Database Management System (DBMS)

CSC403

Subject Incharge

Prachiti Pimple

Assistant Professor

email: prachiti@sfit.ac.in

Entity–Relationship Data Model

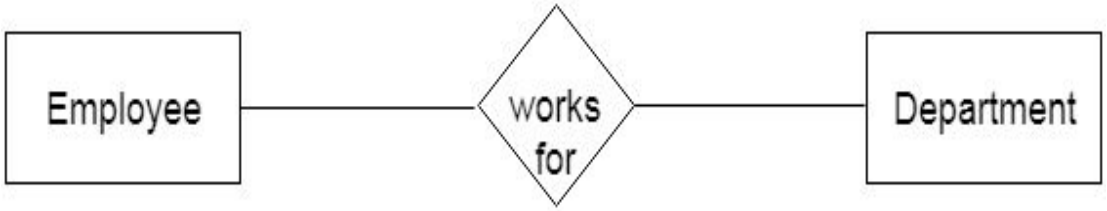
- ER model stands for an Entity-Relationship model.
- It is a high-level data model.
- This model is used to define the data elements and relationship for a specified system.
- It develops a conceptual design for the database. It also develops a very simple and easy to design view of data.

ER diagram basically having three components:

- **Entities** − It is a real-world thing which can be a person, place.
  For Example: Department, Admin, Courses, Teachers, Students, Building, etc are some of the entities of a School Management System.
- **Attributes** − An entity which contains a real-world property called an attribute.
  For Example: The entity employee has the property like employee id, salary, age, etc.
- **Relationship** − Relationship tells how two entities are related.
  For Example: Employee works for a department.

1. Entity

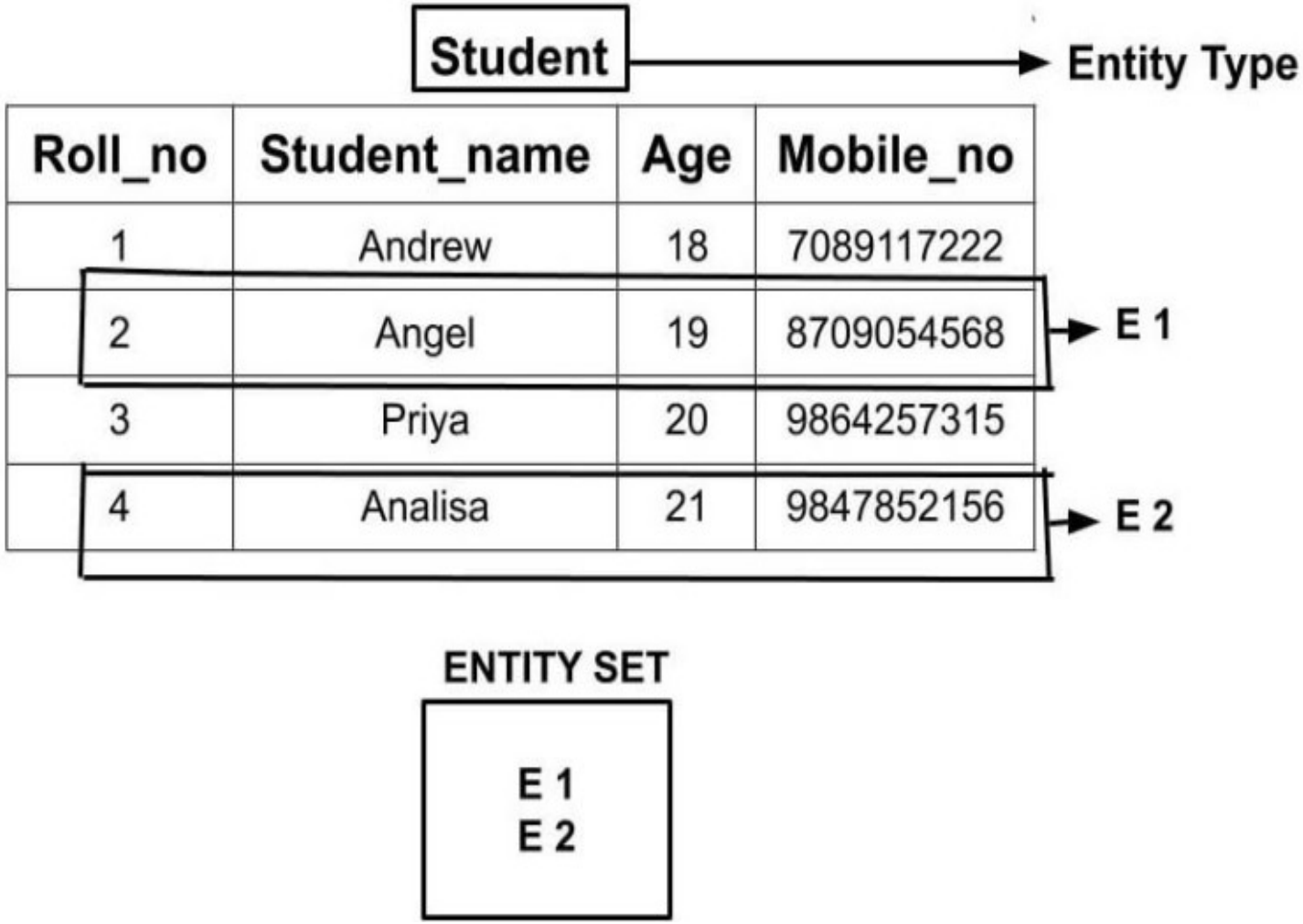
- An entity may be any object, class, person or place. In the ER diagram, an entity can be represented as rectangles.
- Consider an organization as an example- manager, product, employee, department etc. can be taken as an entity.

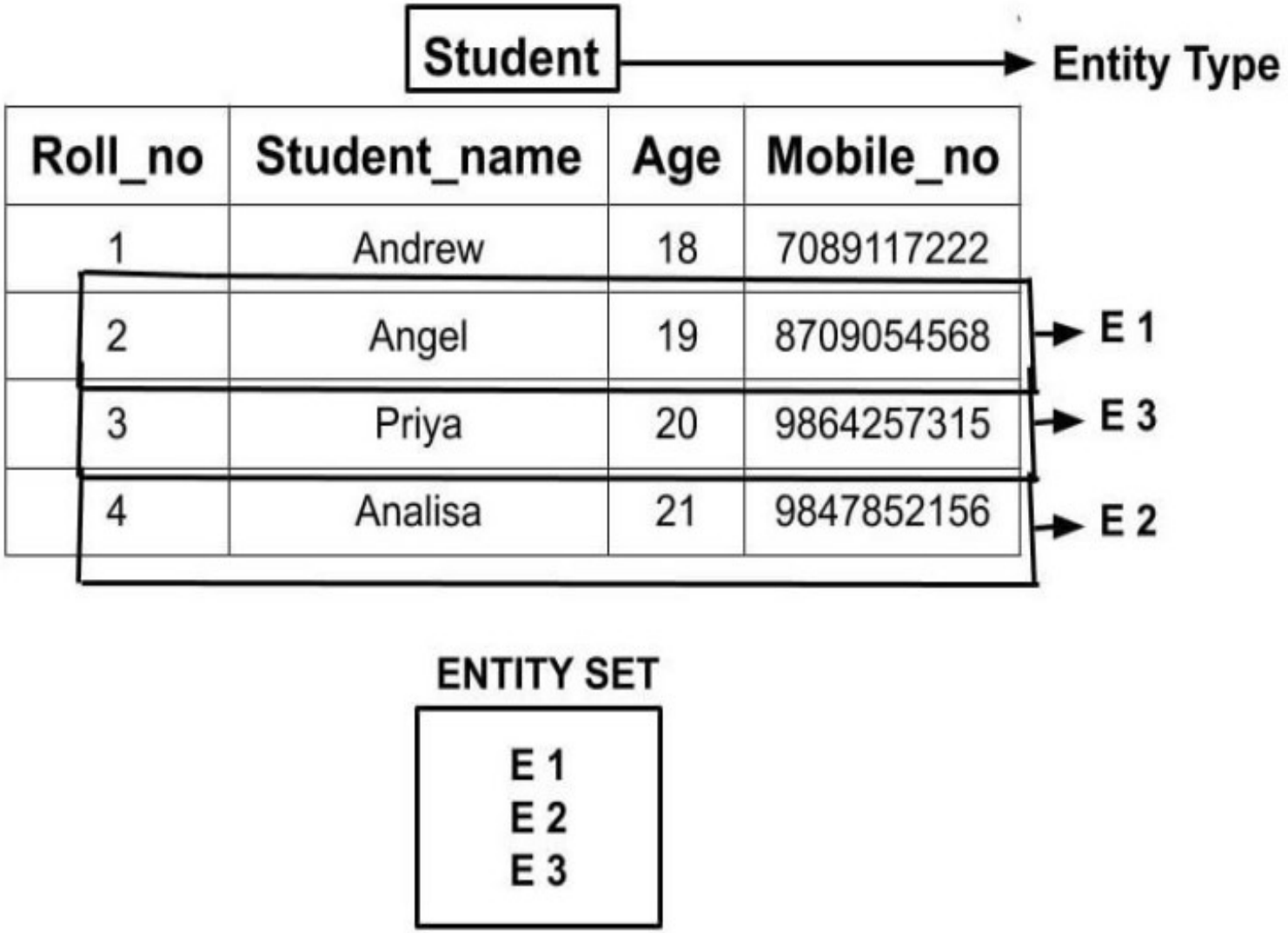


• Entity set

- An entity set is a collection of same type of entities i.e. they share same properties or attributes.
- Consider example of a student. A student has a unique roll no, name, date of birth etc. So an entity set will be set of all those people in a college or school who are students.

Roll No.	Name	Date of Birth
101	Keith	12.12.1992
102	Adrian	14.06.1993
103	Anne	04.09.1993
104	Mathew	07.08.1991


Student
Entity Type
Roll_no
Student_name
Age
Mobile_no
1
Andrew
18
7089117222
2
Angel
19
8709054568
E 1
3
Priya
20
9864257315
4
Analisa
21
9847852156
E 2
ENTITY SET
E 1
E 2


Student
Entity Type
Roll_no
Student_name
Age
Mobile_no
1
Andrew
18
7089117222
2
Angel
19
8709054568
E 1
3
Priya
20
9864257315
E 3
4
Analisa
21
9847852156
E 2
ENTITY SET
E 1
E 2
E 3

Types of Entity:

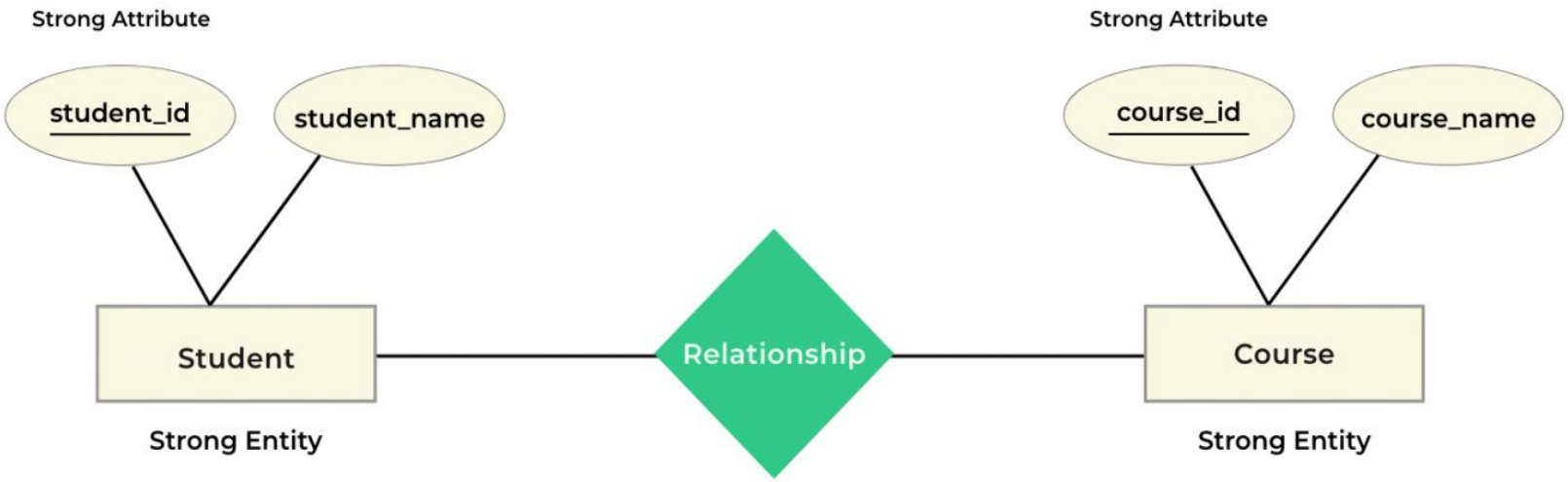
1] strong entity

2] Weak entity

Strong entity

- Strong Entity is independent of any other entity in the schema
- **Example** – A student entity can exist without needing any other entity in the schema or a course entity can exist without needing any other entity in the schema
- A Strong entity is nothing but an entity set having a primary key attribute or a table that consists of a primary key column

Strong Entity



Strong Entities	Student, Course
Strong Attributes	student_id, course_id

Total Participation

Partial Participation

Relationship

Strong Entity

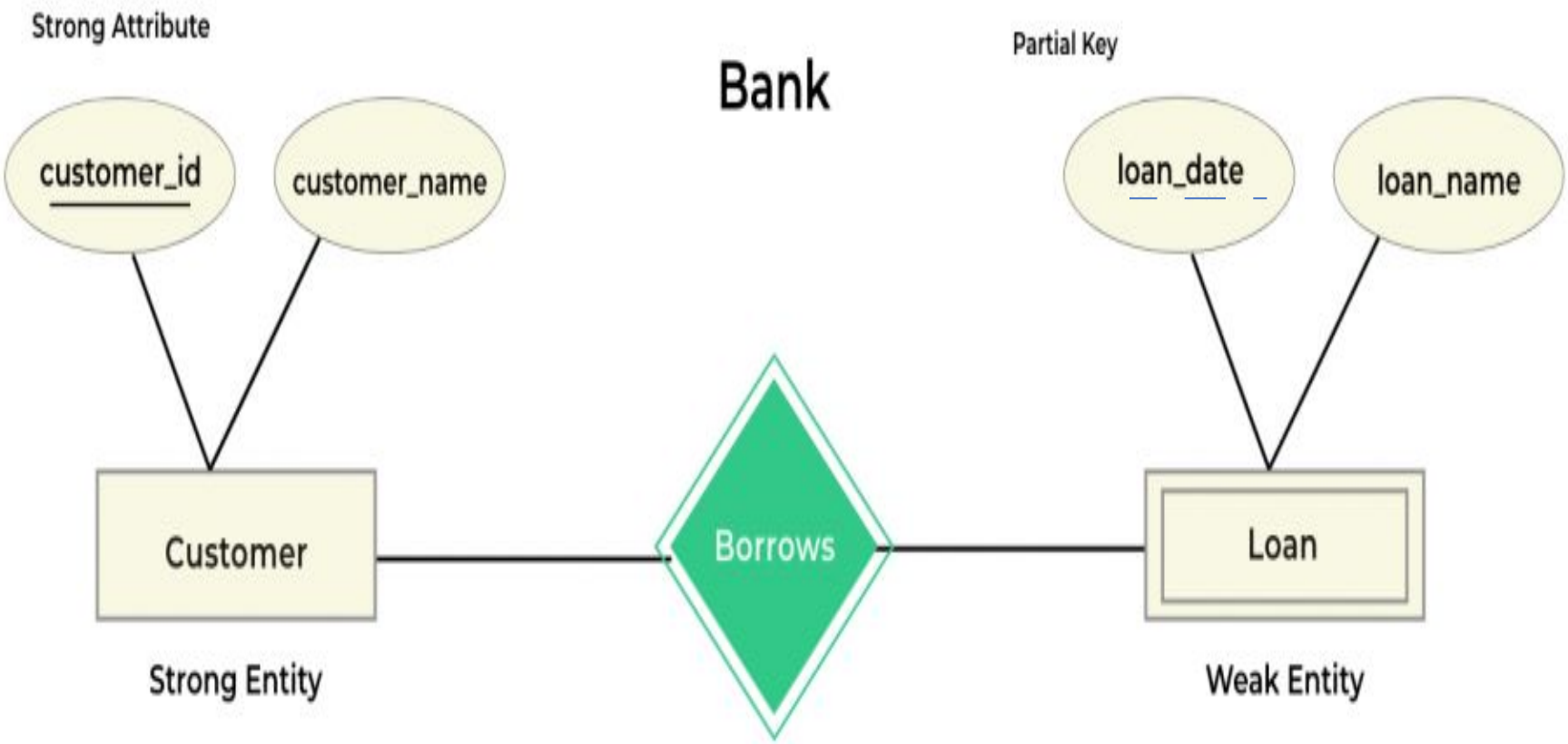
Attribute

- Representation
  - The **strong entity** is represented by a **single rectangle**.
  - **The relationship between two strong entities** is represented by a **single diamond.**

- Consider above the ER diagram which consists of **two entities student and course**
- **Student entity is a strong entity** because it consists of a **primary key** called student id which is enough for accessing each record uniquely
- In the same way, the course entity contains of course ID attribute which is capable of uniquely accessing each row.

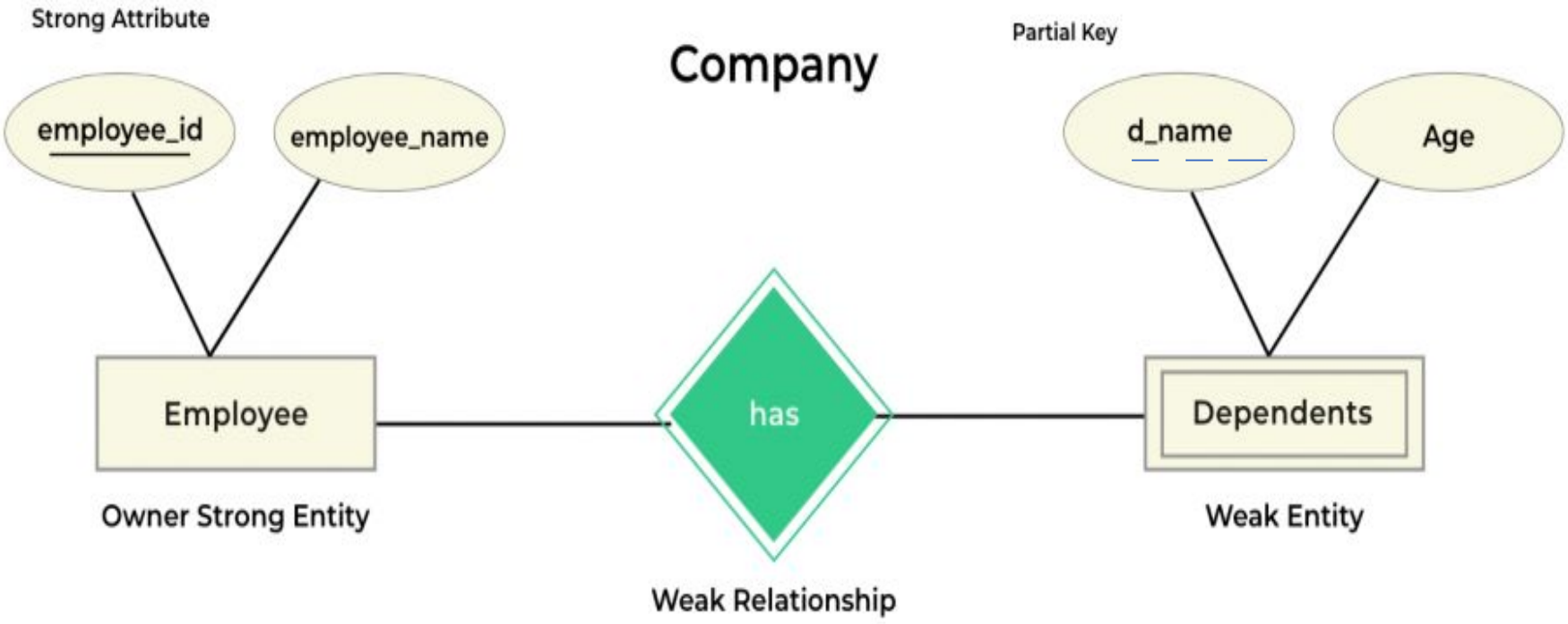
Weak entity

- A weak entity is an entity set that does not have sufficient attributes for Unique Identification of its records.
- An entity that depends on another entity called a weak entity.
- **Example 1 –** A loan entity can not be created for a customer if the customer doesn't exist


Strong Attribute
Bank
Partial Key
customer_id
customer_name
loan_date
loan_name
Customer
Borrows
Loan
Strong Entity
Weak Entity

- a weak entity is nothing but an entity that does not have a primary key attribute
- It **contains** a partial key called a **discriminator which helps in identifying a group of entities from the entity set**
- **A discriminator** is represented by underlining with a **dashed line**

Example 2 – A dependents list entity can not be created if the employee doesn't exist



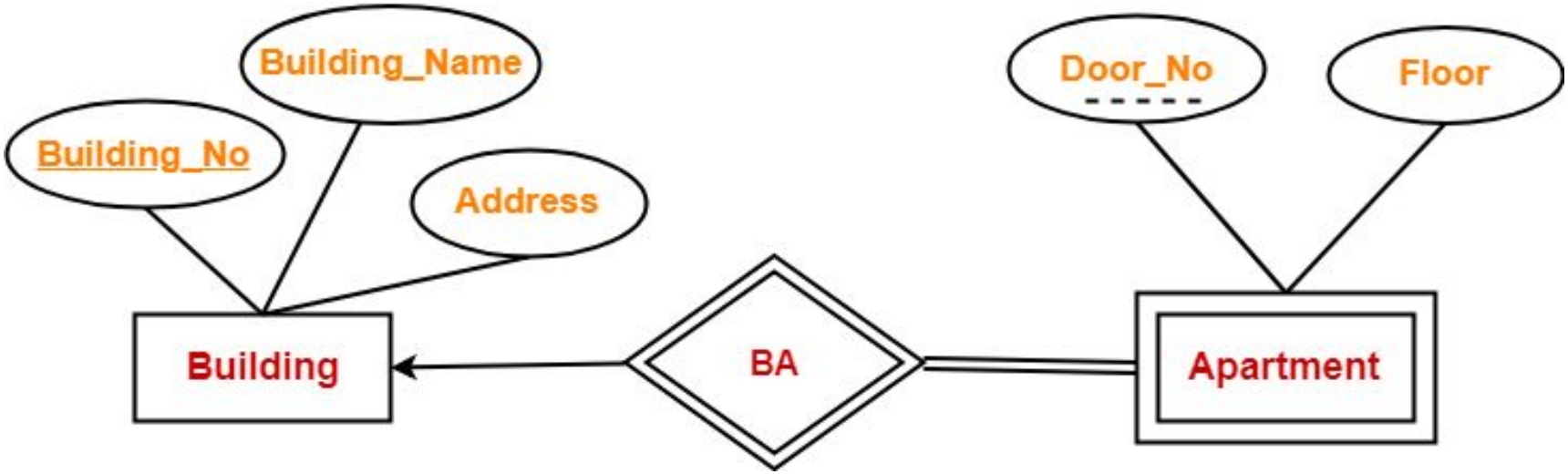
- In the ER diagram, we have **two entities Employee and Dependents.**
- **Employee is a strong entity** because it has a **primary key** attribute called Employee number (Employee_No) which is capable of uniquely identifying all the employee.
- Unlike Employee, **Dependents is weak entity** because it **does not have any primary key .**
- D_Name along with the Employee_No can uniquly identfy the records of Depends. So here the D_Name (Depends Name) is partial key.

- Representation

  - A **double rectangle** is used for representing a **weak entity set**
  - The **double diamond** symbol is used for representing the **relationship between a strong entity and a weak entity** which is known as identifying relationship

Primary key of weak entity set

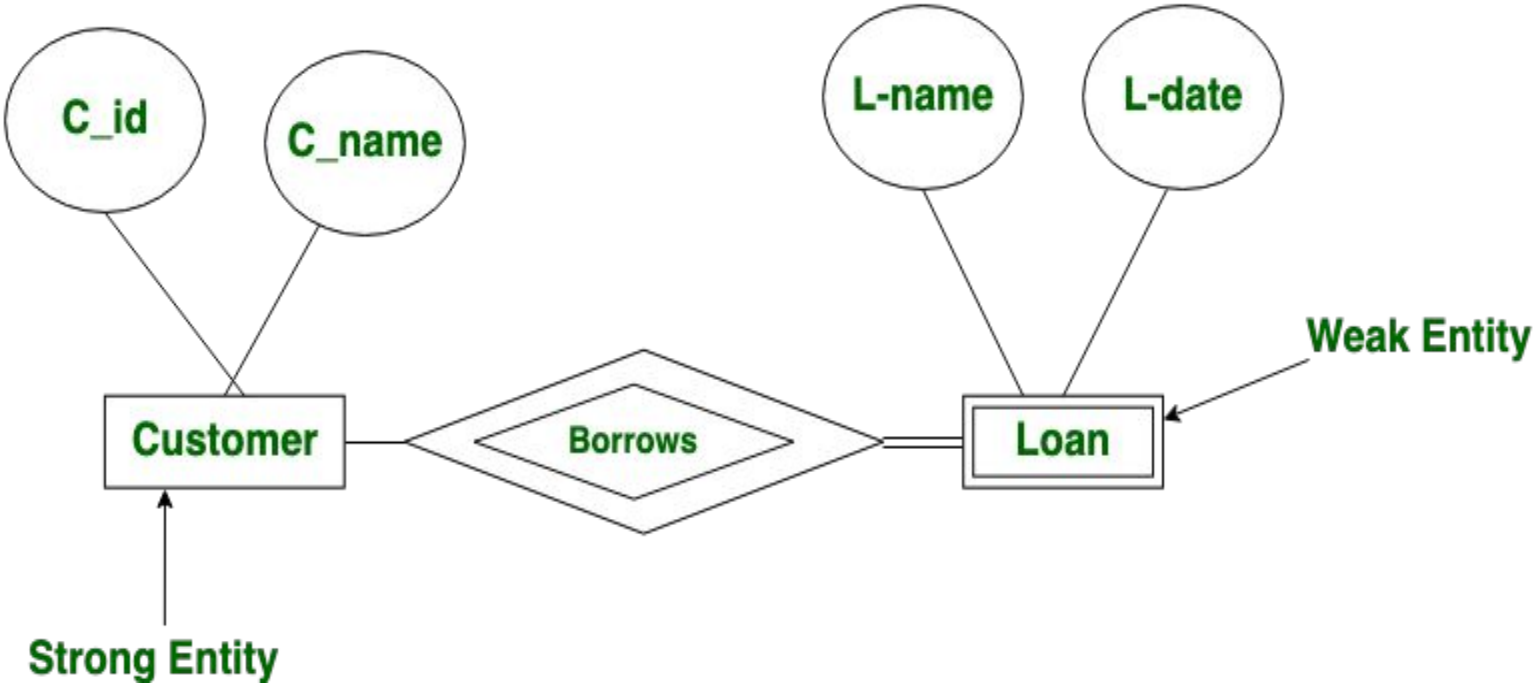
= Its own discriminator + Primary key of strong entity set



- 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.

The relation between one strong and one weak entity is represented by a double diamond. This relationship is also known as **identifying relationship.**



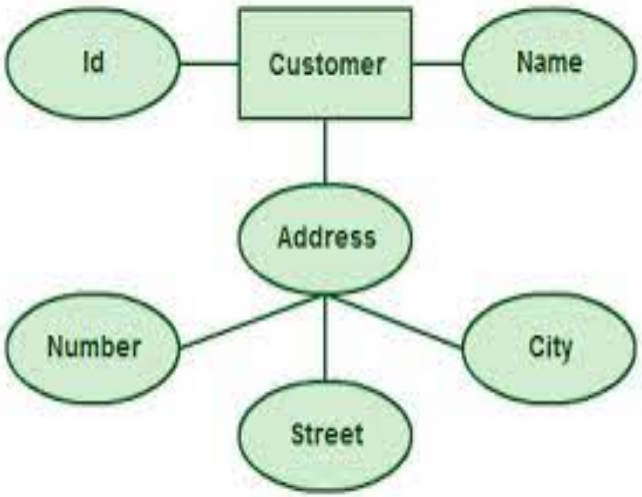
- A weak entity is one that can only exist when owned by another one.
- **Example-1:**
  The existence of rooms is entirely dependent on the existence of a hotel. So room can be seen as the weak entity of the hotel.
- **Example-2:**
  The bank account of a particular bank has no existence if the bank doesn’t exist anymore.

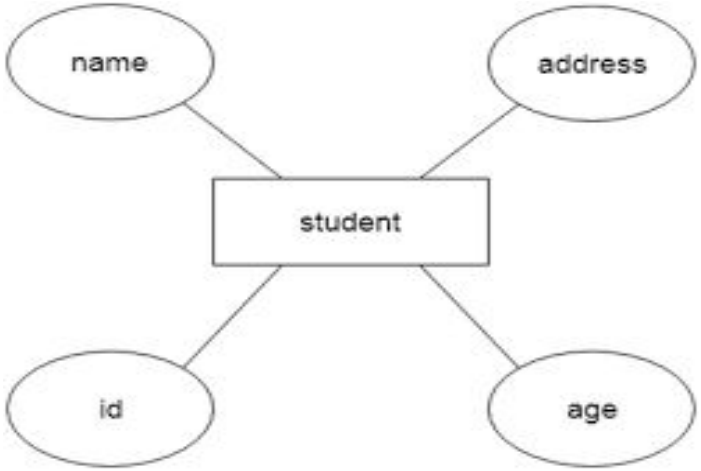
Strong Entity Set	Weak Entity Set
it has its own primary key.	It does not save sufficient attributes to form a primary Key on its own.
It is represented by a rectangle.	It is represented by a double rectangle.
It contains a primary key represented by an underline.	It contains a Partial Key or discriminator represented by a dashed underline.
The member of strong entity set is called as dominant entity set.	The member of weak entity set is called as subordinate entity set.
The Primary Key is one of its attributes which uniquely Identifies its member.	The Primary Key of weak entity set is a combination of partial Key and Primary Key of the strong entity set.
The relationship between two strong entity set is represent by a diamond symbol.	The relationship between one strong and a weak entity set is represented by a double diamond sign. It is known as identifying relationship.
The line connecting strong entity set with the relationship is single	The line connecting weak entity set with the identifying relationship is double.
Total participation in the relationship may or may not exist.	Total participation in the identifying relationship always exists.

2. Attribute

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

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



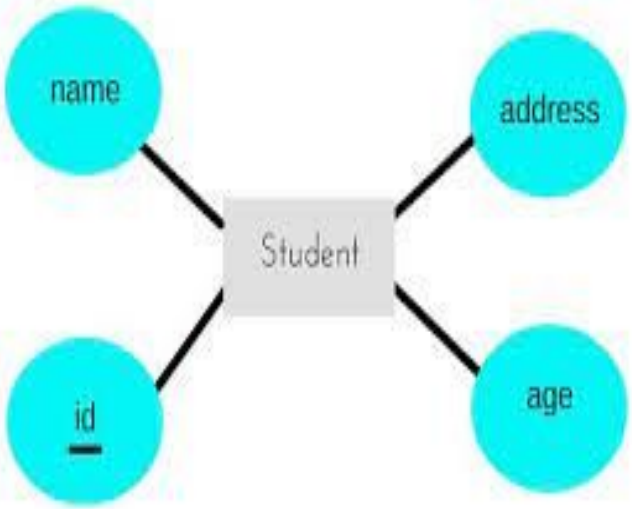


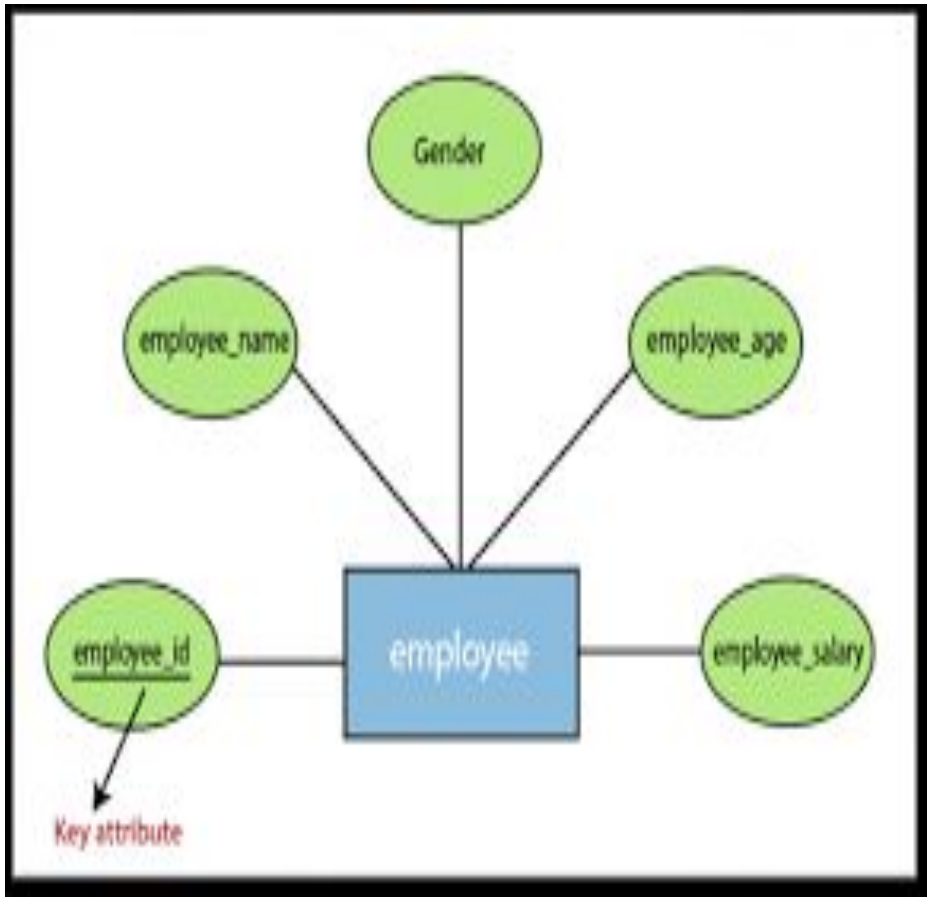
A. Key Attribute

The attribute which **uniquely identifies each entity** in the entity set is called key attribute.

For example, Roll_No or student ID will be unique for each student.

In ER diagram, key attribute is represented by an oval with underlying lines.

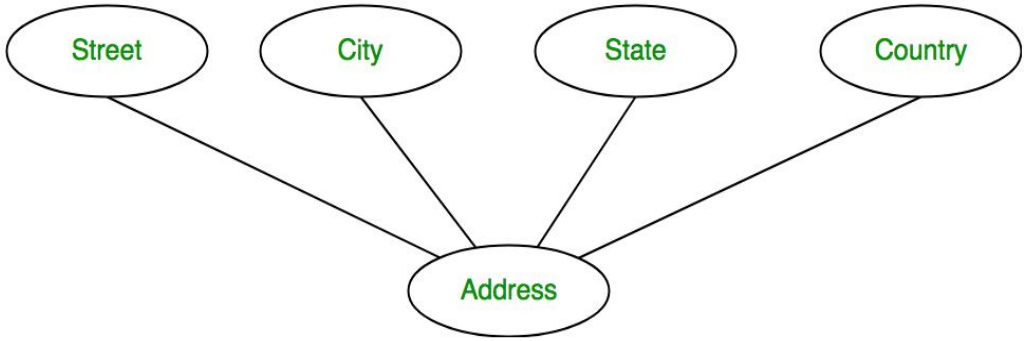

name
address
Student
id
age

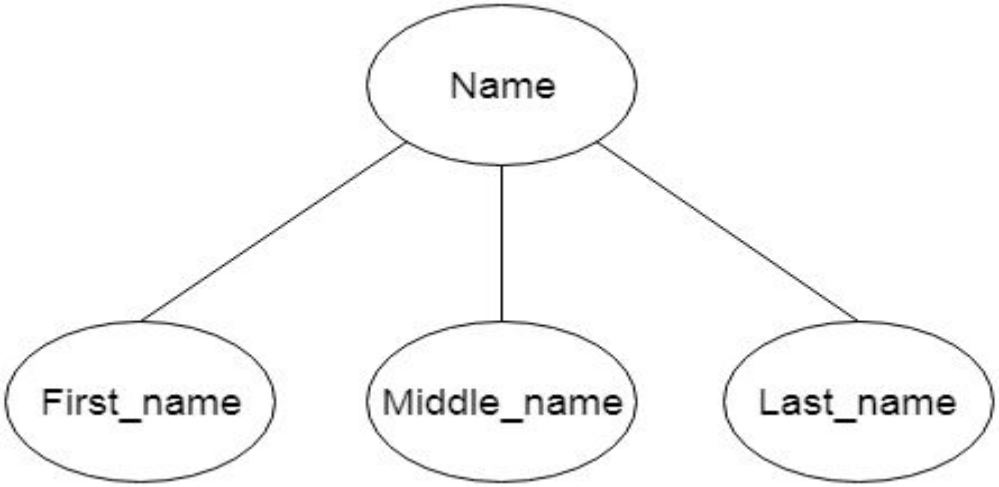

Gender
employee_name
employee_age
employee_id
employee
employee_salary
Key attribute

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.

For example, Address attribute of student Entity type consists of Street, City, State, and Country.

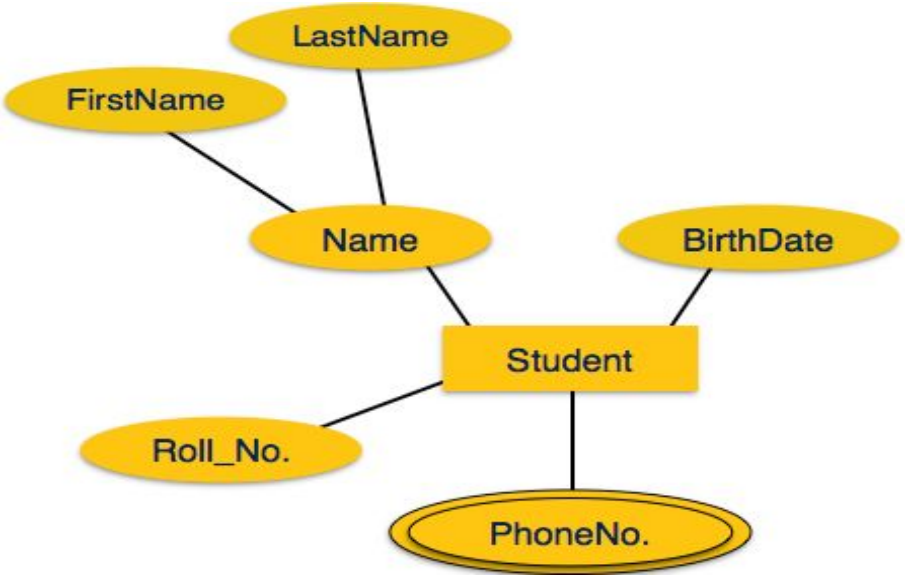


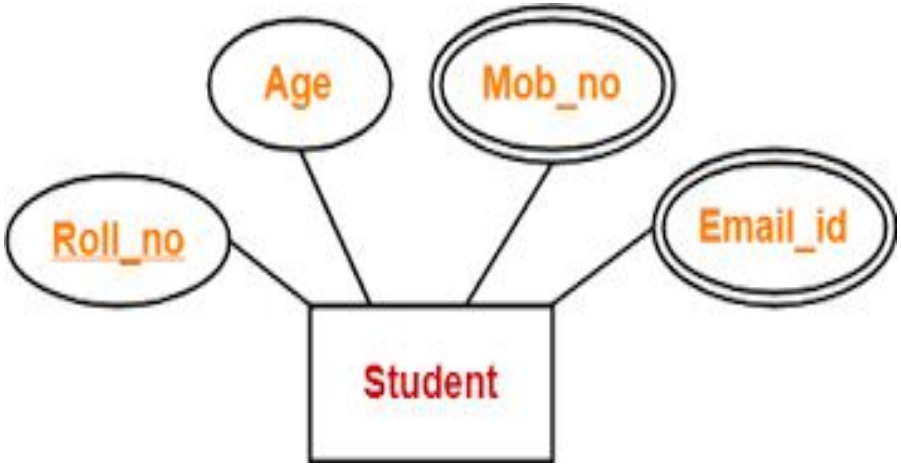

Name
First_name
Middle_name
Last_name

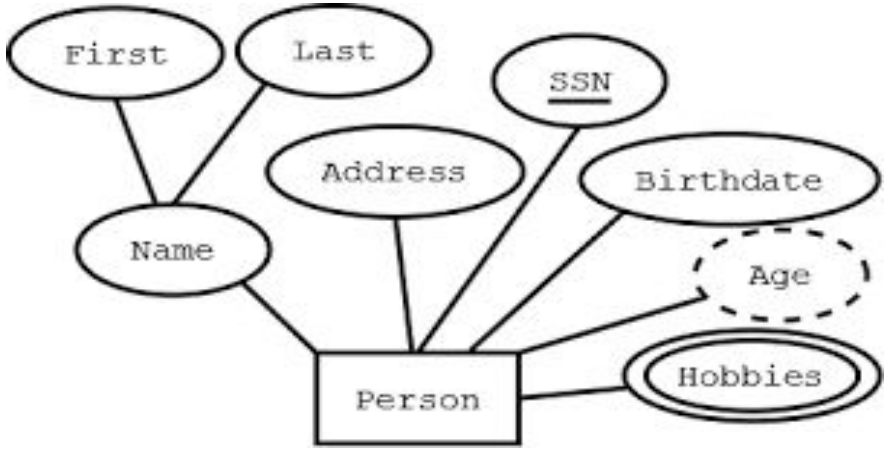
c. Multivalued Attribute

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.



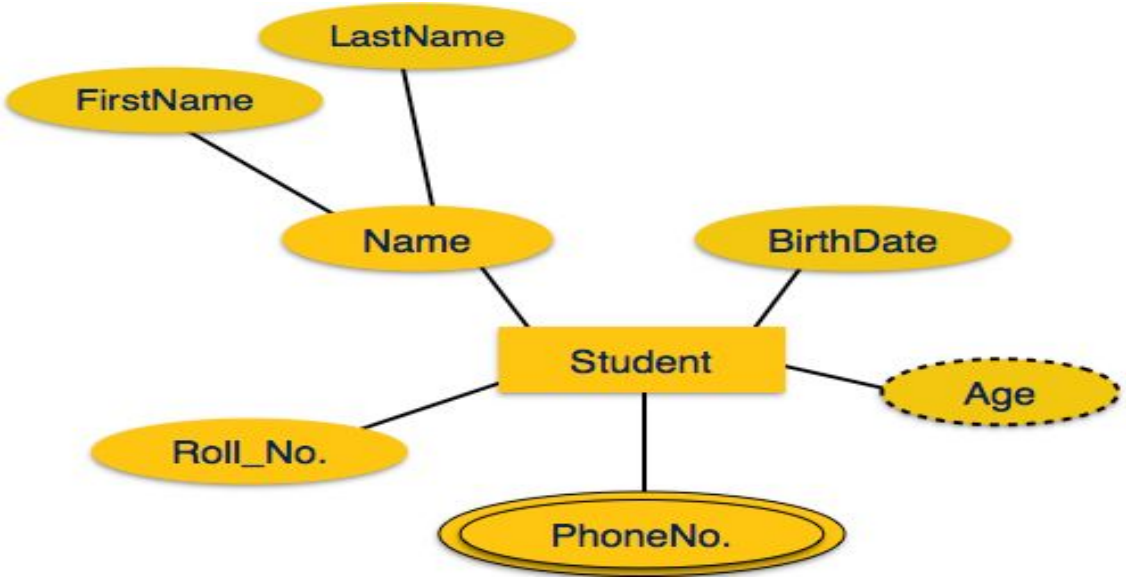

Age
Mob_no
Roll_no
Email_id
Student

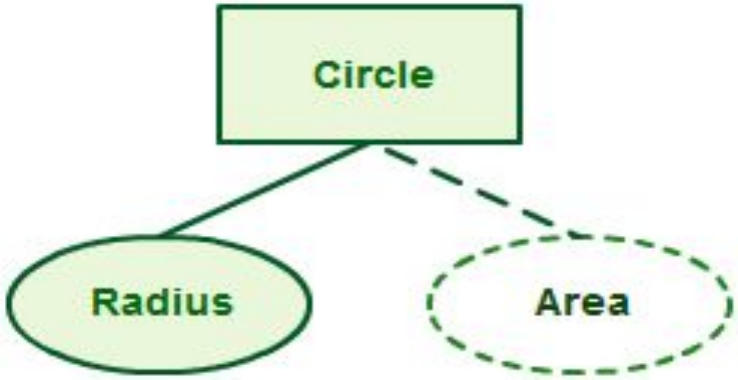

First
Last
SSN
Address
Birthdate
Name
Age
Hobbies
Person

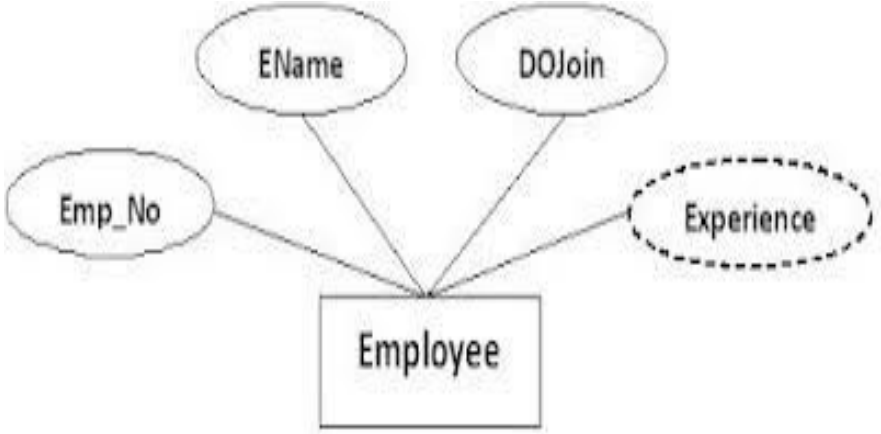
d. Derived Attribute

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.



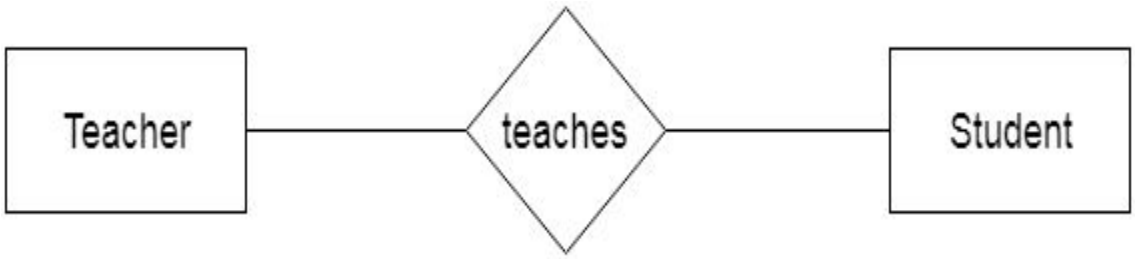

Circle
Radius
Area


EName
DOJoin
Emp_No
Experience
Employee

3. Relationship

A relationship is used to describe the relation between entities.

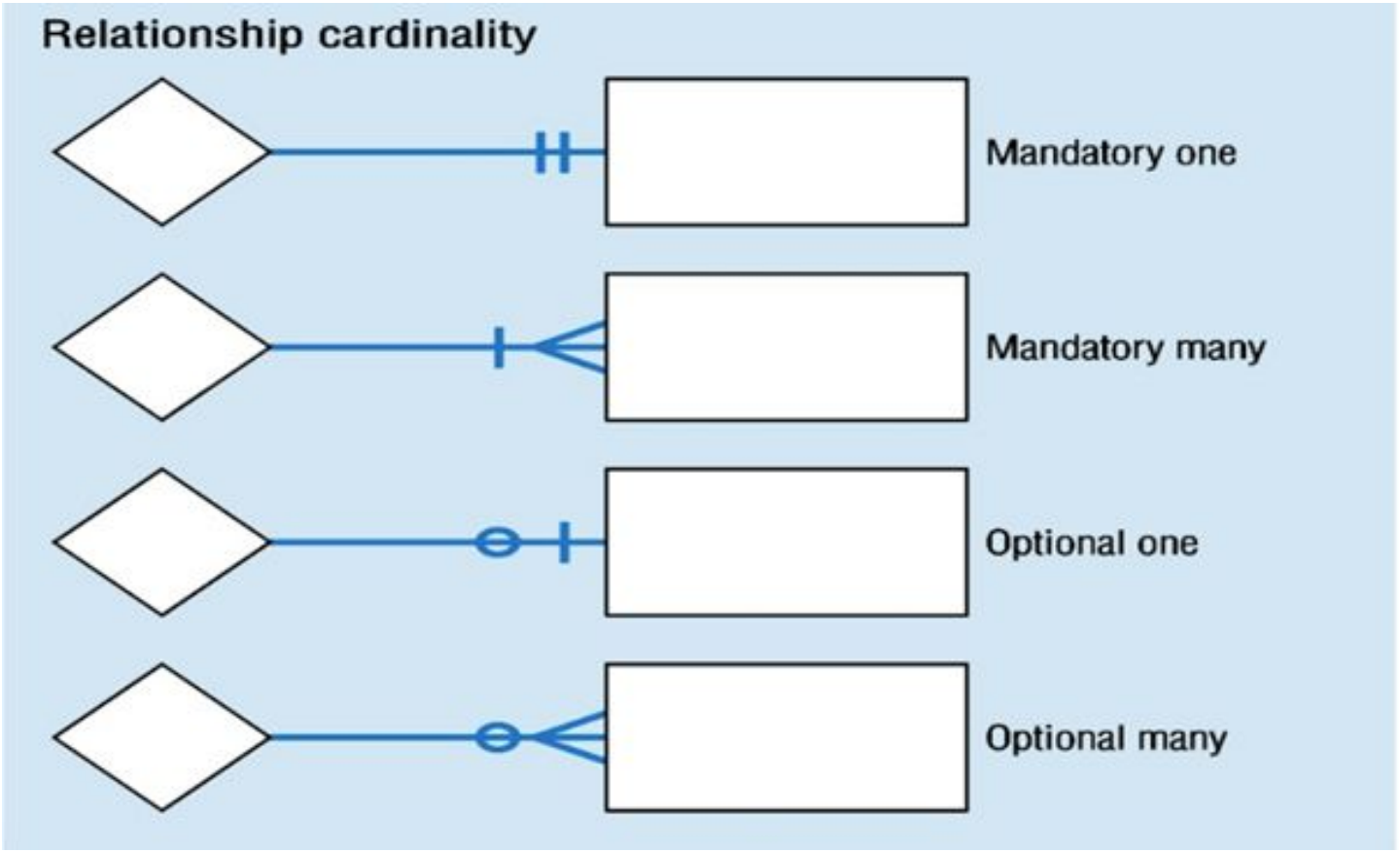
Diamond or rhombus is used to represent the relationship.



Cardinality

- Cardinality defines the number of attributes in one entity set, which can be associated with the number of attributes of another set via a relationship set.
- it refers to the relationship one table can have with the other table.
- Defines the numerical attributes of the relationship between two entities or entity sets.
- Different types of cardinal relationships are:

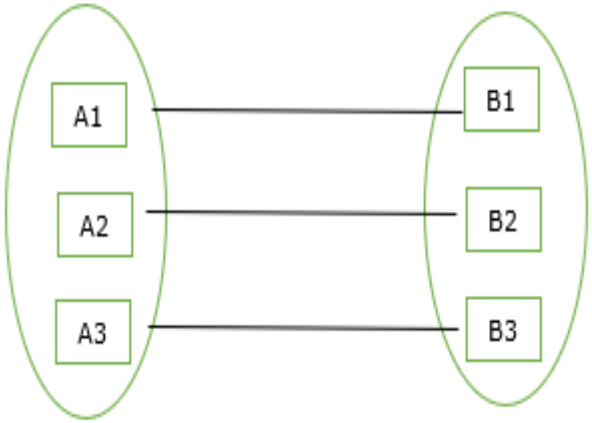
1. One-to-One Relationships
2. One-to-Many Relationships
3. May to One Relationships
4. Many-to-Many Relationships


Relationship cardinality
Mandatory one
Mandatory many
Optional one
Optional many

a. One-to-One Relationship

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.



One entity of A is associated with one entity of B

- **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.

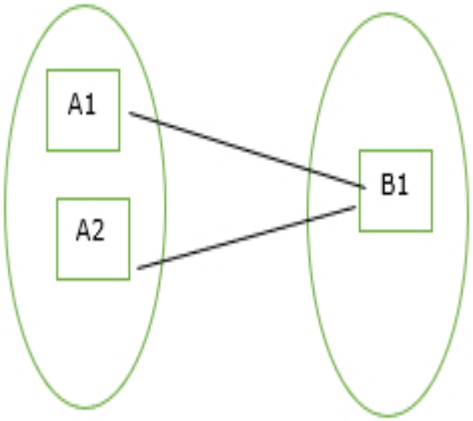
- **Example**

1] Here, one department has one head of the department (HOD).

2] A relationship between person and passport is one to one because a person can have only one passport and a passport can be assigned to only one person.

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.



Many to One Relationship

here

•Many students can enroll in one course.

- **Example**

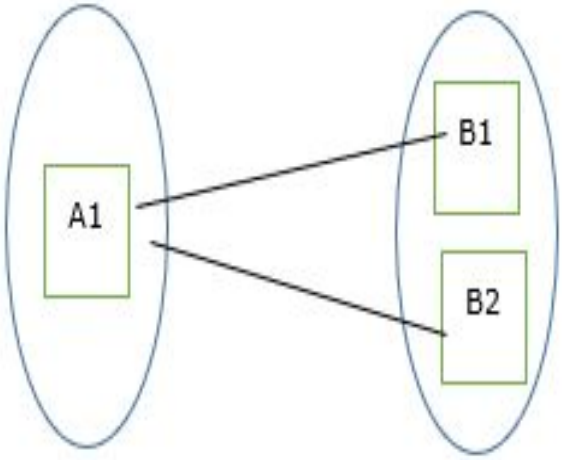
1] Here, many faculties work in one department.

2] Relationship between student and university is many to one because a university can have many students but a student can only study only in single university at a time.

3. One-to-Many Cardinality-

By this cardinality constraint,

- An entity in set A can be associated with any number of entities in set B.
- An entity in set B can be associated with at most one entity in set A.



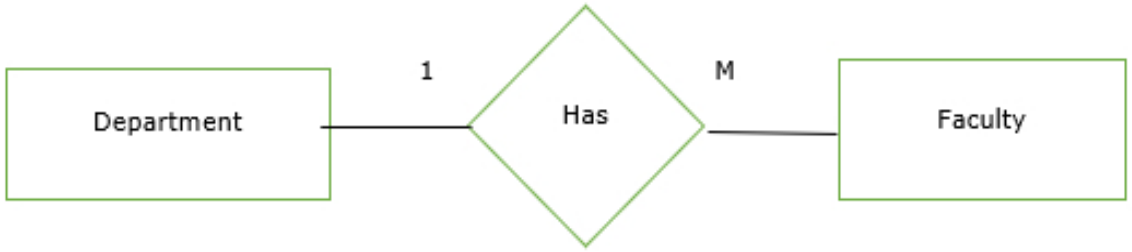
One to Many Relationship

Here,

•One student can enroll in any number of courses.

- **Example**

1] Here, one department has many faculties.

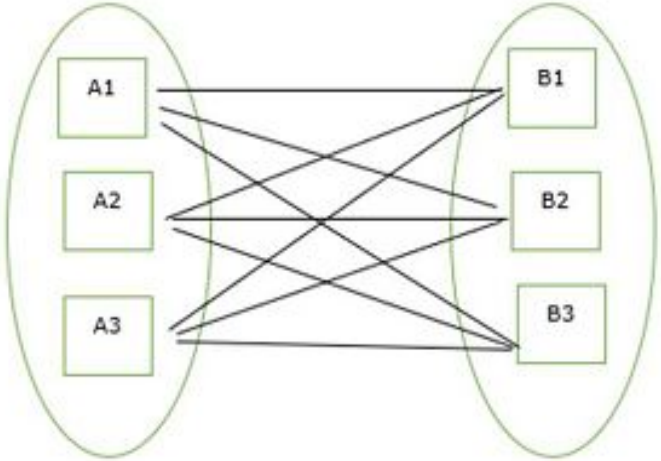


2] Relationship between customer and order is one to many because a customer can place many orders

4. Many-to-Many Cardinality

By this cardinality constraint,

- An entity in set A can be associated with any number of entities in set B.
- An entity in set B can be associated with any number of entities in set A.



Many to Many Relationship

Here,
Any number of student can enroll in any number of courses.

- **Example**

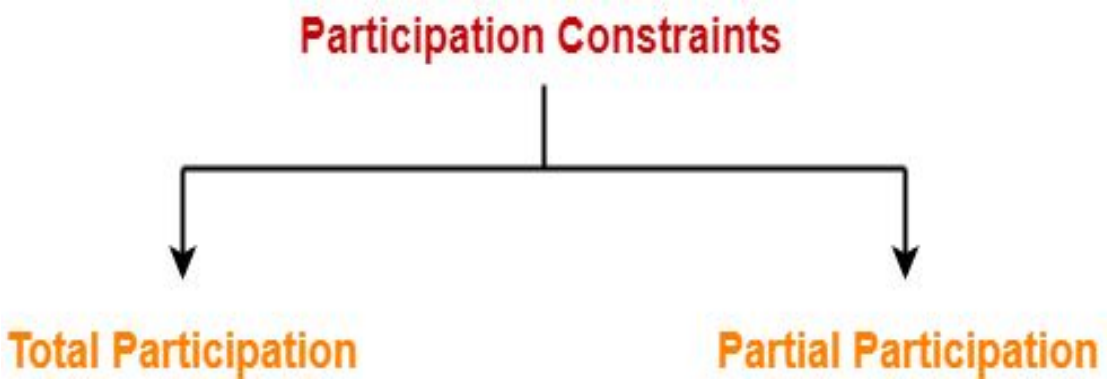
1] Here, many employees work on many projects.

2] A many-to-many relationship exists between customers and products: customers can purchase various products, and products can be purchased by many customers.

Participation

Participation constraints deal with the participation of entities from an entity set in a relationship set

- **Types of Participation Constraints-**



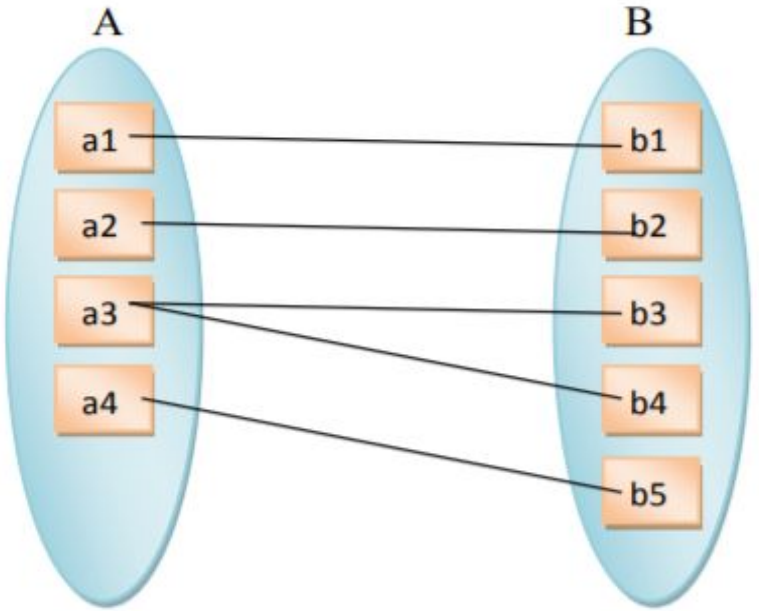
• 1. Total Participation-

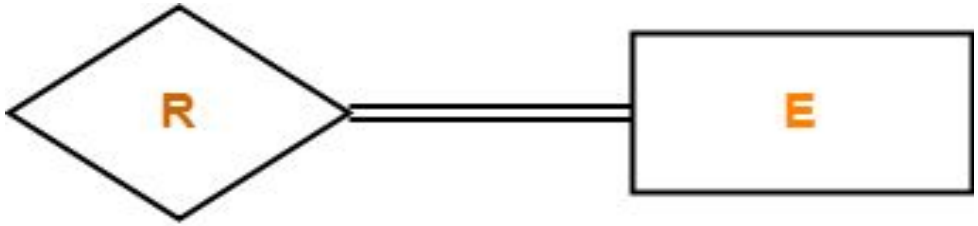
- The Participation of an entity set E in a relationship set R is said to be total if every entity in E participates in at least one relationship in R.

The participation of entity set A in the relationship set is **total** because every entity of A participates in the relationship set.

and

The participation of entity set B in the relationship set is also **total** because every entity of B also participates in the relationship set.





Total Participation

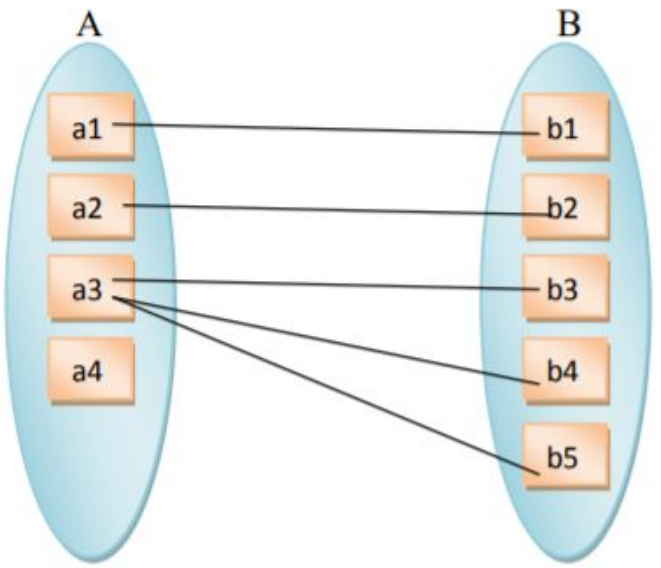
2. Partial Participation-

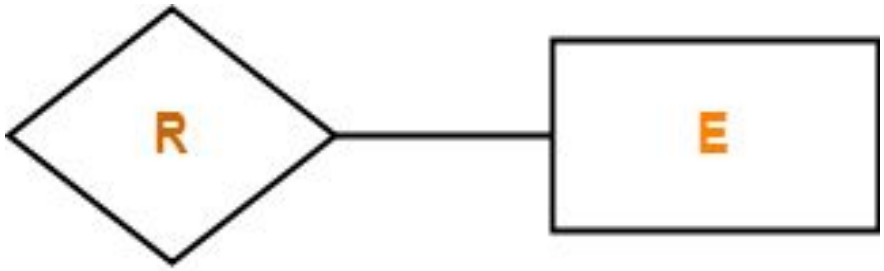
The participation of an entity set E in relationship set R is said to be partial if only some entities in E participate in relationships in R

The participation of entity set A in the relationship set is **partial** because only some entities of A participate in the relationship set.

while

The participation of entity set B in the relationship set is **total** because every entity of B participates in the relationship set.





Partial Participation

- In ERD, the total participation is denoted by doubled-line between entity set and relationship set and partial participation is denoted by single line between entity set and relationship set.

Participation in R: **Partial A**
Total B

- Example:
- Suppose an entity set Student related to an entity set Course through Enrolled relationship set.
- Single line between the entity set “Course” and relationship set “Enrolled in” signifies partial participation.It specifies that there might exist some courses for which no enrollments are made.
- Double line between the entity set “Student” and relationship set “Enrolled in” signifies total participation.It specifies that each student must be enrolled in at least one course.

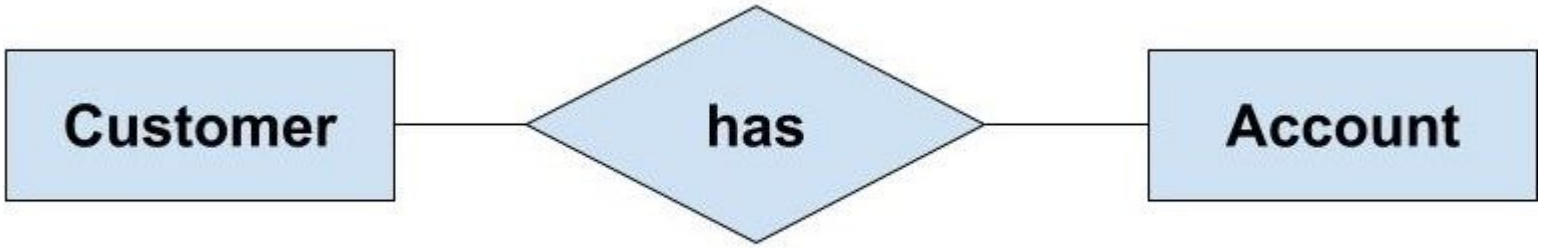
Participation in Enrolled relationship set: **Partial** Course
Total Student

Degree of Relationship

- The degree of a relationship is the number of entity types that participate(associate) in a relationship.
- **The number of an entity type that is connected to a relationship is the degree of that relationship .**

- ***For example,*** If we have two entity type ‘Customer’ and ‘Account’ and they are linked using the primary key and foreign key.

We can say that the degree of relationship is 2 because here two entities are taking part in the relationship.



degree 2

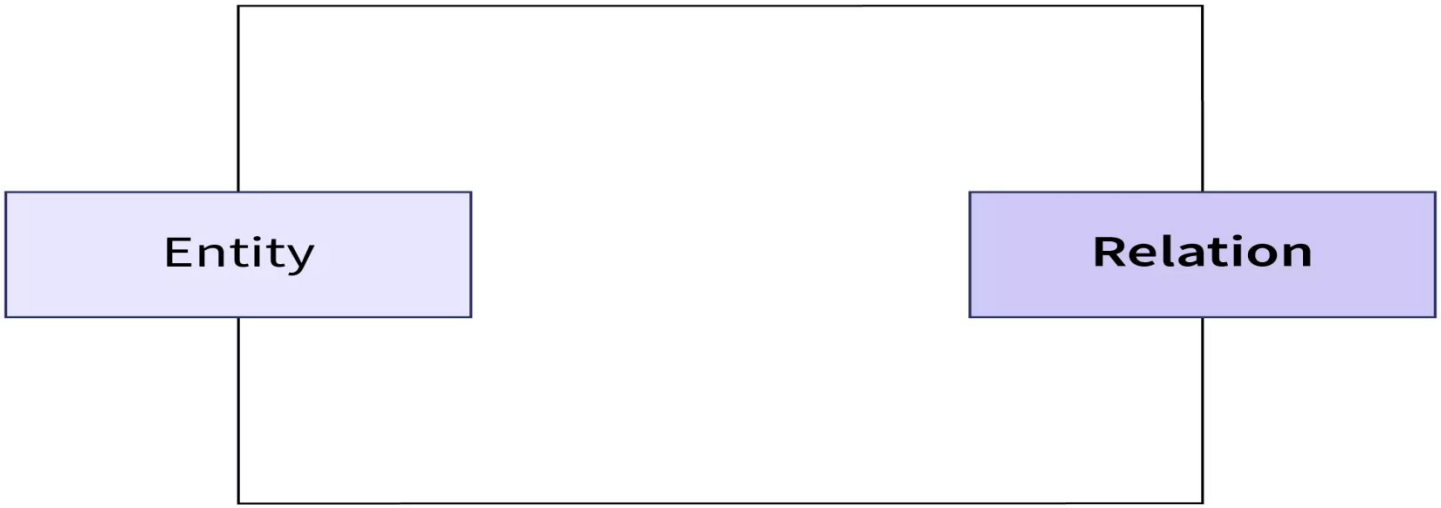
- Based on the number of entity types that are connected we have the following degree of relationships:
- Unary
- Binary
- Ternary
- N-ary

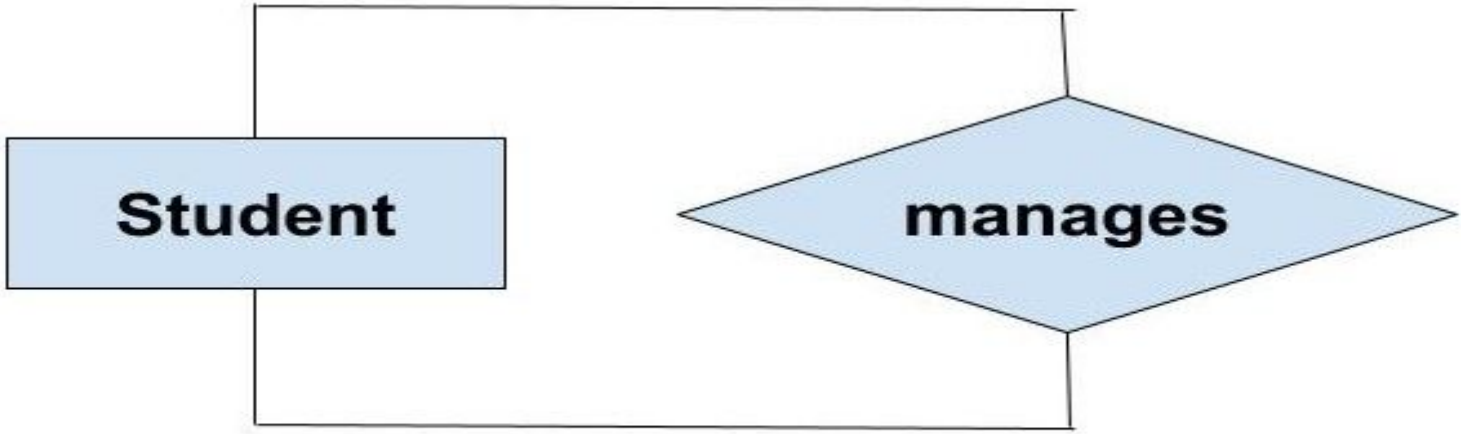
1] Unary (degree 1)

A unary relationship exists when both the participating entity type are the same.

When such a relationship is present we say that the degree of relationship is 1.

Example: In the college presidential elections, a president is chosen among the students. He/she leads the entire community and looks after the student-centric activities. Even though he/she is the president, but after all is a student. So we can say that there is only one entity i.e., the student.



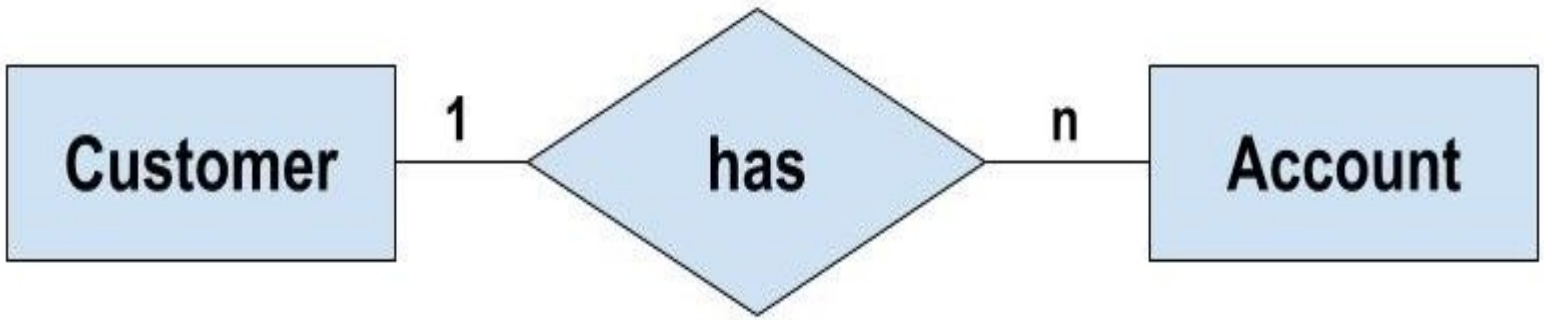


Unary Relationship
(degree 1)

2] Binary (degree 2)

- A binary relationship exists when exactly two entity type participates.
- When such a relationship is present we say that the degree is 2.
- This is the most common degree of relationship.

- **Example-** Each Indian citizen has their own Aadhar Card so we can say that citizen and Aadhar Card are two entities involved.



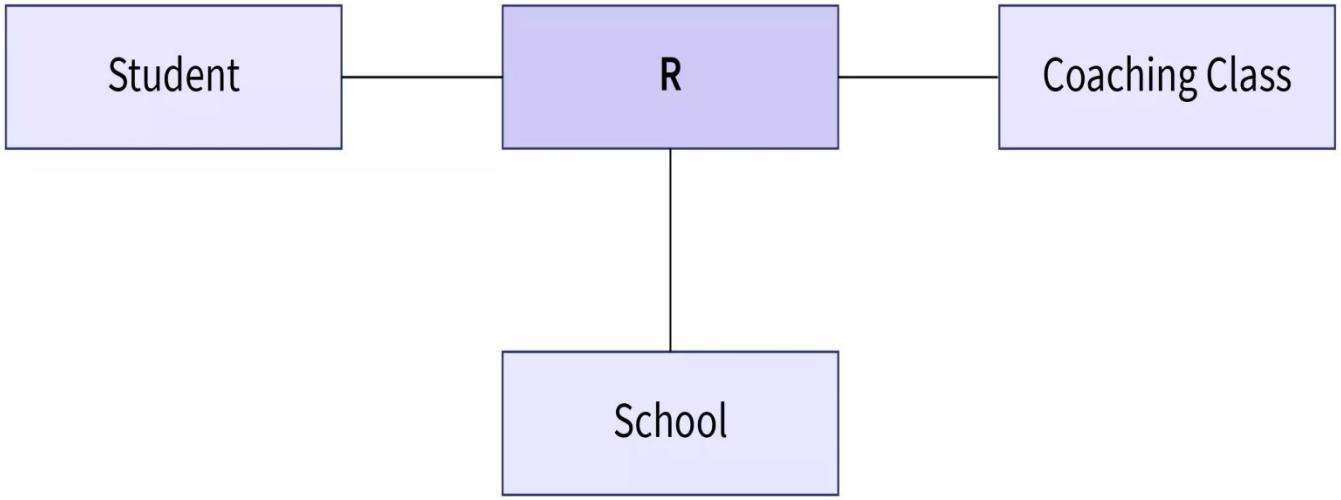
Binary Relationship
(degree 2)

For example, We have two entity type ‘Customer’ and ‘Account’ where each ‘Customer’ has an ‘Account’ which stores the account details of the ‘Customer’. Since we have two entity types participating we call it a binary relationship. Also, one ‘Customer’ can have many ‘Account’ but each ‘Account’ should belong to only one ‘Customer’. We can say that it is a one-to-many binary relationship

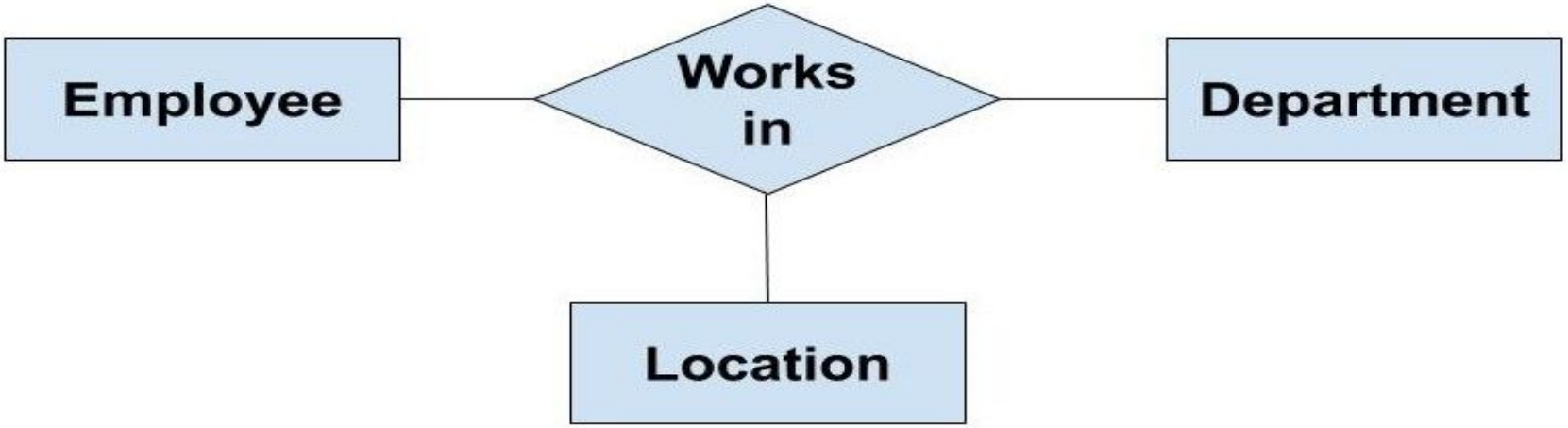
3]Ternary(degree 3)

- A ternary relationship exists when exactly three entity type participates.
- When such a relationship is present we say that the degree is 3.

- **Example-** A student studies in a school, so student and school are two entities. But, the same student also takes some coaching classes, so he has a relation with coaching as well, so coaching class is also an entity. Here as there are 3 entities, so it is a ternary relationship.



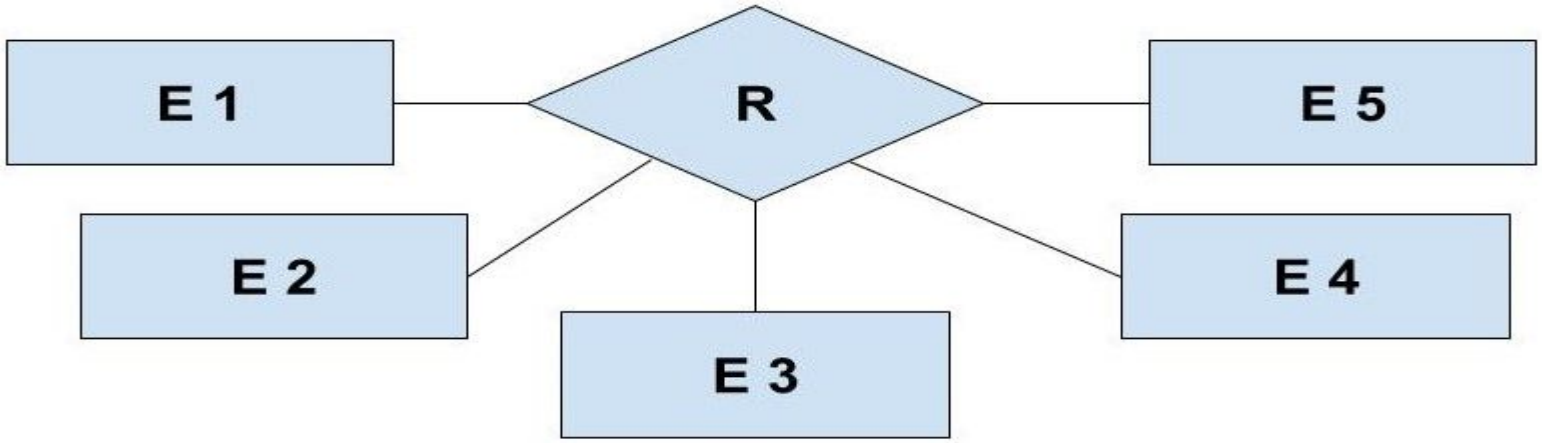
- ***For example,*** We have three entity type ‘Employee’, ‘Department’ and ‘Location’. The relationship between these entities are defined as an employee works in a department, an employee works at a particular location. So, we can see we have three entities participating in a relationship so it is a ternary relationship. The degree of this relation is 3.



Ternary Relationship
(degree 3)

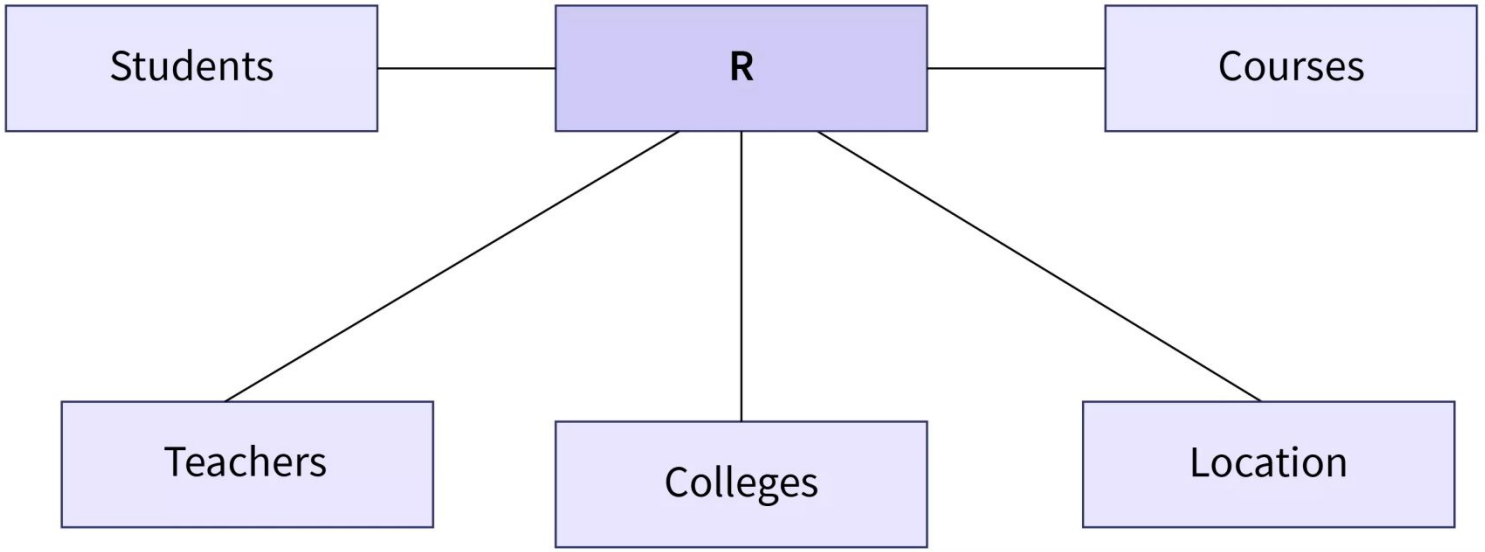
4] N-ary (n degree)

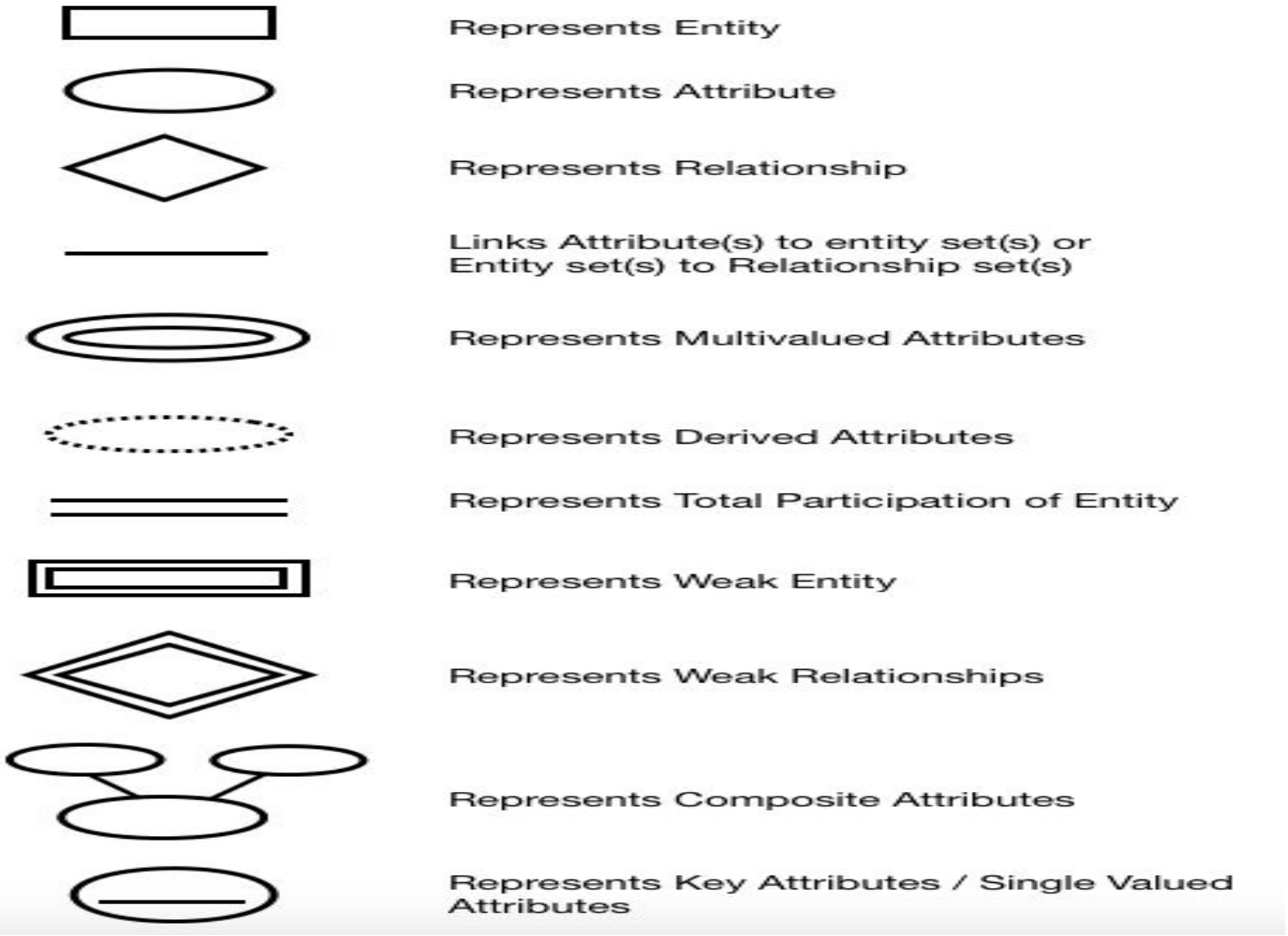
An N-ary relationship exists when ‘n’ number of entities are participating. So, any number of entities can participate in a relationship. There is no limitation to the maximum number of entities that can participate.



N-ary Relationship

- **Example-** Let us consider the example of a university. It has many entities like students, teachers, affiliated colleges, courses, etc. Here, there are many entities associated with the university.





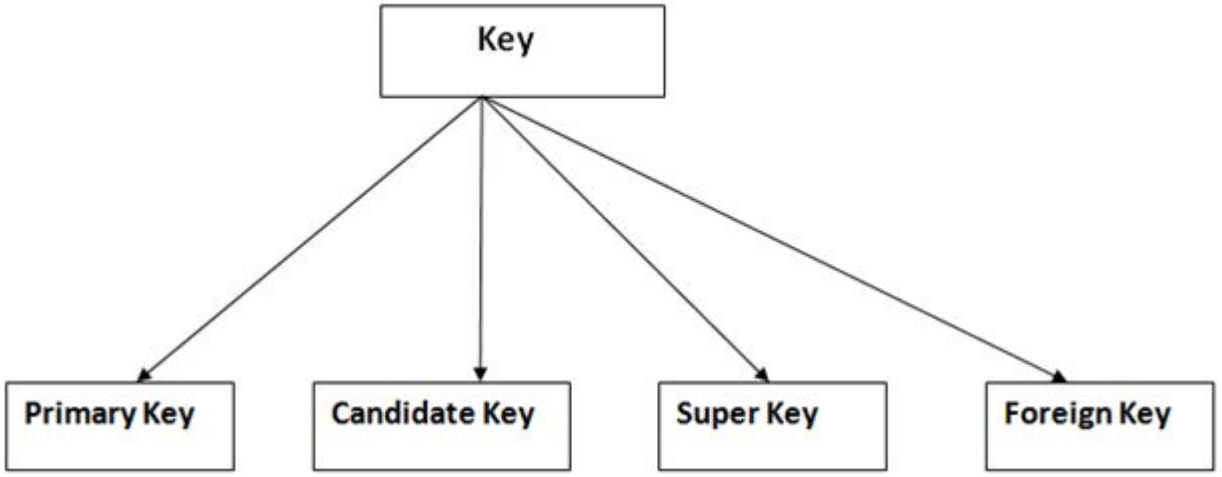
Name of student	marks	course
s	10	CS
d	50	IT
d	35	IT
r	50	IT

Keys

- It is used to uniquely identify any record or row of data from the table.
- It is also used to establish and identify relationships between tables.
- For example: In Student table, ID is used as a key because it is unique for each student. In PERSON table, passport_number, license_number are keys since they are unique for each person.

Emp Id	Emp Name	Mobile No.
E101	Ally	9848785252
E102	Ben	9695943654
E103	Cathy	8170502364

Types of key:



1. Super Key

Super key is a set of an attribute which can uniquely identify a tuple. Super key is a superset of a candidate key.
Super Key can contain multiple attributes that might not be able to independently identify tuples in a table, but when grouped with certain keys, they can identify tuples uniquely.

For example: In the 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.

consider an **EMPLOYEE_DETAIL** table example where we have the following attribute:
Emp_SSN: The SSN number is stored in this field.
Emp_Id: An attribute that stores the value of the employee identification number.
Emp_name: An attribute that stores the name of the employee holding the specified employee id.
Emp_email: An attribute that stores the email id of the specified employees.

Emp_SSN	Emp_Id	Emp_name	Emp_email
11051	01	John	john@email.com
19801	02	Merry	merry@email.com
19801	03	Riddle	riddle@email.com
41201	04	Cary	cary@email.com

Set of super keys obtained

{ Emp_SSN }
{ Emp_Id }
{ Emp_email }
{ Emp_SSN, Emp_Id }
{ Emp_Id, Emp_name }
{ Emp_SSN, Emp_Id, Emp_email }
{ Emp_SSN, Emp_name, Emp_Id }

Consider the following Student schema-

- **Student (roll , name , address)**
- examples of super keys for Student table-

1. (roll , name , address)
2. (roll , Name)
3. (roll , address)
4. (Roll)

- **NOTE-**
- All the attributes in a super key are definitely sufficient to identify each tuple uniquely in the given relation but all of them may not be necessary.

A candidate key is a super key but vice versa is not true

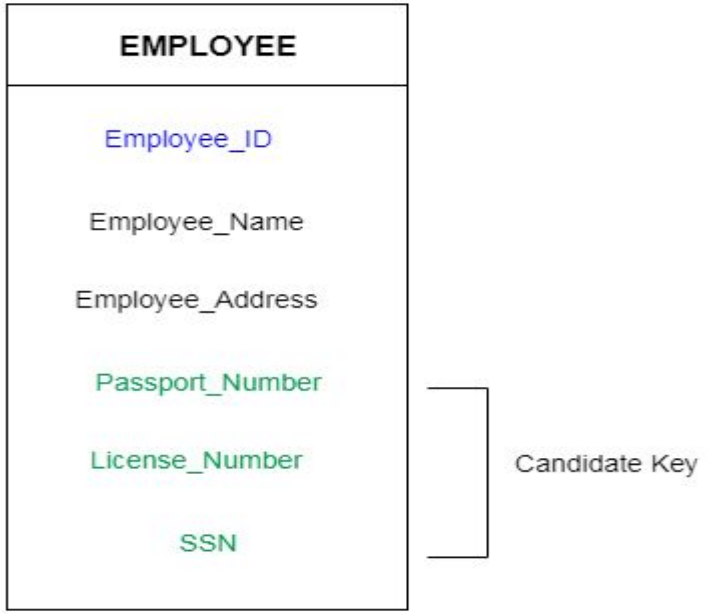
2. Candidate key

A candidate key is an attribute or set of an attribute which can uniquely identify a tuple.

The remaining attributes except for primary key are considered as 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. Rest of the attributes like SSN, Passport_Number, and License_Number, etc. are considered as a candidate key.

A table can have multiple candidate keys but only a single primary key.



- The candidate key can be simple (having only one attribute) or composite as well.

For Example, {STUD_NO, COURSE_NO} is a composite candidate key for relation STUDENT_COURSE.

- Properties of Candidate key:

1. A candiate key can never be NULL or empty. And its value should be unique.
2. There can be more than one candidate keys for a table.
3. A candidate key can be a combination of more than one columns(attributes).

SID	SNAME	MAILID
1	hyyu	ui
2	uiui	iiiu
3	juyiyi	nhhyj

Emp_SSN	Emp_Id	Emp_name	Emp_email
11051	01	John	john@email.com
19801	02	Merry	merry@email.com
19801	03	Riddle	riddle@email.com
41201	04	Cary	cary@email.com

Candidate Keys :

Emp_SSN

Emp_Id

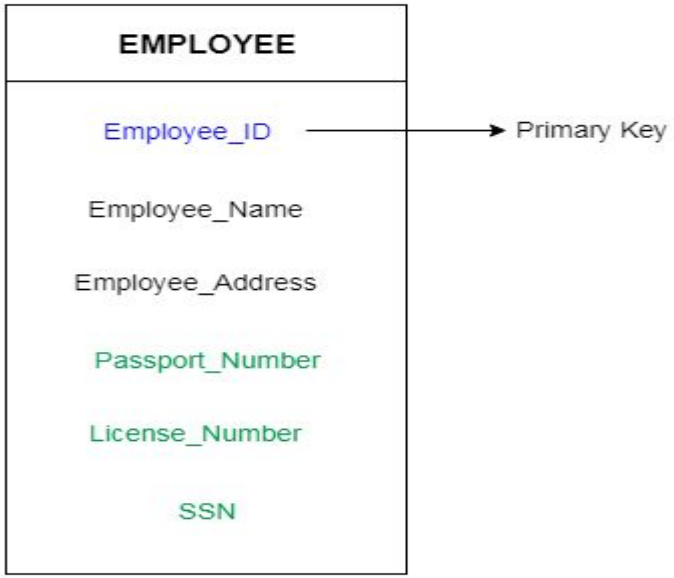
Emp_email

3. Primary key

It is the first key which is used to identify one and only one instance of an entity uniquely.

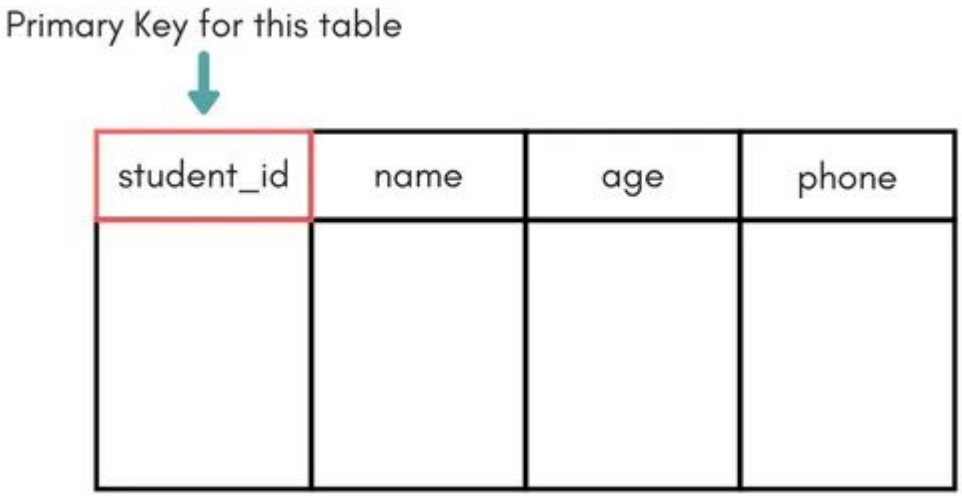
A table cannot have more than one primary key.

In the EMPLOYEE table, ID can be primary key since it is unique for each employee. In the EMPLOYEE table, we can even select License_Number and Passport_Number as primary key since they are also unique.



- Rules for defining Primary key:

1. Two rows can’t have the same primary key value
2. The primary key field cannot be null.



For the table Student we can make the student_id column as the primary key

4. Foreign key

- Foreign keys are the column of the table which is used to point to the primary key of another table.
- A foreign key is the one that is used to link two tables together via the primary key. It means the columns of one table points to the primary key attribute of the other table.

EID	ENAme	aDD	
1	A	gtyt	
2	V	hgtyht	
3	D	yuyu	

DEpt	DNAME	Extension number
101	mca	
102	cs	

Foreign Keys

Employee Table (Referencing relation)

Emp_Id	Name	Aadhar_No	Email_Id	Dept_Id
01	Aman	775762540011	aa@gmail.com	1
02	Neha	876834788522	nn@gmail.com	2
03	Neha	996677898677	ss@gmail.com	2
04	Vimal	796454638800	vv@gmail.com	3

Foreign Key:

In Employee Table

1. Dept_Id

Primary Key

Department Table (Referenced relation)

Dept_Id	Dept_Name
1	Sales
2	Marketing

- Consider two tables **Student** and **Department** having their respective attributes as shown in the below table structure:

Student

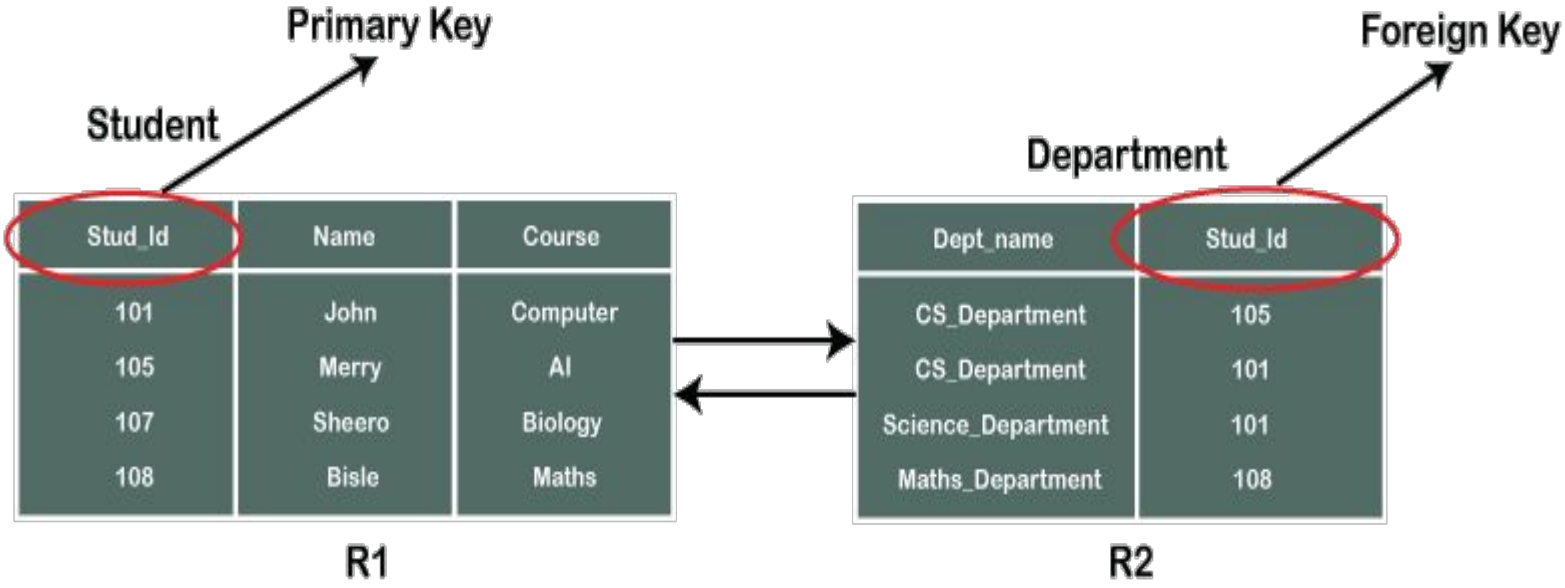
Stud_Id	Name	Course
101	John	Computer
105	Merry	AI
107	Sheero	Biology
108	Bisle	Maths

Department

Dept_name	Stud_Id
CS_Department	105
CS_Department	101
Science_Department	101
Math_Department	108

In the Student table, the field Stud_Id is a primary key because it is uniquely identifying all other fields of the Student table.
On the other hand, Stud_Id is a foreign key attribute for the Department table because it is acting as a primary key attribute for the Student table. It means that both the Student and Department table are linked with one another because of the Stud_Id attribute.

- structure of the relationship between the two tables.



Extended Entity-Relationship (EE-R) Model

EER is a high-level data model that incorporates the extensions to the original ER model.

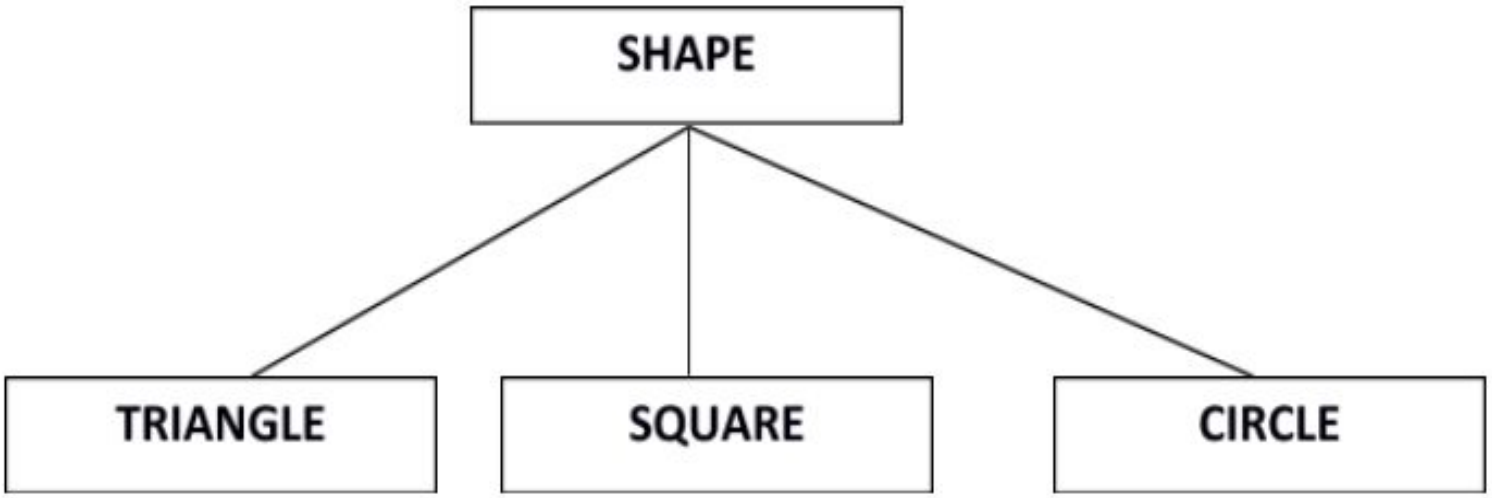
In addition to ER model concepts EE-R includes −

- Subclasses and Super classes.
- Specialization and Generalization.
- Aggregation.

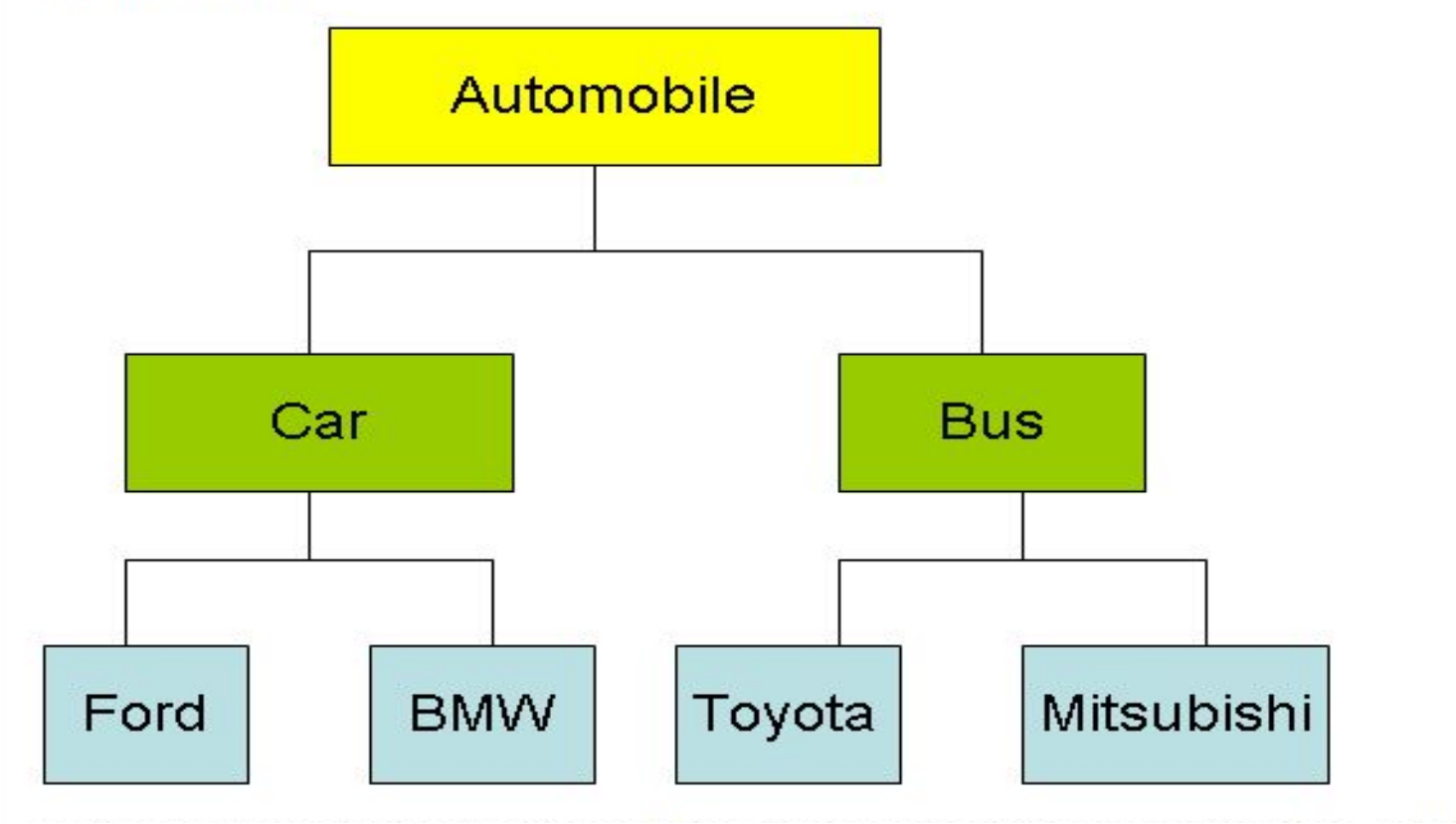
These concepts are used to create EE-R diagrams.

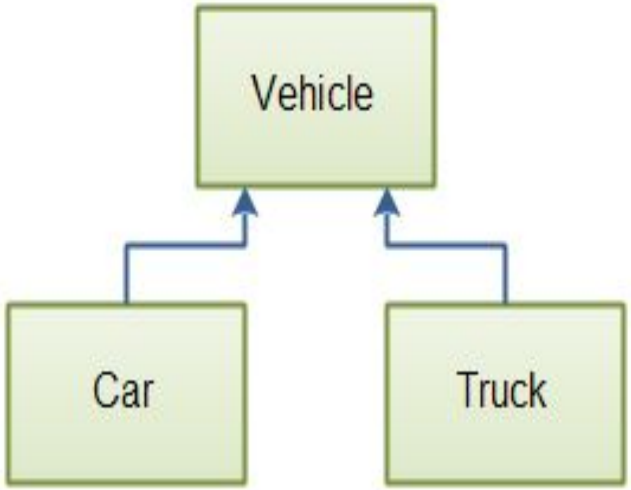
Subclasses and Super class

- Super class is an entity that can be divided into further subtype.
- A superclass is the class from which many subclasses can be created.
- The subclasses inherit the characteristics of a superclass.
- The superclass is also known as the parent class or base class.
- For **example** − consider Shape super class.



- Super class shape has sub groups: Triangle, Square and Circle.
- Sub classes are the group of entities with some unique attributes.
- Sub class inherits the properties and attributes from super class.

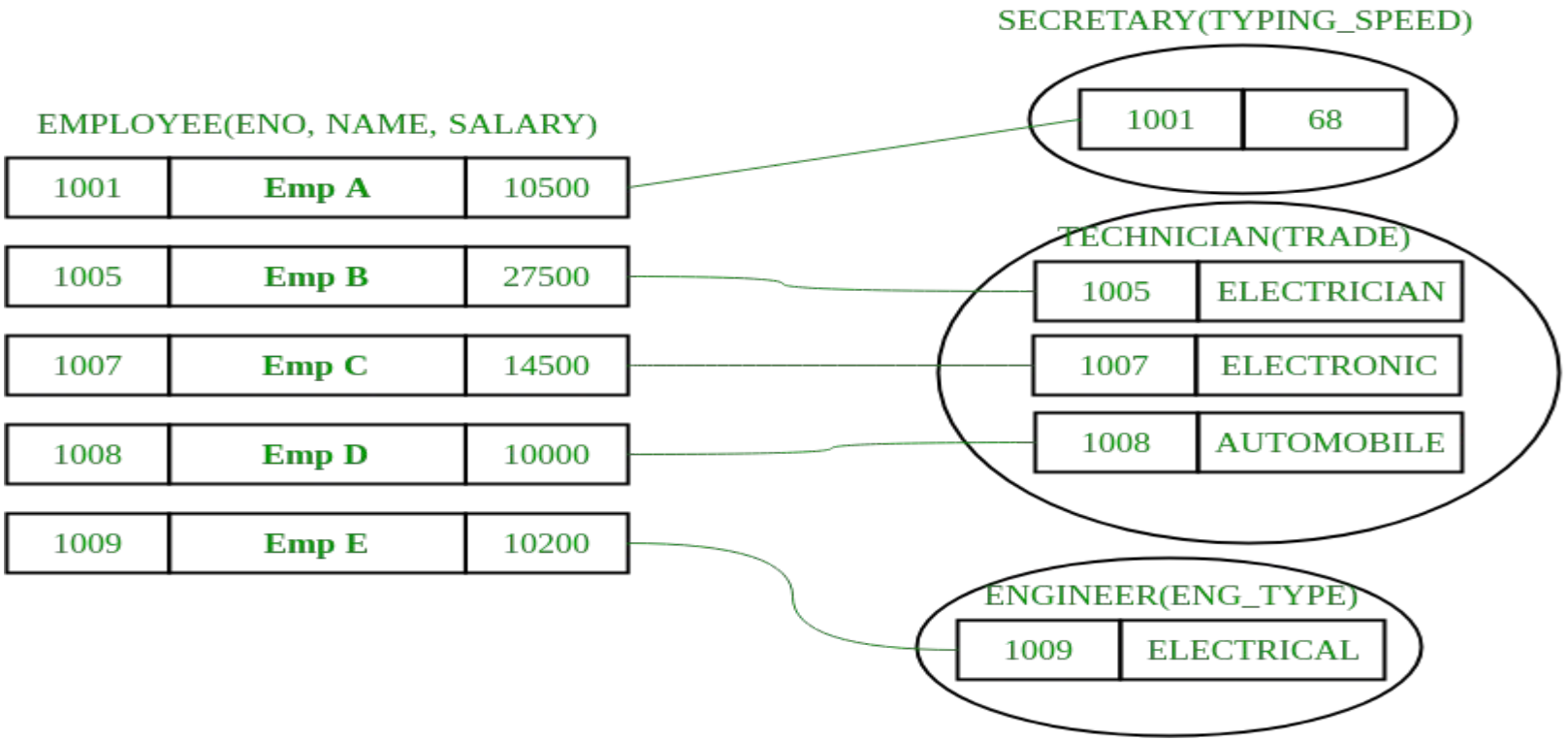



Vehicle
Car
Truck

- Example –

consider we have 3 sets of employees: Secretary, Technician, and Engineer.

The employee is super-class of the rest three sets of individual sub-class is a subset of Employee set.



Specialization

The process of defining subclasses of an entity type is called specialization.

- This entity type is called “superclass” of the specialization.
- Specialization distinguishes between subclasses based on a certain method:-

Example: Salaried Employee and Hourly Employee are grouped together because they are classified based on paying method.

The Employee entity type has 3 specializations:

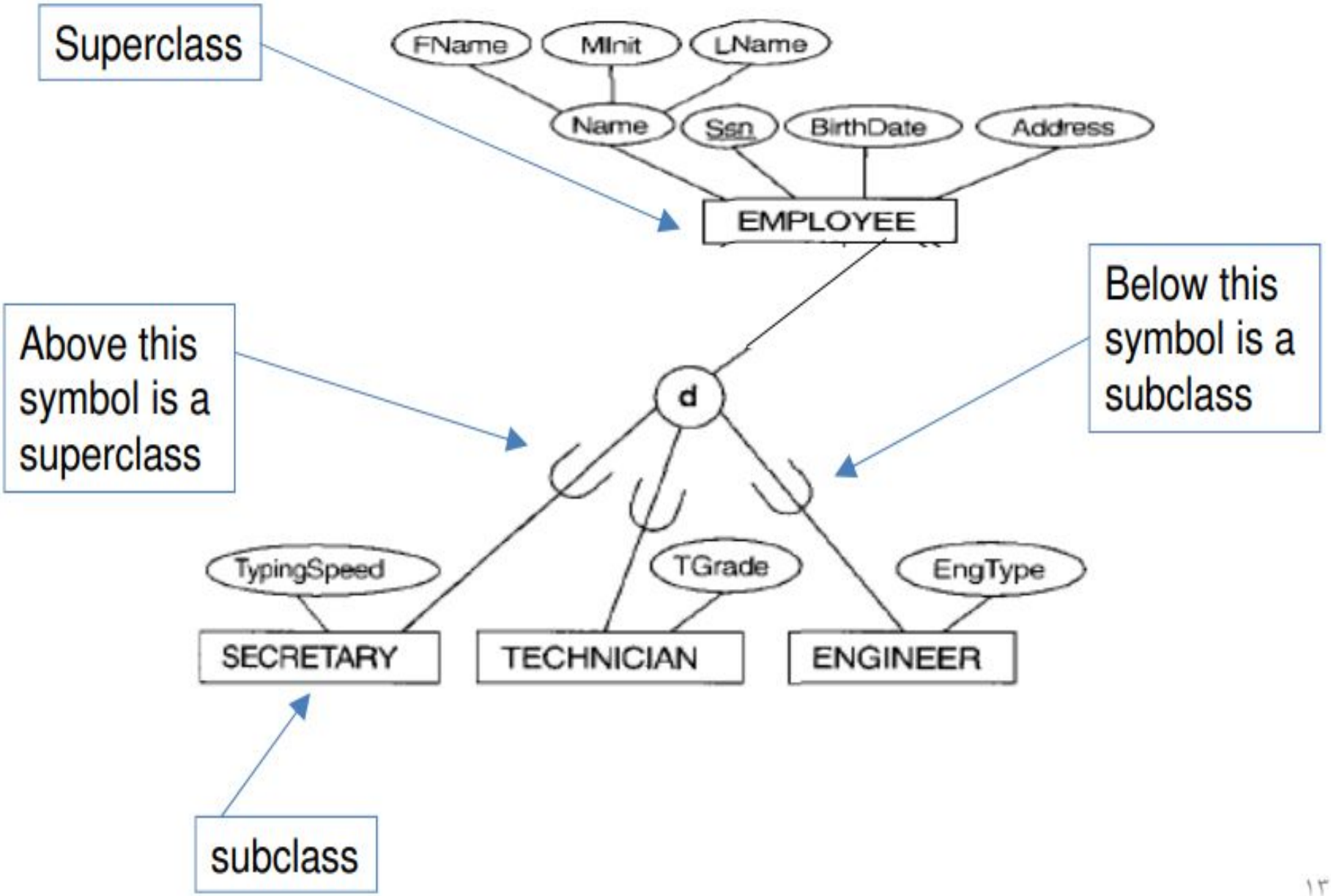
1) {SalariedEmployee,HourlyEmployee}
   - Classified based on paying method.

2) {Secretary, Technician, Engineer}
   - Classified based on job type.

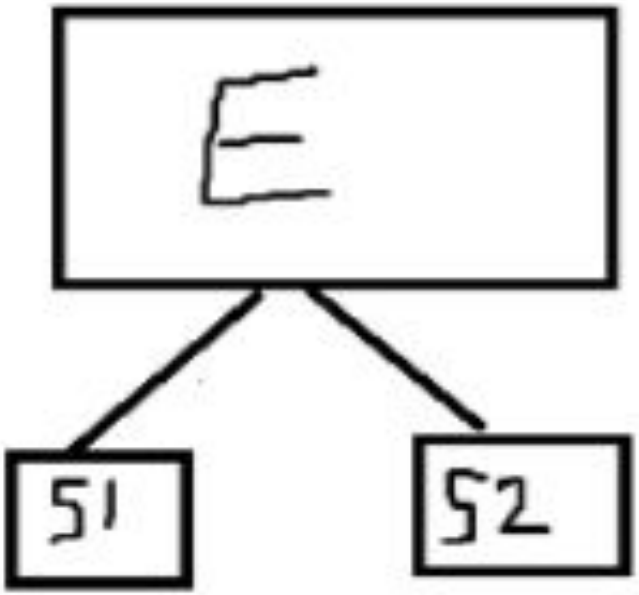
How specialization is represented in EER: –

Subclasses that define a specialization are attached by lines to a circle that represents the specialization, which is connected to the superclass.

- The subset symbol "U" on each line connecting a subclass to the circle indicates the direction of the superclass/subclass relationship.
- If the specialization contains only one subclass, we do not use the symbol ◯ which is used for grouping subclasses.


Superclass
FName
MInit
LName
Name
Ssn
BirthDate
Address
EMPLOYEE
Above this symbol is a superclass
Below this symbol is a subclass
d
TypingSpeed
TGrade
EngType
SECRETARY
TECHNICIAN
ENGINEER
subclass

d means "**disjoint**"- what it tells is the subclasses must have disjoint sets of entities.



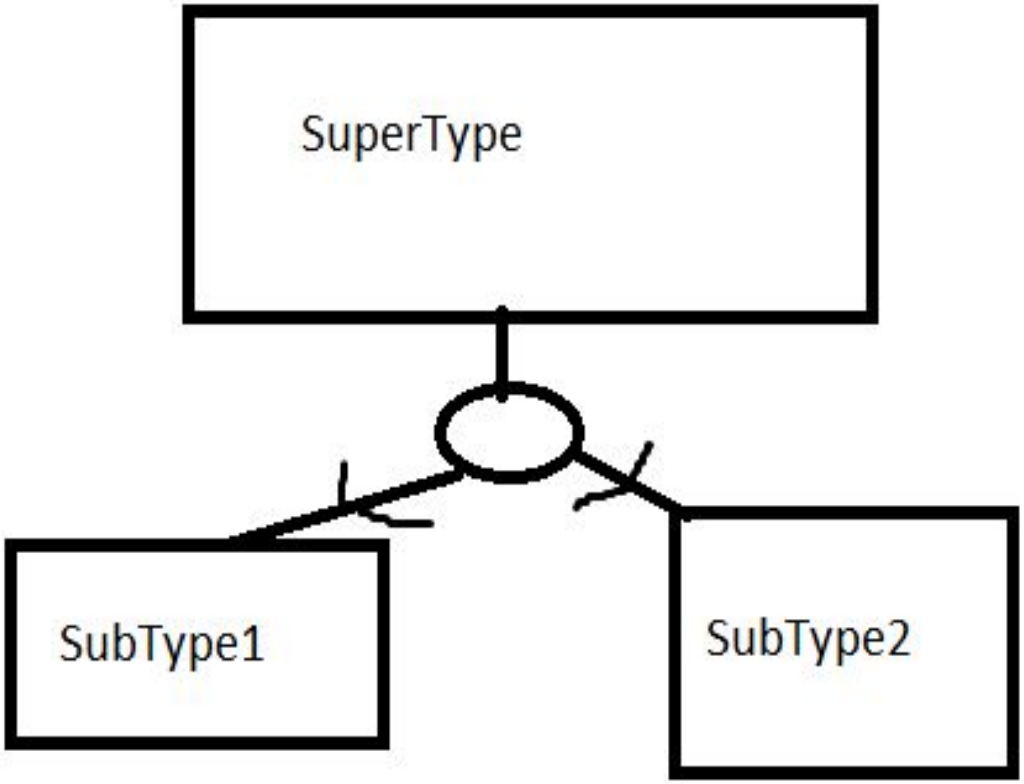
This means the sub classes are totally different.There won't be any one between this two.

Totally Seperated.

A disjoint relationship means that an entity from the superclass can belong to only one of the subclasses

Example :
An employee cannot be a physician assistant as well as a nurse, or, cannot be a nurse as well as a technician

- **U** symbol indicates the Subtype is a subset of the Supertype.

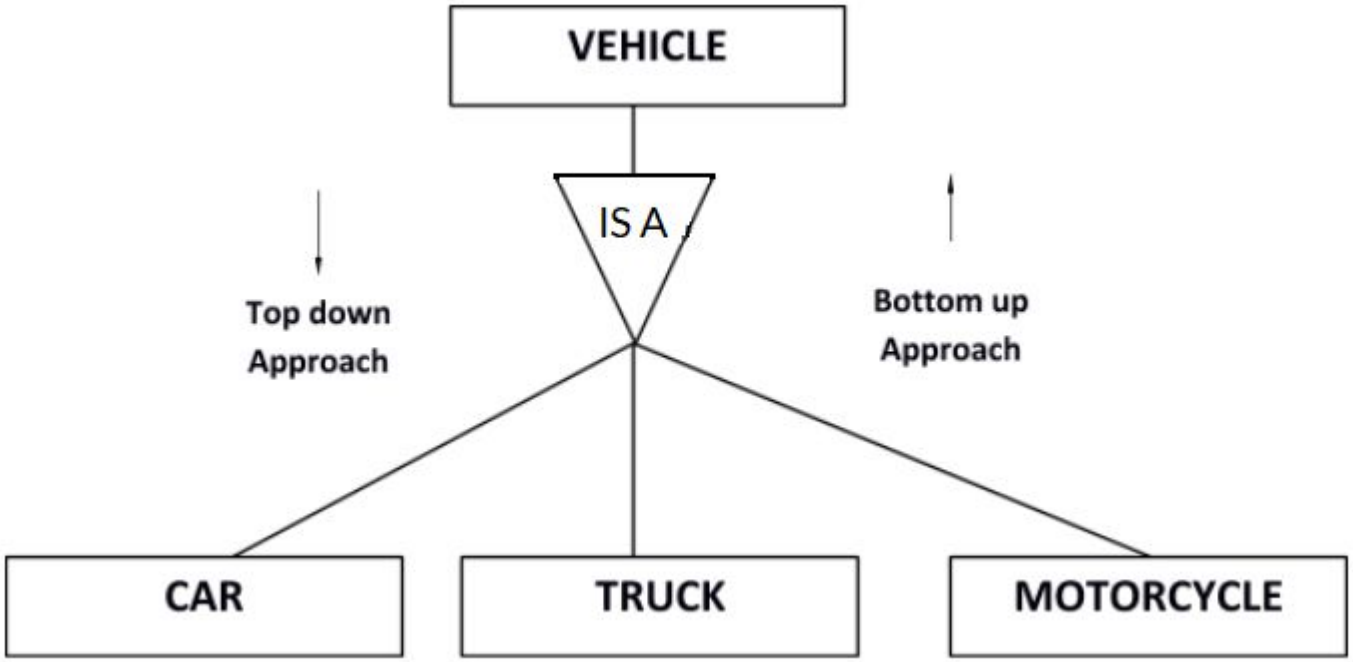


In summary, specialization allows us to: –

1. Define subclasses of entity types.
2. Define specific attributes for subclasses.
3. Define specific relationships between subclasses and other entities or subclasses.

Generalization is a Bottom up process i.e. consider we have 3 sub entities Car, Truck and Motorcycle. Now these three entities can be generalized into one super class named as Vehicle

Specialization is a top down approach in which one entity is broken down into low level entity.



Generalization

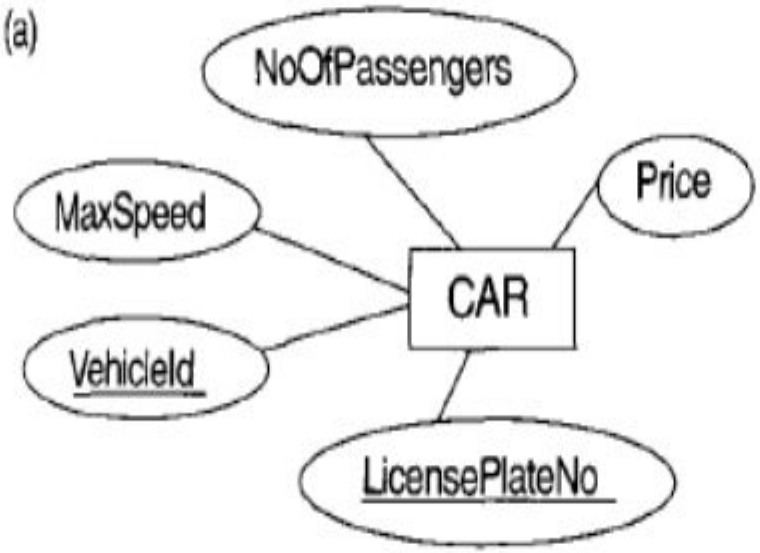
- Generalization is the reverse process of specialization.
- In generalization, you generalize a set of entity types into one superclass entity type. Therefore, the generalized entity types are considered subclasses.
- If you find a set of classes with many common attributes, they can be considered subclasses and generalized to a common entity (superclass).

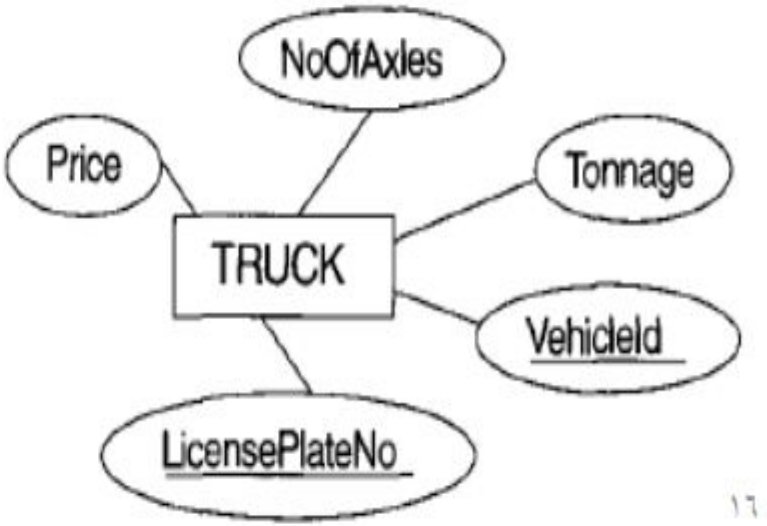
Example: In an ER diagram consider you have two entity types:

- Car
- Truck

How can we generalize this?

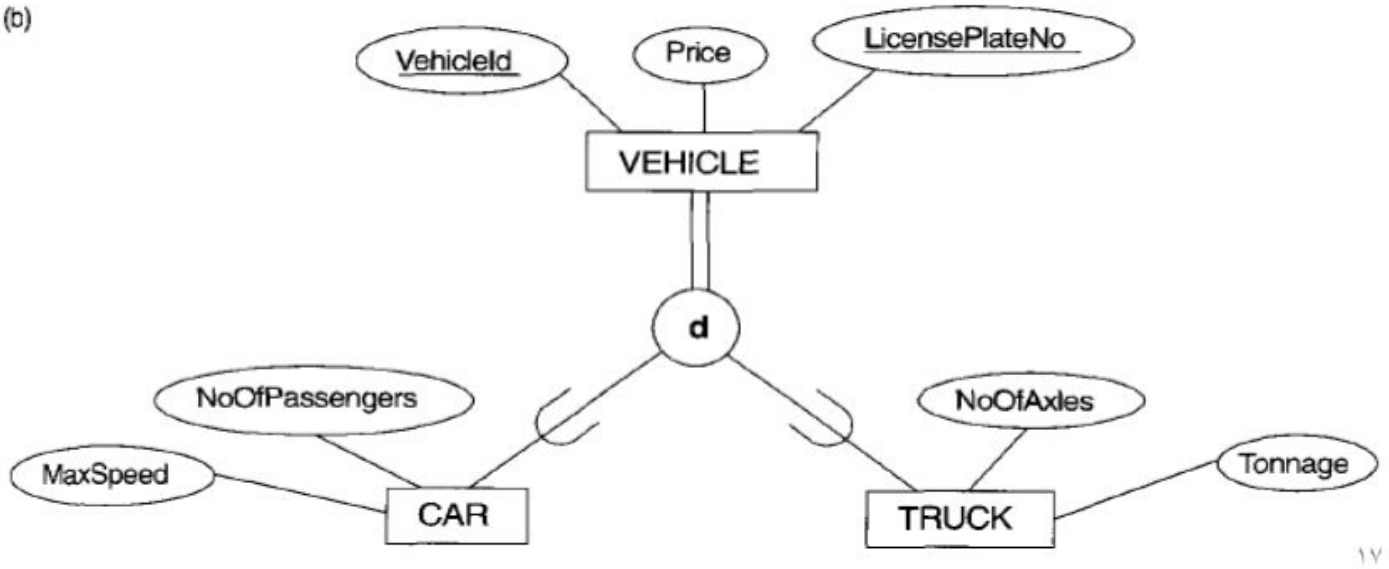
(a)





The common attributes go to the superclass and the current entity types (Car and Truck) become subclasses.

An Vehicle must either be car or truck-total participation



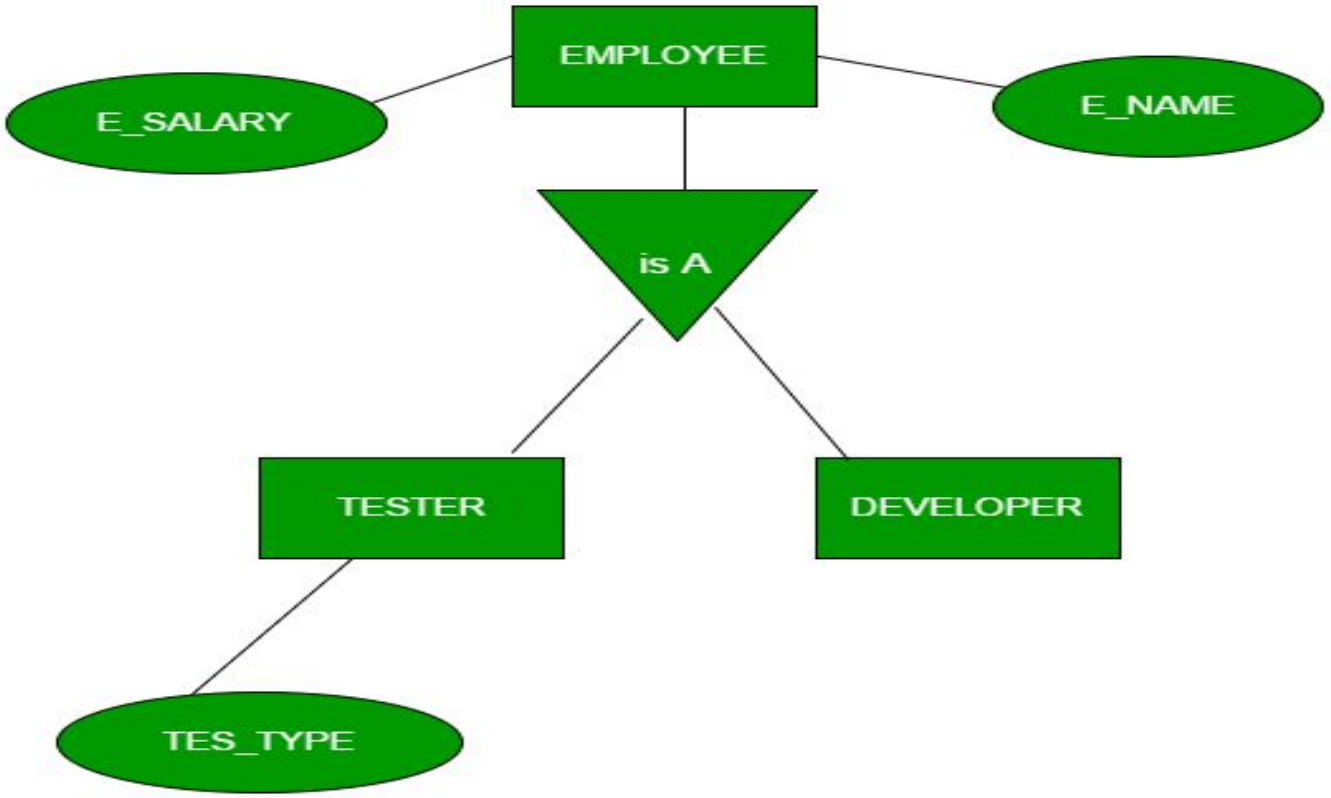
- Generalization is the process of extracting common properties from a set of entities and create a generalized entity from it.

- 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.
- For Example, STUDENT and FACULTY can be generalized to a higher level entity called PERSON. 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. For Example, EMPLOYEE entity in an Employee management system can be specialized into DEVELOPER, TESTER etc.

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).

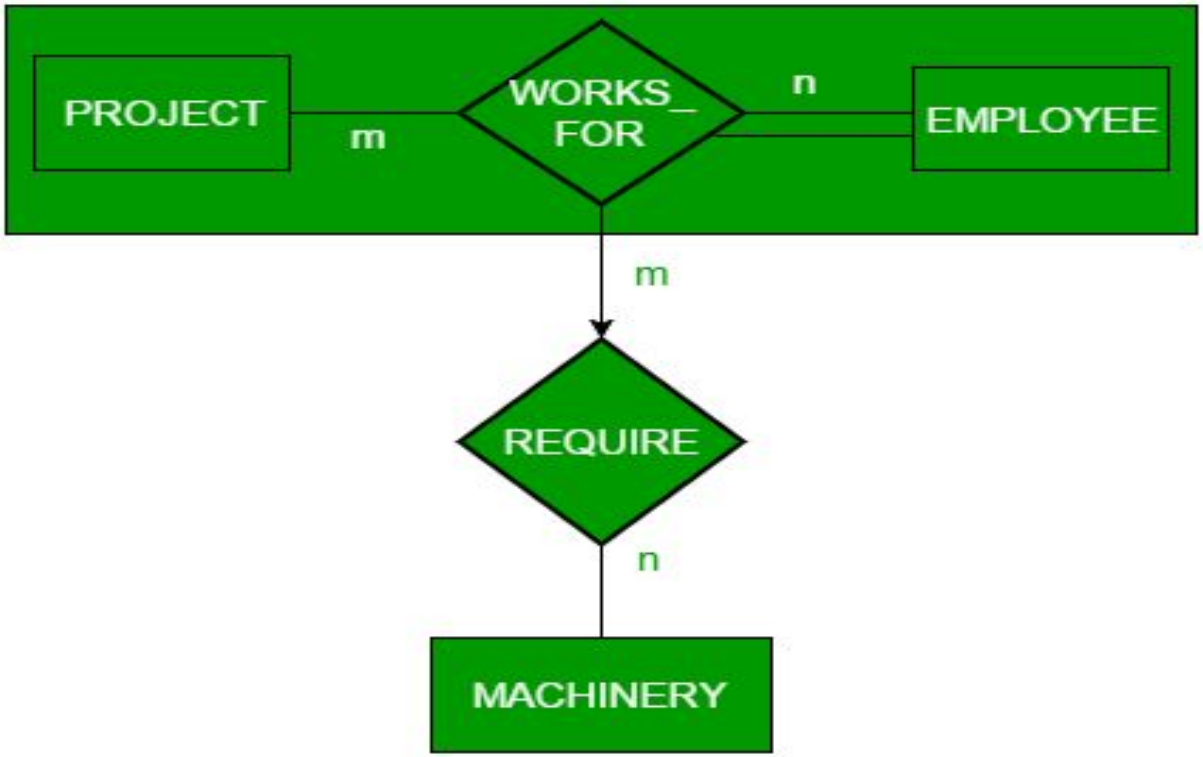


Specialization

Aggregation

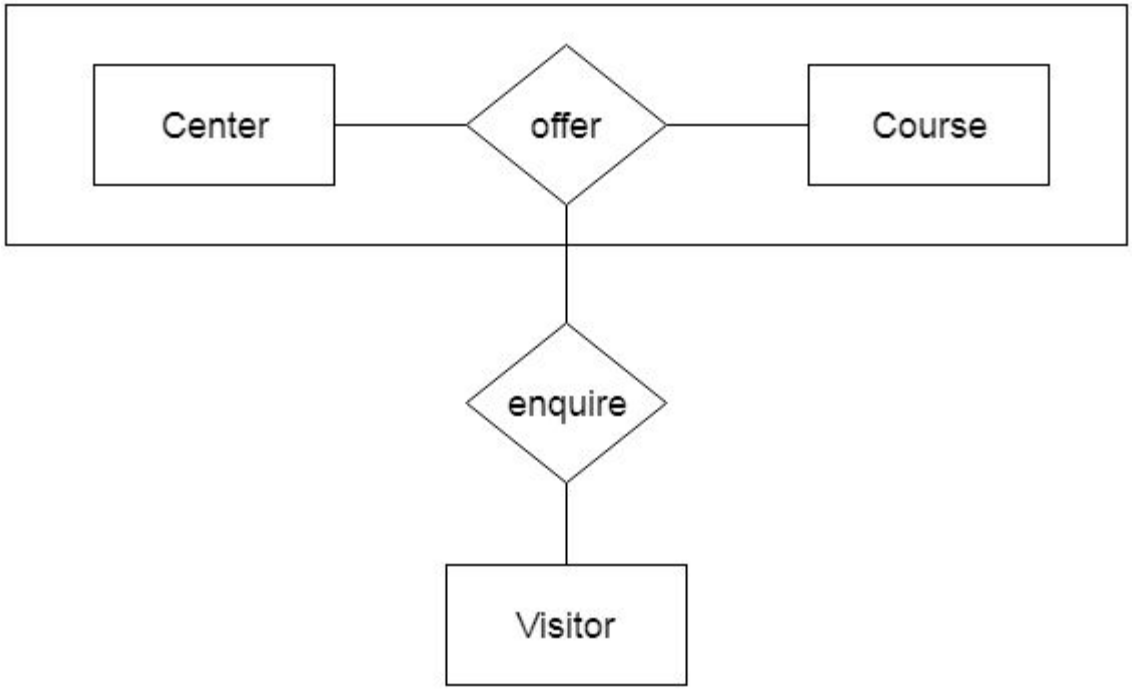
- An ER diagram is not capable of representing relationship between an entity and a relationship which may be required in some scenarios.
- In those cases, a relationship with its corresponding entities is aggregated into a higher level entity.
- Aggregation is an abstraction through which we can represent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

- For 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.

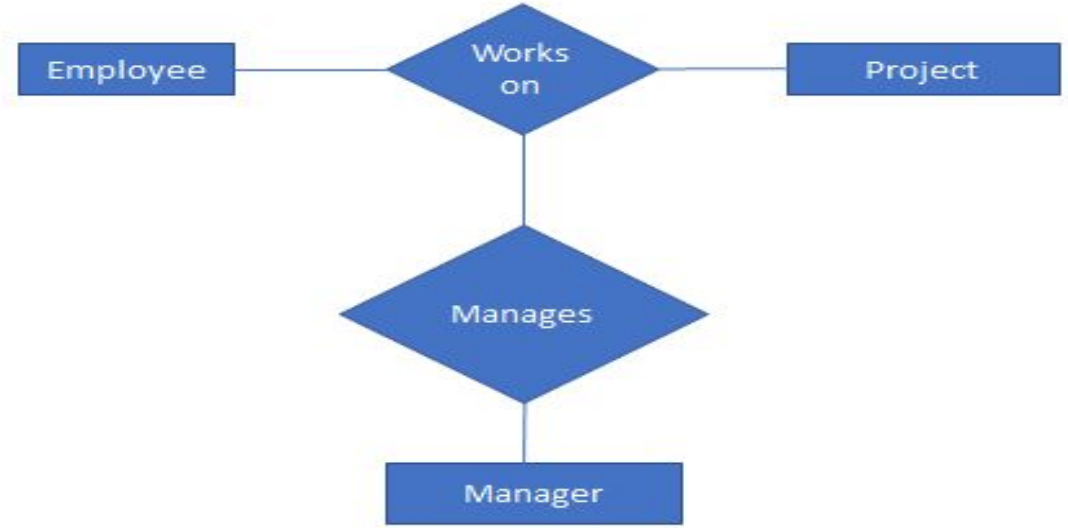


Aggregation

- **For example:** Center entity offers the Course entity act as a single entity in the relationship which is in a relationship with another entity visitor. In the real world, if a visitor visits a coaching center then he will never enquiry about the Course only or just about the Center instead he will ask the enquiry about both.



In real world, we know that a manager not only manages the employee working under them but he has to manage the project as well. In such scenario if entity “Manager” makes a “manages” relationship with either “Employee” or “Project” entity alone then it will not make any sense because he has to manage both. In these cases the relationship of two entities acts as one entity. In our example, the relationship “Works-On” between “Employee” & “Project” acts as one entity that has a relationship “Manages” with the entity “Manager”.



Thank you …