UNIT – I

Database : The database is a collection of interrelated data which is used to retrive, insert and delete the data efficiently. It is also used to organize the data in the form of a table, schema, views and reports, etc.

Example : The college database organizes the data about the admin, staff, students and faculty etc. Using the database, you can easily retrive, insert and delete the information.

Database Management Systems : Database Management System is a software which is used to manage the database.

Example : MYSQL, Oracle, etc are a very popular commercial database which is used in different applications.

- DBMS Provides an interface to perform various operations like database creation, storing data in it updating data, creating a table in the database and a lot more.
- It provides protection and security to the database. In the case of multiple users, it also maintains data consistency.

History of DBMS

Here, are the important landmarks from the history:

- 1960 – Charles Bachman designed first DBMS system
- 1970 – Codd introduced IBM'S Information Management System (IMS)
- 1976- Peter Chen coined and defined the Entity-relationship model also know as the ER model
- 1980 – Relational Model becomes a widely accepted database component
- 1985- Object-oriented DBMS develops.
- 1990s- Incorporation of object-orientation in relational DBMS.
- 1991- Microsoft ships MS access, a personal DBMS and that displaces all other personal DBMS products.
- 1995: First Internet database applications
- 1997: XML applied to database processing. Many vendors begin to integrate XML into DBMS products.

File Systems versus a DBMS

File System	DBMS
A file system is a software that manages and organizes the files in a storage medium. It controls how data is stored and retrieved.	DBMS or Database Management System is a software application. It is used for accessing, creating, and managing databases.
The file system provides the details of data representation and storage of data.	DBMS gives an abstract view of data that hides the details
Storing and retrieving of data can't be done efficiently in a file system.	DBMS is efficient to use as there are a wide variety of methods to store and retrieve data.
It does not offer data recovery processes.	There is a backup recovery for data in DBMS.
The file system doesn't have a crash recovery mechanism.	DBMS provides a crash recovery mechanism
Protecting a file system is very difficult.	DBMS offers good protection mechanism.
In a file management system, the redundancy of data is greater.	The redundancy of data is low in the DBMS system.
Data inconsistency is higher in the file system.	Data inconsistency is low in a database management system.
The file system offers lesser security.	Database Management System offers high security.
File System allows you to stores the data as isolated data files and entities.	Database Management System stores data as well as defined constraints and interrelation.
Not provide support for complicated transactions.	Easy to implement complicated transactions.
The centralization process is hard in File Management System.	Centralization is easy to achieve in the DBMS system.
It doesn't offer backup and recovery of data if it is lost.	DBMS system provides backup and recovery of data even if it is lost.
There is no efficient query processing in the file system.	You can easily query data in a database using the SQL language.
These system doesn't offer concurrency.	DBMS system provides a concurrency facility.

Characteristics of DBMS

Here are the characteristics and properties of Database Management System:

- Provides security and removes redundancy
- Self-describing nature of a database system
- Insulation between programs and data abstraction
- Support of multiple views of the data
- Sharing of data and multiuser transaction processing
- Database Management Software allows entities and relations among them to form tables.
- It follows the ACID concept (Atomicity, Consistency, Isolation, and Durability).
- DBMS supports multi-user environment that allows users to access and manipulate data in parallel.

Advantages of DBMS

Data Independence: Application programs should not, ideally, be exposed to details of data representation and storage. The provides an abstract view of the data that hides such details.

Efficient Data Access: A DBMS utilizes a variety of sophisticated techniques to store and retrieve data efficiently. This feature is especially important if the data is stored on external storage devices.

Data Integrity and Security: If data is always accessed through the DBMS, the DBMS can enforce integrity constraints. For example, before inserting salary information for an employee, the DBMS can check that the department budget is not exceeded. Also, it can enforce *access controls* that govern what data is visible to different classes of users.

Data Administration: When several users share the data, centralizing the administration of data can offer sig11ificant improvements. Experienced professionals who understand the nature of the data being managed, and how different groups of users use it, can be responsible for organizing the data representation to minimize redundancy and for fine-tuning the storage of the data to make retrieval efficient.

Concurrent Access and Crash Recovery: A DBMS schedules concurrent accesses to the data in such a manner that users can think of the data as being accessed by only one user at a time. Further, the DBMS protects users from the effects of system failures.

Reduced Application Development Time: Clearly, the DBMS supports important functions that are common to many applications accessing data in the DBMS. This, in conjunction with the high-level interface to the data, facilitates quick application development. DBMS applications are also likely to be more robust than similar stand-alone applications because many important tasks are handled by the DBMS (and do not have to be debugged and tested in the application).

Disadvantage of DBMS

DBMS may offer plenty of advantages but, it has certain flaws-

- Cost of Hardware and Software of a DBMS is quite high which increases the budget of your organization.
- Most database management systems are often complex systems, so the training for users to use the DBMS is required.
- In some organizations, all data is integrated into a single database which can be damaged because of electric failure or database is corrupted on the storage media
- Use of the same program at a time by many users sometimes lead to the loss of some data.
- DBMS can't perform sophisticated calculations

Applications of DBMS

Below are the popular database system applications:

Sector	Use of DBMS
Banking	For customer information, account activities, payments, deposits, loans, etc.
Airlines	For reservations and schedule information.
Universities	For student information, course registrations, colleges and grades.
Telecommunication	It helps to keep call records, monthly bills, maintaining balances, etc.
Finance	For storing information about stock, sales, and purchases of financial instruments like stocks and bonds.
Sales	Use for storing customer, product & sales information.
Manufacturing	It is used for the management of supply chain and for tracking production of items. Inventories status in warehouses.
HR Management	For information about employees, salaries, payroll, deduction, generation of paychecks, etc.

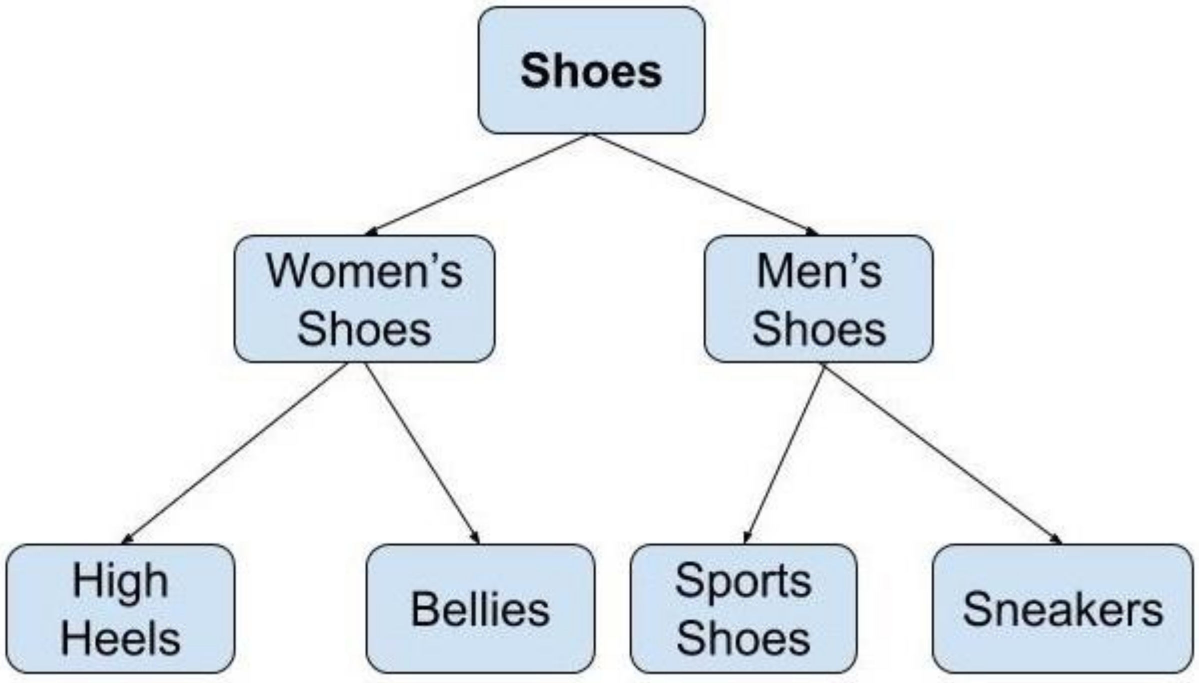
Data Models

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 organisation can communicate and understand it. Though there are many data models being used now a days but the Relational model is the most widely used model. Apart from the Relational model, there are many other types of data models.

1. Hierarchical Model
2. Network Model
3. Entity-Relationship Model
4. Relational Model

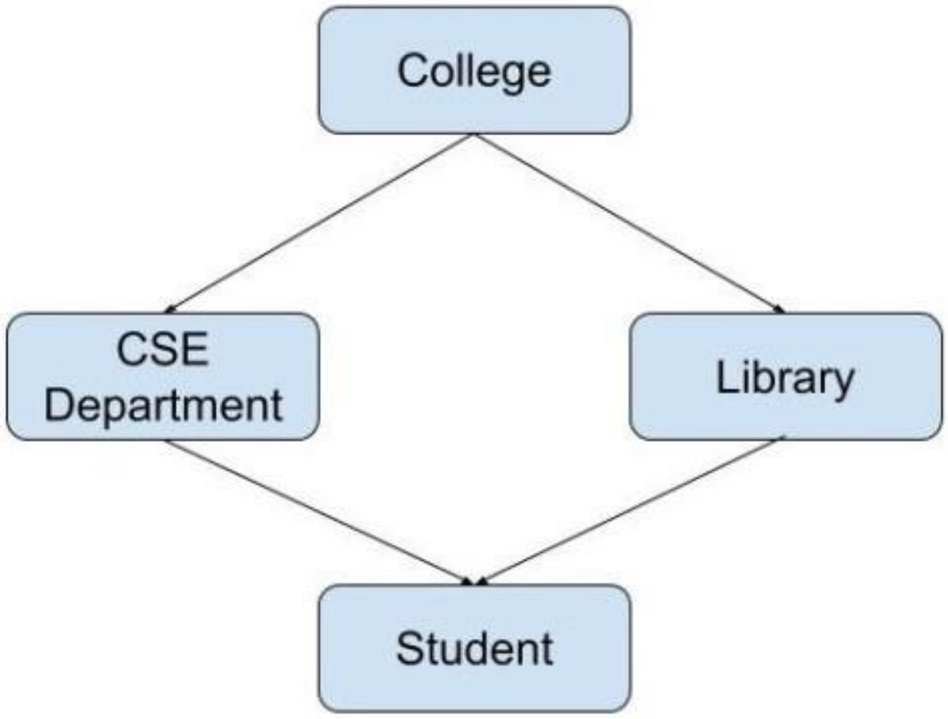
1. Hierarchical Model : Hierarchical Model was the first DBMS model. This model organises the data in the hierarchical tree structure. The hierarchy starts from the root which has root data and then it expands in the form of a tree adding child node to the parent node. This model easily represents some 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:**



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.



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.

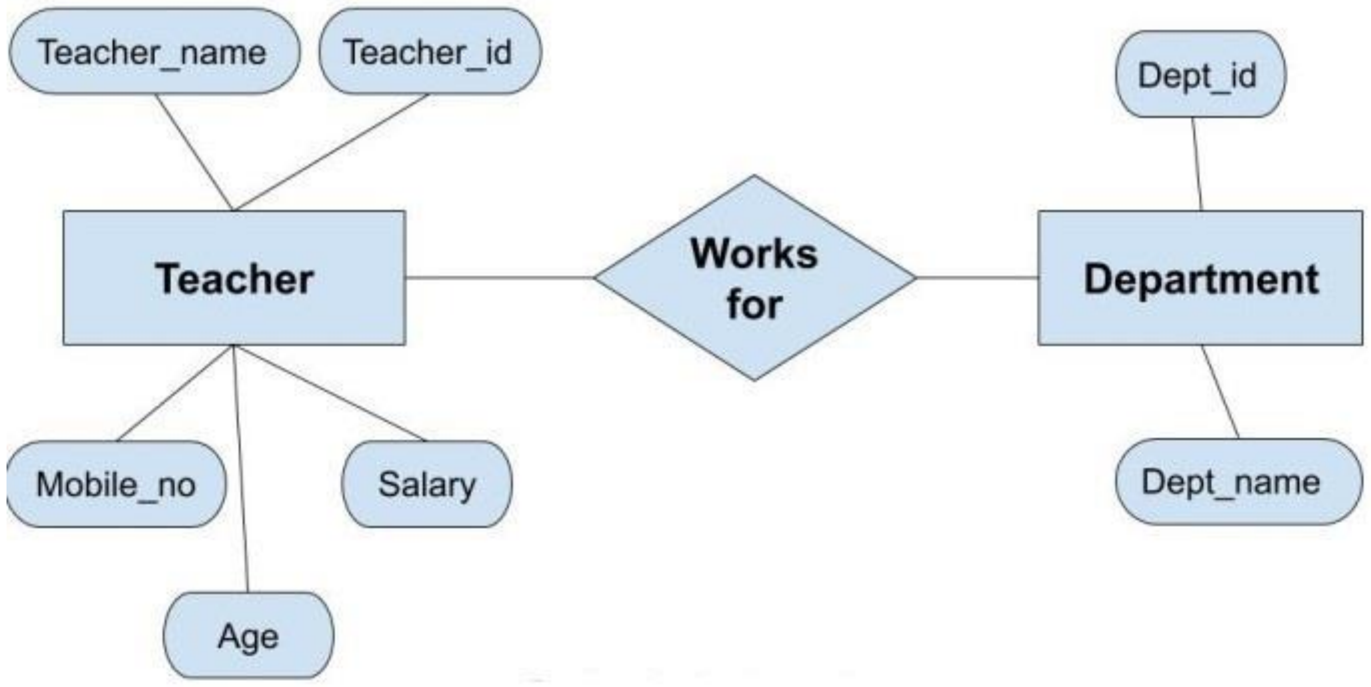
ER diagram has the following three components:

- ***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.



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.

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

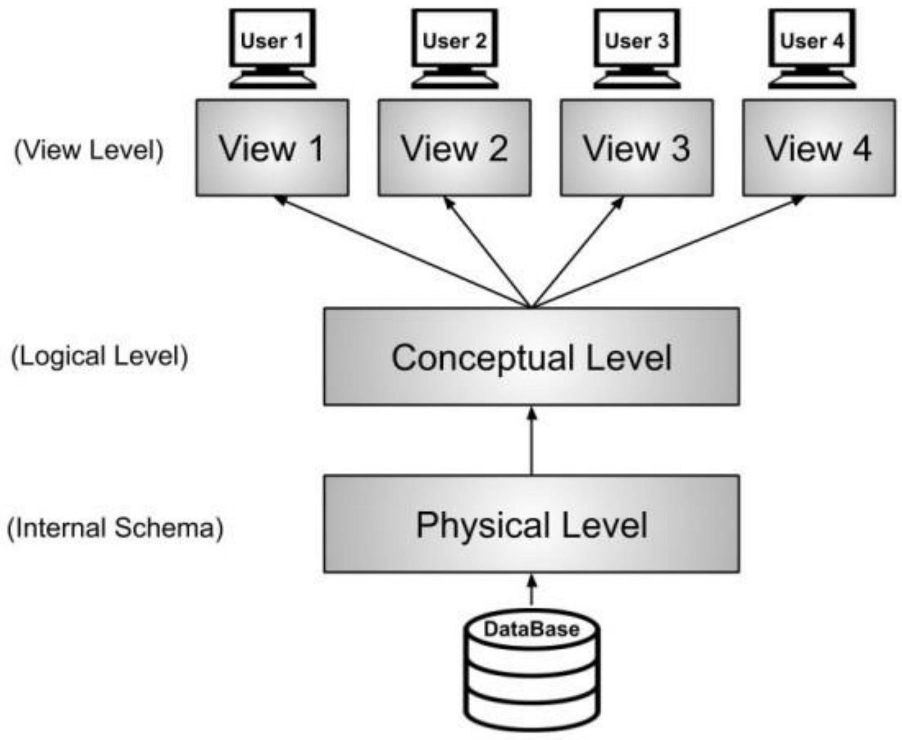
Levels of Abstraction in a DBMS :

Data Abstraction : Data Abstraction refers to the process of hiding irrelevant details from the user.

Example: If we want to access any mail from our Gmail then we don't know where that data is physically stored i.e is the data present in India or USA or what data model has been used to store that data? We are not concerned about these things. We are only concerned with our email. So, information like these i.e. location of data and data models are irrelevant to us and in data abstraction, we do this only. Apart from the location of data and data models, there are other factors that we don't care of. We hide the unnecessary data from the user and this process of hiding unwanted data is called Data Abstraction.

There are mainly three levels of data abstraction and we divide it into three levels in order to achieve *Data Independence*. Data Independence means users and data should not directly interact with each other. The user should be at a different level and the data should be present at some other level. By doing so, Data Independence can be achieved.

1. View Level
2. Conceptual Level
3. Physical Level



Levels of Data Abstraction

1. View Level or External Schema :

This level tells the application about how the data should be shown to the user.

Example: If we have a login-id and password in a university system, then as a student, we can view our marks, attendance, fee structure, etc. But the faculty of the university will have a different view. He will have options like salary, edit marks of a student, enter attendance of the students, etc. So, both the student and the faculty have a different view. By doing so, the security of the system also increases. In this example, the student can't edit his marks but the faculty who is authorized to edit the marks can edit the student's marks. Similarly, the dean of the college or university will have some more authorization and accordingly, he will has his view. So, different users will have a different view according to the authorization they have.

2. Conceptual Level or Logical Level :

This level tells how the data is actually stored and structured. We have different data models by which we can store the data .

Example: Let us take an example where we use the relational model for storing the data. We have to store the data of a student, the columns in the student table will be student_name, age, mail_id, roll_no etc. We have to define all these at this level while we are creating the database. Though the data is stored in the database but the structure of the tables like the student table, teacher table, books table, etc are defined here in the conceptual level or logical level. Also, how the tables are related to each other are defined here. Overall, we can say that we are creating a blueprint of the data at the conceptual level.

3. Physical Level or Internal Schema :

As the name suggests, the Physical level tells us that where the data is actually stored i.e. it tells the actual location of the data that is being stored by the user. The Database Administrators(DBA) decide that which data should be kept at which particular disk drive, how the data has to be fragmented, where it has to be stored etc. They decide if the data has to be centralized or distributed. Though we see the data in the form of tables at view level the data here is actually stored in the form of files only. It totally depends on the DBA, how he/she manages the database at the physical level.

So, the Data Abstraction provides us with a different view and help in achieving Data Independence.

Data Independence : Data independence refers to the property of DBMS through which we can modify the schema definition at any level without changing the schema definition at any higher level. We have two levels of data independence that are defined on the basis of these three levels of abstraction.

1. Physical Data Independence
2. Logical Data Independence

1. Physical Data Independence :

Physical Data Independence refers to the characteristic of changing the physical level without affecting the logical level or conceptual level. Using this property we can easily change the storage device of the database without affecting the logical schema.

Example: Suppose you want to replace the storage device form hard disk to SSD or magnetic tape then it should not affect the data stored at the logical level.

The changes in the physical level may include changes like:

1. Using a new storage device like SSD, magnetic tape, hard disk, etc.
2. Using a new data structure for storage.
3. Using a different data access method or using an alternative file organization technique.
4. Changing the location(like changing the drive) of the database.

2. Logical Data Independence :

It refers to the characteristics of changing the logical level without affecting the external or view level. This also helps in separating the logical level from the view level. If we do any changes in the logical level then the user view of the data remains unaffected. The changes in the logical level are required whenever there is a change in the logical structure of the database.

The changes in the logical level may include:

1. Changing the data definition.
2. Adding, deleting, or updating any new attribute, entity or relationship in the database.

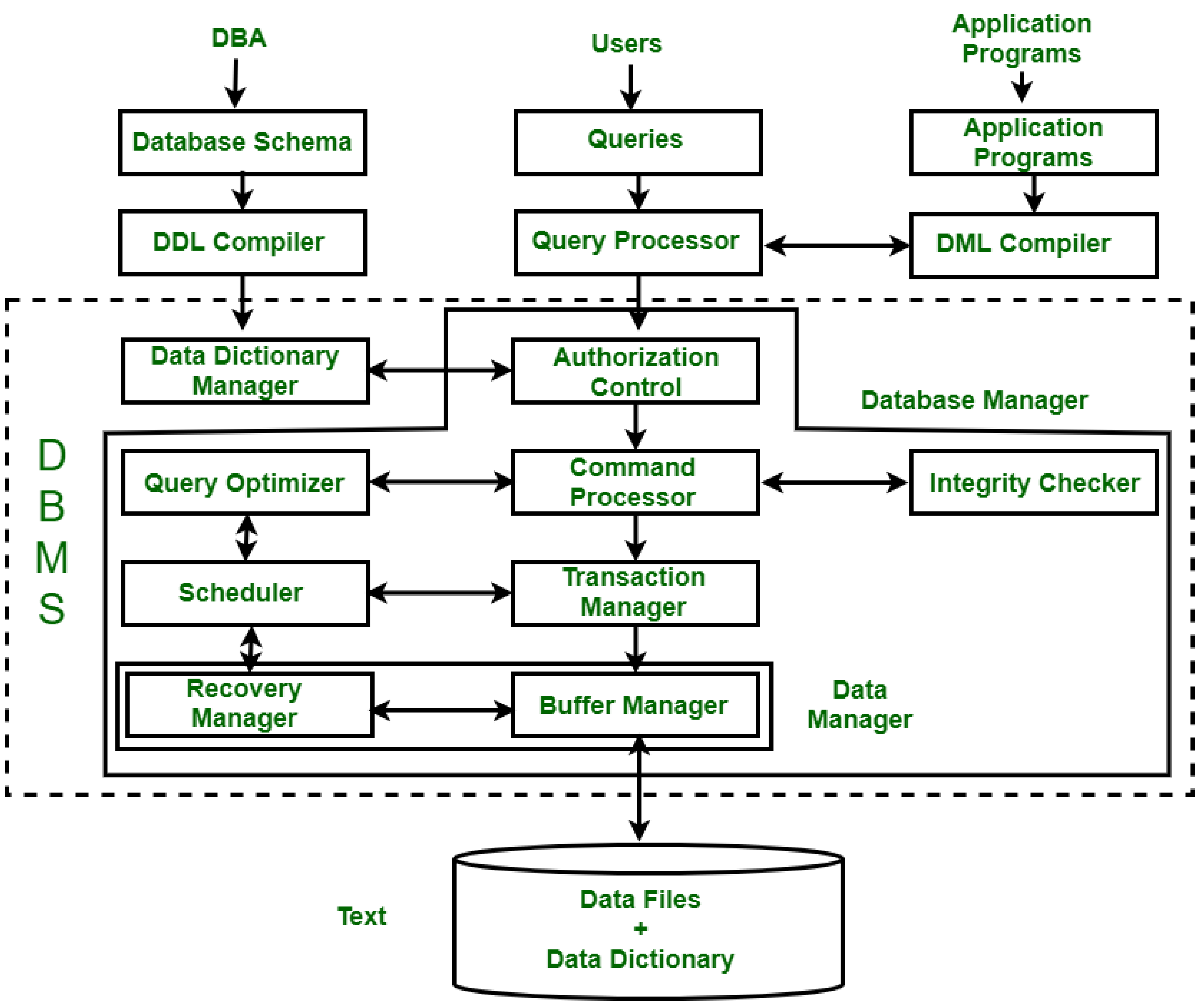
Structure of Database Management System

Database Management System is a software that allows access to data stored in a database and provides an easy and effective method of –

- Defining the information.
- Storing the information.
- Manipulating the information.
- Protecting the information from system crashes or data theft.
- Differentiating access permissions for different users.

The database system is divided into three components:

1. Query Processor
2. Storage Manager
3. Disk Storage



1. Query Processor : It interprets the requests (queries) received from end user via an application program into instructions. It also executes the user request which is received from the DML compiler.

Query Processor contains the following components –

- **DML Compiler** : It processes the DML statements into low level instruction (machine language), so that they can be executed.
- **DDL Interpreter** : It processes the DDL statements into a set of table containing meta data (data about data).
- **Embedded DML Pre-compiler** : It processes DML statements embedded in an application program into procedural calls.
- **Query Optimizer** : It executes the instruction generated by DML Compiler.

2. Storage Manager : Storage Manager is a program that provides an interface between the data stored in the database and the queries received. It is also known as Database Control System. It maintains the consistency and integrity of the database by applying the constraints and executes the DCL statements. It is responsible for updating , storing, deleting, and retrieving data in the database.

It contains the following components –

- **Authorization Manager** : It ensures role-based access control, i.e,. checks whether the particular person is privileged to perform the requested operation or not.
- **Integrity Manager** : It checks the integrity constraints when the database is modified.
- **Transaction Manager** : It controls concurrent access by performing the operations in a scheduled way that it receives the transaction. Thus, it ensures that the database remains in the consistent state before and after the execution of a transaction.
- **File Manager** : It manages the file space and the data structure used to represent information in the database.
- **Buffer Manager** : It is responsible for cache memory and the transfer of data between the secondary storage and main memory.

3. Disk Storage : It contains the following components –

- **Data Files** : It stores the data.
- **Data Dictionary** : It contains the information about the structure of any database object. It is the repository of information that governs the metadata.
- **Indices** : It provides faster retrieval of data item.

Introduction to Database Design

Database Design and ER Diagrams

- ER diagram or Entity Relationship diagram shows the Relationship among entity set and an entity set is a group of similar entities and these entities can have attributes.
- It shows all the constraints and relationships that exist among the different components.

Example-

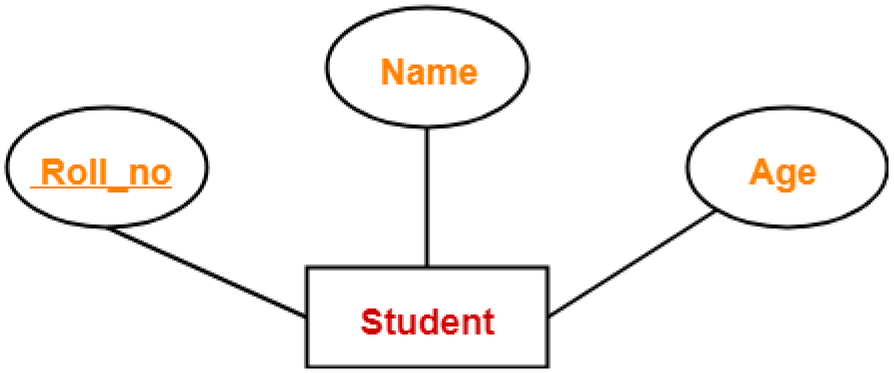
Consider the following Student table-

Roll_no	Name	Age
1	Akshay	20
2	Rahul	19
3	Pooja	20
4	Aarti	19

This complete table is referred to as "Student Entity Set" and each row represents an "entity".

Representation as ER Diagram-

The above table may be represented as ER diagram as-



Here,

- Roll_no is a primary key that can identify each entity uniquely.
- Thus, by using student's roll number, a student can be identified uniquely.

Components of ER diagram (OR) ER Diagram Symbols

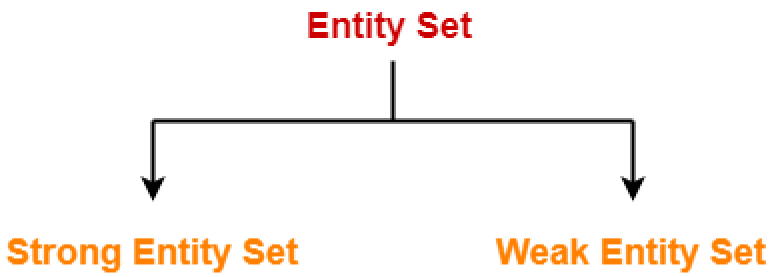
An ER diagram is mainly composed of following three components-

1. Entity Sets
2. Attributes
3. Relationship Set

1. Entity Sets: An entity set is a set of same type of entities. An entity refers to any object having-

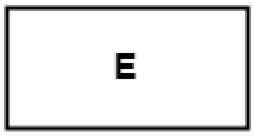
- Either a physical existence such as a particular person, office, house or car.
- Or a conceptual existence such as a school, a university, a company or a job.

An entity set may be of the following two types-



1. Strong Entity Set :

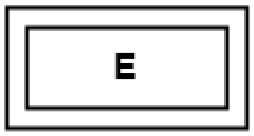
- A strong entity set possess its own primary key.
- It is represented using a single rectangle.



Strong Entity Set

2. Weak Entity Set :

- A weak entity set do not possess its own primary key.
- It is represented using a double rectangle.

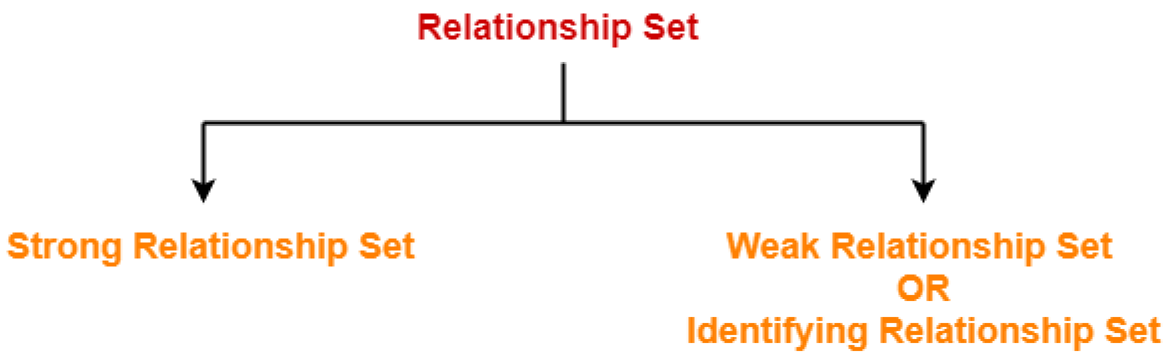


Weak Entity Set

2. Relationship Sets-

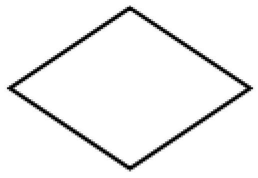
- Relationship defines an association among several entities.
- A relationship set is a set of same type of relationships.

A relationship set may be of the following two types-



1. Strong Relationship Set-

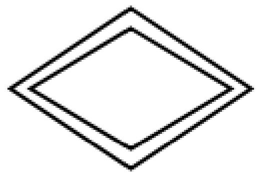
- A strong relationship exists between two strong entity sets.
- It is represented using a diamond symbol.



Strong Relationship Set

2. Weak Relationship Set-

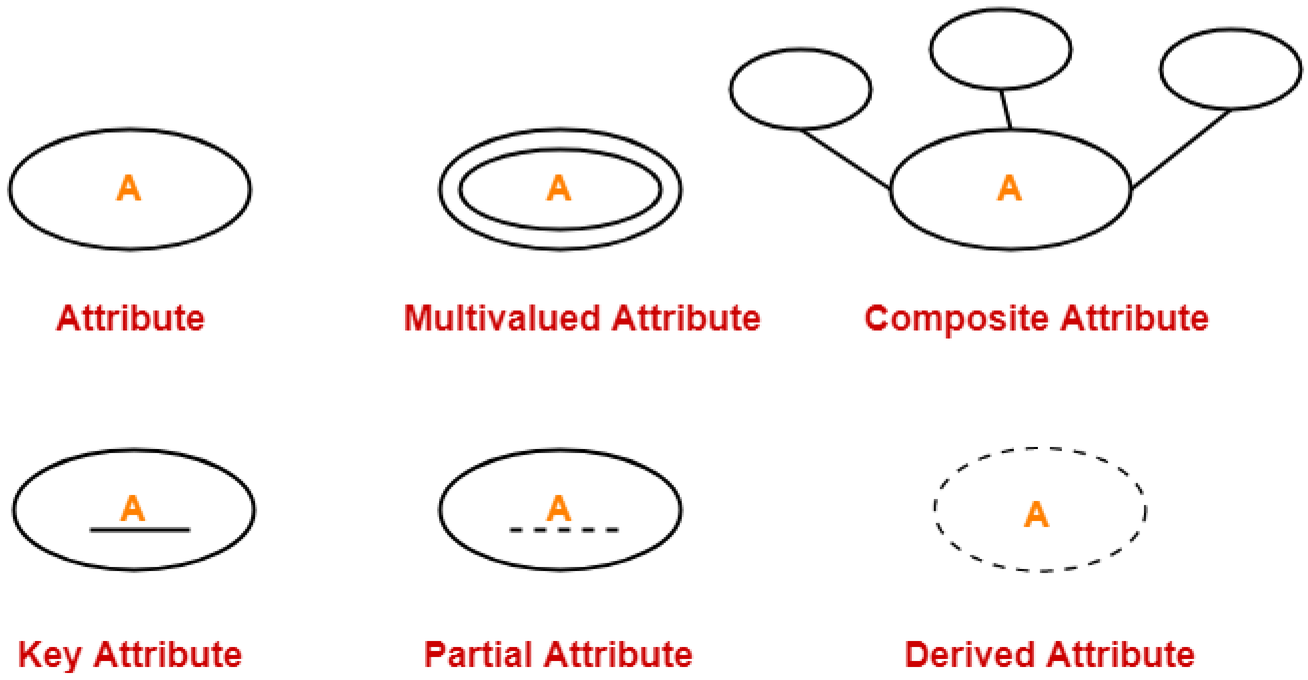
- A weak or identifying relationship exists between the strong and weak entity set.
- It is represented using a double diamond symbol.



Weak or Identifying Relationship Set

3. Attributes-

- Attributes are the properties which describes the entities of an entity set.
- There are several types of attributes.



Entity Set in DBMS: An entity set is a set of same type of entities. An entity refers to any object having-

- Either a physical existence such as a particular person, office, house or car.
- Or a conceptual existence such as a school, a university, a company or a job.

In ER diagram,

- Attributes are associated with an entity set.
- Attributes describe the properties of entities in the entity set.
- Based on the values of certain attributes, an entity can be identified uniquely.

1. Strong Entity Set-

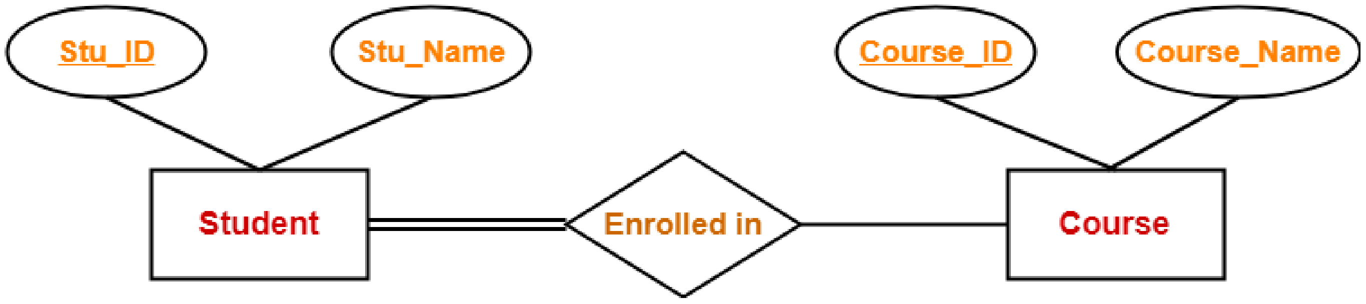
- A strong entity set is an entity set that contains sufficient attributes to uniquely identify all its entities.
- In other words, a primary key exists for a strong entity set.
- Primary key of a strong entity set is represented by underlining it.

Symbols Used-

- 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. Weak Entity Set-

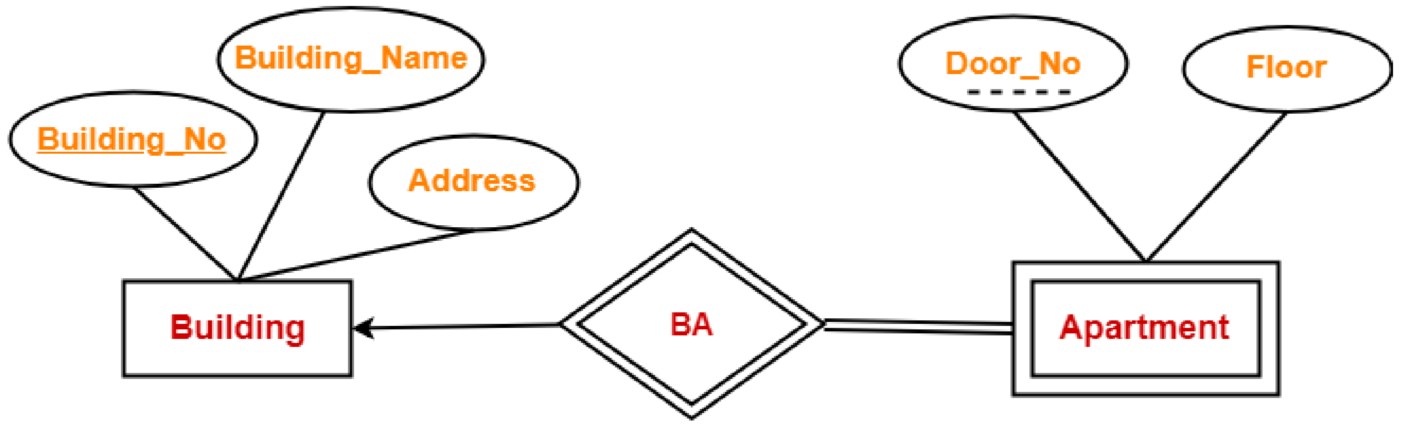
- A weak entity set is an entity set that does not contain sufficient attributes to uniquely identify its entities.
- In other words, a primary key does not exist for a weak entity set.
- However, it contains a partial key called as a **discriminator.**
- Discriminator can identify a group of entities from the entity set.
- Discriminator is represented by underlining with a dashed line.

Symbols Used-

- A double rectangle is used for representing a weak entity set.
- A double diamond symbol is used for representing the relationship that exists between the strong and weak entity sets and this relationship is known as **identifying relationship**.
- A double line is used for representing the connection of the weak entity set with the relationship set.
- Total participation always exists in the identifying relationship.

Example-

Consider the following ER diagram-



In this ER diagram,

- One strong entity set "**Building**" and one weak entity set "**Apartment**" are related to each other.
- Strong entity set "Building" has building number as its primary key.
- Door number is the discriminator of the weak entity set "Apartment".
- This is because door number alone can not identify an apartment uniquely as there may be several other buildings having the same door number.
- Double line between Apartment and relationship set signifies total participation.
- It suggests that each apartment must be present in at least one building.
- Single line between Building and relationship set signifies partial participation.
- It suggests that there might exist some buildings which has no apartment.

To uniquely identify any apartment,

- First, building number is required to identify the particular building.
- Secondly, door number of the apartment is required to uniquely identify the apartment.

Thus,

Primary key of Apartment = Primary key of Building + Its own discriminator

= Building number + Door number

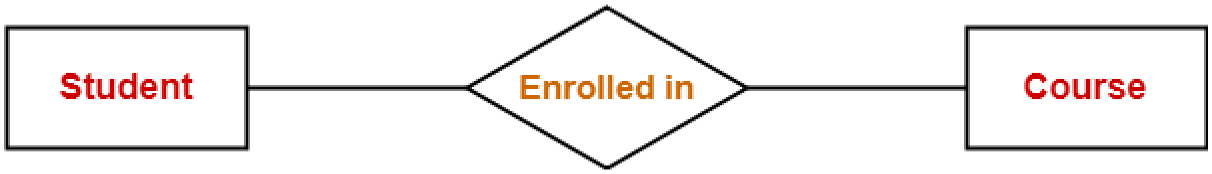
Differences between Strong entity set and Weak entity set-

Strong entity set	Weak entity set
A single rectangle is used for the representation of a strong entity set.	A double rectangle is used for the representation of a weak entity set.
It contains sufficient attributes to form its primary key.	It does not contain sufficient attributes to form its primary key.
A diamond symbol is used for the representation of the relationship that exists between the two strong entity sets.	A double diamond symbol is used for the representation of the identifying relationship that exists between the strong and weak entity set.
A single line is used for the representation of the connection between the strong entity set and the relationship.	A double line is used for the representation of the connection between the weak entity set and the relationship set.
Total participation may or may not exist in the relationship.	Total participation always exists in the identifying relationship.

Relationship in DBMS: A relationship is defined as an association among several entities.

Example-

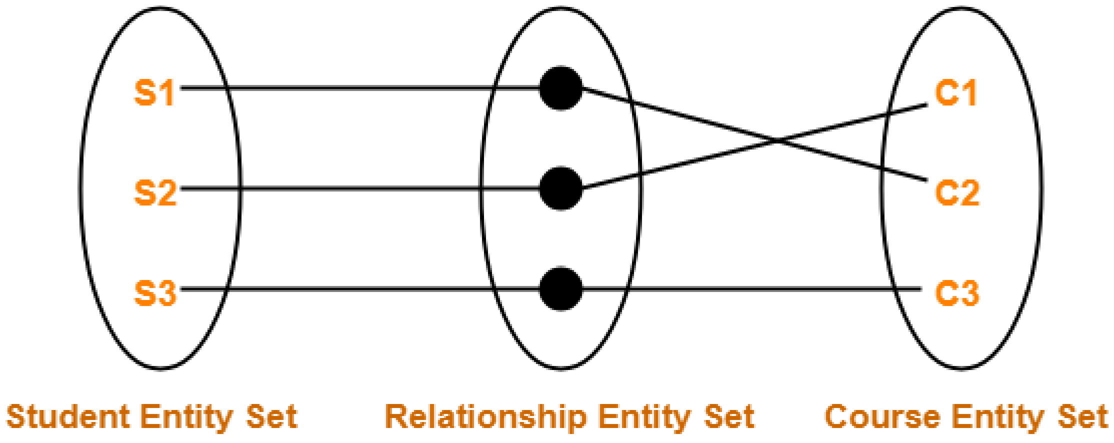
'Enrolled in' is a relationship that exists between entities Student and Course.



Relationship Set : A relationship set is a set of relationships of same type.

Example-

Set representation of above ER diagram is-



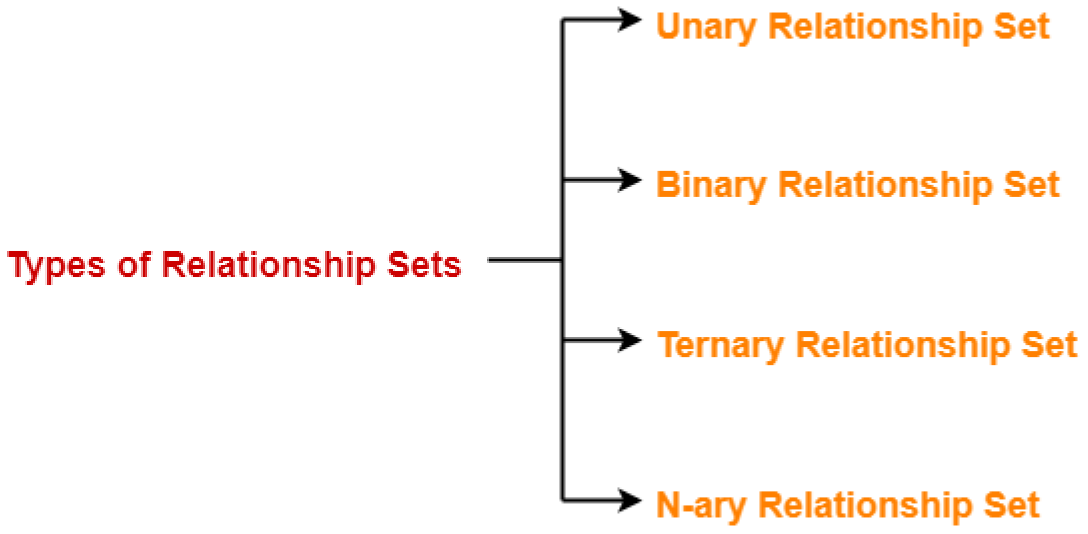
Set Representation of ER Diagram

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,

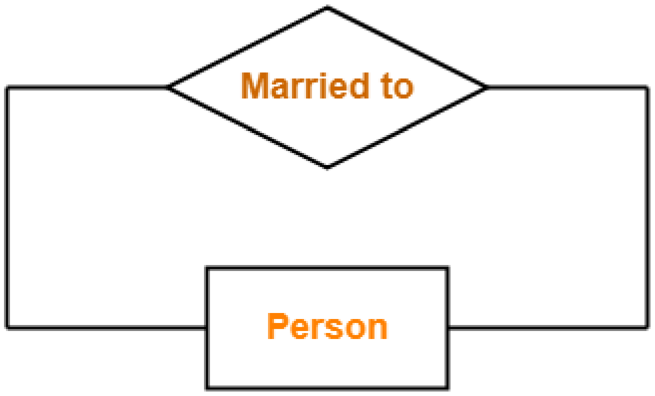
Degree of a relationship set = Number of entity sets participating in a relationship set.

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



1. Unary Relationship Set : Unary relationship set is a relationship set where only one entity set participates in a relationship set.

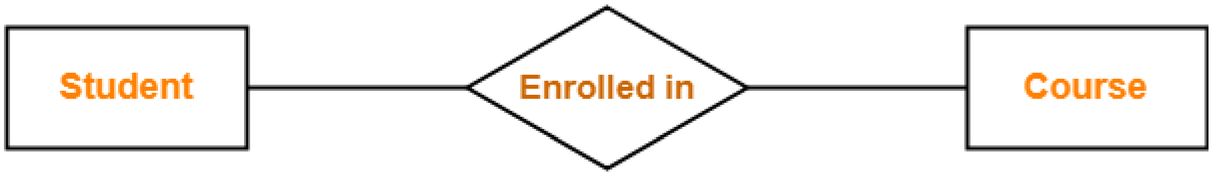
Example: One person is married to only one person.



Unary Relationship Set

2. Binary Relationship Set : Binary relationship set is a relationship set where two entity sets participate in a relationship set.

Example :

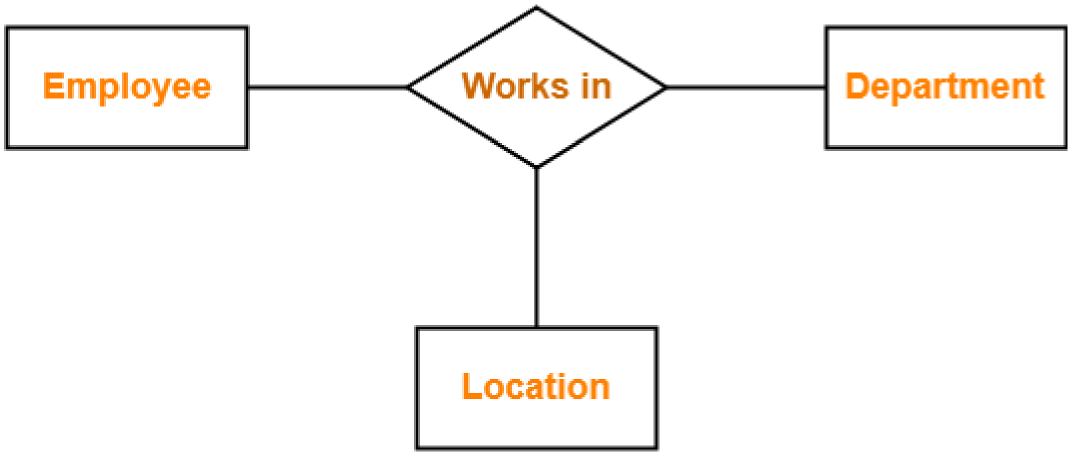


Binary Relationship Set

3. Ternary Relationship Set-

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

Example-



Ternary Relationship Set

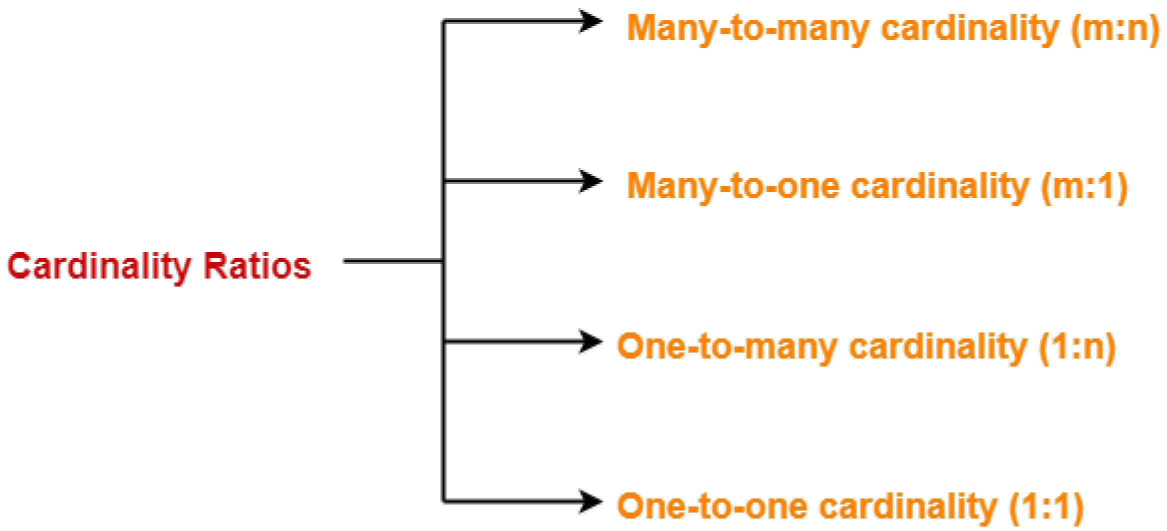
4. N-ary Relationship Set-

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

Cardinality Constraints / Ratios: Cardinality constraint defines the maximum number of relationship instances in which an entity can participate.

Types of Cardinality Ratios :

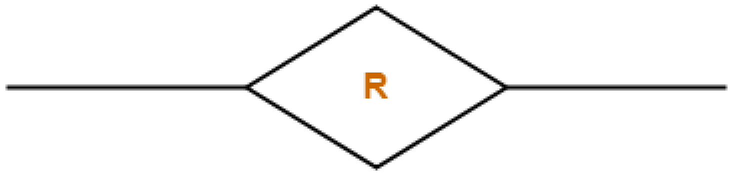
There are 4 types of cardinality ratios-


Cardinality Ratios
Many-to-many cardinality (m:n)
Many-to-one cardinality (m:1)
One-to-many cardinality (1:n)
One-to-one cardinality (1:1)

1. Many-to-Many Cardinality: By this cardinality constraint,

- An entity in set A can be associated with any number (zero or more) of entities in set B.
- An entity in set B can be associated with any number (zero or more) of entities in set A.

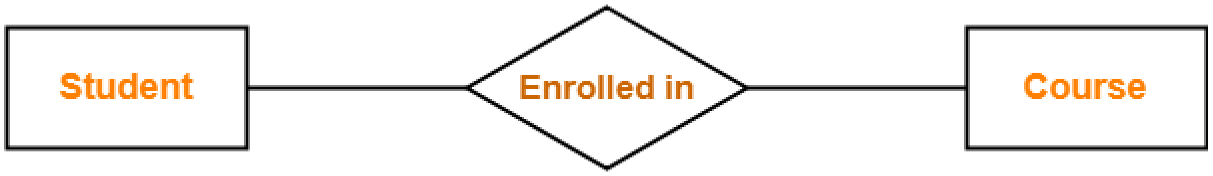
Symbol Used-



Cardinality Ratio = m : n

Example-

Consider the following ER diagram-



Many to Many Relationship

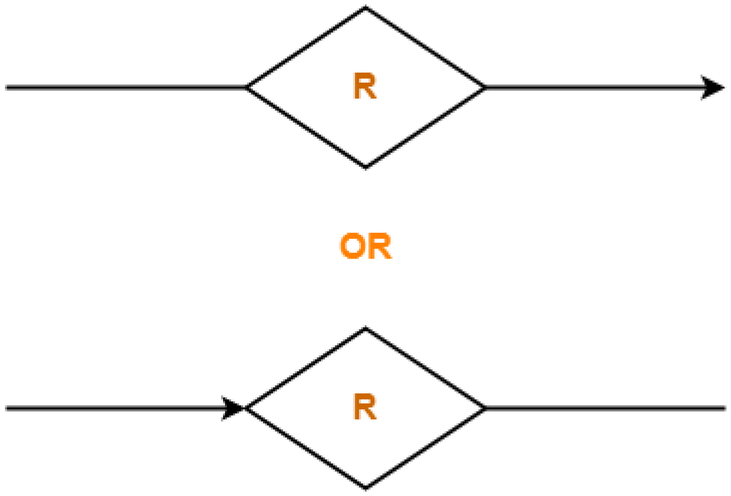
Here,

- One student can enroll in any number (zero or more) of courses.
- One course can be enrolled by any number (zero or more) of students.

2. Many-to-One Cardinality: By this cardinality constraint,

- An entity in set A can be associated with at most one entity in set B.
- An entity in set B can be associated with any number (zero or more) of entities in set A.

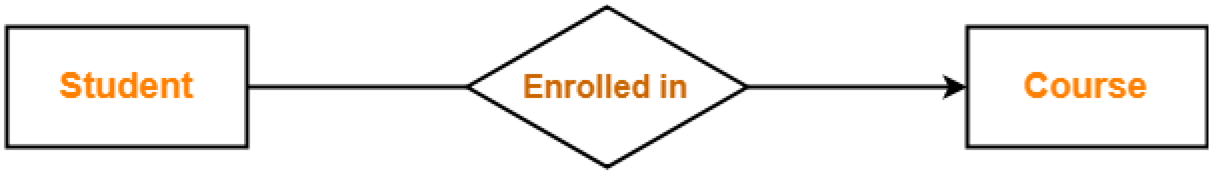
Symbol Used-



Cardinality Ratio = m : 1

Example-

Consider the following ER diagram-



Many to One Relationship

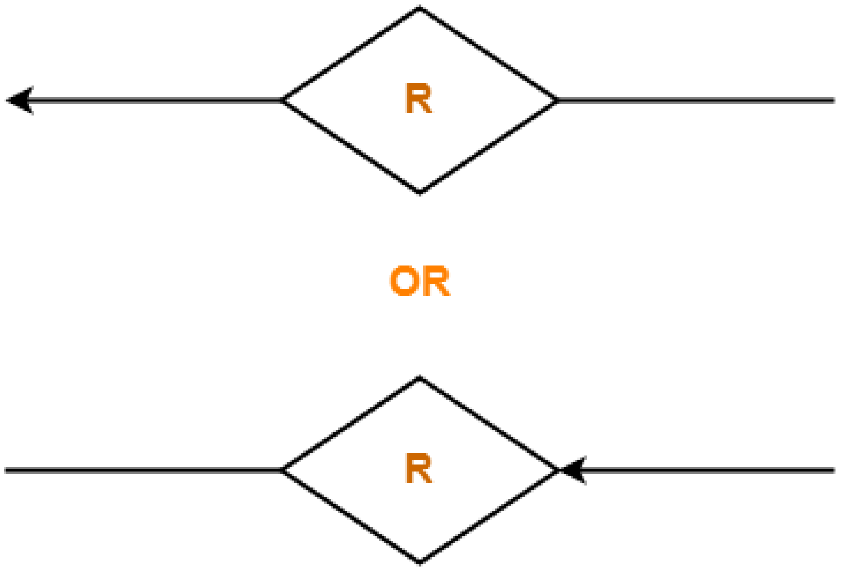
Here,

- One student can enroll in at most one course.
- One course can be enrolled by any number (zero or more) of students

3. One-to-Many Cardinality: By this cardinality constraint,

- An entity in set A can be associated with any number (zero or more) of entities in set B.
- An entity in set B can be associated with at most one entity in set A.

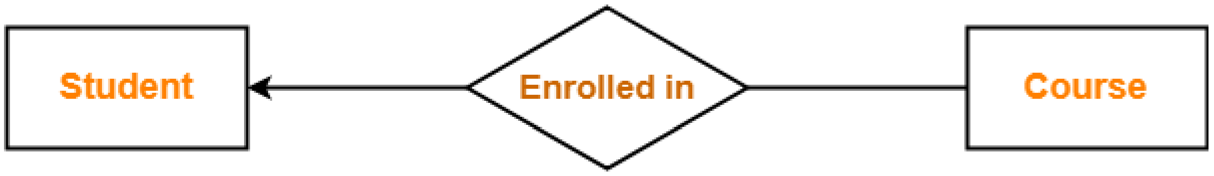
Symbol Used-



Cardinality Ratio = 1 : n

Example-

Consider the following ER diagram-



One to Many Relationship

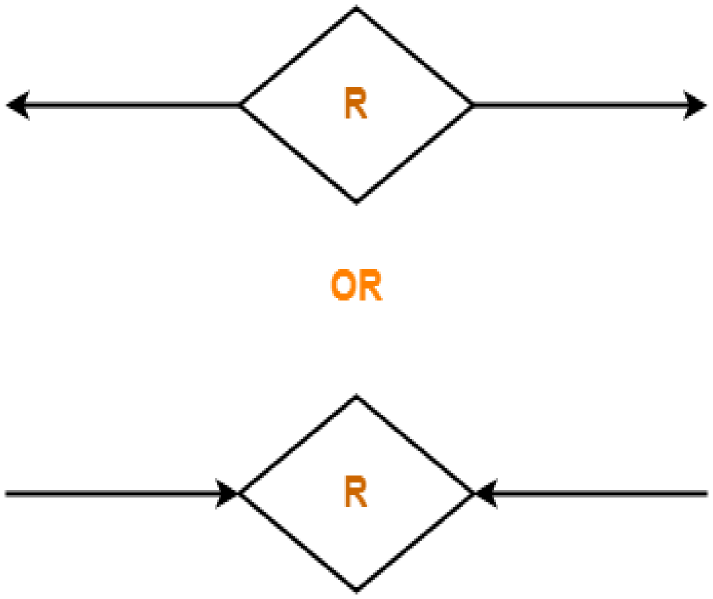
Here,

- One student can enroll in any number (zero or more) of courses.
- One course can be enrolled by at most one student.

4. One-to-One Cardinality: By this cardinality constraint,

- An entity in set A can be associated with at most one entity in set B.
- An entity in set B can be associated with at most one entity in set A.

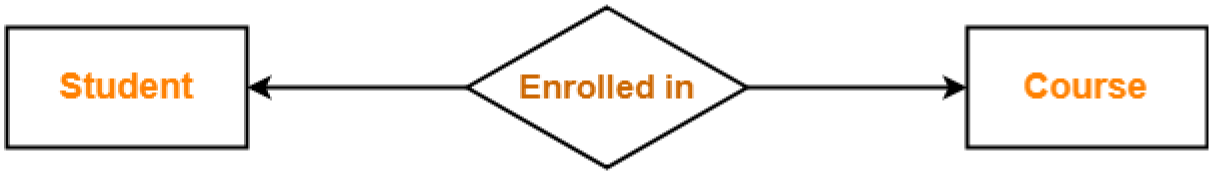
Symbol Used :



Cardinality Ratio = 1 : 1

Example-

Consider the following ER diagram-



One to One Relationship

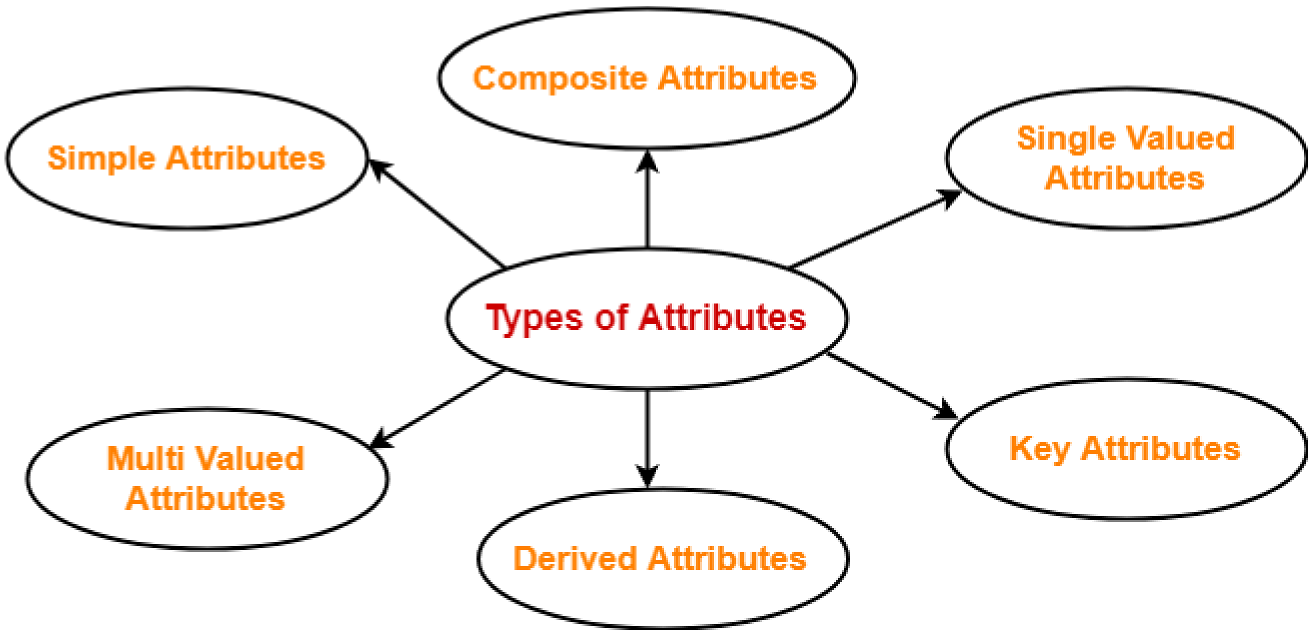
Here,

- One student can enroll in at most one course.
- One course can be enrolled by at most one student.

Attributes in ER Diagram:

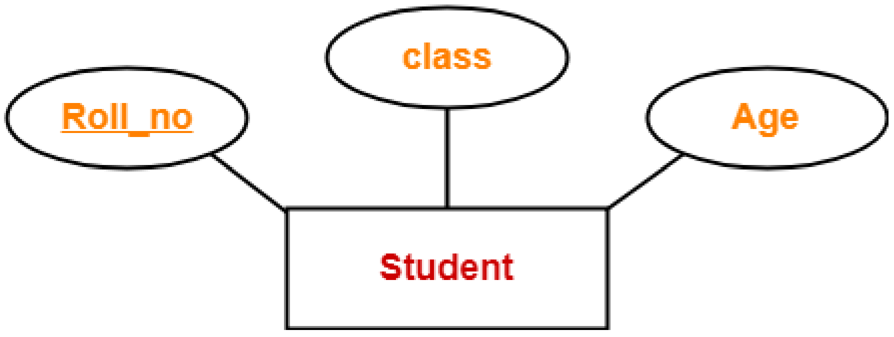
- Attributes are the descriptive properties which are owned by each entity of an Entity Set.
- There exist a specific domain or set of values for each attribute from where the attribute can take its values.

Types of Attributes: In ER diagram, attributes associated with an entity set may be of the following types-



1. Simple Attributes: Simple attributes are those attributes which can not be divided further.

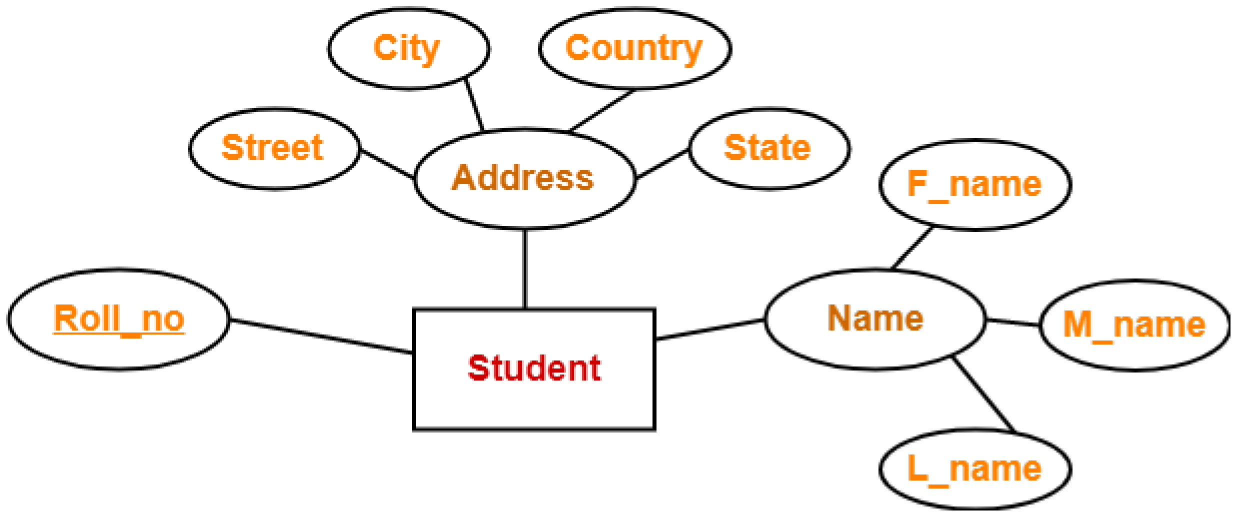
Example :



Here, all the attributes are simple attributes as they cannot be divided further.

2. Composite Attributes: Composite attributes are those attributes which are composed of many other simple attributes.

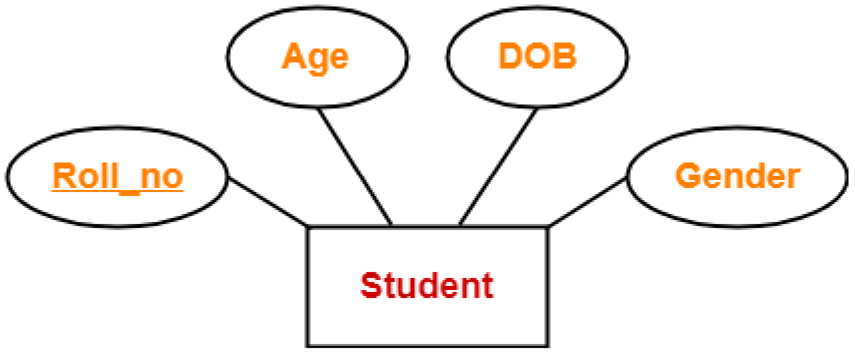
Example:



Here, the attributes "Name" and "Address" are composite attributes as they are composed of many other simple attributes.

3. Single Valued Attributes: Single valued attributes are those attributes which can take only one value for a given entity from an entity set.

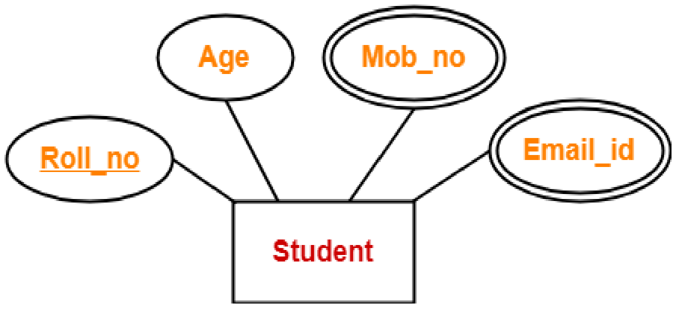
Example:



Here, all the attributes are single valued attributes as they can take only one specific value for each entity.

4. Multi Valued Attributes: Multi valued attributes are those attributes which can take more than one value for a given entity from an entity set.

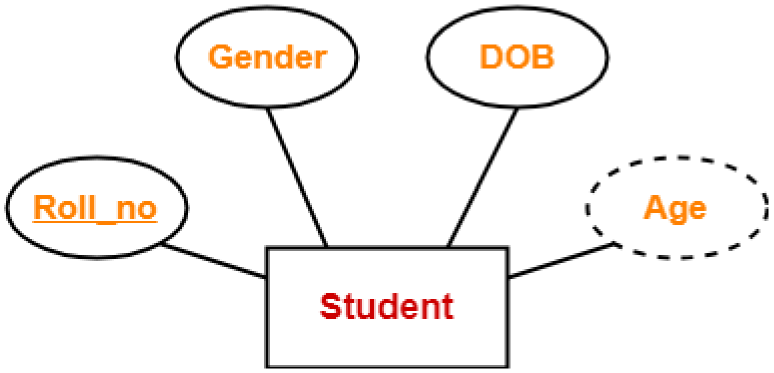
Example:



Here, the attributes “Mob_no” and “Email_id” are multi valued attributes as they can take more than one values for a given entity.

5. Derived Attributes : Derived attributes are those attributes which can be derived from other attribute(s).

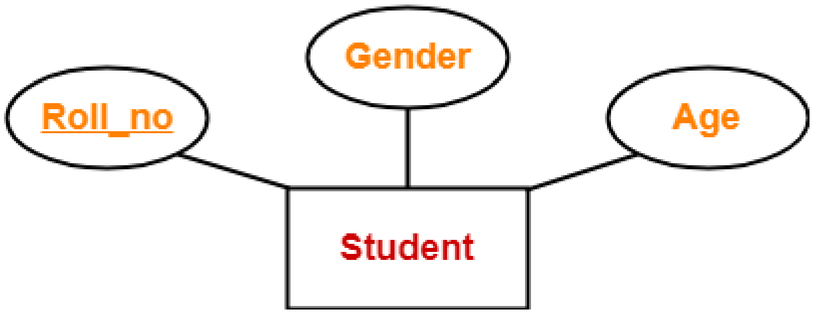
Example-



Here, the attribute “Age” is a derived attribute as it can be derived from the attribute “DOB”.

6. Key Attributes: Key attributes are those attributes which can identify an entity uniquely in an entity set.

Example-

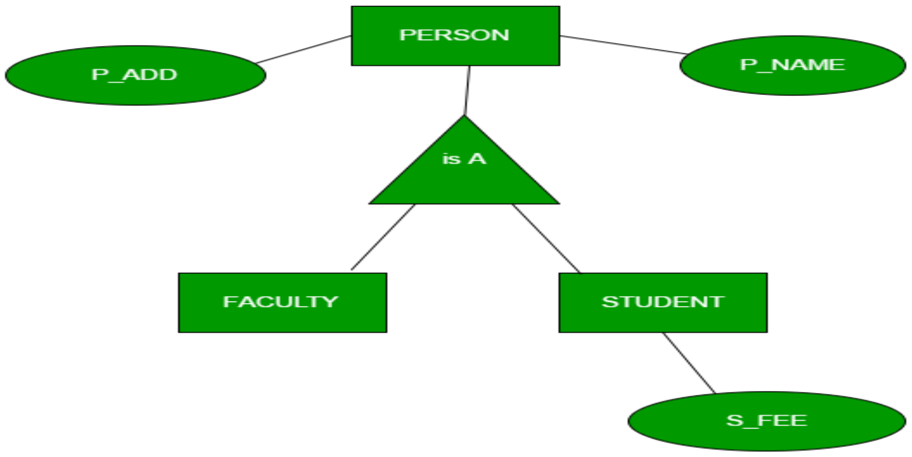


Here, the attribute “Roll_no” is a key attribute as it can identify any student uniquely.

Additional Features of the ER Model

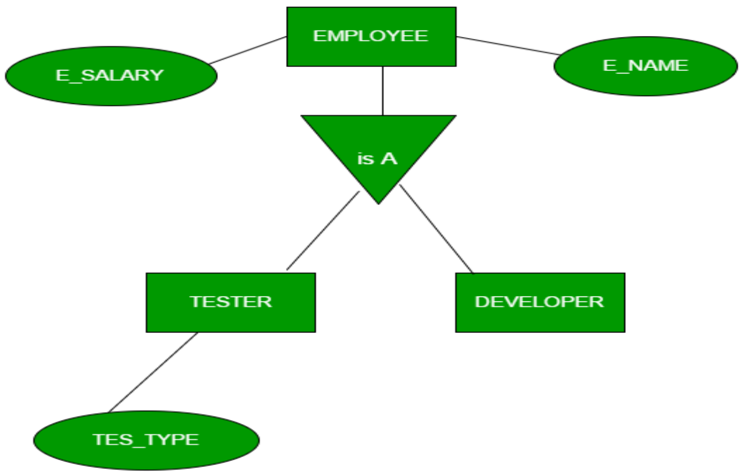
Generalization : Generalization is the process of extracting common properties from a set of entities and create a generalized entity from it. It is a bottom-up approach in which two or more entities can be generalized to a higher level entity if they have some attributes in common.

Example : STUDENT and FACULTY can be generalized to a higher level entity called PERSON as shown in Figure 1. In this case, common attributes like P_NAME, P_ADD become part of higher entity (PERSON) and specialized attributes like S_FEE become part of specialized entity (STUDENT).



Specialization : In specialization, an entity is divided into sub-entities based on their characteristics. It is a top-down approach where higher level entity is specialized into two or more lower level entities.

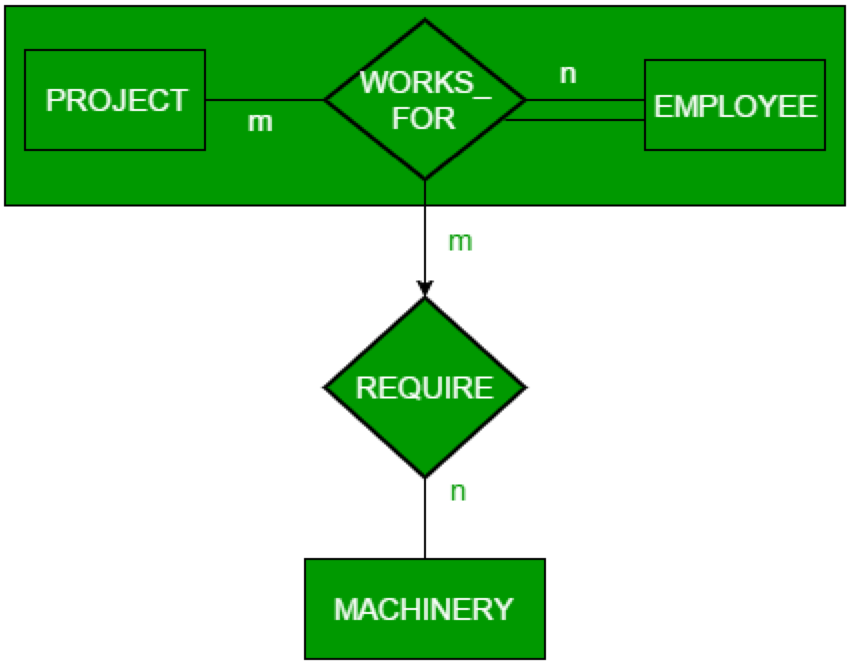
Example : EMPLOYEE entity in an Employee management system can be specialized into DEVELOPER, TESTER etc. as shown in Figure 2. In this case, common attributes like E_NAME, E_SAL etc. become part of higher entity (EMPLOYEE) and specialized attributes like TES_TYPE become part of specialized entity (TESTER).



Aggregation :

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

Example : Employee working for a project may require some machinery. So, REQUIRE relationship is needed between relationship WORKS_FOR and entity MACHINERY. Using aggregation, WORKS_FOR relationship with its entities EMPLOYEE and PROJECT is aggregated into single entity and relationship REQUIRE is created between aggregated entity and MACHINERY.



CONCEPTUAL DESIGN WITH THE ER MODEL :

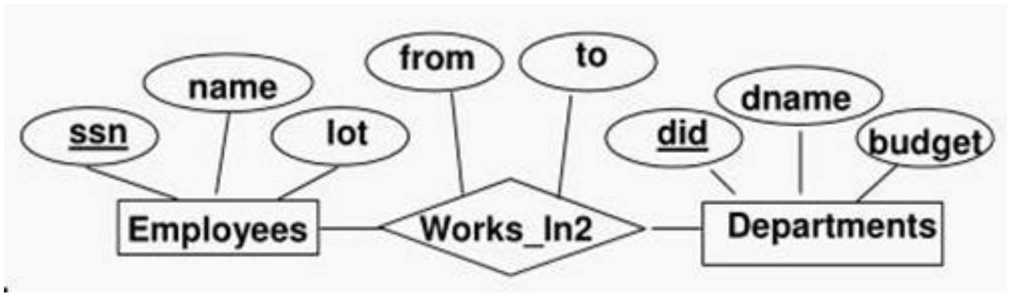
Developing an ER diagram presents several choices,including the following:

- Should a concept be modeled as an entity or an attribute?
- Should a concept be modeled as an entity or a relationship?
- What are the relationship sets and their participating entity sets? Shouls we use binary or ternary relationships?
- Should we use aggregation.

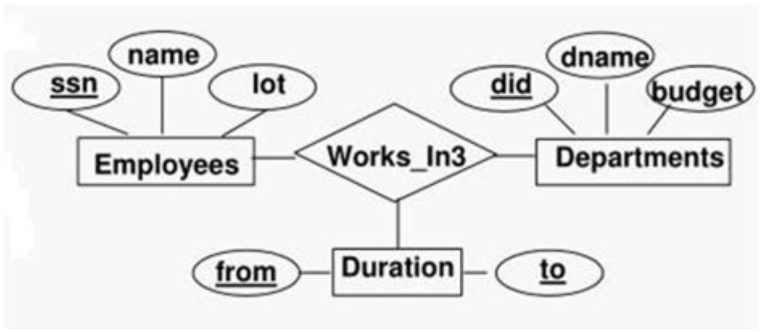
Entity versus Attribute

- Should address be an attribute of Employees or an entity (connected to Employees by a relationship)?
- Depends upon the use we want to make of address information and the semantics of the data.
    - If we have several addresses per employee, address must be an entity (since attributes cannot be set-valued).
    - If the structure (city, street, etc) is important, e.g., we want to retrive employees in a given city, address must be modeled as an entity (since attribute values are atomic).

Works_In2 does not allow an employee to work in a department for two or more periods.

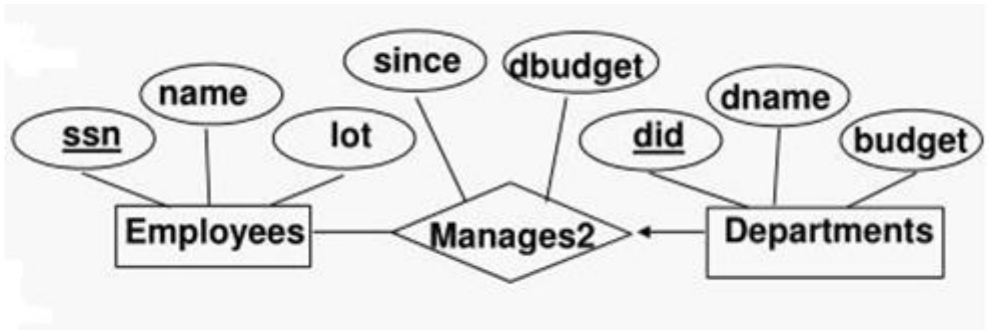


Similar to the problem of wanting to record several addresses for an employee: we want to record several values of the descriptive attributes for each instance of this relationship.



Entity versus Relationship

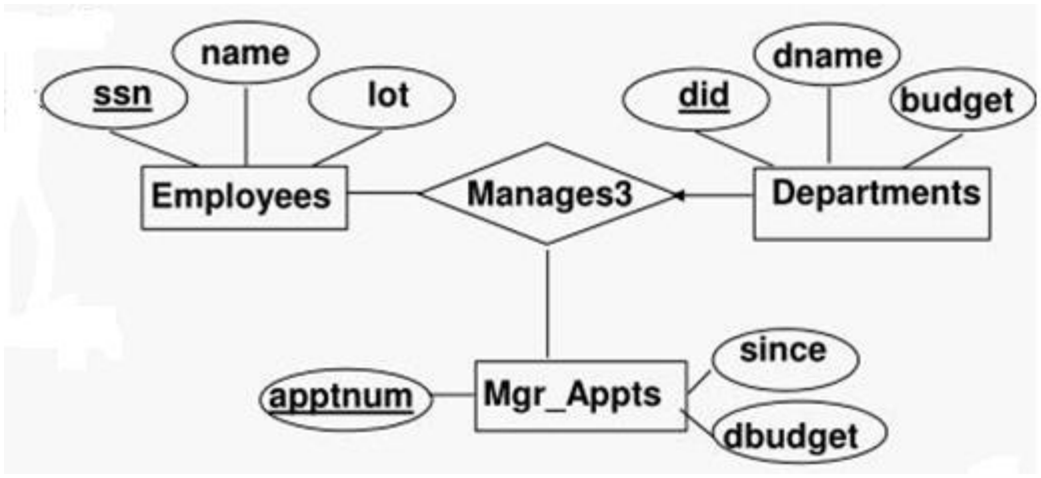
First ER diagram OK if a manager gets a separate discretionary budget for each department.



What if a manager gets a discretionary budget that covers all managed departments?

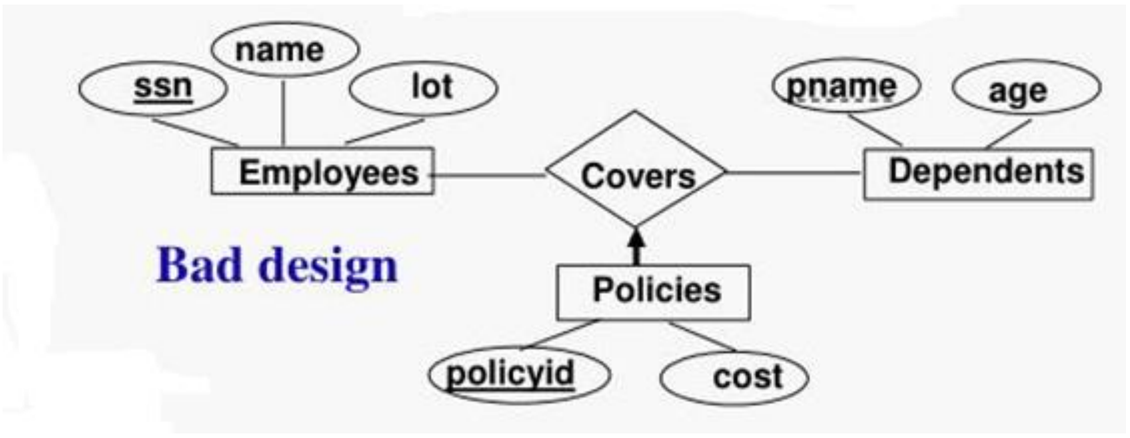
Redundancy of dbudget, which is stored for each dept managed by the manager.

Misleading : Suggests dbudget tied to managed dept.



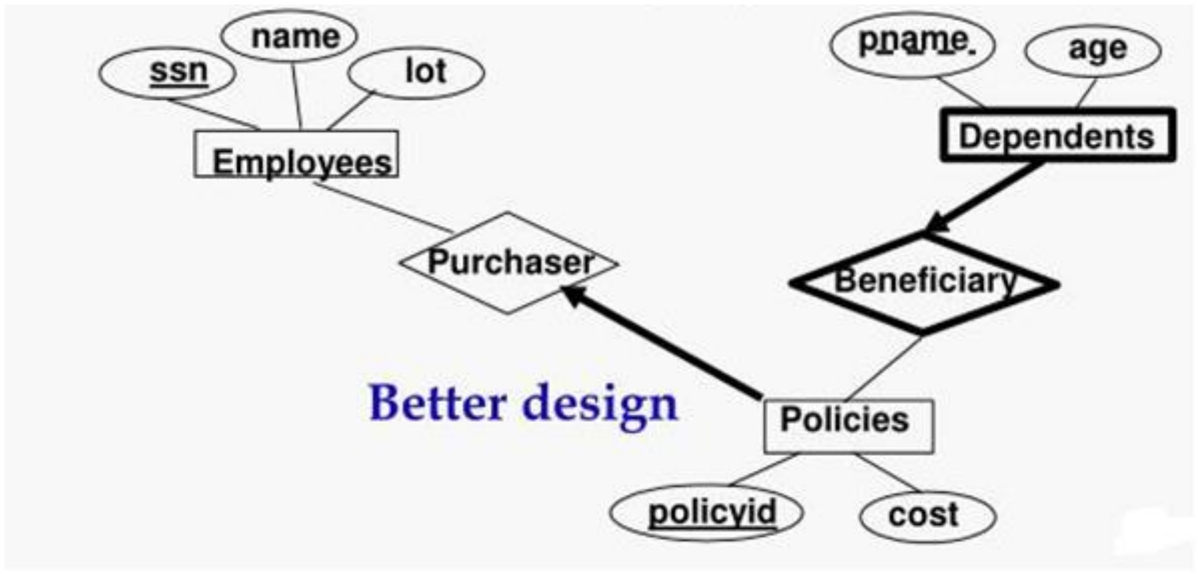
Binary versus Ternary Relationships

- Suppose that a policy cannot be owned jointly by two or more Employee
- Key constraint on policies would mean policy can only cover 1 dependant?
- Suppose that Every policy must be owned by some employee.
- Acceptable if each policy covers at least one dependant.



What are the additional constraints in this 2nd diagram?

- Policy cannot be owned jointly by two or more employees.
- Every policy must be owned by some employee.
- Dependent is a weak entity set, and each dependent entity is uniquely identified by taking **pname** in conjuction with the **policyid** of a policy.
- Dependents cannot be beneficiary of more than one policy.



- Previous example illustrated a case when two binary relationships were better than one ternary relationship.
- An example in the other direction: a ternary relation **Contracts** relates entity sets **Parts**, **Departments** and **Suppliers** and has descriptive attribute *qty*.
- No combination of binary relationships is an adequate substitute for a least two reasons:
    - S "can-supply" P, D "needs" P, and D "deals-with" S does not imply that D has agreed to buy P from S.
    - How do we record *qty*?