **Attribute or field:** Attributes are the property which defines the table or relation. The values of the attribute should be from the same domain. In the above example, we have different attributes of the employee like Salary, Mobile_no, etc.

Advnatages of Relational Model

- **Simple:** This model is more simple as compared to the network and hierarchical model.
- **Scalable:** This model can be easily scaled as we can add as many rows and columns we want.
- **Structural Independence:** We can make changes in database structure without changing the way to access the data. When we can make changes to the database structure without affecting the capability to DBMS to access the data we can say that structural independence has been achieved.

Disadvantages of Relatinal Model

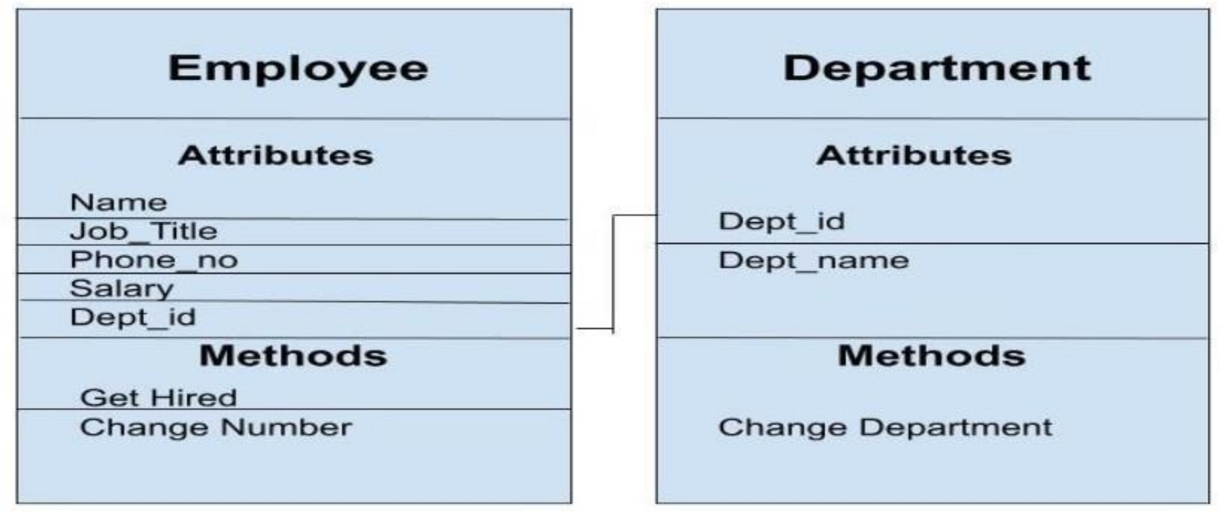
- **Hardware Overheads:** For hiding the complexities and making things easier for the user this model requires more powerful hardware computers and data storage devices.
- **Bad Design:** As the relational model is very easy to design and use. So the users don't need to know how the data is stored in order to access it. This ease of design can lead to the development of a poor database which would slow down if the database grows.

But all these disadvantages are minor as compared to the advantages of the relational model. These problems can be avoided with the help of proper implementation and organisation.

5. Object-Oriented Data Model

- The real-world problems are more closely represented through the object-oriented data model. In this model, both the data and relationship are present in a single structure known as an object.

- We can store audio, video, images, etc in the database which was not possible in the relational model(although you can store audio and video in relational database, it is adviced not to store in the relational database). In this model, two are more objects are connected through links.
- We use this link to relate one object to other objects. This can be understood by the example given below.



Object_Oriented_Model

- In the above example, we have two objects Employee and Department.
- All the data and relationships of each object are contained as a single unit.
- The attributes like Name, Job_title of the employee and the methods which will be performed by that object are stored as a single object.
- The two objects are connected through a common attribute i.e the Department_id and the communication between these two will be done with the help of this common id.

6. Flat Data Model

- It is a simple model in which the database is represented as a table consisting of rows and columns. To access any data, the computer has to read the entire table. This makes the modes slow and inefficient.

7. Semi-Structured Model

- Semi-structured model is an evolved form of the relational model.
- We cannot differentiate between data and schema in this model.
- **Example:** Web-Based data sources which we can't differentiate between the schema and data of the website.
- In this model, some entities may have missing attributes while others may have an extra attribute.
- This model gives flexibility in storing the data. It also gives flexibility to the attributes. **Example:** If we are storing any value in any attribute then that value can be either atomic value or a collection of values.

8. Associative Data Model

- Associative Data Model is a model in which the data is divided into two parts. Everything which has independent existence is called as an entity and the relationship among these entities are called association. The data divided into two parts are called items and links.
    - **Item:** Items contain the name and the identifier(some numeric value).

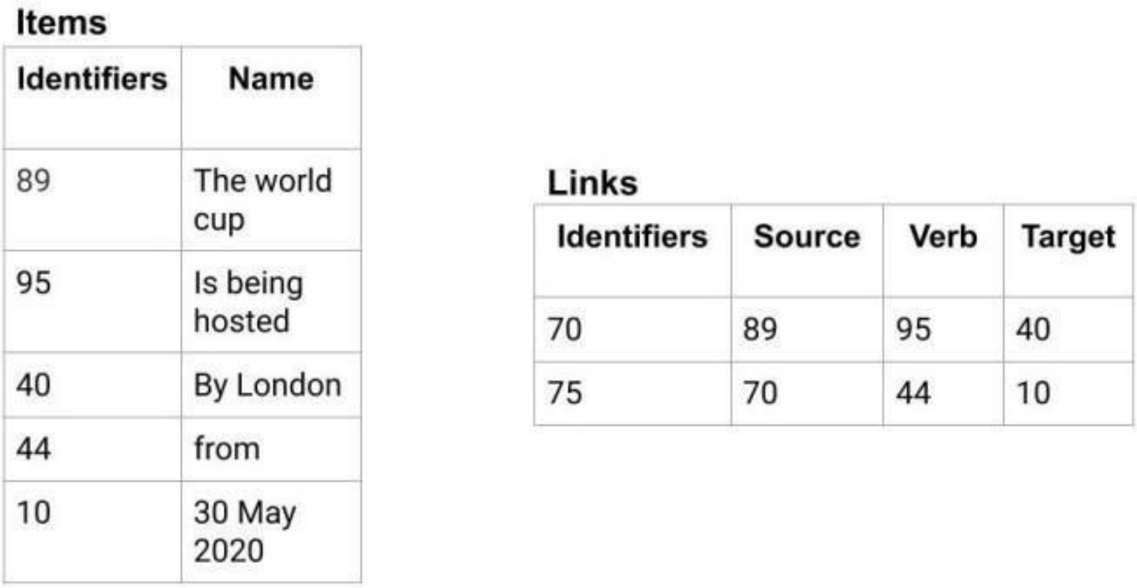
- **Links:** Links contain the identifier, source, verb and subject.

Example: Let us say we have a statement "The world cup is being hosted by London from 30 May 2020". In this data two links need to be stored:

1. The world cup is being hosted by London. The source here is 'the world cup', the verb 'is being' and the target is 'London'.

2.from 30 May 2020. The source here is the previous link, the verb is 'from' and the target is '30 May 2020'.

This is represented using the table as follows:



Items

Identifiers	Name
89	The world cup
95	Is being hosted
40	By London
44	from
10	30 May 2020

Links

Identifiers	Source	Verb	Target
70	89	95	40
75	70	44	10

ASSOCIATIVE MODEL

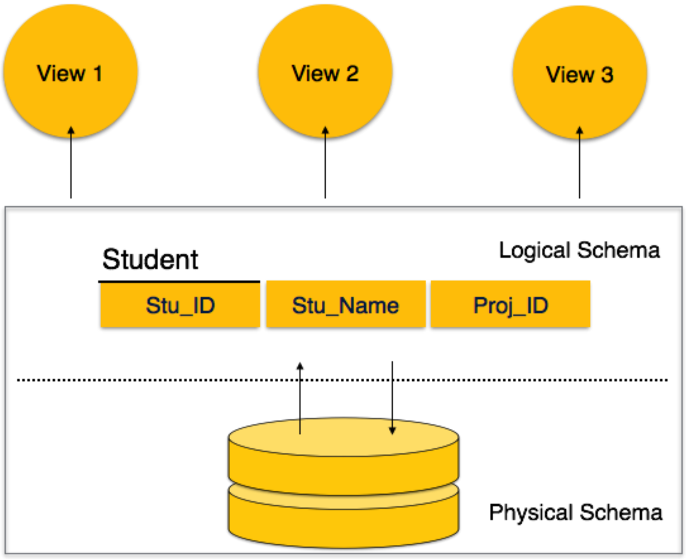
9. Context Data Model

- Context Data Model is a collection of several models. This consists of models like network model, relational models etc. Using this model, we can do various types of tasks which are not possible using any model alone.

Schema and Instances:

Database Schema:

- A database schema is the skeleton structure that represents the logical view of the entire database. It defines how the data is organized and how the relations among them are associated. It formulates all the constraints that are to be applied on the data.
- A database schema defines its entities and the relationship among them. It contains a descriptive detail of the database, which can be depicted by means of schema diagrams. It’s the database designers who design the schema to help programmers understand the database and make it useful.



A database schema can be divided broadly into two categories −

- **Physical Database Schema** − This schema pertains to the actual storage of data and its form of storage like files, indices, etc. It defines how the data will be stored in a secondary storage.
- **Logical Database Schema** − This schema defines all the logical constraints that need to be applied on the data stored. It defines tables, views, and integrity constraints.

Database Instance:

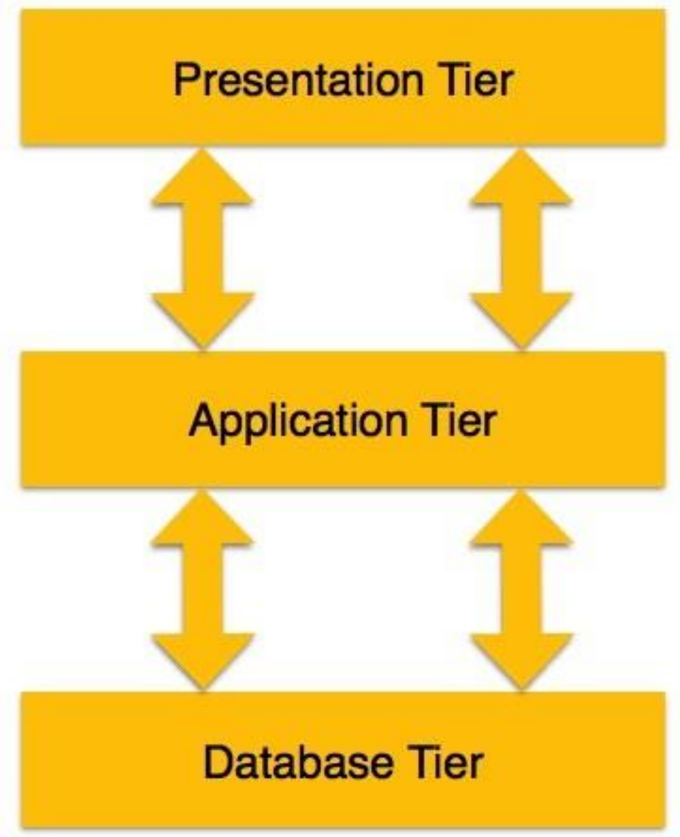
- It is important that we distinguish these two terms individually. Database schema is the skeleton of database. It is designed when the database doesn't exist at all. Once the database is operational, it is very difficult to make any changes to it.
- A database schema does not contain any data or information.
- A database instance is a state of operational database with data at any given time. It contains a snapshot of the database.
- Database instances tend to change with time. A DBMS ensures that its every instance (state) is in a valid state, by diligently following all the validations, constraints, and conditions that the database designers have imposed.

DBMS Architecture and Data Independence

- The design of a DBMS depends on its architecture.
- It can be centralized or decentralized or hierarchical.
- The architecture of a DBMS can be seen as either single tier or multi-tier.
- An n-tier architecture divides the whole system into related but independent **n** modules, which can be independently modified, altered, changed, or replaced.
- In 1-tier architecture, the DBMS is the only entity where the user directly sits on the DBMS and uses it. Any changes done here will directly be done on the DBMS itself.
- It does not provide handy tools for end-users. Database designers and programmers normally prefer to use single-tier architecture.
- If the architecture of DBMS is 2-tier, then it must have an application through which the DBMS can be accessed. Programmers use 2-tier architecture where they access the DBMS by means of an application.
- Here the application tier is entirely independent of the database in terms of operation, design, and programming.

3-tier Architecture

A 3-tier architecture separates its tiers from each other based on the complexity of the users and how they use the data present in the database. It is the most widely used architecture to design a DBMS.

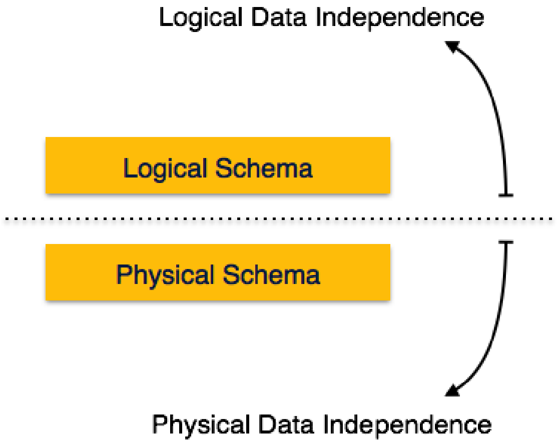


- **Database (Data) Tier** − At this tier, the database resides along with its query processing languages. We also have the relations that define the data and their constraints at this level.
- **Application (Middle) Tier** − At this tier reside the application server and the programs that access the database. For a user, this application tier presents an abstracted view of the database. End-users are unaware of any existence of the database beyond the application. At the other end, the database tier is not aware of any other user beyond the application tier. Hence, the application layer sits in the middle and acts as a mediator between the end-user and the database.
- **User (Presentation) Tier** − End-users operate on this tier and they know nothing about any existence of the database beyond this layer. At this layer, multiple views of the database can be provided by the application. All views are generated by applications that reside in the application tier.

Multiple-tier database architecture is highly modifiable, as almost all its components are independent and can be changed independently.

Data Independence

- A database system normally contains a lot of data in addition to users’ data. For example, it stores data about data, known as metadata, to locate and retrieve data easily.
- It is rather difficult to modify or update a set of metadata once it is stored in the database.
- But as a DBMS expands, it needs to change over time to satisfy the requirements of the users. If the entire data is dependent, it would become a tedious and highly complex job.



- Metadata itself follows a layered architecture, so that when we change data at one layer, it does not affect the data at another level. This data is independent but mapped to each other.

Logical Data Independence

- Logical data is data about database, that is, it stores information about how data is managed inside. For example, a table (relation) stored in the database and all its constraints, applied on that relation.
- Logical data independence is a kind of mechanism, which liberalizes itself from actual data stored on the disk. If we do some changes on table format, it should not change the data residing on the disk.

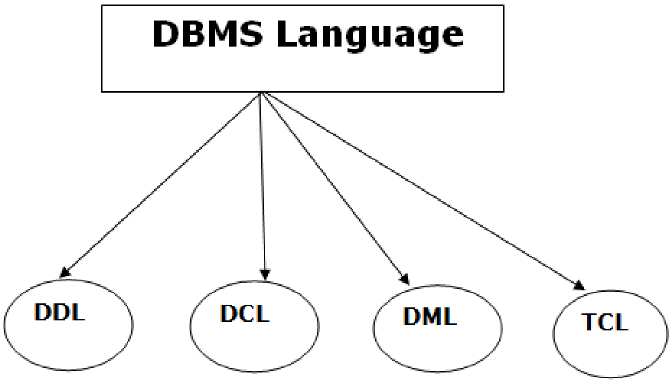
Physical Data Independence

- All the schemas are logical, and the actual data is stored in bit format on the disk. Physical data independence is the power to change the physical data without impacting the schema or logical data.
- For example, in case we want to change or upgrade the storage system itself − suppose we want to replace hard-disks with SSD − it should not have any impact on the logical data or schemas.

Database Languages:

- A DBMS has appropriate languages and interfaces to express database queries and updates.
- Database languages can be used to read, store and update the data in the database.

Types of Database Language



1. Data Definition Language

- **DDL** stands for **D**ata **D**efinition **L**anguage. It is used to define database structure or pattern.
- It is used to create schema, tables, indexes, constraints, etc. in the database.
- Using the DDL statements, you can create the skeleton of the database.

- Data definition language is used to store the information of metadata like the number of tables and schemas, their names, indexes, columns in each table, constraints, etc.

Here are some tasks that come under DDL:

  - **Create:** It is used to create objects in the database.
  - **Alter:** It is used to alter the structure of the database.
  - **Drop:** It is used to delete objects from the database.
  - **Truncate:** It is used to remove all records from a table.
  - **Rename:** It is used to rename an object.
  - **Comment:** It is used to comment on the data dictionary.

These commands are used to update the database schema that's why they come under Data definition language.

2. Data Manipulation Language

- **DML** stands for **D**ata **M**anipulation **L**anguage. It is used for accessing and manipulating data in a database. It handles user requests.

Here are some tasks that come under DML:

  - **Select:** It is used to retrieve data from a database.
  - **Insert:** It is used to insert data into a table.
  - **Update:** It is used to update existing data within a table.
  - **Delete:** It is used to delete all records from a table.
  - **Merge:** It performs UPSERT operation, i.e., insert or update operations.
  - **Call:** It is used to call a structured query language or a Java subprogram.
  - **Explain Plan:** It has the parameter of explaining data.
  - **Lock Table:** It controls concurrency.

3. Data Control Language

- **DCL** stands for **D**ata **C**ontrol **L**anguage. It is used to retrieve the stored or saved data.
- The DCL execution is transactional. It also has rollback parameters.
  (But in Oracle database, the execution of data control language does not have the feature of rolling back.)

Here are some tasks that come under DCL:

  - **Grant:** It is used to give user access privileges to a database.
  - **Revoke:** It is used to take back permissions from the user.

There are the following operations which have the authorization of Revoke:
CONNECT, INSERT, USAGE, EXECUTE, DELETE, UPDATE and SELECT.

4. Transaction Control Language

- TCL is used to run the changes made by the DML statement. TCL can be grouped into a logical transaction.

Here are some tasks that come under TCL:

  - **Commit:** It is used to save the transaction on the database.
  - **Rollback:** It is used to restore the database to original since the last Commit.

Database Interfaces:

- A database management system (DBMS) interface is a user interface which allows for the ability to input queries to a database without using the query language itself.

User-friendly interfaces provide by DBMS may include the following:

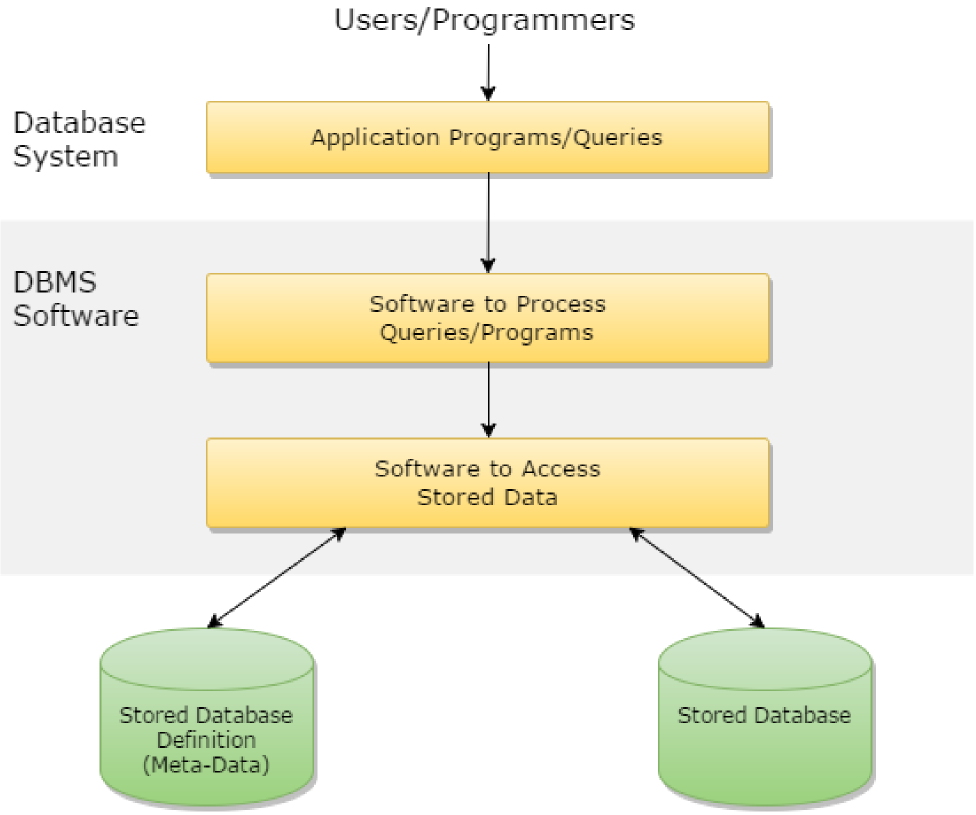
1. **Menu-Based Interfaces for Web Clients or Browsing**
   - These interfaces present the user with lists of options (called menus) that lead the user through the formation of a request.
   - Basic advantage of using menus is that they removes the tension of remembering specific commands and syntax of any query language, rather than query is basically composed step by step by collecting or picking options from a menu that is basically shown by the system.
   - Pull-down menus are a very popular technique in *Web based interfaces*.
   - They are also often used in *browsing interface* which allow a user to look through the contents of a database in an exploratory and unstructured manner.
2. **Forms-Based Interfaces**
   - A forms-based interface displays a form to each user.
   - Users can fill out all of the form entries to insert a new data, or they can fill out only certain entries, in which case the DBMS will redeem same type of data for other remaining entries.
   - This type of forms is usually designed or created and programmed for the users that have no expertise in operating system.
   - Many DBMSs have *forms specification languages* which are special languages that help specify such forms.
   - Example: SQL* Forms is a form-based language that specifies queries using a form designed in conjunction with the relational database schema.b>
3. **Graphical User Interface**
   - A GUI typically displays a schema to the user in diagrammatic form.
   - The user then can specify a query by manipulating the diagram. In many cases, GUI's utilize both menus and forms.
   - Most GUIs use a pointing device such as mouse, to pick certain part of the displayed schema diagram.
4. **Natural language Interfaces**
   - These interfaces accept request written in English or some other language and attempt to understand them.
   - A Natural language interface has its own schema, which is similar to the database conceptual schema as well as a dictionary of important words.
   - The natural language interface refers to the words in its schema as well as to the set of standard words in a dictionary to interpret the request.
   - If the interpretation is successful, the interface generates a high-level query corresponding to the natural language and submits it to the DBMS for processing, otherwise a dialogue is started with the user to clarify any provided condition or request.
   - The main disadvantage with this is that the capabilities of this type of interfaces are not that much advance.
5. **Speech Input and Output**
   - There is an limited use of speech say it for a query or an answer to a question or being a result of a request it is becoming commonplace.
   - Applications with limited vocabularies such as inquiries for telephone directory, flight arrival/departure, and bank account information are allowed speech for input and output to enable ordinary folks to access this information.

- The Speech input is detected using a predefined words and used to set up the parameters that are supplied to the queries. For output, a similar conversion from text or numbers into speech take place.

6. **Interfaces for DBA**
   - Most database system contains privileged commands that can be used only by the DBA's staff.
   - These include commands for creating accounts, setting system parameters, granting account authorization, changing a schema, reorganizing the storage structures of a databases.

Database System Environment

- One of the primary aims of a database is to supply users with an abstract view of data, hiding a certain element of how data is stored and manipulated.
- Therefore, the starting point for the design of a database should be an abstract and general description of the information needs of the organization that is to be represented in the database.
- And hence you will require an environment to store data and make it work as a database. In this chapter, you will learn about the database environment and its architecture.


Users/Programmers
Database System
Application Programs/Queries
DBMS Software
Software to Process Queries/Programs
Software to Access Stored Data
Stored Database Definition (Meta-Data)
Stored Database

Database System Environment

- A database environment is a collective system of components that comprise and regulates the group of data, management, and use of data, which consist of software, hardware, people, techniques of handling database, and the data also.
- Here, the hardware in a database environment means the computers and computer peripherals that are being used to manage a database, and the software means the whole thing right from the operating system (OS) to the application programs that include database management software like M.S.
- Access or SQL Server. Again the people in a database environment include those people who administrate and use the system. The techniques are the rules, concepts, and instructions given to

both the people and the software along with the data with the group of facts and information positioned within the database environment.

Data Modeling Using the Entity-Relationship Model

Data Modeling

- Data modeling is a technique to document a software system using diagrams and symbols. It is used to represent communication of data.
- The highest level of abstraction for the data model is called the Entity Relationship Diagram (ERD). It is a graphical representation of data requirements for a database.

Entity Relationship Diagram

- The main value of carefully constructing an ERD is that it can readily be converted into a database structure.

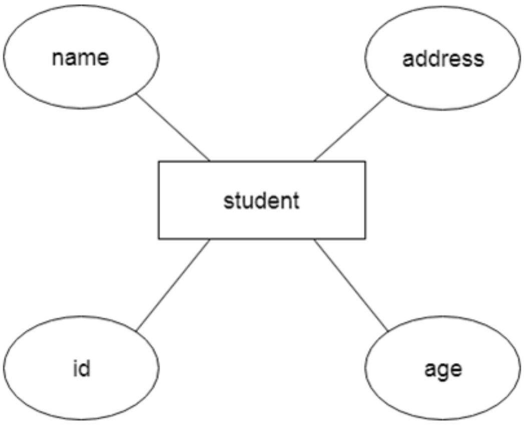
There are three components in ERD.

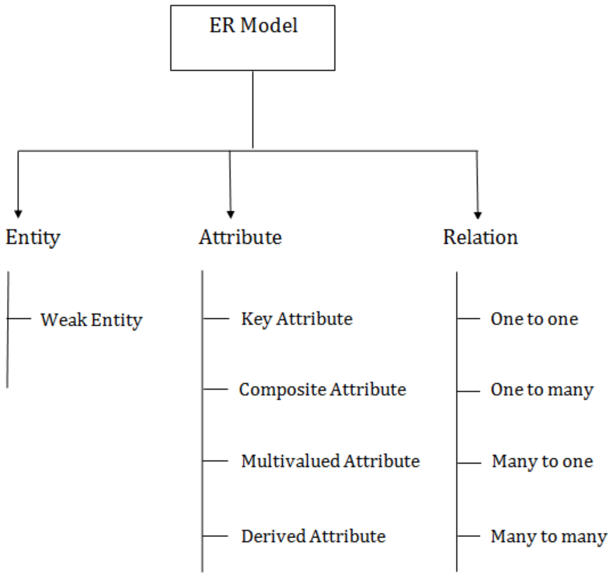
- Entities: Number of tables you need for your database.
- Attributes: Information such as property, facts you need to describe each table.
- Relationships: How tables are linked together.

ER model

- 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.

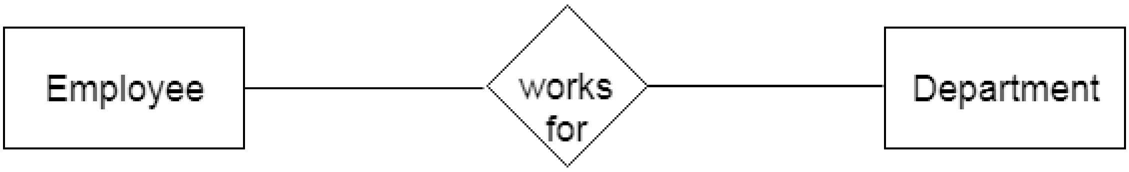




Component of ER Diagram

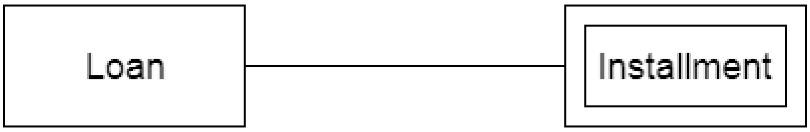
1. Entity:

- 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.

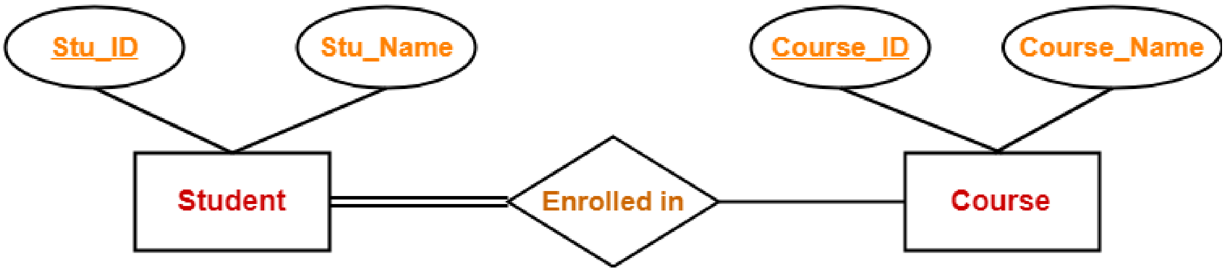


b. Strong Entity

- A single rectangle is used for representing a strong entity set.
- A diamond symbol is used for representing the relationship that exists between two strong entity sets.
- A single line is used for representing the connection of the strong entity set with the relationship set.
- A double line is used for representing the total participation of an entity set with the relationship set.
- Total participation may or may not exist in the relationship.

Example-

Consider the following ER diagram-



In this ER diagram,

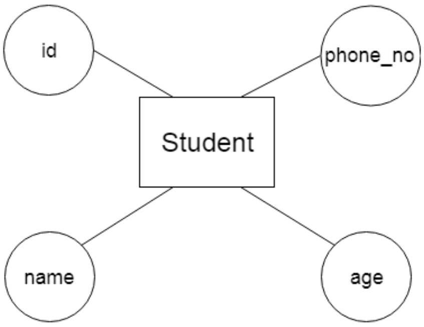
Two strong entity sets “**Student**” and “**Course**” are related to each other.

- Student ID and Student name are the attributes of entity set “Student”.
- Student ID is the primary key using which any student can be identified uniquely.
- Course ID and Course name are the attributes of entity set “Course”.
- Course ID is the primary key using which any course can be identified uniquely.
- Double line between Student and relationship set signifies total participation.
- It suggests that each student must be enrolled in at least one course.

- Single line between Course and relationship set signifies partial participation.
- It suggests that there might exist some courses for which no enrollments are made.

2. Attribute

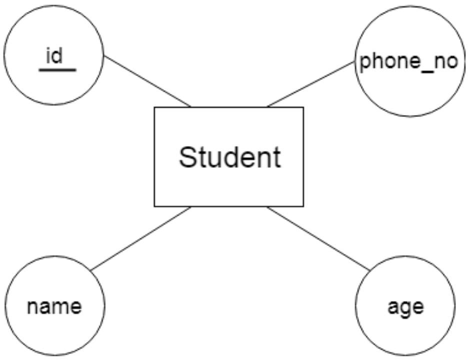
The attribute is used to describe the property of an entity. Eclipse is used to represent an attribute.



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

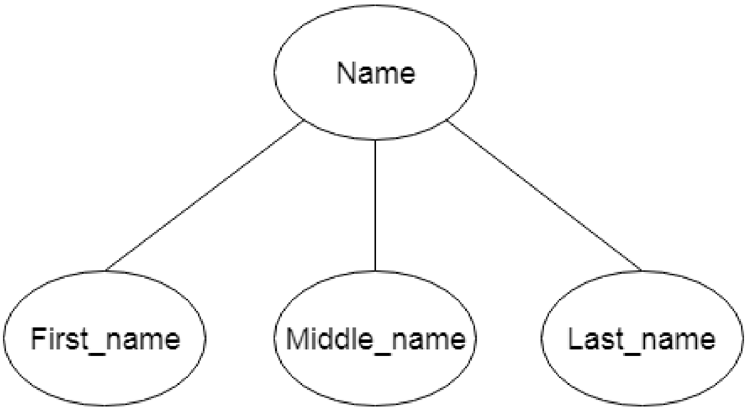
a. Key Attribute

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

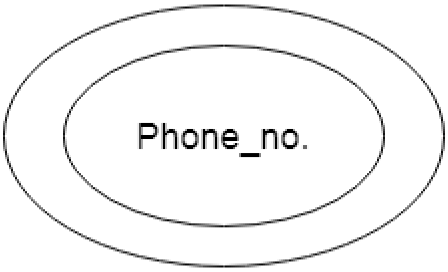
An attribute that composed of many other attributes is known as a composite attribute. The composite attribute is represented by an ellipse, and those ellipses are connected with an ellipse.



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.

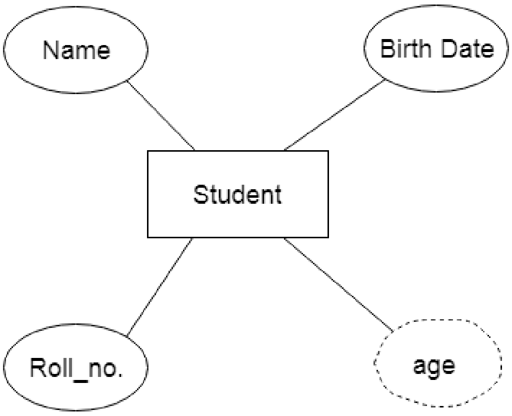
For example, a student can have more than one phone number.



d. Derived Attribute

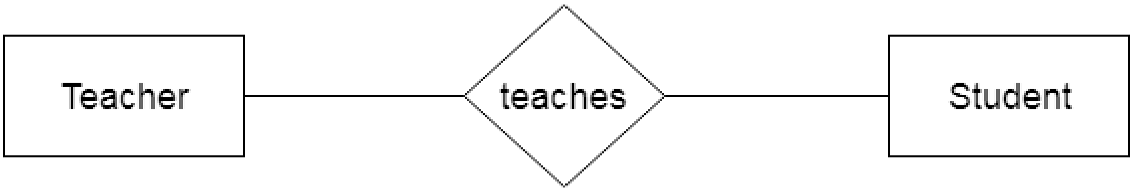
An attribute that can be derived from other attribute is known as a derived attribute. It can be represented by a dashed ellipse.

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.

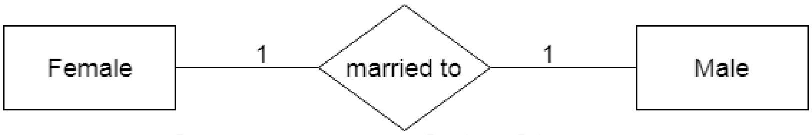


Types of relationship are as follows:

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.

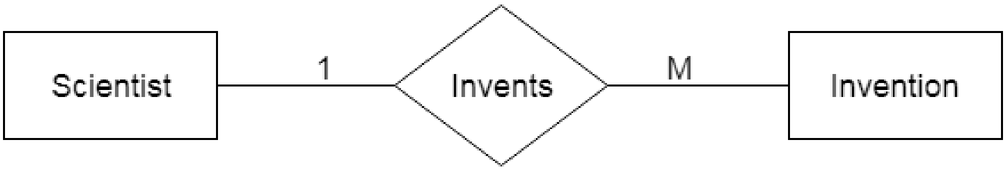
For example, A female can marry to one male, and a male can marry to one female.



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.

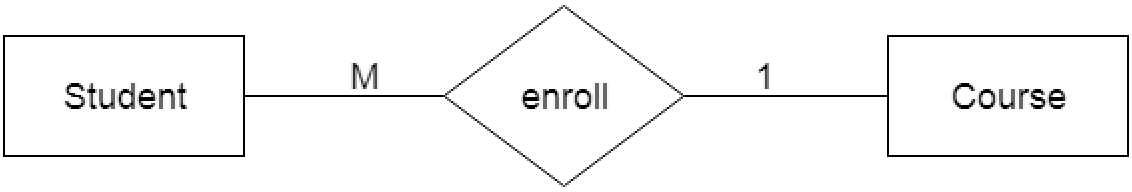
For example, Scientist can invent many inventions, but the invention is done by the only specific scientist.



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.

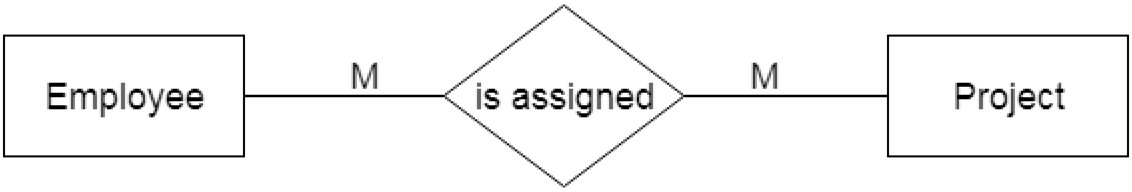
For example, Student enrolls for only one course, but a course can have many students.



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.

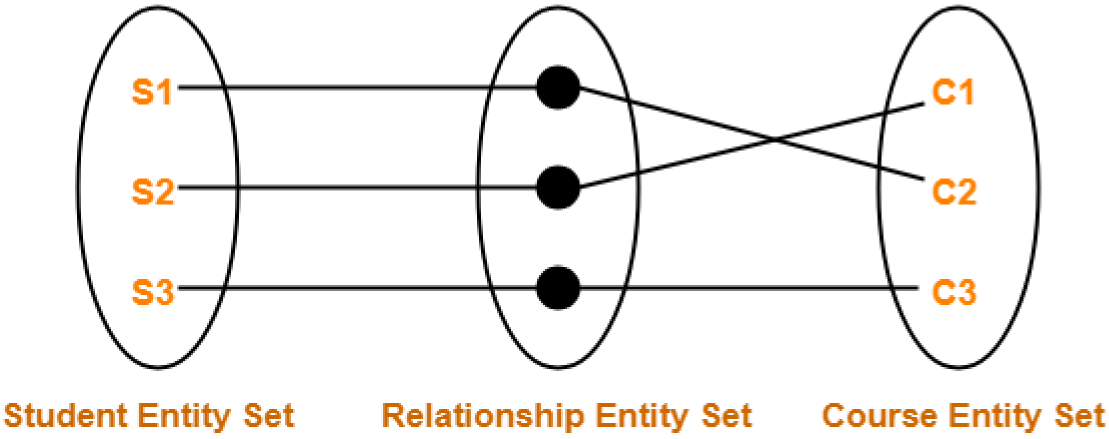


Relationship Set-

A relationship set is a set of relationships of same type.

Example-

Set representation of above ER diagram is-



Set Representation of ER Diagram

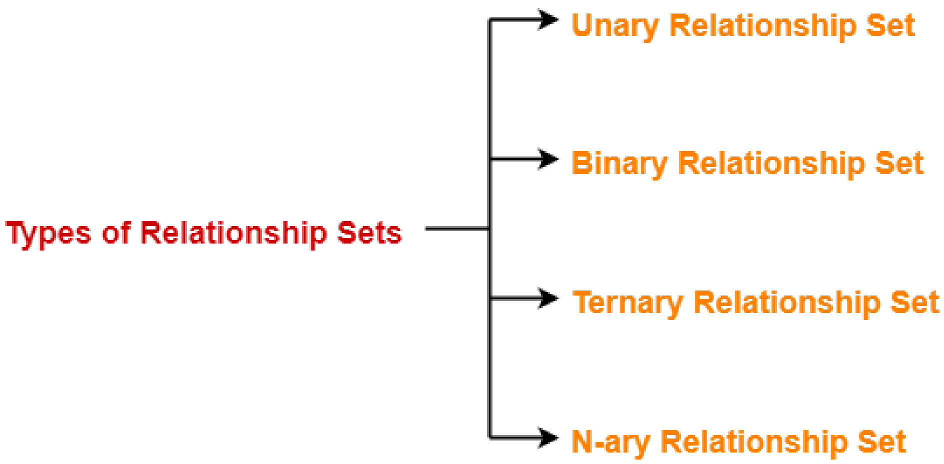
Degree of a Relationship Set-

The number of entity sets that participate in a relationship set is termed as the degree of that relationship set. Thus,

Degree of a relationship set = Number of entity sets participating in a relationship set

<u>Types of Relationship Sets-</u>

On the basis of degree of a relationship set, a relationship set can be classified into the following types-



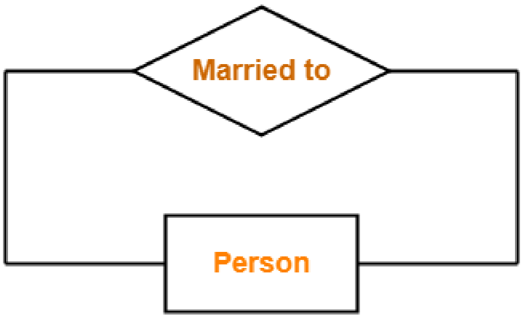
1. Unary relationship set
2. Binary relationship set
3. Ternary relationship set
4. N-ary relationship set

1. Unary Relationship Set-

Unary relationship set is a relationship set where only one entity set participates in a relationship set.

Example-

One person is married to only one person



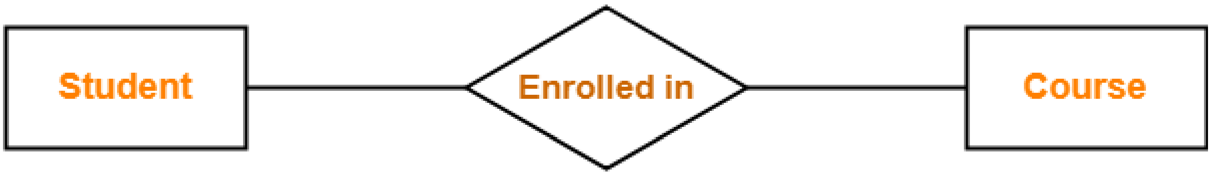
Unary Relationship Set

2. Binary Relationship Set-

Binary relationship set is a relationship set where two entity sets participate in a relationship set.

Example-

Student is enrolled in a Course

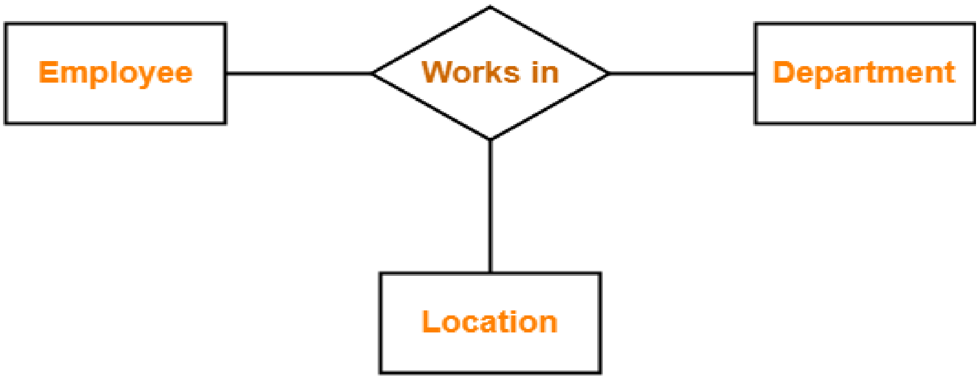


Binary Relationship Set

3. Ternary Relationship Set-

Ternary relationship set is a relationship set where three entity sets participate in a relationship set.

Example-



Ternary Relationship Set

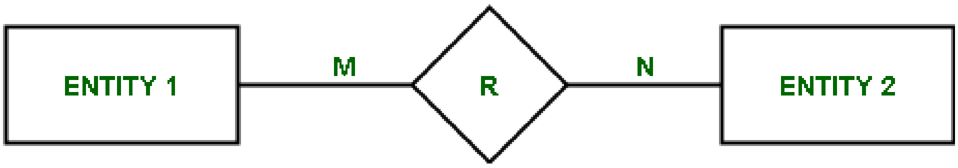
4. N-ary Relationship Set-

N-ary relationship set is a relationship set where ‘n’ entity sets participate in a relationship set.

Roles, and Structural Constraints

Cardinality Ratios of relationships:

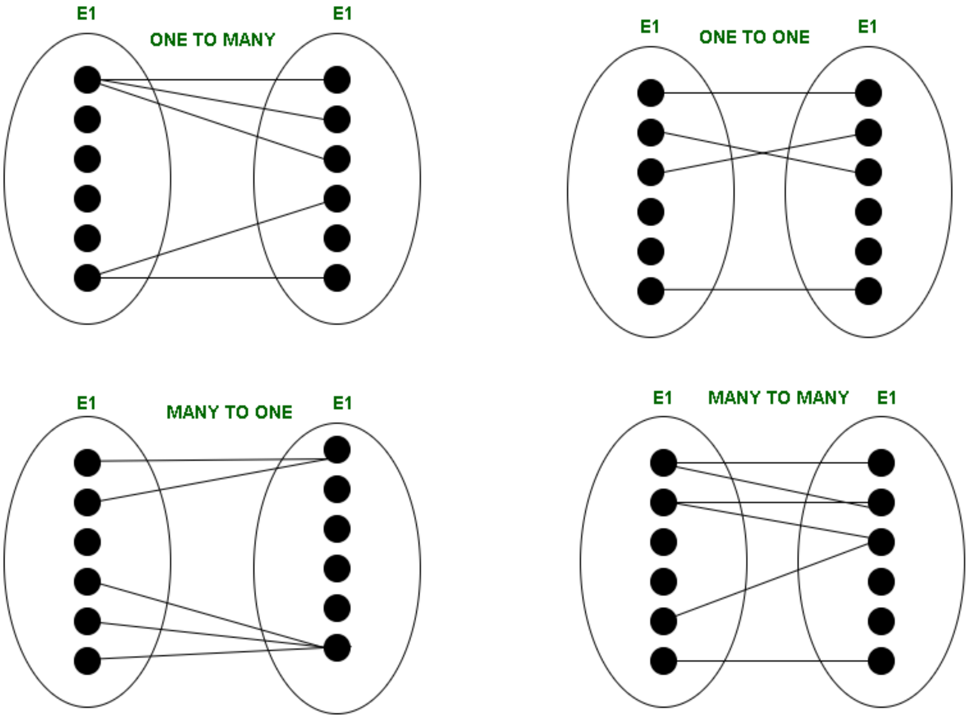
The entities are denoted by rectangle and relationships by diamond.



There are numbers (represented by M and N) written above the lines which connect relationships and entities. These are called cardinality ratios. These represent the maximum number of entities that can be associated with each other through relationship, R.

Types of Cardinality :

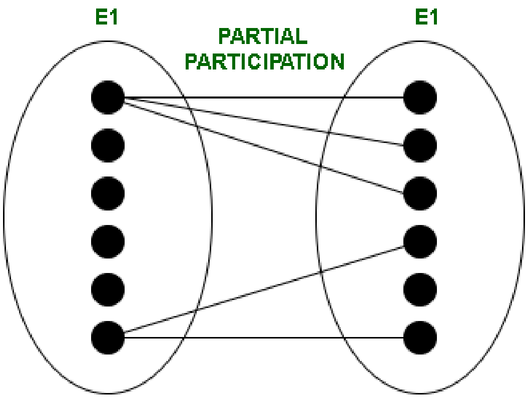
There can be 4 types of cardinality –

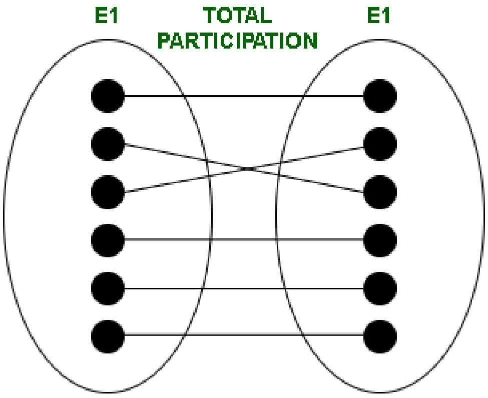


1. **One-to-one (1:1) –**
   When one entity in each entity set takes part at most once in the relationship, the cardinality is one-to-one.
2. **One-to-many (1: N) –**
   If entities in the first entity set take part in the relationship set at most once and entities in the second entity set take part many times (at least twice), the cardinality is said to be one-to-many.
3. **Many-to-one (N:1) –**
   If entities in the first entity set take part in the relationship set many times (at least twice), while entities in the second entity set take part at most once, the cardinality is said to be many-to-one.
4. **Many-to-many (N: N) –**
   The cardinality is said to be many to many if entities in both the entity sets take part many times (at least twice) in the relationship set.

Participation Constraints:

Participation Constraints tell us that that the participation in a relationship can either be total or partial.

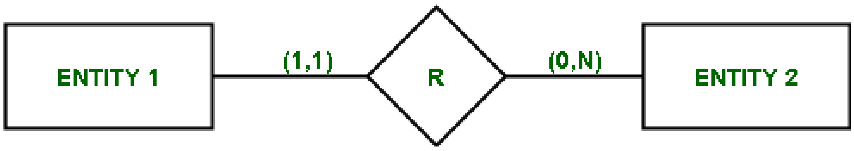




When each entity in an entity set participates in a relation, it is called *Total Participation*.
However, when all entities in the given entity set do not participate in a relation, it is called *Partial Participation*.

Structural Constraints:

- Structural Constraints are also called Structural properties of a database management system (DBMS).
- Cardinality Ratios and Participation Constraints taken together are called Structural Constraints. The name constraints refer to the fact that such limitations must be imposed on the data, for the DBMS system to be consistent with the requirements.



- The Structural constraints are represented by **Min-Max notation**.
- This is a pair of numbers (m, n) that appear on the connecting line between the entities and their relationships.

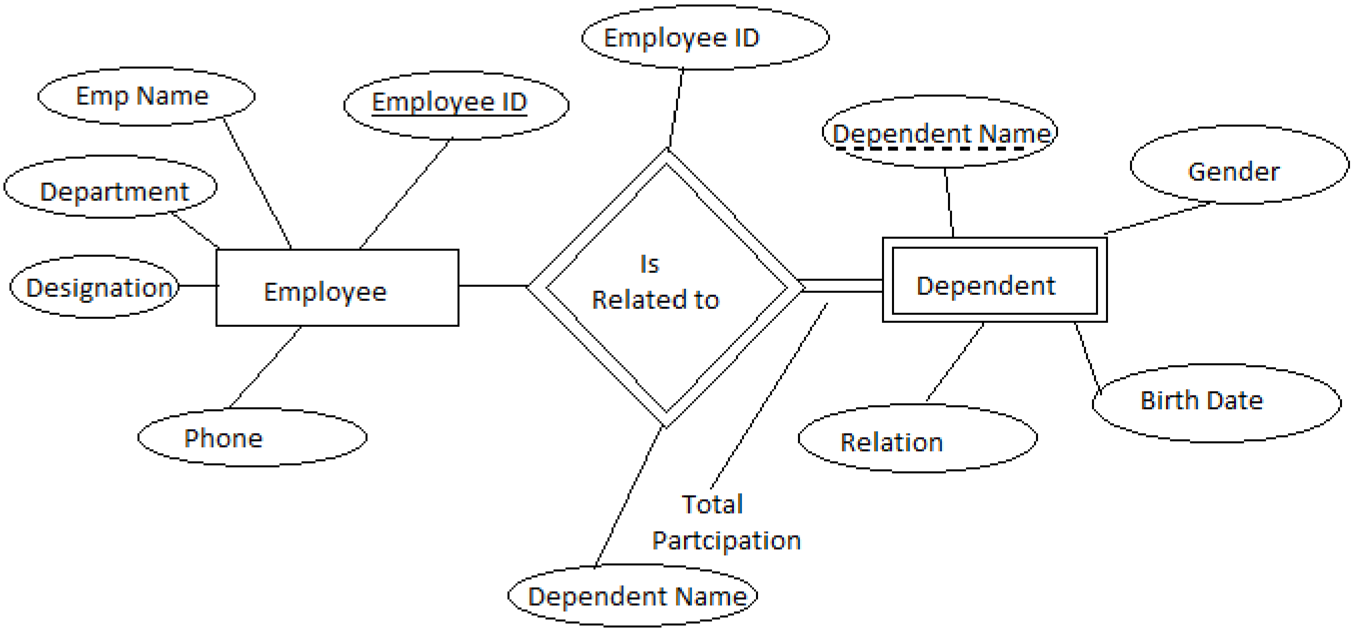
- The minimum number of times an entity can appear in a relation is represented by m whereas, the maximum time it is available is denoted by n.
- If m is 0 it signifies that the entity is participating in the relation partially, whereas, if m is either greater than or equal to 1, it denotes total participation of the entity.

Note –

Number of times an entity participates in a relationship is same as the number appearance of the entity in the tuples.

Weak Entity Types

- A **Weak Entity** has to depend on a strong entity for its existence. So, it creates a total participation in the identifying relationship.
- A weak entity has existential dependency on its owner entity. A **Strong Entity** can have a partial participation in the **Identifying Relationship.**
- In a weak Entity the Primark Key includes the key attribute of the related Strong Entity along with its own discriminating attribute.
- In the example Dependent Entity Type Primary Key is {Employee ID, Dependent Name}



Identifying Relationship set attributes are {Employee ID, Dependent Name}

Difference between Strong entity and Weak Entity

- **Key Attribute:** Strong Entity has a Key attribute whereas the weak entity doesn't have a key attribute. Instead, a weak entity has partial key or discriminators. These discriminators along with the key attribute of the strong entity act as a primary key for the weak entity. Also, the key attribute of the strong entity is represented by underline whereas the discriminator of the weak entity is represented by the dashed line.
- **Existence:** A weak entity depends on a strong entity for its existence whereas a strong entity has its own independent existence. It means that we can uniquely identify the data from the strong entity using its own attributes only. But, for identifying any data from a weak entity uniquely we need the key attributes of the strong entity.

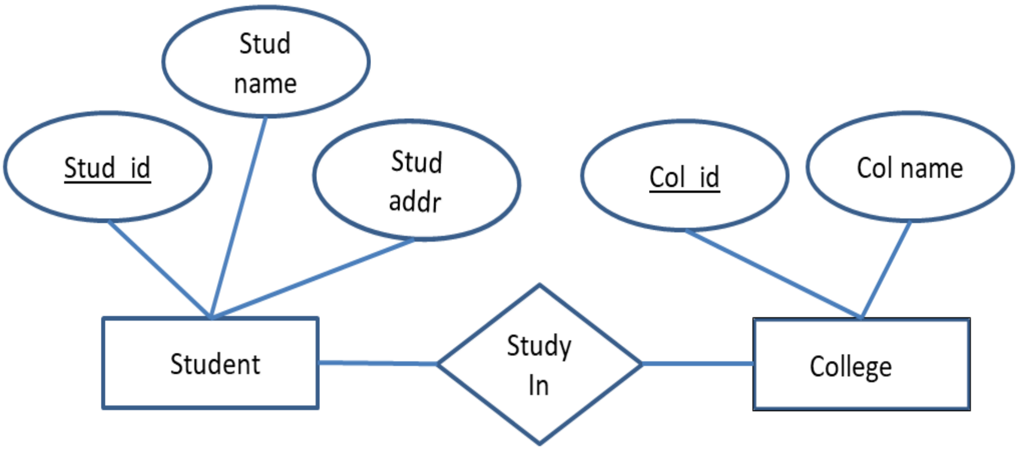
- **Notation:** A strong entity is represented by a single outlined rectangle whereas a weak entity is represented by a double outlined rectangle.
- **Relationship**: Relationship between two strong entity type is represented by a diamond shape. It is simply called a relationship. But, the relationship between a strong and weak entity is represented by a double diamond shape. It is called Identifying Relationship.
- **Participation:** Total Participation may or may not exist in relationship between strong entity but the weak entity always shows total participation in identifying relationship. Also, the line connecting the strong entity with the relationship is single. But, the line connecting a weak entity with the identifying relationship is double.

ER Diagram

- An **Entity–relationship model (ER model)** describes the structure of a database with the help of a diagram, which is known as **Entity Relationship Diagram (ER Diagram)**.
- An ER model is a design or blueprint of a database that can later be implemented as a database. The main components of E-R model are: entity set and relationship set.
- An ER diagram shows the relationship among entity sets. An entity set is a group of similar entities and these entities can have attributes.
- In terms of DBMS, an entity is a table or attribute of a table in database, so by showing relationship among tables and their attributes, ER diagram shows the complete logical structure of a database.

A simple ER Diagram:

- In the following diagram we have two entities Student and College and their relationship.
- The relationship between Student and College is many to one as a college can have many students however a student cannot study in multiple colleges at the same time.
- Student entity has attributes such as Stu_Id, Stu_Name & Stu_Addr and College entity has attributes such as Col_ID & Col_Name.



- Here are the geometric shapes and their meaning in an E-R Diagram.

Rectangle: Represents Entity sets.

Ellipses: Attributes

Diamonds: Relationship Set

Lines: They link attributes to Entity Sets and Entity sets to Relationship Set
Double Ellipses: Multivalued Attributes
Dashed Ellipses: Derived Attributes
Double Rectangles: Weak Entity Sets
Double Lines: Total participation of an entity in a relationship set

Naming Conventions and Design Aspects

the basic design issues of an ER database schema in the following points:

1) Use of Entity Set vs Attributes

- The use of an entity set or attribute depends on the structure of the real-world enterprise that is being modelled and the semantics associated with its attributes.
- It leads to a mistake when the user use the primary key of an entity set as an attribute of another entity set.
- Instead, he should use the relationship to do so. Also, the primary key attributes are implicit in the relationship set, but we designate it in the relationship sets.

2) Use of Entity Set vs. Relationship Sets

- It is difficult to examine if an object can be best expressed by an entity set or relationship set.
- To understand and determine the right use, the user need to designate a relationship set for describing an action that occurs in-between the entities.
- If there is a requirement of representing the object as a relationship set, then its better not to mix it with the entity set.

3) Use of Binary vs n-ary Relationship Sets

- Generally, the relationships described in the databases are binary relationships. However, non-binary relationships can be represented by several binary relationships.
- For example, we can create and represent a ternary relationship 'parent' that may relate to a child, his father, as well as his mother. Such relationship can also be represented by two binary relationships i.e, mother and father, that may relate to their child.
- Thus, it is possible to represent a non-binary relationship by a set of distinct binary relationships.

4) Placing Relationship Attributes

- The cardinality ratios can become an affective measure in the placement of the relationship attributes.
- So, it is better to associate the attributes of one-to-one or one-to-many relationship sets with any participating entity sets, instead of any relationship set.
- The decision of placing the specified attribute as a relationship or entity attribute should possess the characteristics of the real world enterprise that is being modelled.
- **For example**, if there is an entity which can be determined by the combination of participating entity sets, instead of determine it as a separate entity.
- Such type of attribute must be associated with the many-to-many relationship sets.
- Thus, it requires the overall knowledge of each part that is involved inb desgining and modelling an ER diagram.
- The basic requirement is to analyses the real-world enterprise and the connectivity of one entity or attribute with other.

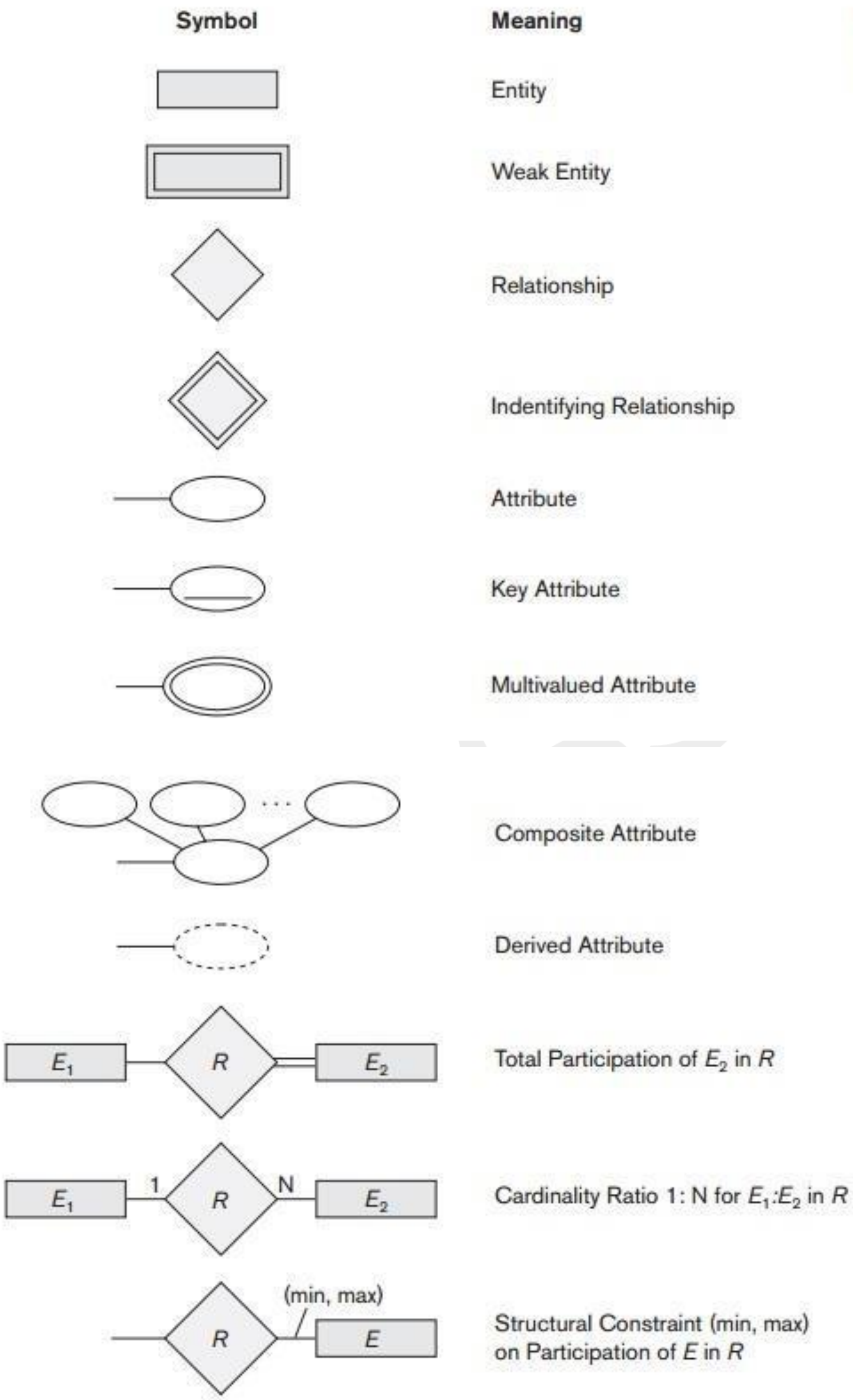

Symbol
Meaning
Entity
Weak Entity
Relationship
Indentifying Relationship
Attribute
Key Attribute
Multivalued Attribute
Composite Attribute
Derived Attribute
E_1
R
E_2
Total Participation of E_2 in R
E_1
1
R
N
E_2
Cardinality Ratio 1: N for E_1:E_2 in R
(min, max)
R
E
Structural Constraint (min, max) on Participation of E in R

Figure 7.14
Summary of the notation for ER diagrams.