**Syllabus:**

**The Entity-Relationship (ER) Model**

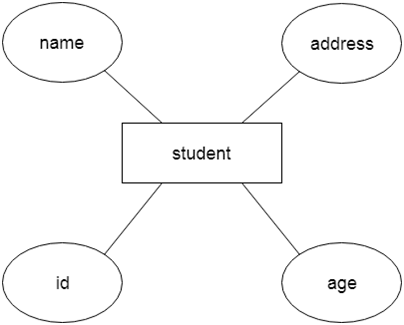
**Entity Types, Entity Sets, Attributes, and Keys,**

**Relationship Types, Relationship Sets, Weak Entity Types**

**Generalization, Specialization and Aggregation,**

**Extended Entity Relationship (EER) Model.**

* ***The Entity-Relationship (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.



* Advantages are-

1. Simplicity and Understandability:
2. Visual Clarity:
3. Database Design:
4. Data Integrity:
5. Scalability:

* Disadvantages are-

1. Limited Expressivity:
2. Ambiguity:
3. Lack of Implementation Details:
4. Time and Effort:
5. Evolution and Maintenance:

* Applications are-

1. Database design.
2. System documentation
3. Conceptual design.
4. Data migration.
5. Query optimization.

* Symbols used are-

1. **Rectangles:**

* Rectangles represent Entities in the ER Model.

1. **Ellipses:**

* Ellipses represent Attributes in the ER Model.

1. **Diamond:**

* Diamonds represent Relationships among Entities.

1. **Lines:**

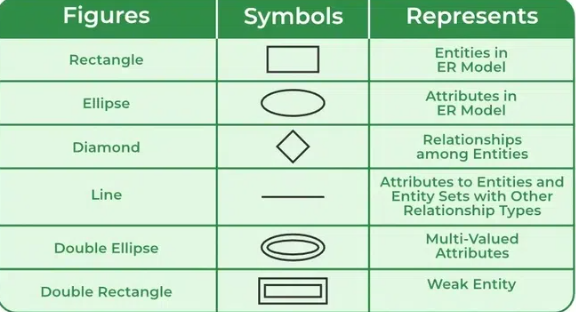
* Lines represent attributes to entities and entity sets with other relationship types.

1. **Double Ellipse:**

* Double Ellipses represent Multi-Valued Attributes.

1. **Double Rectangle:**

* Double Rectangle represents a Weak Entity.

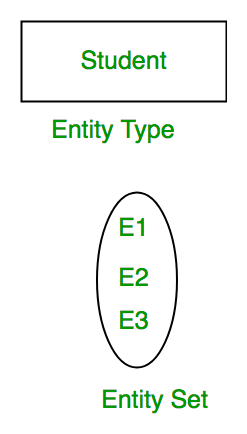


* Component of ER Diagram



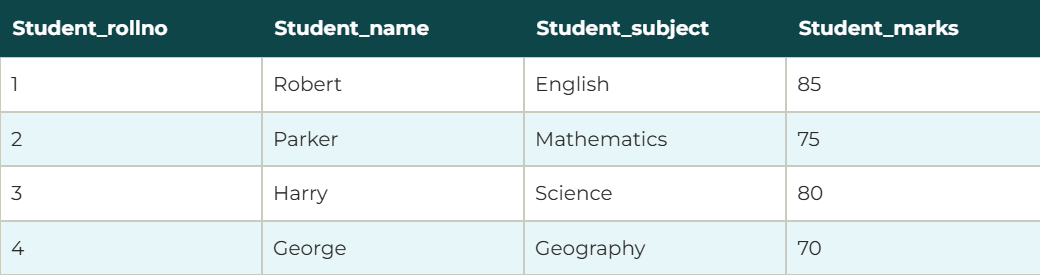
1. Entity:

* An entity may be any object, class, person or place. In the ER diagram, an entity can be represented as rectangles.
* It may be an object with physical and conceptual existence ie. Physical existence means person house car employee is also called tangible (physical object that exists in a real world) and conceptual existence means company job university course is also called intangible (intangible means that cannot be physically touched but still need to be represented in database).
* Entity set-
* An Entity is an object of Entity Type and a set of all entities is called an entity set. For Example, E1 is an entity having Entity Type Student and the set of all students is called Entity Set. In ER diagram, Entity Type is represented as:



* We can represent the entity set in ER Diagram but can’t represent entity in ER Diagram because entity is row and column in the relation and ER Diagram is graphical representation of data.
* Consider an organization as an example- manager, product, employee, department etc. can be taken as an entity.



* 

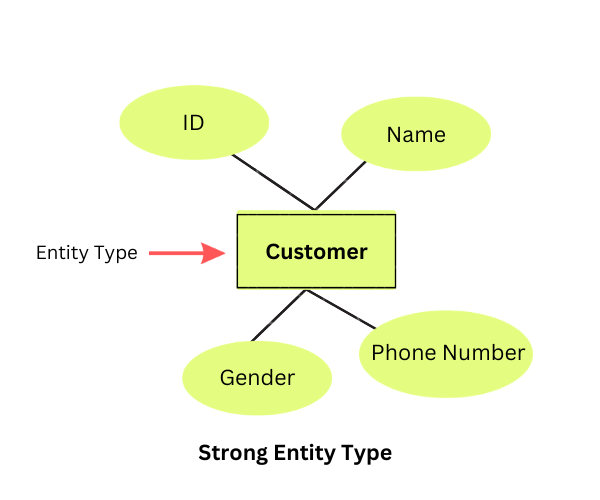
🡺**Types of entity are-**

1. **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.
* For ExampleA company may store the information of dependents (Parents, Children, Spouse) of an Employee. But the dependents can’t exist without the employee. So Dependent will be a Weak Entity Typeand Employee will be Identifying Entity type for Dependent, which means it is Strong Entity Type .
* The participation of weak entity types is always total. The relationship between the weak entity type and its identifying strong entity type is called identifying relationship and it is represented by a double diamond.



1. **Strong entity**

* It is an entity that has its own existence and is independent.The entity relationship diagram represents a strong entity type with the help of a single rectangle.
* 
* In the above example, the "Customer" is the entity type with attributes such as ID, Name, Gender, and Ph….one Number. Customer is a strong entity type as it has a unique ID for each customer.

1. 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.

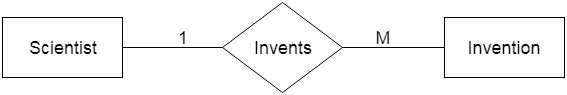
For example, A female can marry to one male, and a male can marry to one female.



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.

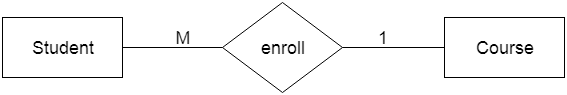
For example, Scientist can invent many inventions, but the invention is done by the only specific scientist.



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.

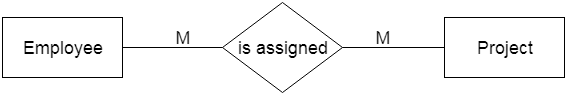


Advertisement

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.



🡺Components of relationships

1. Name
2. Degree
3. Structural constrain.

**Name –**

* Every relationship must have unique name.

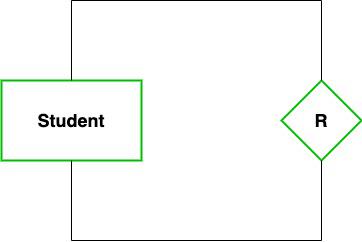
**Degree of relationship set-**

* A degree of relationship represents the number of entity types that are associated with a relationship.
* For example, we have two entities, one is a student and the other is a bag and they are connected with the primary key and foreign key. So, here we can see that the degree of relationship is 2 as 2 entities are associating in a relationship.
* Types of degree

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

## Unary (Degree 1)

* In this type of relationship, both the associating entity types are the same. So, we can say that unary relationships exist when both entity types are the same and we call them the degree of relationship is 1.
* In other words, in a relation only one entity set is participating then such type of relationship is known as a unary relationship.
* **Example:** In a particular class, we have many students, there are monitors too. So, here class monitors are also students. Thus, we can say that only students are participating here. So the degree of such type of relationship is 1.



## Binary (Degree 2)

🡺In a Binary relationship, there are two types of entity associates. So, we can say that a Binary relationship exists when there are two types of entity and we call them a degree of relationship is 2.

🡺In other words, in a relation when two entity sets are participating then such type of relationship is known as a binary relationship. This is the most used relationship and one can easily be converted into a relational table.

🡺**Example:**We have two entity types ‘Student’ and ‘ID’ where each ‘Student’ has his ‘ID’. So, here two entity types are associating we can say it is a binary relationship. Also, one ‘Father’ can have many ‘daughters’ but each ‘daughter’ should belong to only one ‘father. We can say that it is a one-to-many binary relationship.

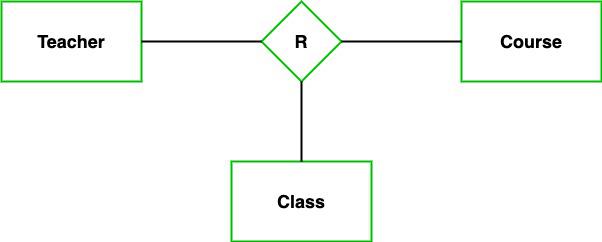


## Ternary (Degree 3)

🡺In the Ternary relationship, there are three types of entity associates. So, we can say that a Ternary relationship exists when there are three types of entity and we call them a degree of relationship is 3.

🡺Since the number of entities increases due to this, it becomes very complex to turn E-R into a relational table.

🡺**Example:** We have three entity types ‘Teacher’, ‘Course’, and ‘Class’. The relationship between these entities is defined as the teacher teaching a particular course, also the teacher teaches a particular class. So, here three entity types are associating we can say it is a ternary relationship.

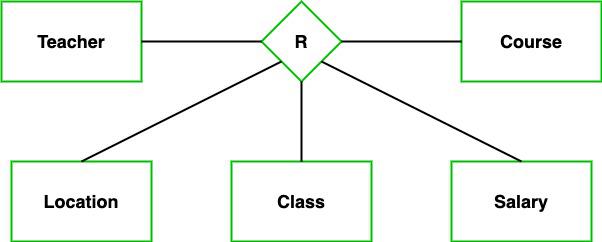


## N-ary (n Degree)

🡺In the N-ary relationship, there are n types of entity that associates. So, we can say that an N-ary relationship exists when there are n types of entities.

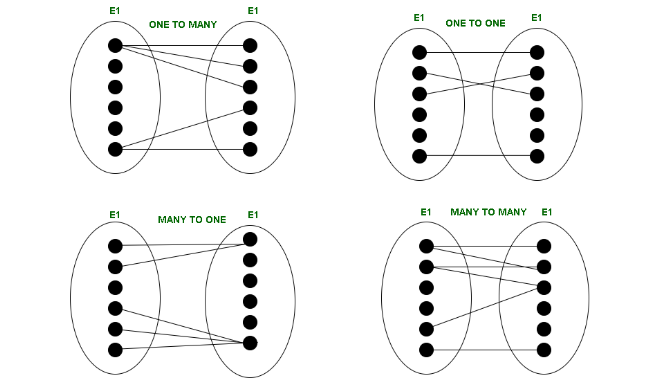
🡺There is one limitation of the N-ary relationship, as there are many entities so it is very hard to convert into an entity, rational table. So, this is very uncommon, unlike binary which is very much popular.

**🡺Example:**We have 5 entities Teacher, Class, Location, Salary, Course. So, here five entity types are associating we can say an n-ary relationship is 5.



**Structural constrain-**

* Structural constraints specify and determine how the entities take part in the relationships and this gives an outline of how the interactions between the entities can be designed in a database.
* Two types of constraints are
* **cardinality** and
* **participation**.
  1. Cardinality
* it specify the number of instances in a relationship.
* It is defined as one-to-one, one-to-many, or many-to-many.
* 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.
* Cardinality ratio is the entities which is denoted by rectangle and relationship by diamonds.
* Types of cardinality ratio are-

****

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.

1. **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.

1. **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.

1. **Many-to-many (N: N)**

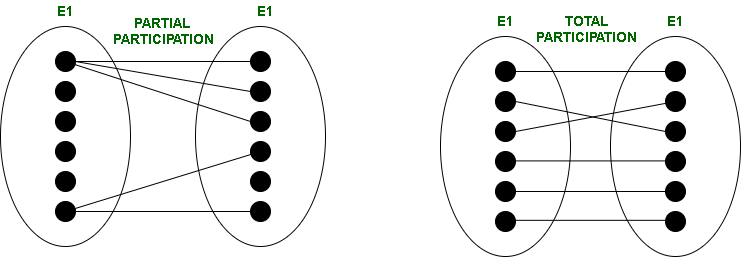
* The cardinality is said to be many to many if entities in both the entity sets take part many times (at least twice) in the relationship set.
  1. Participation
* Participation can be total or partial.

### Total Participation

* **When each entity in an entity set participates in a relation, it is called *Total Participation*.**
* In ER modeling, total participation exists when the instances of the entity must necessarily participate in at least one relationship instance.
* For example, in a university system, where every student must be enrolled in at least one course, total participation exists in the relationship between “Student” and “Course”. Similarly, if each professor teaches at least one course, then the relationship between “Professor” and “Course” also displays total participation. The notion here is that no entity can exist without being involved in the relationship.

### 🡺Partial Participation

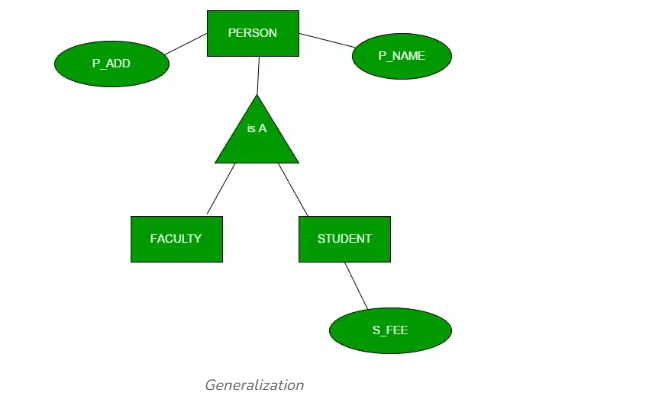
* **When all entities in the given entity set do not participate in a relation, it is called *Partial Participation*.**
* Contrary to full participation, partial participation allows some components of a particular entity to be present without being part of any relation.
* Some entities may or may not be part of the relationship, and, in case of a single line in the ER diagram, that is indicated. For instance, in the university system, if some professors do not teach any courses, then their participation would be partial in the “Teaches” relationship.



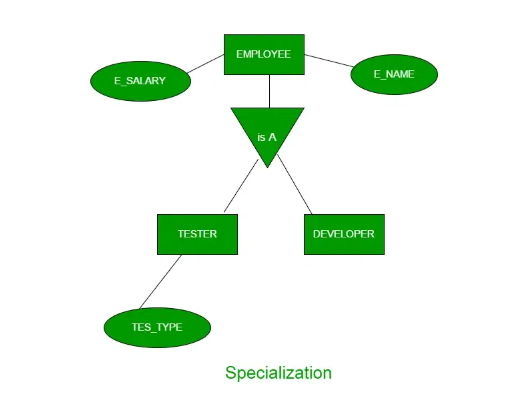
* ***Conversion of ER to Relational model:***
* From notebook.
* ***Relational dbms:***
* From notebook.
* ***Generalization, Specialization and Aggregation***

**Generalization**

1. Generalization is the process of extracting common properties from a set of entities and creating a generalized entity from it.
2. 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.
3. For 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, and P\_ADD become part of a higher [entity](https://www.geeksforgeeks.org/difference-between-entity-and-object/) (PERSON), and specialized [attributes](https://www.geeksforgeeks.org/types-of-attributes-in-er-model/) like S\_FEE become part of a specialized entity (STUDENT).

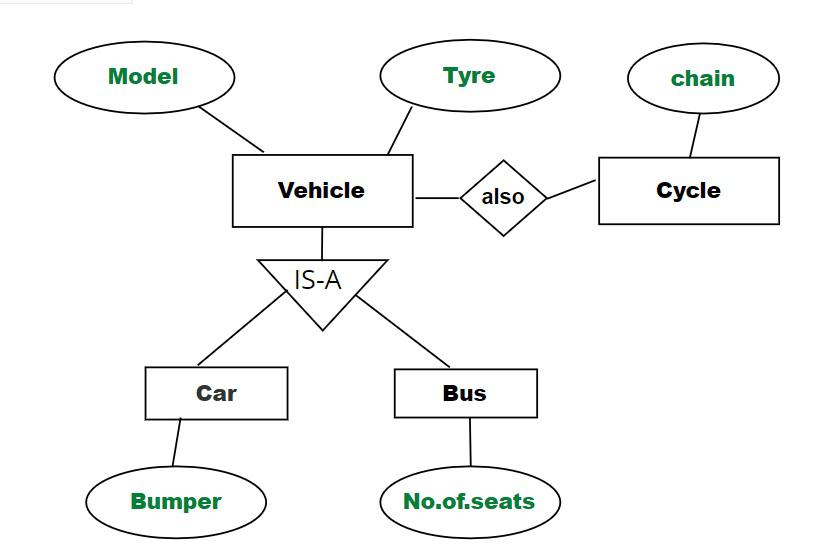


**Specialization**

1. In specialization, an entity is divided into sub-entities based on its characteristics.
2. It is a top-down approach where the higher-level entity is specialized into two or more lower-level [entities](https://www.geeksforgeeks.org/difference-between-entity-entity-set-and-entity-type/).
3. For Example, an 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 a higher entity (EMPLOYEE), and specialized attributes like TES\_TYPE become part of a specialized entity (TESTER).
4. 

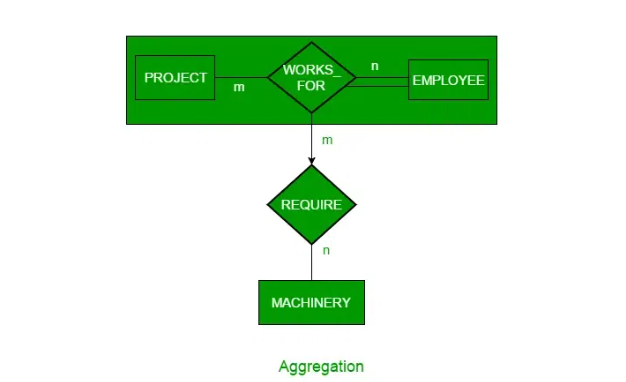
**Inheritance:**

* It is an important feature of generalization and specialization
* **Attribute inheritance**:
* It allows lower level entities to inherit the attributes of higher level entities and vice versa. In diagram Carentity is an inheritance of Vehicleentity ,So Car can acquire attributes of Vehicle**.**Example:car can acquire Modelattribute of Vehicle**.**
* **Participation inheritance:**
* Participation inheritance in ER modeling refers to the inheritance of participation constraints from a higher-level entity (superclass) to a lower-level entity (subclass).
* It ensures that subclasses adhere to the same participation rules in relationships, although attributes and relationships themselves are inherited differently. In diagram Vehicle entity has an relationship with Cycle entity, but it would not automatically acquire the relationship itself with the Vehicle entity. Participation inheritance only refers to the inheritance of participation constraints, not the actual relationships between entities.



*Example of Relation*

**Aggregation**

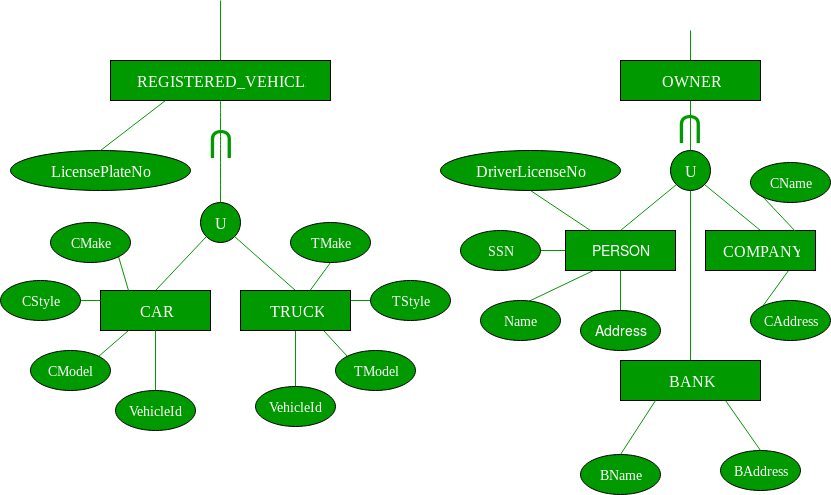
1. An ER diagram is not capable of representing the relationship between an entity and a relationship which may be required in some scenarios.
2. In those cases, a relationship with its corresponding entities is aggregated into a higher-level entity. Aggregation is an abstraction through which we can represent relationships as higher-level entity sets.
3. For Example, an Employee working on a project may require some machinery. So, REQUIRE relationship is needed between the relationship WORKS\_FOR and entity MACHINERY. Using aggregation, WORKS\_FOR relationship with its entities EMPLOYEE and PROJECT is aggregated into a single entity and relationship REQUIRE is created between the aggregated entity and MACHINERY.
4. 

The**ER model**forms the backbone of database design, especially in defining relationships through generalization, specialization, and aggregation.

* ***Extended Entity Relationship (EER) Model.***

1. Enhanced ERDs are high-level models that represent the requirements and complexities of complex databases.
2. The EER model includes all modelling concepts of the ER model.
3. In addition, EER includes the following concepts.
   1. Subclasses and Superclasses

* **A superclass** is a high-level entity that can be further segmented into subclasses or subsets. It is also referred to as a Parent class.
* **A subclass** can be referred to as a child or derived class.
* **Example:**Science is a Super class which has subclasses like Physics, Chemistry, Biology.
  1. Specialization and Generalization
  2. Category or Union type
* Set of Library Members is **UNION**of Faculty, Student, and Staff. A union relationship indicates either type; for example, a library member is either Faculty or Staff or Student.
* Below are two examples that show how **UNION**can be depicted in ERD – Vehicle Owner is UNION of PERSON and Company, and RTO Registered Vehicle is UNION of Car and Truck.

1. 
   1. Attribute and relationship inheritance

## ****Key Features of the EER Model****

1. **Subtypes and Supertypes:**

* The EER model allows for the creation of subtypes and supertypes. A supertype is a generalization of one or more subtypes, while a subtype is a specialization of a supertype. For example, a vehicle could be a supertype, while car, truck, and motorcycle could be subtypes.

1. **Generalization and Specialization:**

* Generalization is the process of identifying common attributes and relationships between entities and creating a supertype based on these common features. Specialization is the process of identifying unique attributes and relationships between entities and creating subtypes based on these unique features.

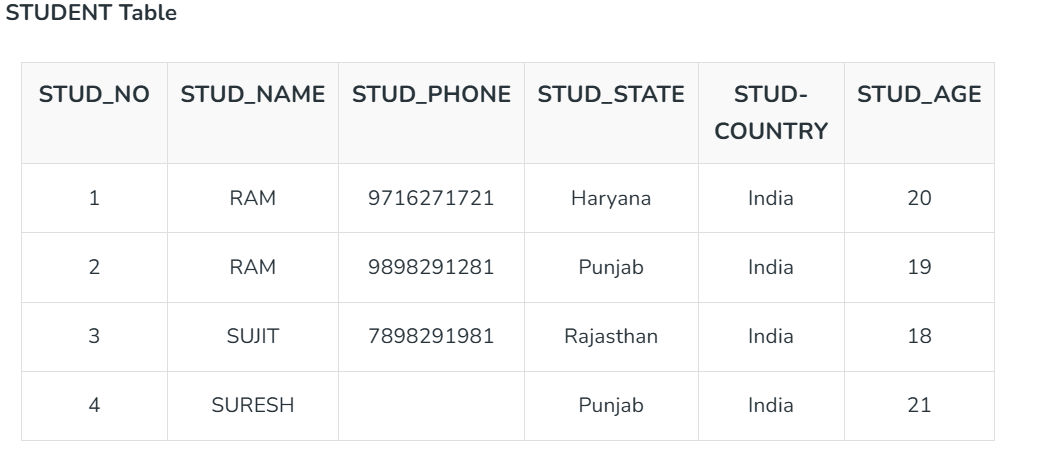
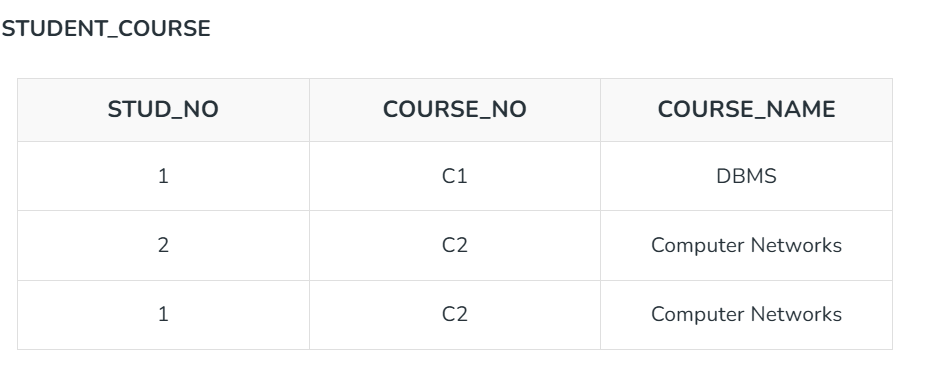
1. **Inheritance:**

* Inheritance is a mechanism that allows subtypes to inherit attributes and relationships from their supertype. This means that any attribute or relationship defined for a supertype is automatically inherited by all its subtypes.

1. **Constraints:**

* The EER model allows for the specification of constraints that must be satisfied by entities and relationships. Examples of constraints include cardinality constraints, which specify the number of relationships that can exist between entities, and participation constraints, which specify whether an entity is required to participate in a relationship.
* **Relational dbms:**
* RDBMS stands for *Relational Database Management System.*
* All modern database management systems like SQL, MS SQL Server, IBM DB2, ORACLE, My-SQL, and Microsoft Access are based on RDBMS.
* It is called Relational Database Management System (RDBMS) because it is based on the relational model introduced by E.F. Codd in 1970.
* Row== tupple,entity, instance,record,cardinality.

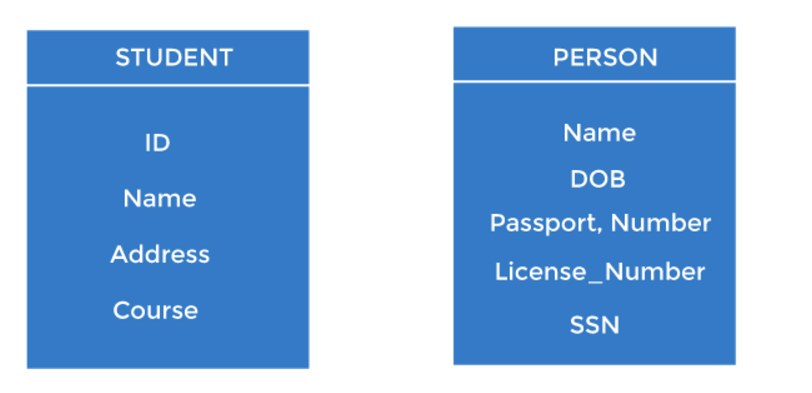
Column==attribute,field,column,domain,degree,etc.

* Properties are:
  1. Each relation has a unique name by which it is identified in the database.
  2. Relation does not contain duplicate tuples.
  3. The tuples of a relation have no specific order.
  4. All attributes in a relation are atomic, i.e., each cell of a relation contains exactly one value.
* Anomalies in the relational model refer to inconsistencies or errors that can arise when working with relational databases, specifically in the context of data insertion, deletion, and modification.
* Database anomalies are the faults in the database caused due to poor management of storing everything in the flat database. It can be removed with the process of [Normalization](https://www.geeksforgeeks.org/normal-forms-in-dbms/), which generally splits the database which results in reducing the anomalies in the database.
* There are different types of anomalies that can occur in referencing and referenced relations:
* These anomalies can be categorized into three types:
* Insertion Anomalies
* Deletion Anomalies
* Update Anomalies.
* 
* 

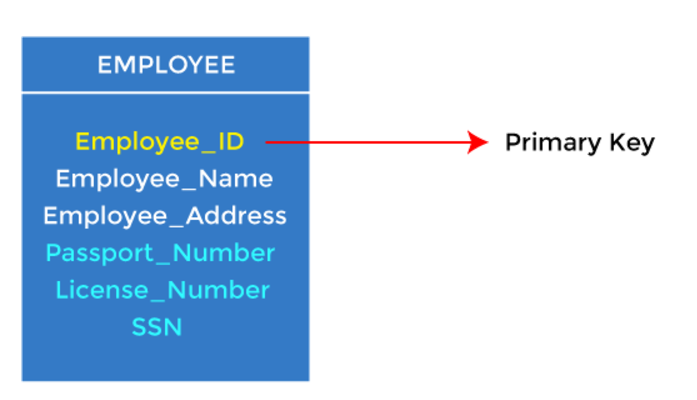
1. **Insertion Anomaly:**

* If a tuple is inserted in referencing relation and referencing attribute value is not present in referenced attribute, it will not allow insertion in referencing relation.
* **Example:**If we try to insert a record in STUDENT\_COURSE with STUD\_NO =7, it will not allow it.

1. **Deletion and Updation Anomaly:**

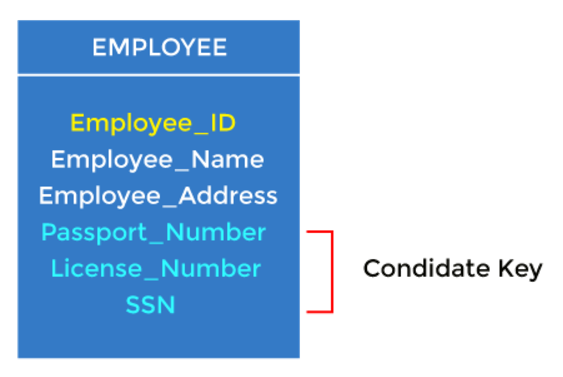
* If a tuple is deleted or updated from referenced relation and the referenced attribute value is used by referencing attribute in referencing relation, it will not allow deleting the tuple from referenced relation.
* **Example:**If we want to update a record from STUDENT\_COURSE with STUD\_NO =1, We have to update it in both rows of the table. If we try to delete a record from STUDENT with STUD\_NO =1, it will not allow it.
* To avoid this, the following can be used in query:
* **ON DELETE/UPDATE SET NULL:** If a tuple is deleted or updated from referenced relation and the referenced attribute value is used by referencing attribute in referencing relation, it will delete/update the tuple from referenced relation and set the value of referencing attribute to NULL.
* **ON DELETE/UPDATE CASCADE:** If a tuple is deleted or updated from referenced relation and the referenced attribute value is used by referencing attribute in referencing relation, it will delete/update the tuple from referenced relation and referencing relation as well.
* **Advantages Anomalies in Relational Model**
  1. **Data Integrity:** Relational databases enforce data integrity through various constraints such as primary keys, foreign keys, and referential integrity rules, ensuring that the data is accurate and consistent.
  2. **Scalability:**Relational databases are highly scalable and can handle large amounts of data without sacrificing performance.
  3. **Flexibility:**The relational model allows for flexible querying of data, making it easier to retrieve specific information and generate reports.
  4. **Security:**Relational databases provide robust security features to protect data from unauthorized access.
* **Disadvantages of Anomalies in Relational Model**
  1. **Redundancy:** When the same data is stored in various locations, a relational architecture may cause data redundancy. This can result in inefficiencies and even inconsistent data.
  2. **Complexity:**Establishing and keeping up a relational database calls for specific knowledge and abilities and can be difficult and time-consuming.
  3. **Performance:** Because more tables must be joined in order to access information, performance may degrade as a database gets larger.
  4. **Incapacity to manage unstructured data:**Text documents, videos, and other forms of semi-structured or unstructured data are not well-suited for the relational paradigm.
* Keys in dbms:
* Key is an attributes(or a set of attributes) that helps to uniquely identify a row/record in table.
* Keys play an important role in the relational database.
* **For example,** ID is used as a key in the Student table because it is unique for each student. In the PERSON table, passport\_number, license\_number, SSN are keys since they are unique for each person.
* 
* Types of keys-

1. Primary key

* It is the first key used to identify one and only one instance of an entity uniquely. An entity can contain multiple keys, as we saw in the PERSON table. The key which is most suitable from those lists becomes a primary key.
* In the EMPLOYEE table, ID can be the primary key since it is unique for each employee. In the EMPLOYEE table, we can even select License\_Number and Passport\_Number as primary keys since they are also unique.
* For each entity, the primary key selection is based on requirements and developers.
* 

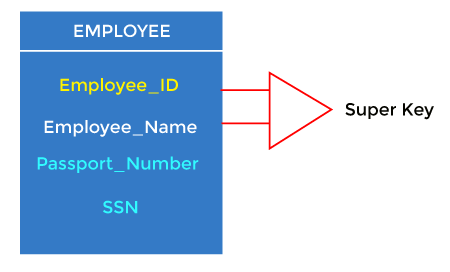
1. Candidate key

* A candidate key is an attribute or set of attributes that can uniquely identify a tuple.
* Except for the primary key, the remaining attributes are considered a candidate key. The candidate keys are as strong as the primary key.
* **For example:** In the EMPLOYEE table, id is best suited for the primary key. The rest of the attributes, like SSN, Passport\_Number, License\_Number, etc., are considered a candidate key.



1. Super Key

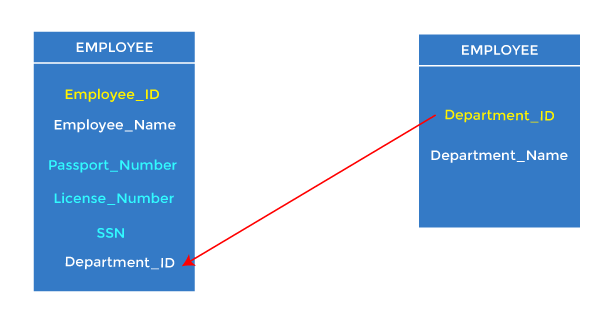
* Super key is an attribute set that can uniquely identify a tuple. A super key is a superset of a candidate key.



* **For example:** In the above EMPLOYEE table, for(EMPLOEE\_ID, EMPLOYEE\_NAME), the name of two employees can be the same, but their EMPLYEE\_ID can't be the same. Hence, this combination can also be a key.
* The super key would be EMPLOYEE-ID (EMPLOYEE\_ID, EMPLOYEE-NAME), etc.

1. Foreign key

* Foreign keys are the column of the table used to point to the primary key of another table.
* Every employee works in a specific department in a company, and employee and department are two different entities. So we can't store the department's information in the employee table. That's why we link these two tables through the primary key of one table.
* We add the primary key of the DEPARTMENT table, Department\_Id, as a new attribute in the EMPLOYEE table.
* In the EMPLOYEE table, Department\_Id is the foreign key, and both the tables are related.



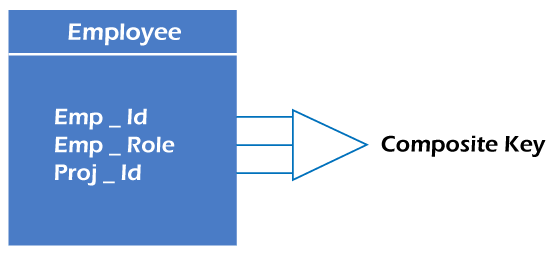
1. Alternate key

* There may be one or more attributes or a combination of attributes that uniquely identify each tuple in a relation. These attributes or combinations of the attributes are called the candidate keys.
* One key is chosen as the primary key from these candidate keys, and the remaining candidate key, if it exists, is termed the alternate key.
* **In other words,** the total number of the alternate keys is the total number of candidate keys minus the primary key. The alternate key may or may not exist. If there is only one candidate key in a relation, it does not have an alternate key.
* **For example,** employee relation has two attributes, Employee\_Id and PAN\_No, that act as candidate keys. In this relation, Employee\_Id is chosen as the primary key, so the other candidate key, PAN\_No, acts as the Alternate key.

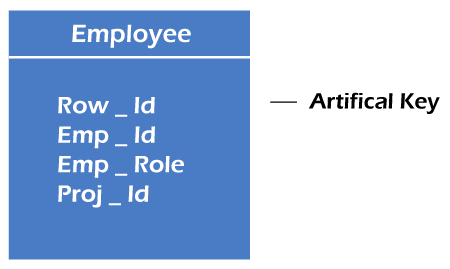


1. Composite key

* Whenever a primary key consists of more than one attribute, it is known as a composite key. This key is also known as Concatenated Key.



* **For example,** in employee relations, we assume that an employee may be assigned multiple roles, and an employee may work on multiple projects simultaneously.
* So the primary key will be composed of all three attributes, namely Emp\_ID, Emp\_role, and Proj\_ID in combination. So these attributes act as a composite key since the primary key comprises more than one attribute.



1. Artificial key

* The key created using arbitrarily assigned data are known as artificial keys. These keys are created when a primary key is large and complex and has no relationship with many other relations. The data values of the artificial keys are usually numbered in a serial order.
* **For example,** the primary key, which is composed of Emp\_ID, Emp\_role, and Proj\_ID, is large in employee relations. So it would be better to add a new virtual attribute to identify each tuple in the relation uniquely.
* Functional dependency:
* The functional dependency is a relationship that exists between two attributes.
* It typically exists between the primary key and non-key attribute within a table.

X   →   Y

The left side of FD is known as a determinant, the right side of the production is known as a dependent.

* **For example:**

Assume we have an employee table with attributes: Emp\_Id, Emp\_Name, Emp\_Address.

Here Emp\_Id attribute can uniquely identify the Emp\_Name attribute of employee table because if we know the Emp\_Id, we can tell that employee name associated with it.

Functional dependency can be written as:

Emp\_Id → Emp\_Name

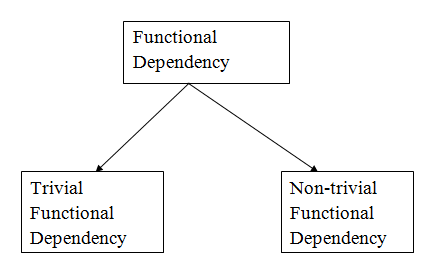
We can say that Emp\_Name is functionally dependent on Emp\_Id.

* **Advantages of Functional Dependencies**

1. Through the identification and removal of redundant or unneeded data, they aid in the reduction of data redundancy in databases.
2. By guaranteeing that data is correct and consistent throughout the database, they enhance data integrity.
3. They make it simpler to add, edit, and remove data, which helps with database management.

* **Disadvantages of Functional Dependencies**

1. The process of identifying functional dependencies can be time-consuming and complex, especially in large databases with many tables and relationships.
2. Overly restrictive functional dependencies can result in slow query performance or data inconsistencies, as data that should be related may not be properly linked.
3. Functional dependencies do not take into account the semantic meaning of data, and may not always reflect the true relationships between data elements.

* Types of Functional dependency;  
  

1. Trivial functional dependency

🡺A → B has trivial functional dependency if B is a subset of A.

🡺The following dependencies are also trivial like: A → A, B → B

**🡺Example:**

Consider a table with two columns Employee\_Id and Employee\_Name.

{Employee\_id, Employee\_Name}   →    Employee\_Id is a trivial functional dependency as

Employee\_Id is a subset of {Employee\_Id, Employee\_Name}.

Also, Employee\_Id → Employee\_Id and Employee\_Name   →    Employee\_Name are trivial dependencies too.

2. Non-trivial functional dependency

🡺A → B has a non-trivial functional dependency if B is not a subset of A.

🡺When A intersection B is NULL, then A → B is called as complete non-trivial.

**🡺Example:**

ID   →    Name,

Name   →    DOB

* Attribute closure/closure on attribute set/closure set of attributes:
* Attribute closure of an attribute set can be defined as set of attributes which can be functionally determined from it.
* It is denoted by F+/C+.
* **How to find attribute closure of an attribute set?**

To find attribute closure of an attribute set:

* Add elements of attribute set to the result set.
* Recursively add elements to the result set which can be functionally determined from the elements of the result set.

### Advantages of Attribute Closure

* 1. Attribute closures help to identify all possible attributes that can be derived from a set of given attributes.
  2. They facilitate database design by identifying relationships between attributes and tables, which can help to optimize query performance.
  3. They ensure data consistency by identifying all possible combinations of attributes that can exist in the database.

### Disadvantages of Attribute Closure

1. The process of calculating attribute closures can be computationally expensive, especially for large datasets.
2. Attribute closures can become too complex to manage, especially as the number of attributes and tables in a [database](https://www.geeksforgeeks.org/what-is-database/) grows.
3. Attribute closures do not take into account the semantic meaning of data, and may not always accurately reflect the relationships between data elements.

* Armstrong axioms/inference rule:
* The Armstrong's axioms are the basic inference rule.
* Armstrong's axioms are used to conclude functional dependencies on a relational database.
* The inference rule is a type of assertion. It can apply to a set of FD(functional dependency) to derive other FD.
* Using the inference rule, we can derive additional functional dependency from the initial set.
* The Functional dependency has 6 types of inference rule:

1. Reflexive Rule (IR1)

* In the reflexive rule, if Y is a subset of X, then X determines Y.

If X ⊇ Y then X  →    Y

* **Example:**

X = {a, b, c, d, e}

Y = {a, b, c}

1. Augmentation Rule (IR2)

* The augmentation is also called as a partial dependency. In augmentation, if X determines Y, then XZ determines YZ for any Z.

If X    →  Y then XZ   →   YZ

* **Example:**

For R(ABCD),  **if** A   →   B then AC  →   BC

1. Transitive Rule (IR3)

* In the transitive rule, if X determines Y and Y determine Z, then X must also determine Z.

If X   →   Y and Y  →  Z then X  →   Z

1. Union Rule (IR4)

* Union rule says, if X determines Y and X determines Z, then X must also determine Y and Z.

If X    →  Y and X   →  Z then X  →    YZ

🡺

X → Y (given)

X → Z (given)  
X → XY (using IR2 on 1 by augmentation with X. Where XX = X)  
XY → YZ (using IR2 on 2 by augmentation with Y)  
X → YZ (using IR3 on 3 and 4)

1. Decomposition Rule (IR5)

* Decomposition rule is also known as project rule. It is the reverse of union rule.
* This Rule says, if X determines Y and Z, then X determines Y and X determines Z separately.

If X   →   YZ then X   →   Y and X  →    Z

**🡺**

X → YZ (given)  
 YZ → Y (using IR1 Rule)  
 X → Y (using IR3 on 1 and 2)

1. Pseudo transitive Rule (IR6)

* In Pseudo transitive Rule, if X determines Y and YZ determines W, then XZ determines W.

If X   →   Y and YZ   →   W then XZ   →   W

🡺

X → Y (given)  
WY → Z (given)  
WX → WY (using IR2 on 1 by augmenting with W)  
WX → Z (using IR3 on 3 and 2)