Database Management System (DBMS)

ITC303

Subject Incharge

Mrs. Priyanka Patil

Assistant Professor

email: priyankapatil@sfit.ac.in

Module 2

The Entity-Relationship Model

Contents

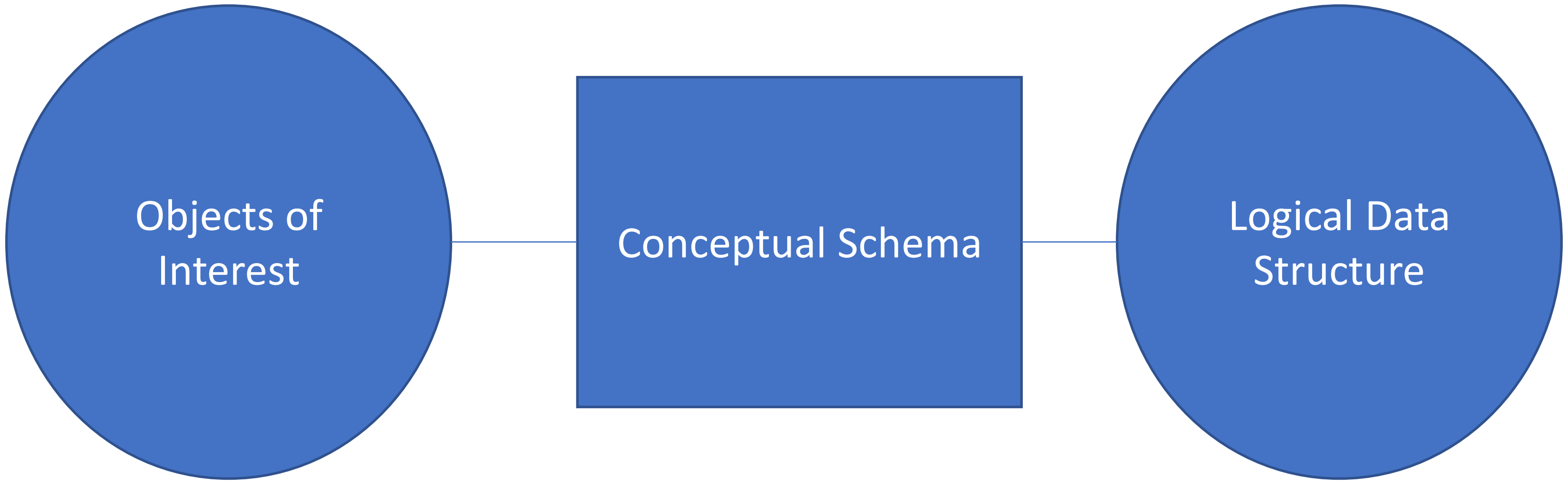
- ❑Conceptual Modeling of a database
- ❑The Entity-Relationship (ER) Model
- ❑Entity Type
- ❑Entity Sets
- ❑Attributes and Keys
- ❑Relationship Types,
- ❑Relationship Sets
- ❑Weak entity Types
- ❑Generalization, Specialization and Aggregation,
- ❑Extended Entity-Relationship (EER) Model.

Conceptual Modeling of a database

- The process of **analysis of data objects** and **their relationships** to other data objects is known as **data modeling.**
- It is the conceptual representation of data in database. It is the first step in database designing.
- Data models define **how data is connected** to each other and **how they are processed** and **stored** inside the system.
- A data model provides a way to describe the design of a database at the physical, logical and view levels.
- There are different types of Data models like **Relational model, Network database model, Object Oriented model** and **Entity Relationship model**

The Entity-Relationship (ER) Model

ER model represents **Overall Logical Structure of a database**



Two phases of database modeling

The Entity-Relationship (ER) Model

- Widely used ***conceptual level data model***
  - proposed by ***Peter P Chen*** in 1970s
- Data model to describe the database system at the requirements collection stage
  - high level description.
  - easy to understand for the enterprise managers.
  - rigorous enough to be used for system building.
- Concepts available in the model
  - ***entities*** and ***attributes*** of entities.
  - ***relationships*** between entities.
  - diagrammatic notation.

E/R (Entity/Relationship) Model

- A **conceptual level** data model.
- Provides the concepts of ***entities, relationships*** and ***attributes.***

The University Database Context

Entities: *student*, *faculty member*, *course*, *departments* etc.

Relationships: *enrollment* relationship between student & course,
employment relationship between faculty member, department etc.

Attributes: *name*, *rollNumber*, *address* etc., of *student* entity,
name, *empNumber*, *phoneNumber* etc., of faculty entity etc.

More details will be given in the E/R Model Module.

The Entity-Relationship (ER) Model

ER diagram basically having three components:

- ❑ Entities
- ❑ Attributes
- ❑ Relationship

Entities

- *Entity* - a thing (animate or inanimate) of independent physical or conceptual existence and *distinguishable*.

  In the University database context, an individual *student*, *faculty member*, a *class room*, a *course* are entities.

• The Entity-Relationship (ER) Model

Entity : An entity is a thing in a real-world with independent existence. An entity can exist independently and is distinguishable from other objects. It can be **identified uniquely**.

Example:

- A **student** with a particular **roll number** is an entity.
- A **company** with a particular **registration number** is an entity.

An entity can be of two types :

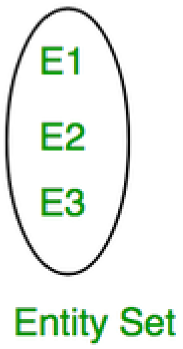
- **Tangible Entity :** Entities that **exist in the real world** physically. Example: Person, car, etc.
- **Intangible Entity :** Entities that **exist only logically** and have no physical existence. Example: Bank Account, etc.

Entity Set or Entity Type

Entity Set - Collection of entities all having the **<u>same properties</u>**. The **type** of all the entities should be the same.

For Example* : *Student* entity set – collection of all *student entities.

Course* entity set** – **collection of all *course entities.



Entity Set or Entity Type(Cont…)

Attributes

- Each entity is described by a **set of attributes/properties** that have associated values
- For Example **Student** entity
  - ▪StudName – name of the student.
  - ▪RollNumber – the roll number of the student.
  - ▪Gender – the gender of the student etc.
  - ▪All entities in an Entity set/type have the same set of attributes. Chosen set of attributes – amount of detail in modeling.

Attributes:

1. Attributes are properties which define a entity type
2. Attributes is a mapping from an entity set to its domain value.

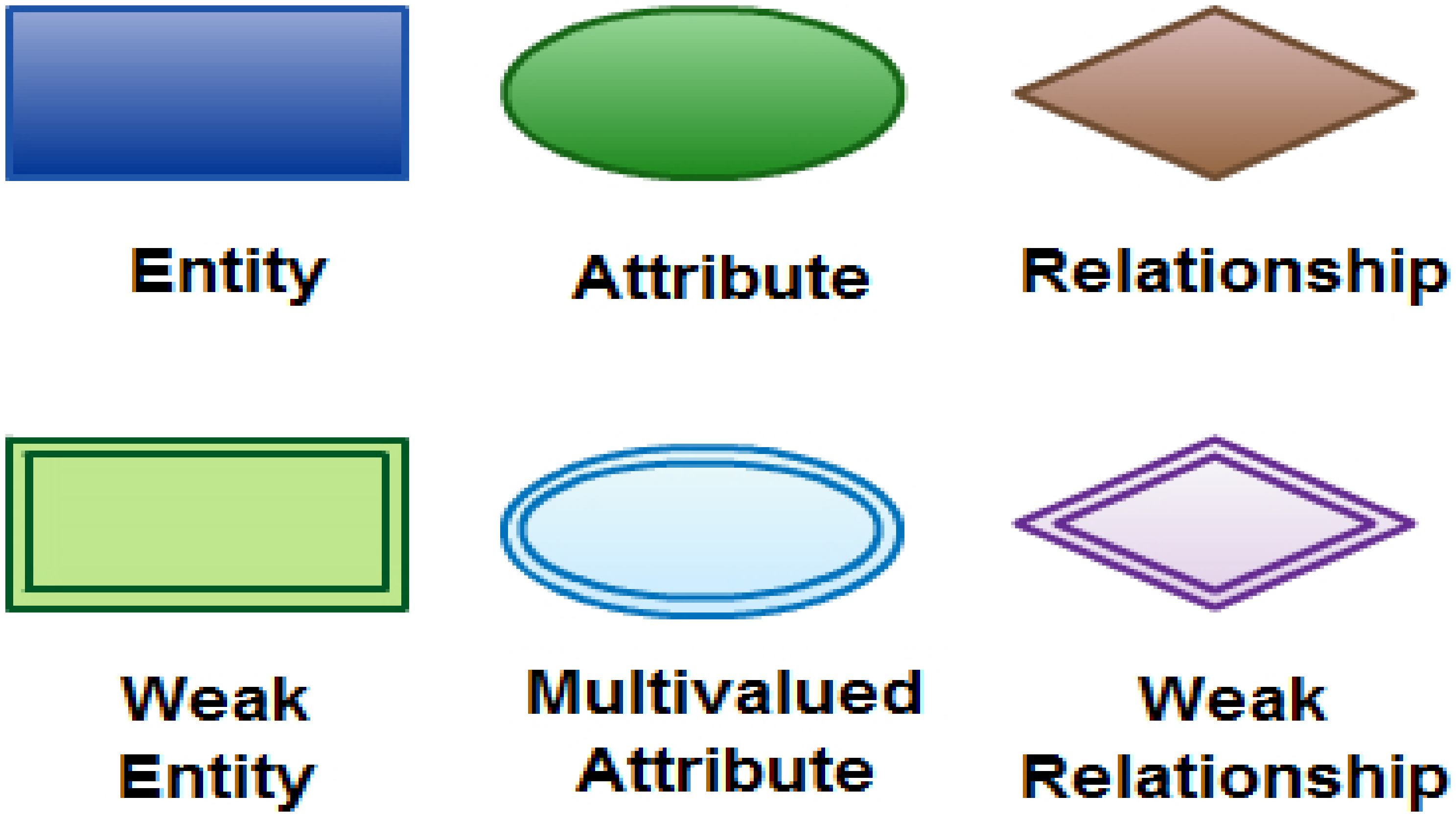
Types of Attributes

- **Simple Attributes**
  - having **atomic or indivisible** values.
- example: *Dept* – a string
  - *PhoneNumber* – a ten digit number
- **Composite Attributes**
  - having several **components** in the value.
- example: *Qualification* with components
- (*DegreeName*, *Year*, *UniversityName*)
- **Derived Attributes**
  - Attribute value is **dependent** on some other attribute.
- example: *Age* depends on *DateOfBirth*. So age is a derived attribute.

Types of Attributes (Cont..)

- **Single-valued**
  - having only one value rather than a set of values.
  - for instance, *PlaceOfBirth* – single string value.
- **Multi-valued**
  - having a set of values rather than a single value.
- for instance,
- *CoursesEnrolled* attribute for student
- *EmailAddress* attribute for student
- *PreviousDegree* attribute for student.
- Attributes can be:
  - simple single-valued, simple multi-valued,
  - composite single-valued or composite multi-valued.

Diagrammatic Notation for Entities

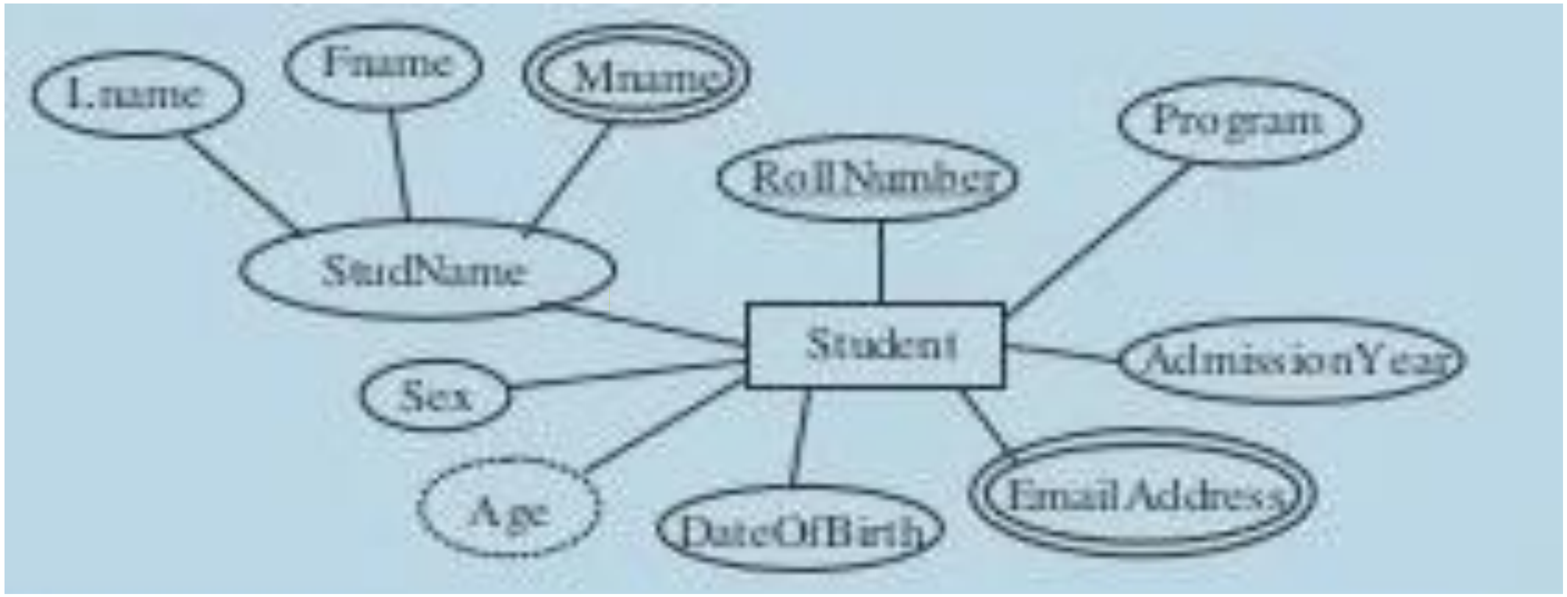


Diagrammatic Notation for Entities

entity - rectangle

attribute – ellipse/Oval connected to rectangle

multi-valued attribute - double ellipse
composite attribute - ellipse connected to ellipse
derived attribute - dashed ellipse



Entity Sets and Key Attributes

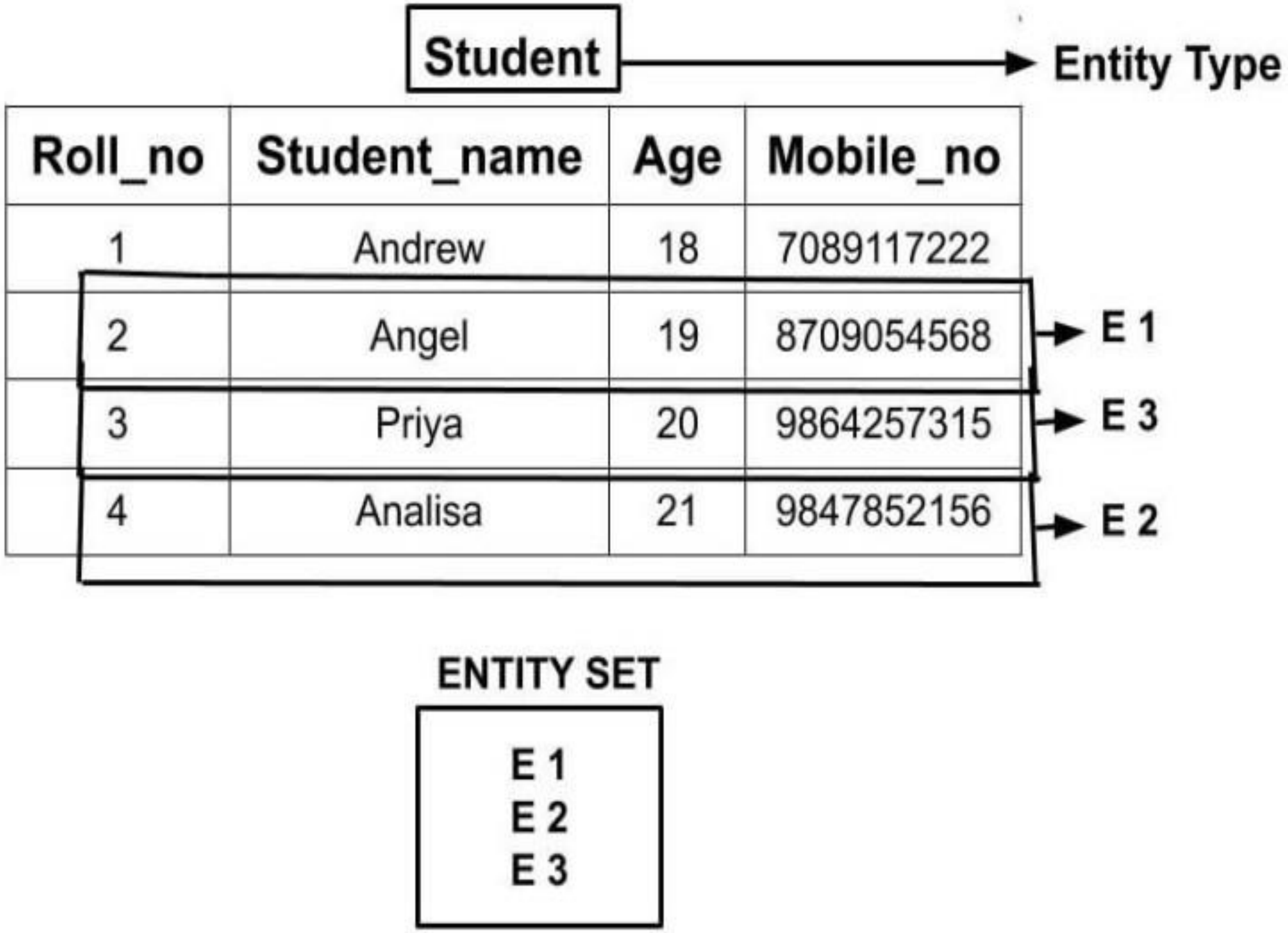
- ***Key*** – an attribute or a collection of attributes whose value(s) **uniquely identify** an entity in the entity set.
  - For **example,**
    - *RollNumber* - Key for *Student* entity set
    - *EmpID* - Key for *Faculty* entity set
    - *HostelName, RoomNo* - Key for *Student* entity set (assuming that each student gets to stay in a single room)
  - A key for an entity set may have **more than one attribute**.
  - An entity set may **have more than one key.**
- Keys can be determined only from the meaning of the attributes in the entity type.
  - Determined by the **designers**

Relationships

- When two or more entities are associated with each other, we have an instance of a ***Relationship***.
  - E.g.: *student* Ramesh *enrolls* in Discrete Mathematics *course*
  - Relationship *enrolls* has *Student* and *Course* as the *participating* entity sets.

- 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

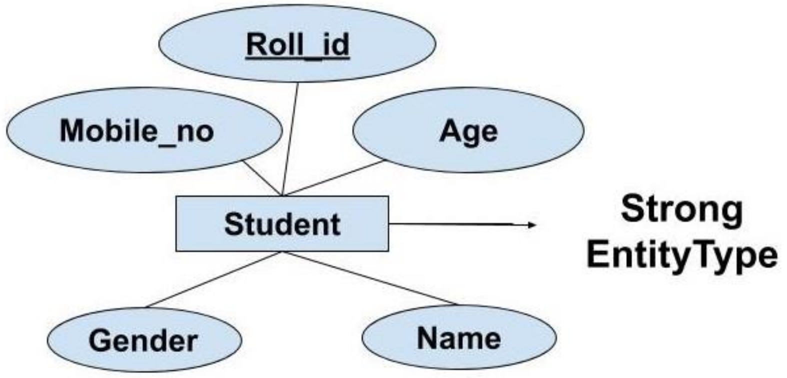


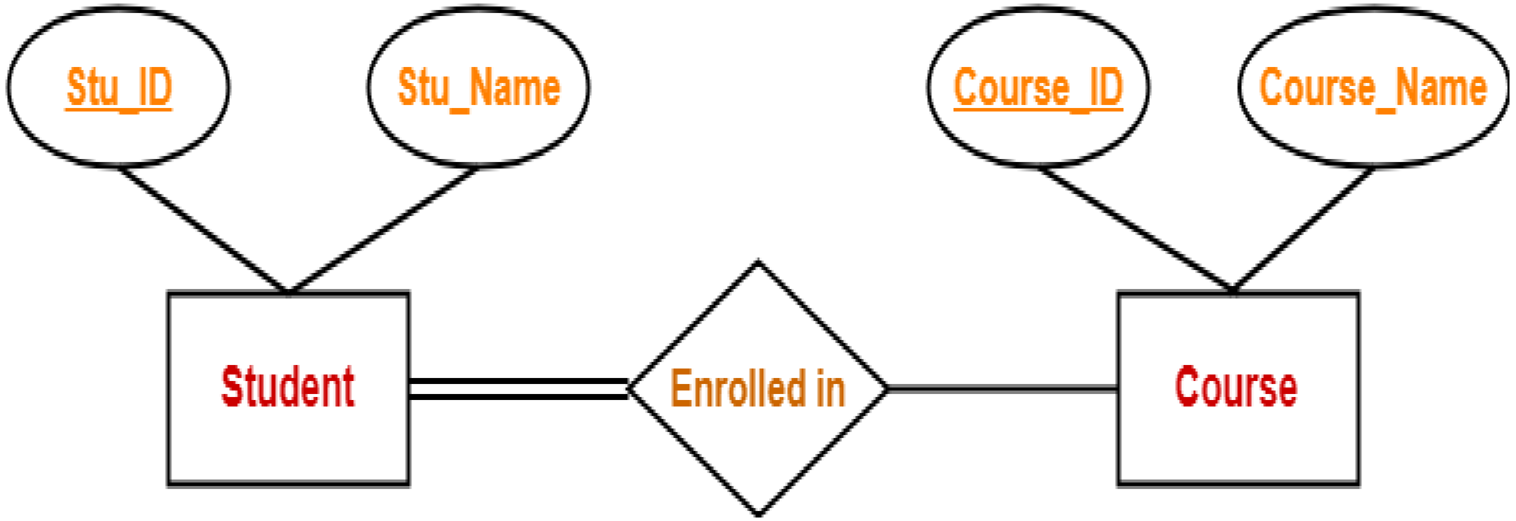
Roll_no	Student_name	Age	Mobile_no
1	Andrew	18	7089117222
2	Angel	19	8709054568
3	Priya	20	9864257315
4	Analisa	21	9847852156

Entity Type

Strong Entity

- A strong entity is **not dependent** on any other entity in the schema.
- A strong entity will always have a primary key.
- Strong_entities are represented by a **single rectangle.**
- The relationship of two strong entities is represented by a single diamond.



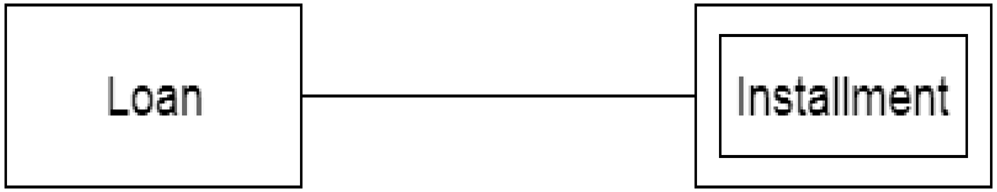

Stu_ID
Stu_Name
Course_ID
Course_Name
Student
Enrolled in
Course

- 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.

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.



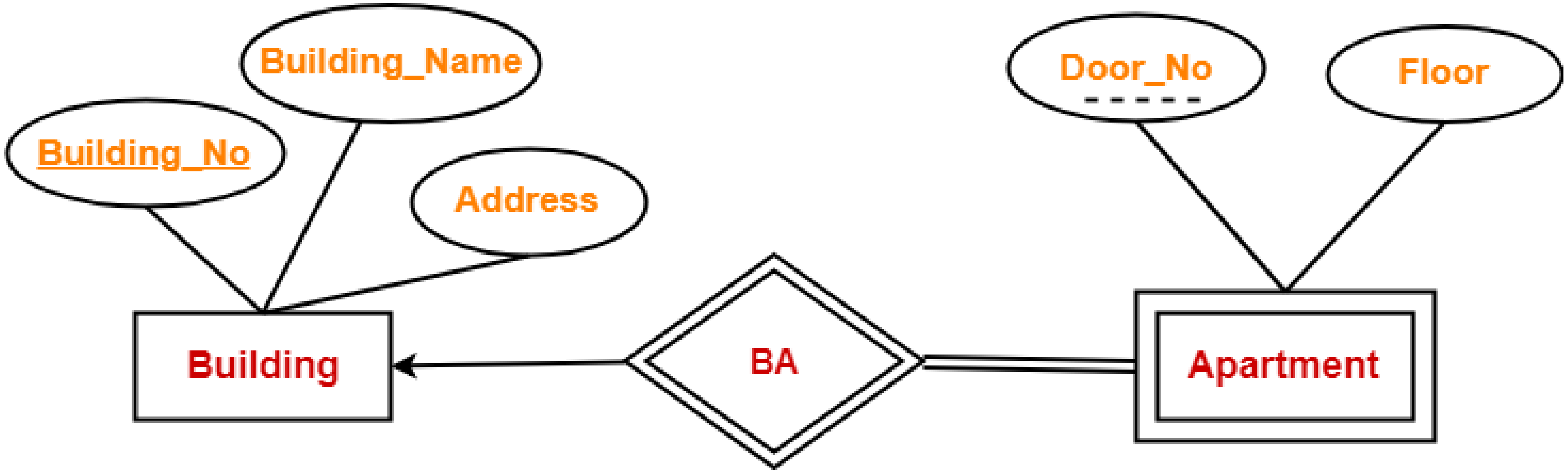
- A weak entity set is an entity set that does not contain sufficient attributes to uniquely identify its entities.
- In other words, a primary key does not exist for a weak entity set.
- However, it contains a **partial key** called as a **discriminator.**
- Discriminator can identify a group of entities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

- **NOTE-**

- The combination of discriminator and primary key of the strong entity set makes it possible to uniquely identify all entities of the weak entity set.
- Thus, this combination serves as a primary key for the weak entity set.
- Clearly, this primary key is not formed by the weak entity set completely.

Primary key of weak entity set

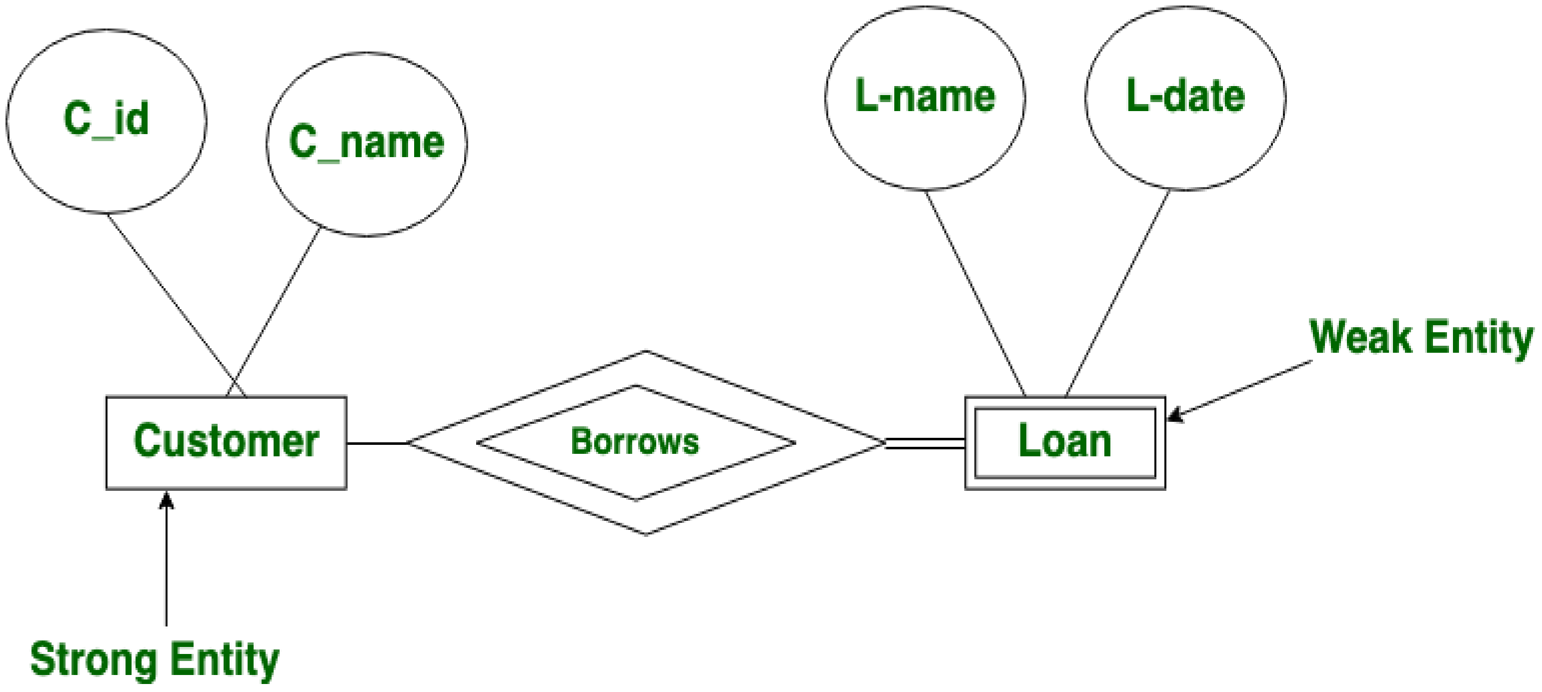
= Its own discriminator + Primary key of strong entity set



- 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.

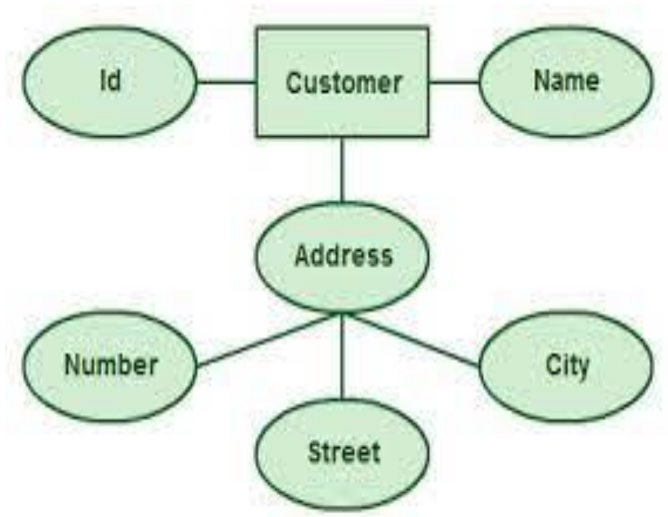
The relation between one strong and one weak entity is represented by a double diamond. This relationship is also known as **identifying relationship.**

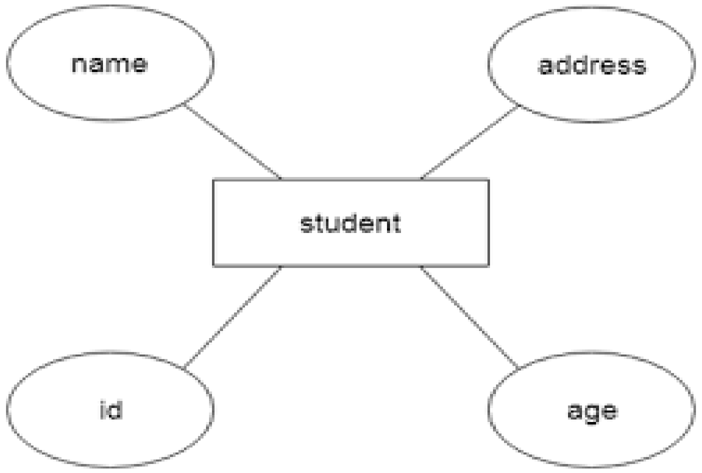


- A weak entity is one that can only exist when owned by another one.
- **Example-1:**
  The existence of rooms is entirely dependent on the existence of a hotel. So room can be seen as the weak entity of the hotel.
- **Example-2:**
  The bank account of a particular bank has no existence if the bank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

2. Attribute

The attribute is used **to describe the property of an entity**. Eclipse is used to represent an attribute.
For example, id, age, contact number, name, etc. can be attributes of a student.





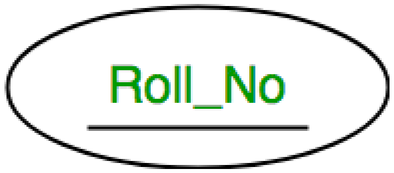
1st August 2022

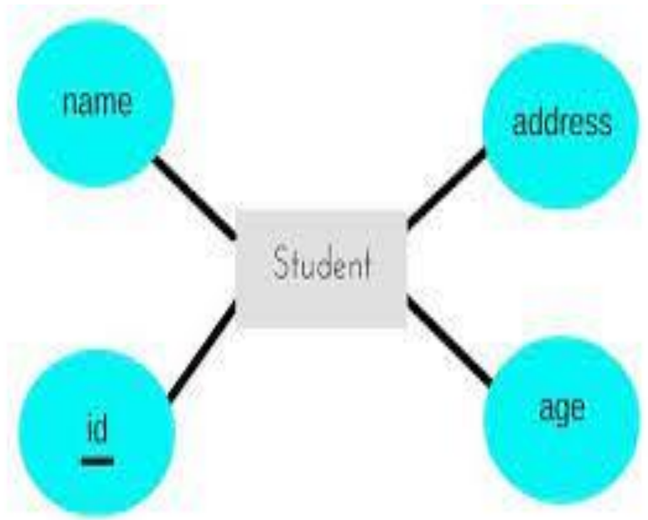
A. Key Attribute

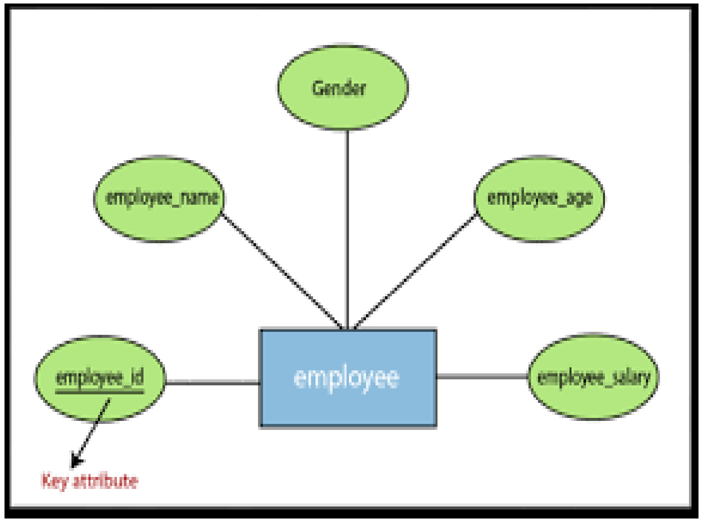
The attribute which **uniquely identifies each entity** in the entity set is called key attribute.

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.



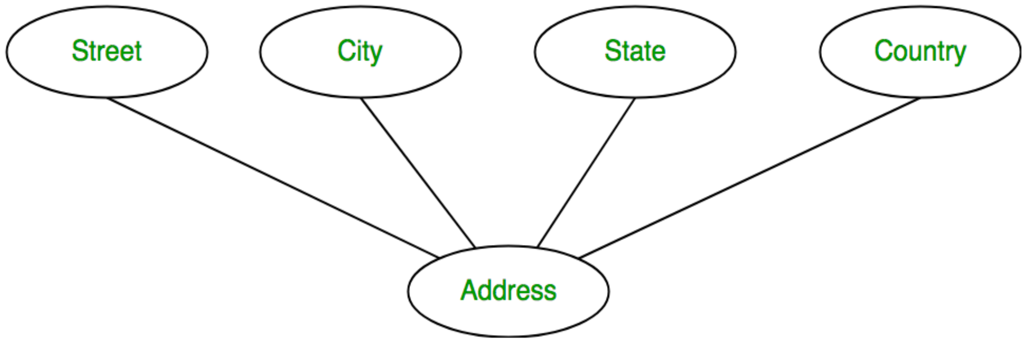

name
address
Student
id
age


Gender
employee_name
employee_age
employee_id
employee
employee_salary
Key attribute

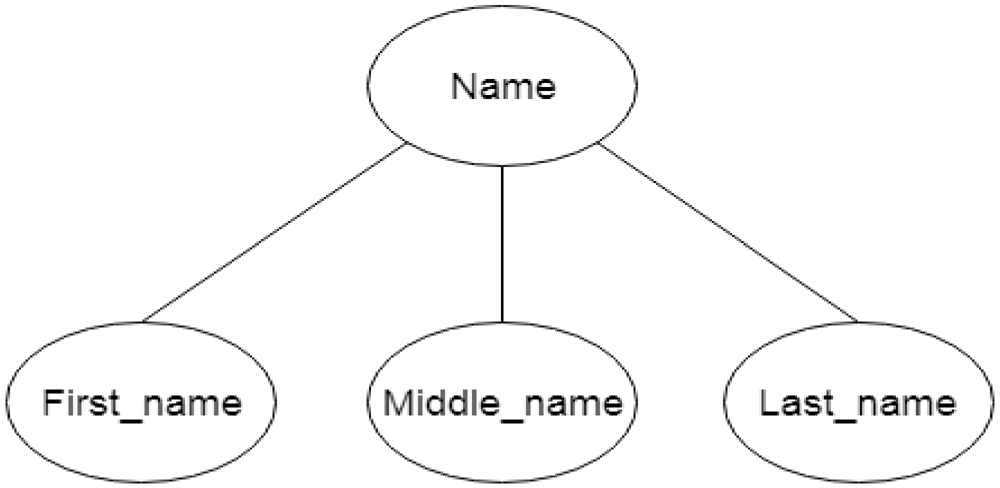
b. Composite Attribute

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.



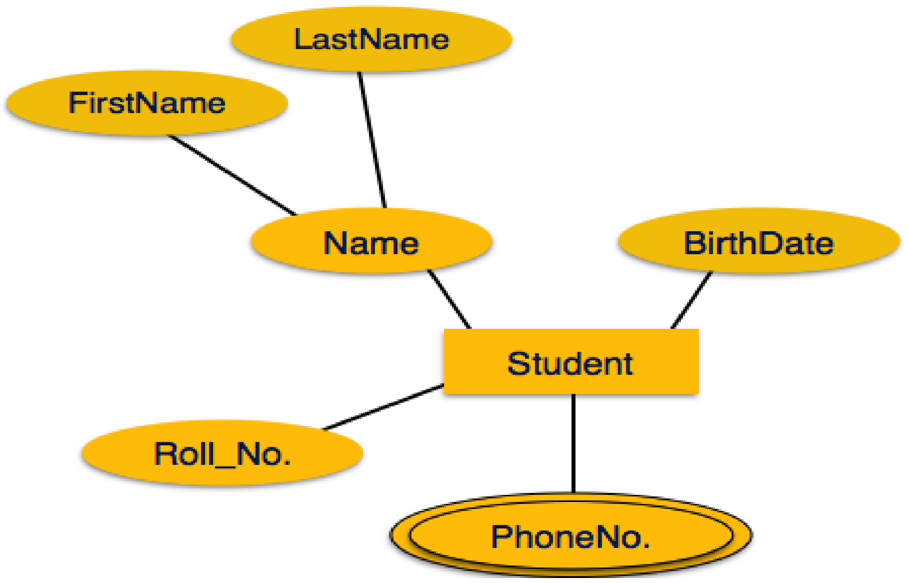
Other Example

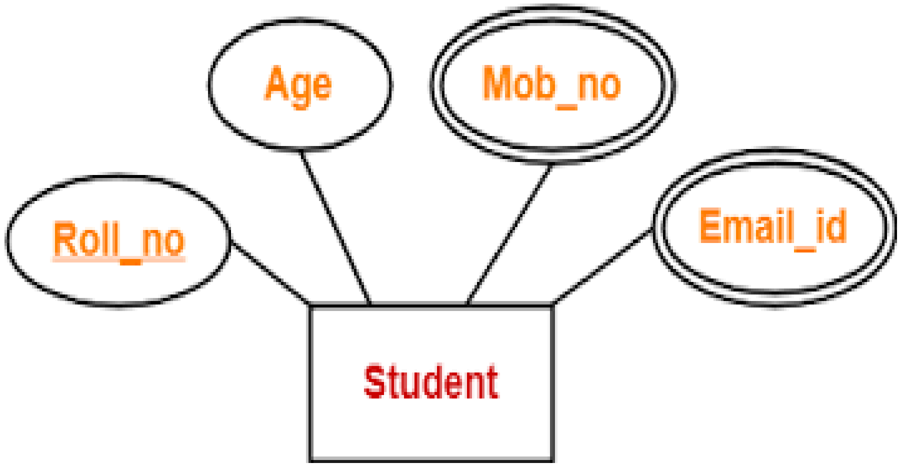


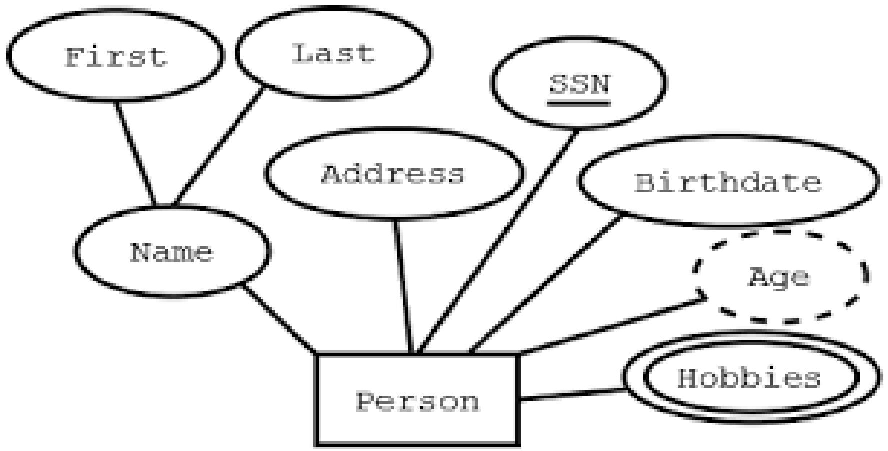
c. Multivalued Attribute

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.



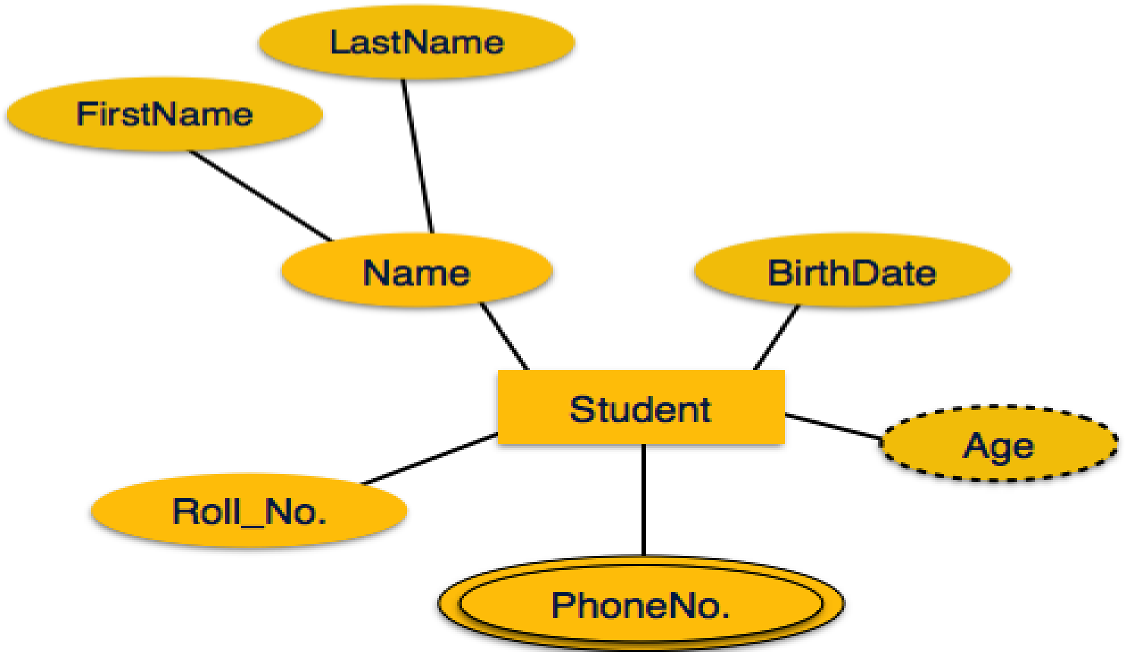

Age
Mob_no
Roll_no
Email_id
Student

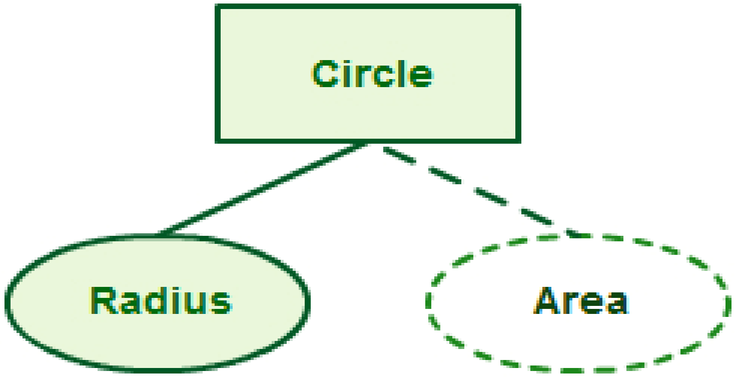

First
Last
SSN
Address
Birthdate
Name
Age
Hobbies
Person

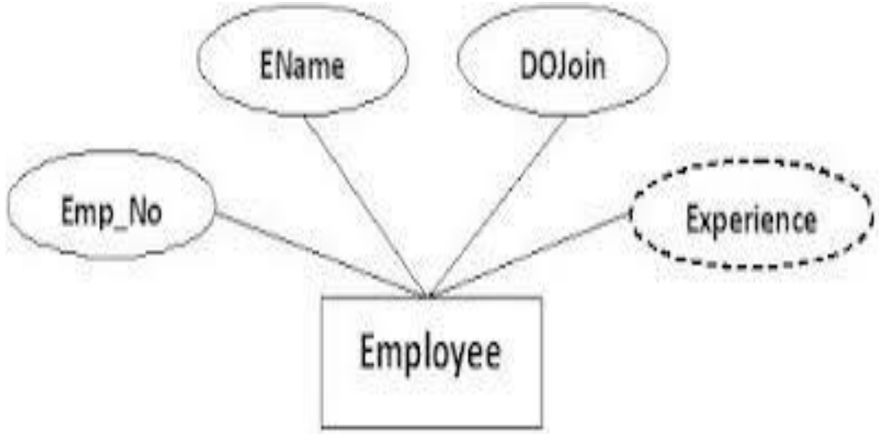
d. Derived Attribute

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.



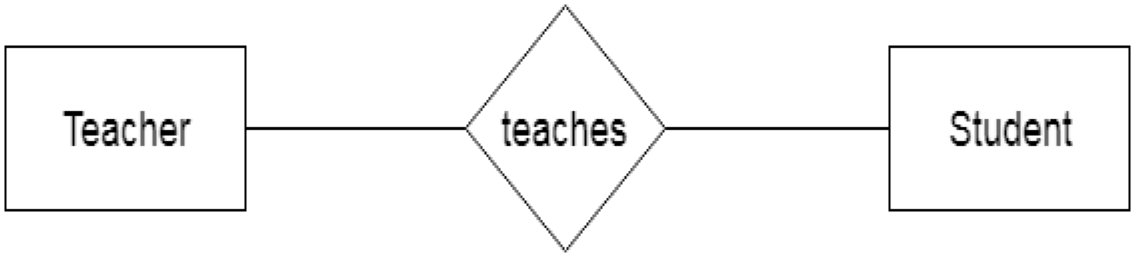

Circle
Radius
Area


EName
DOJoin
Emp_No
Experience
Employee

3. Relationship

A relationship is used to describe the relation between entities.

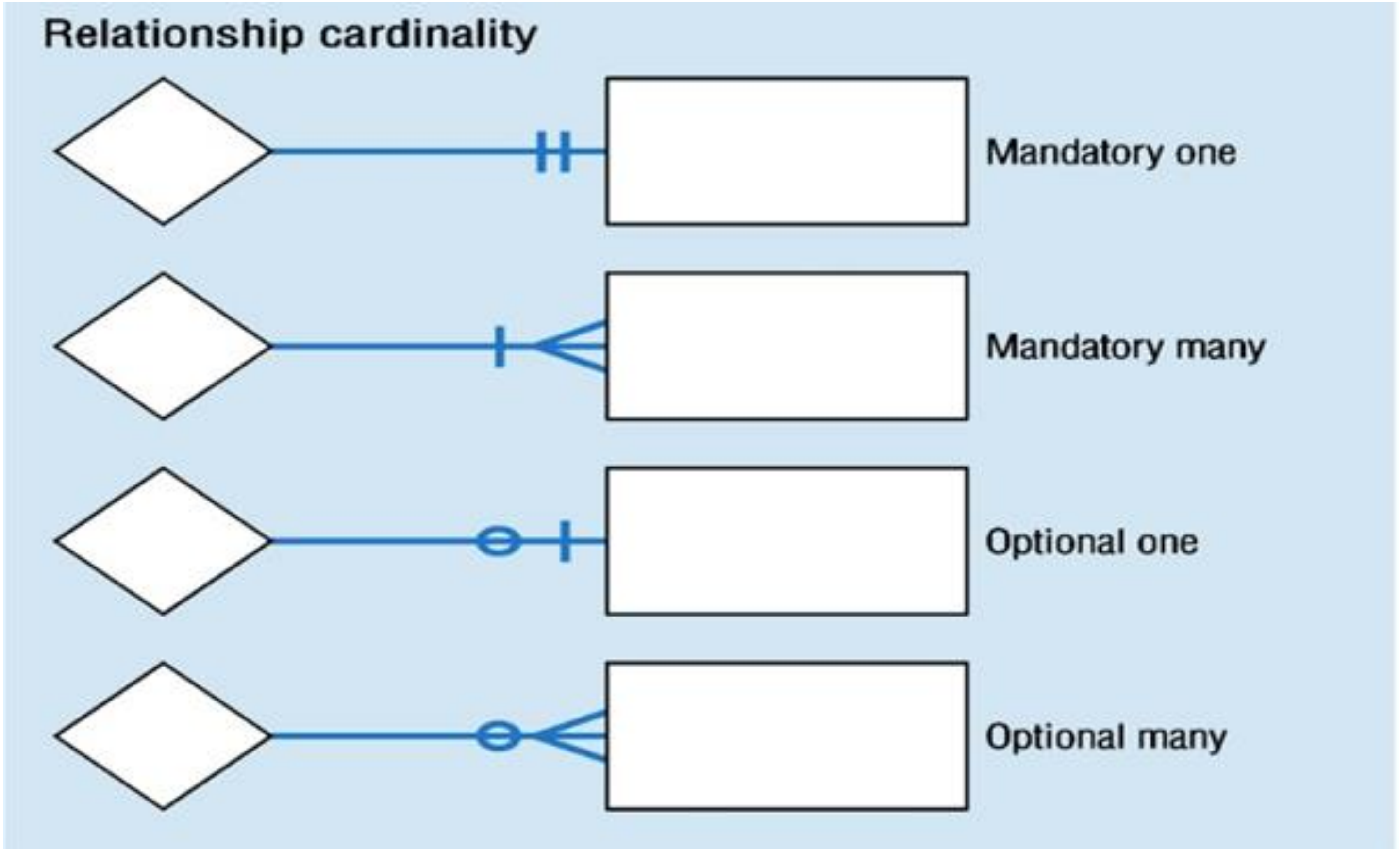
Diamond or rhombus is used to represent the relationship.



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

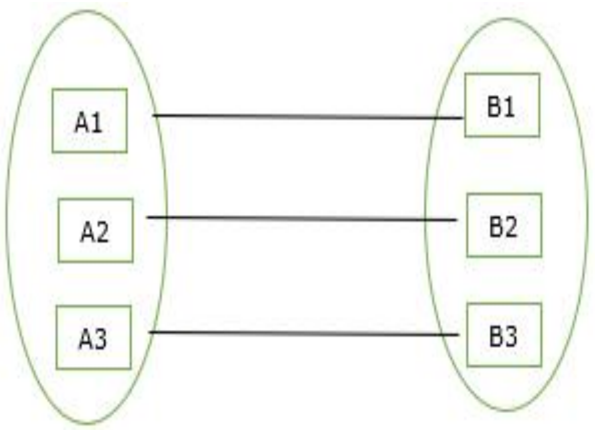

Relationship cardinality
Mandatory one
Mandatory many
Optional one
Optional many

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.

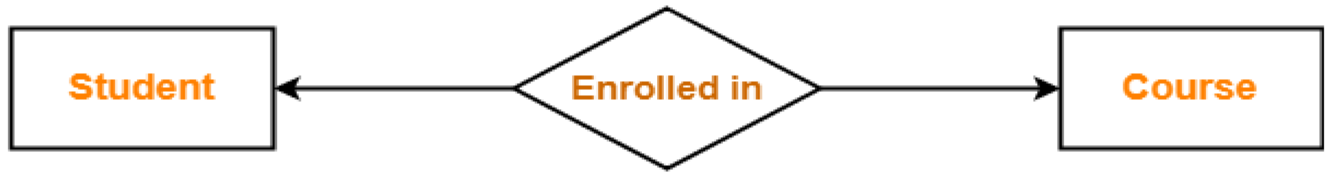
•An entity in set B can be associated with at most one entity in set A.



One entity of A is associated with one entity of B

- **Example-**

- Consider the following ER diagram-



One to One Relationship

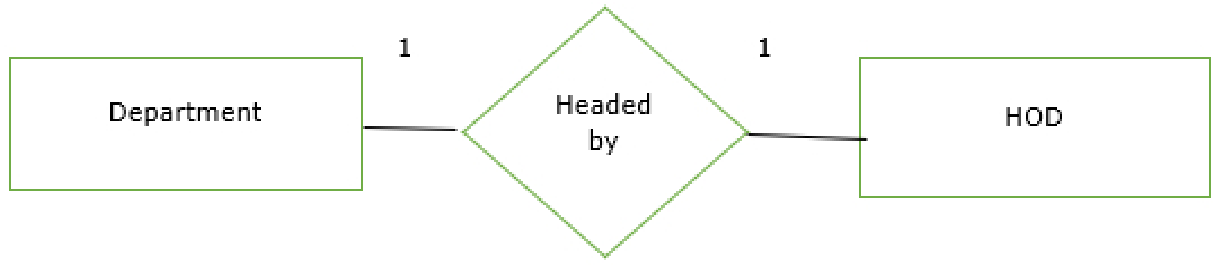
Here,

•One student can enroll in at most one course.

•One course can be enrolled by at most one student.

- **Example**

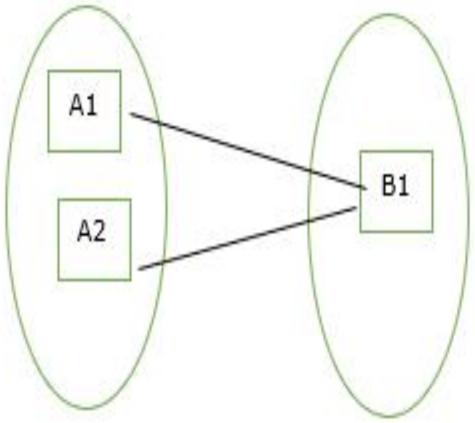
1] Here, one department has one head of the department (HOD).

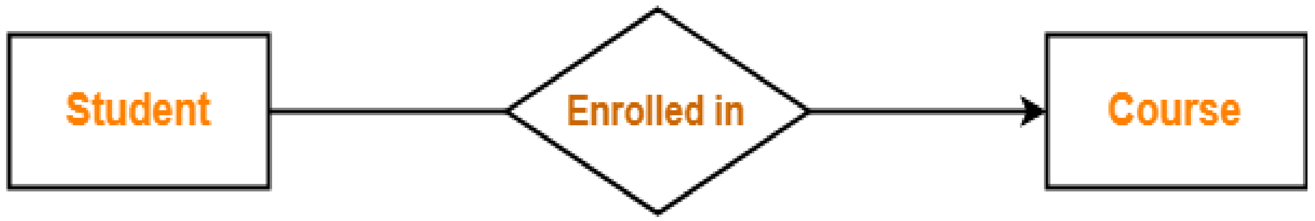


2] A relationship between person and passport is one to one because a person can have only one passport and a passport can be assigned to only one person.

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.





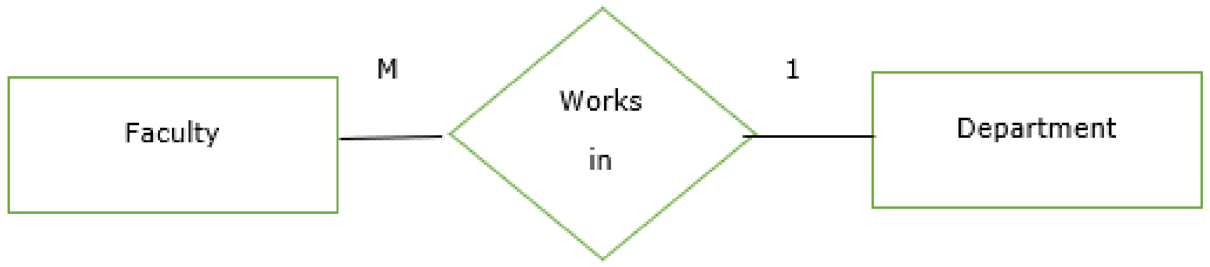
Many to One Relationship

here

- Many students can enroll in one course.

- **Example**

1] Here, many faculties work in one department.

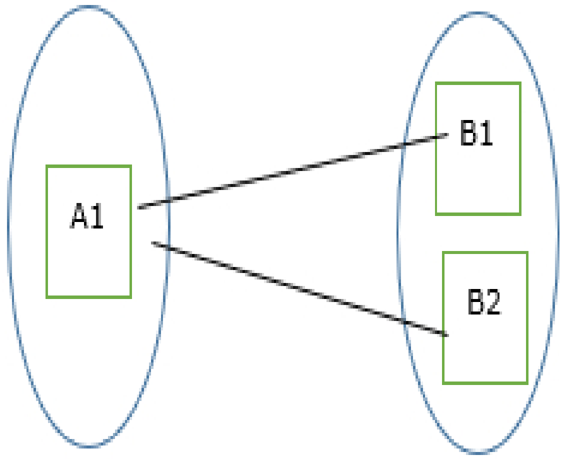


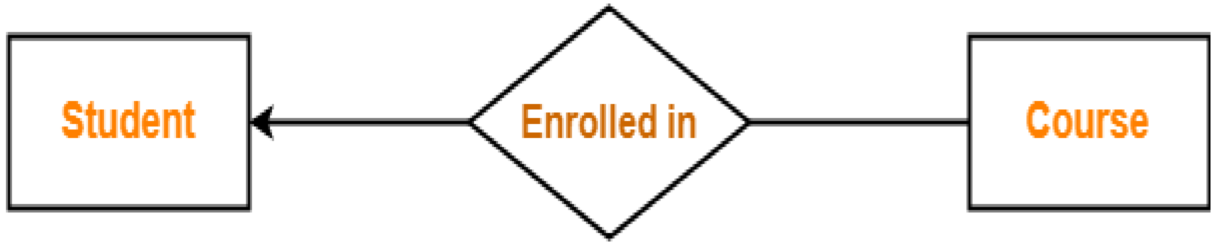
2] Relationship between student and university is many to one because a university can have many students but a student can only study only in single university at a time.

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.





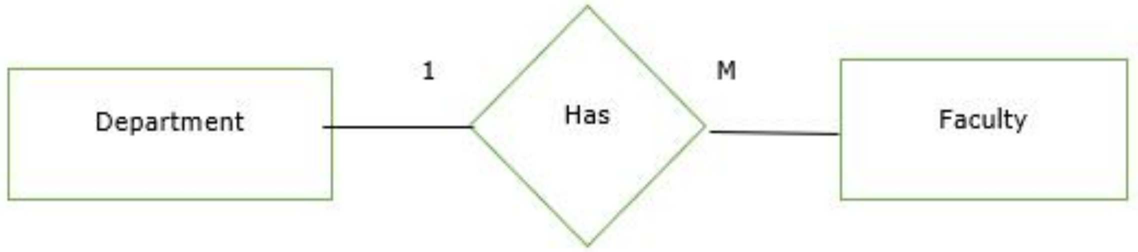
One to Many Relationship

Here,

•One student can enroll in any number of courses.

- **Example**

1] Here, one department has many faculties.

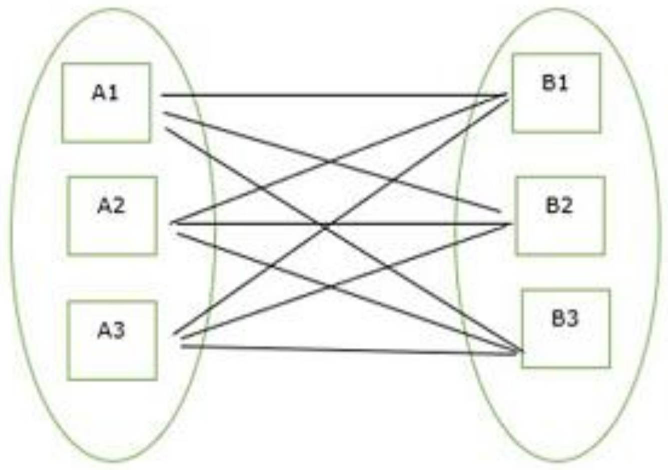


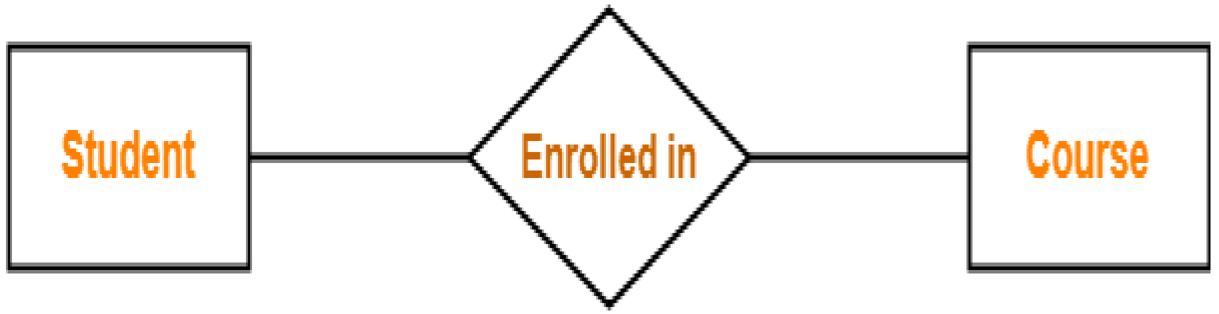
2] Relationship between customer and order is one to many because a customer can place many orders

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.



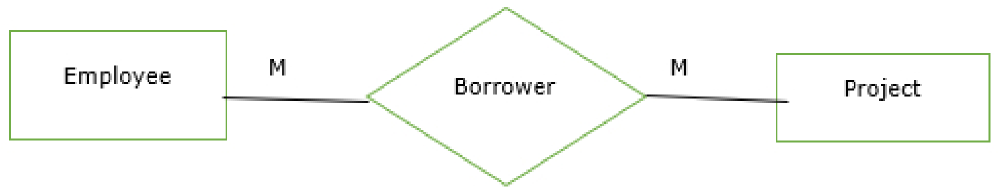


Many to Many Relationship

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

- **Example**

1] Here, many employees work on many projects.

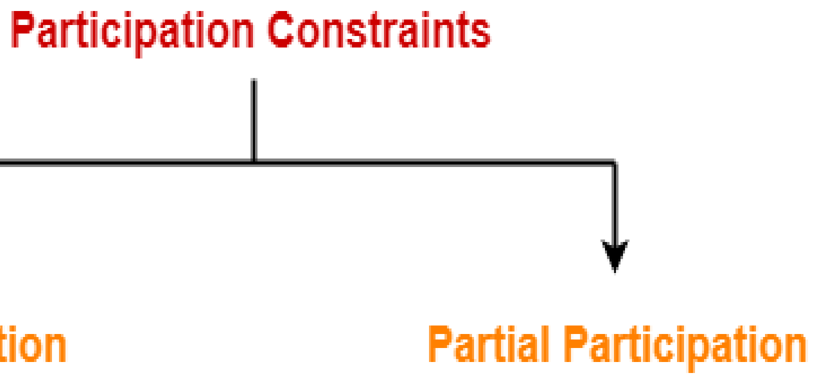


2] A many-to-many relationship exists between customers and products: customers can purchase various products, and products can be purchased by many customers.

Participation

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

- **Types of Participation Constraints-**

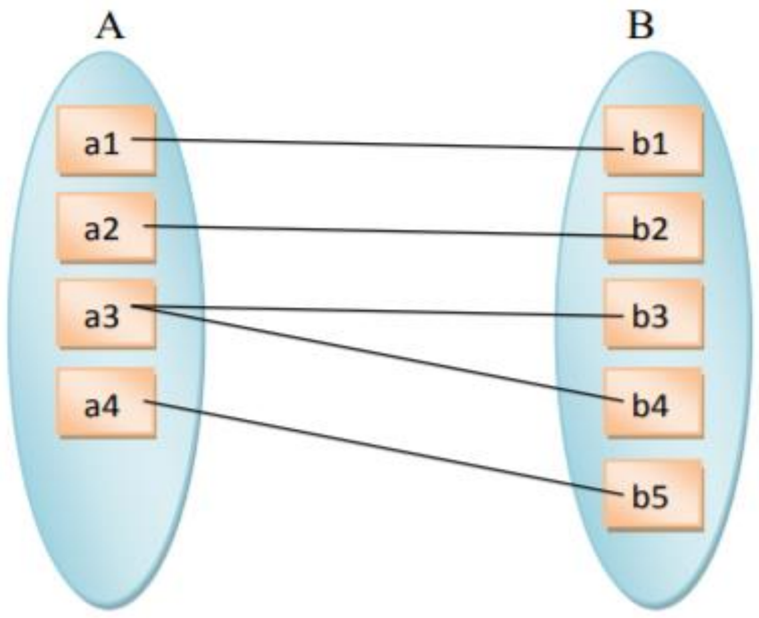


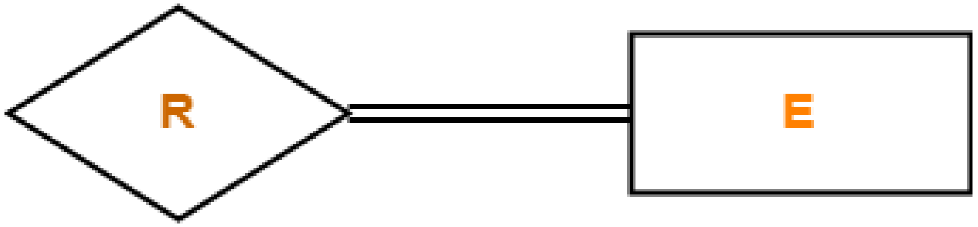
- **1. Total Participation-**
- The Participation of an entity set E in a relationship set R is said to be total if every entity in E participates in at least one relationship in R.

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

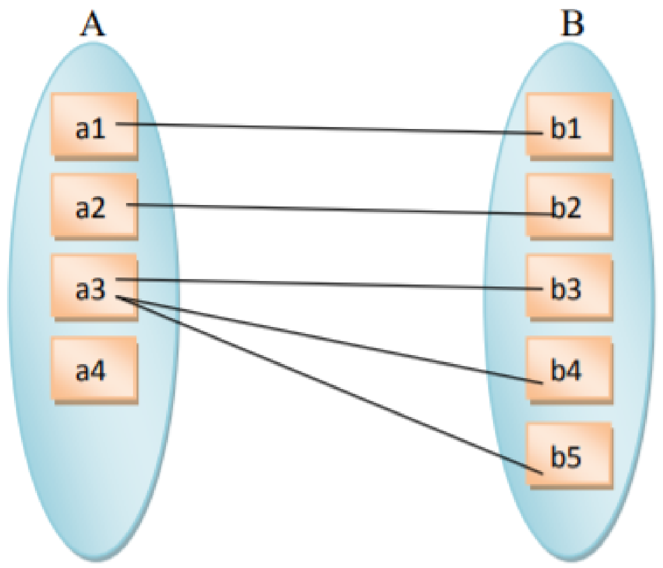
2. Partial Participation-

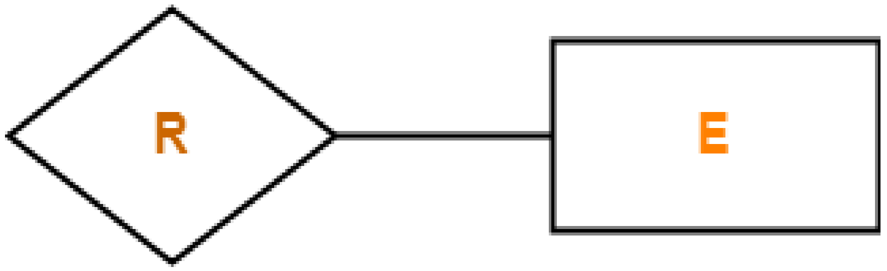
The participation of an entity set E in relationship set R is said to be partial if only some entities in E participate in relationships in R

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

while

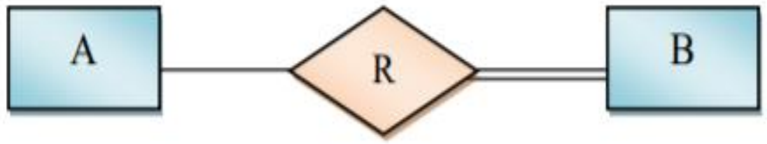
The participation of entity set B in the relationship set is **total** because every entity of B participates in the relationship set.





Partial Participation

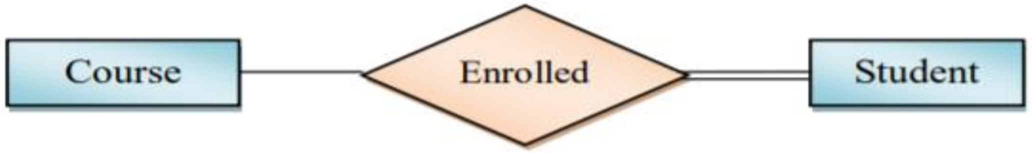
- In ERD, the total participation is denoted by doubled-line between entity set and relationship set and partial participation is denoted by single line between entity set and relationship set.



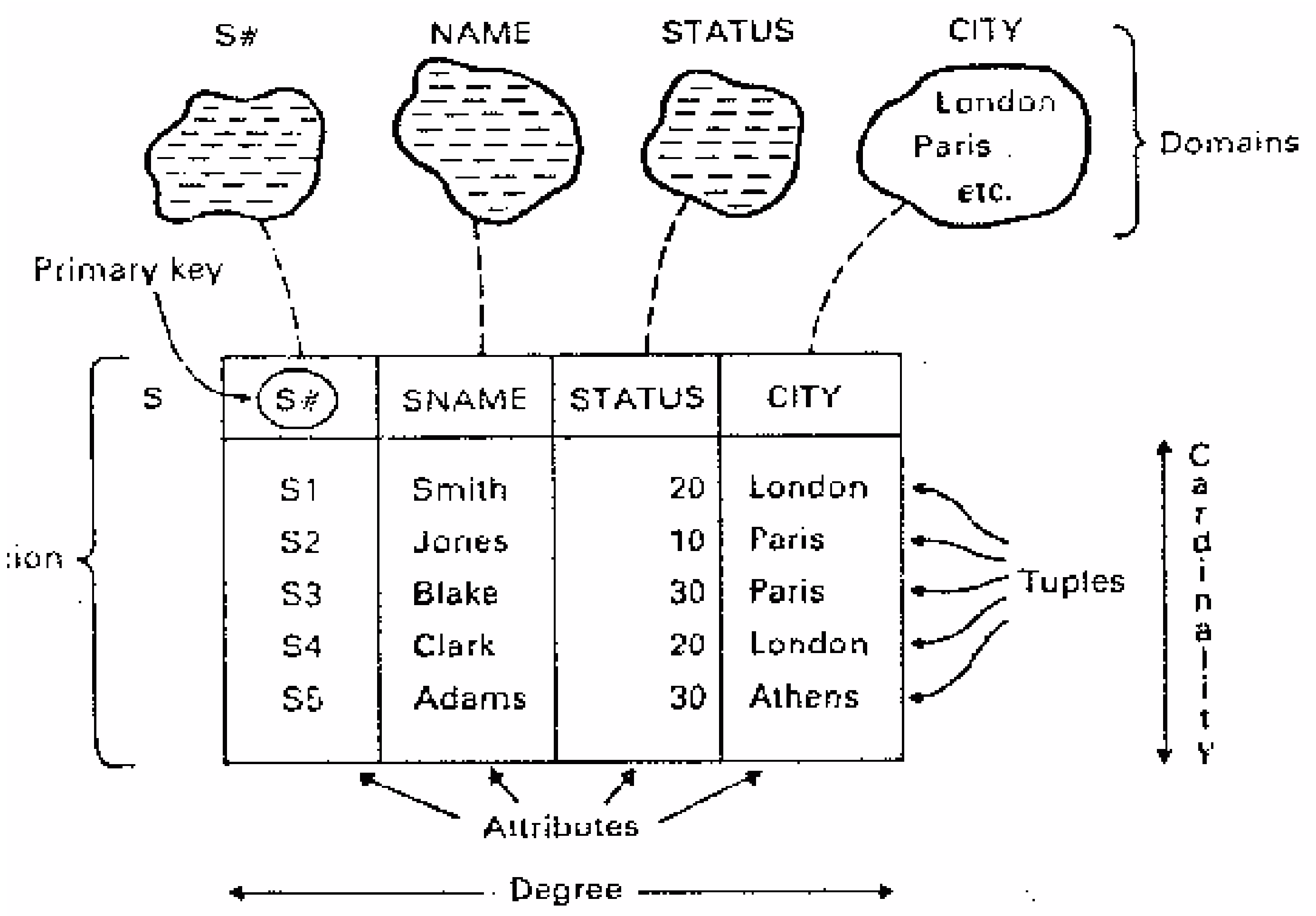
Participation in R: **Partial A**
Total B

- Example:

Suppose an entity set Student related to an entity set Course through Enrolled relationship set. The participation of entity set course in enrolled relationship set is partial because a course may or may not have students enrolled in. It is possible that only some of the course entities are related to the student entity set through the enrolled relationship set. The participation of entity set student in enrolled relationship set is total because every student is expect to relate at least one course through the enrolled relationship set



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

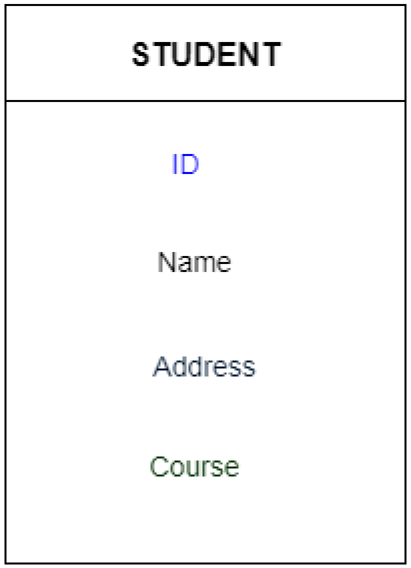


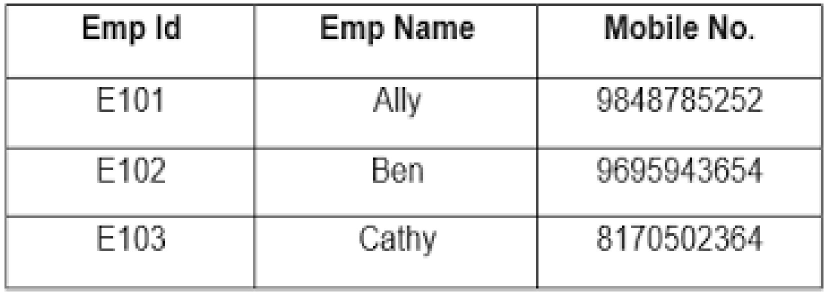
S#	SNAME	STATUS	CITY
S1	Smith	20	London
S2	Jones	10	Paris
S3	Blake	30	Paris
S4	Clark	20	London
S5	Adams	30	Athens

Cardinality = No. Of Tuples(rows)
Degree=No. Of Columns

Keys

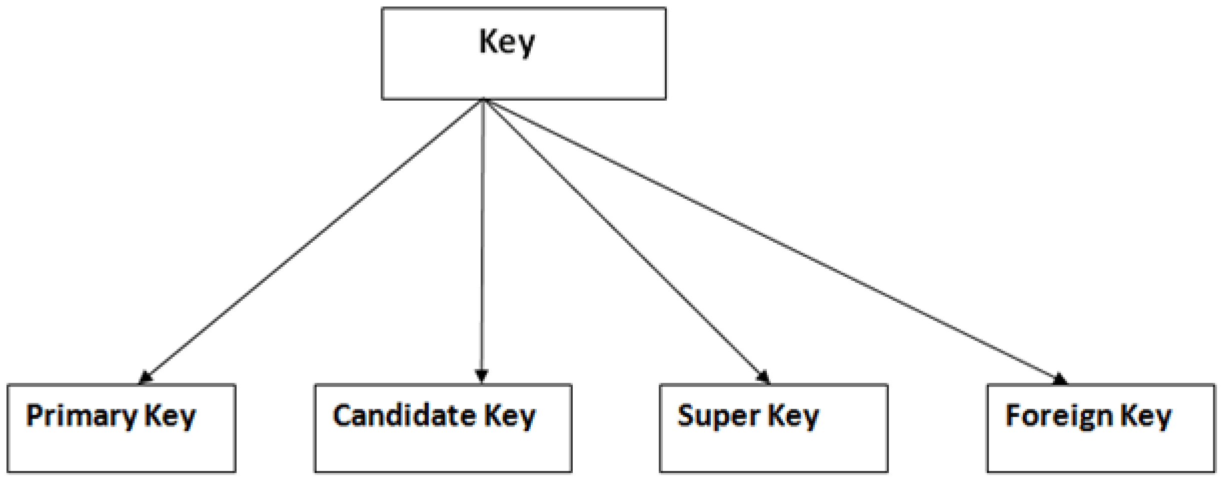
- It is used to **uniquely identify any record or row of data** from the table.
- It is also used to establish and identify relationships between tables.
- For example: In Student table, **ID** is used as a key because it is unique for each student. In PERSON table, **passport_number**, **license_number**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





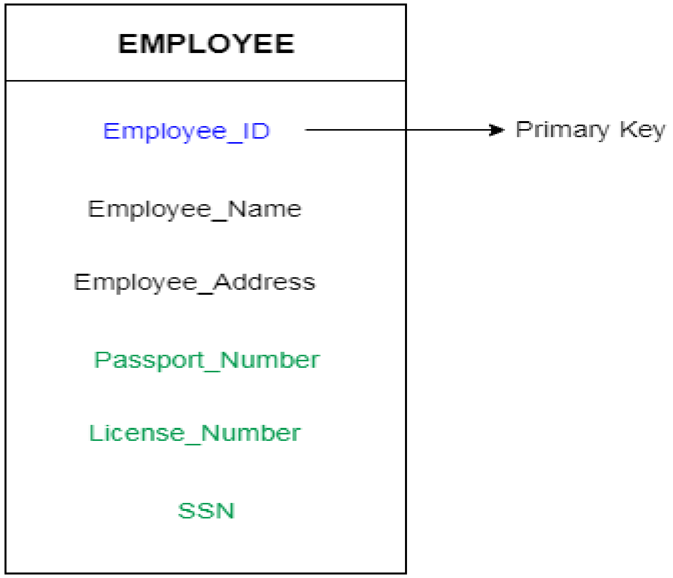
Emp Id	Emp Name	Mobile No.
E101	Ally	9848785252
E102	Ben	9695943654
E103	Cathy	8170502364

Types of key:



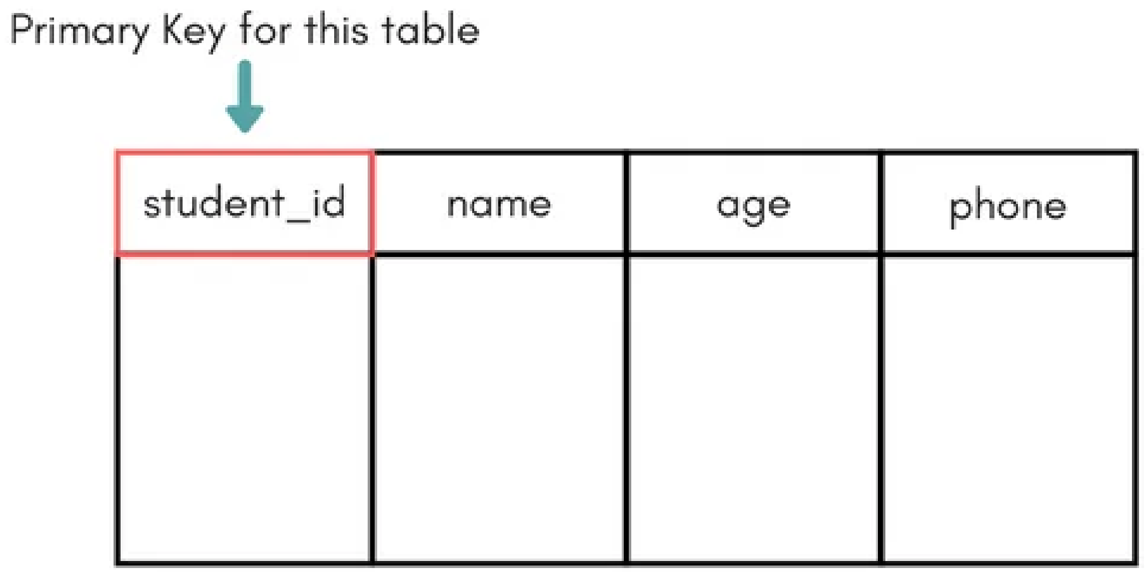
1. Primary key

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

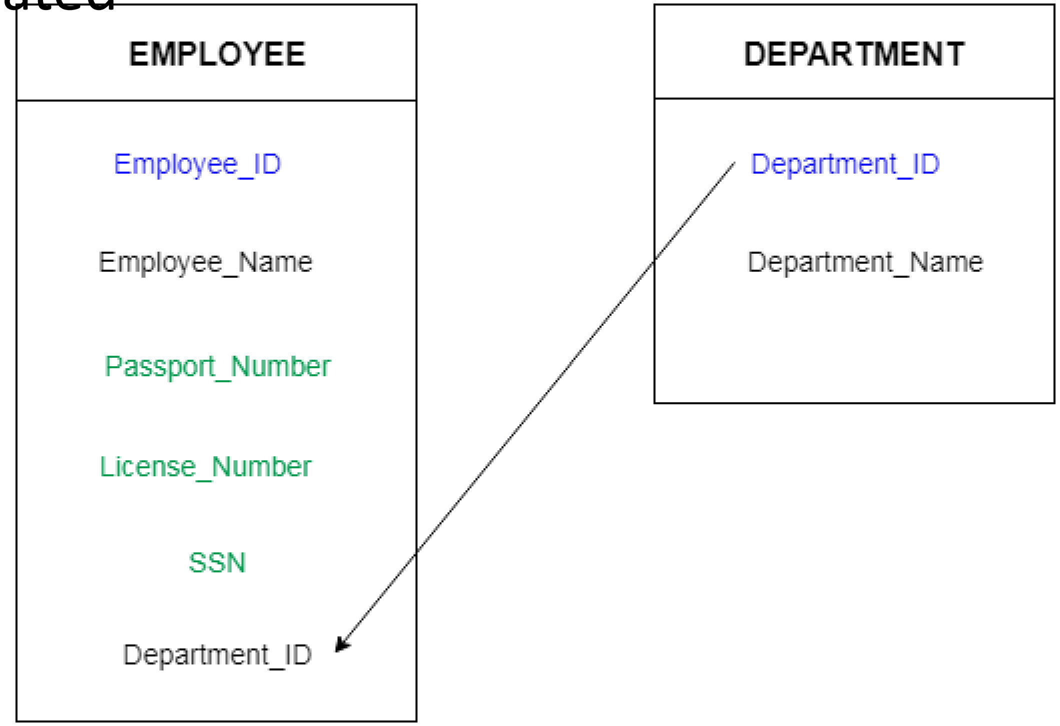
2. Foreign key

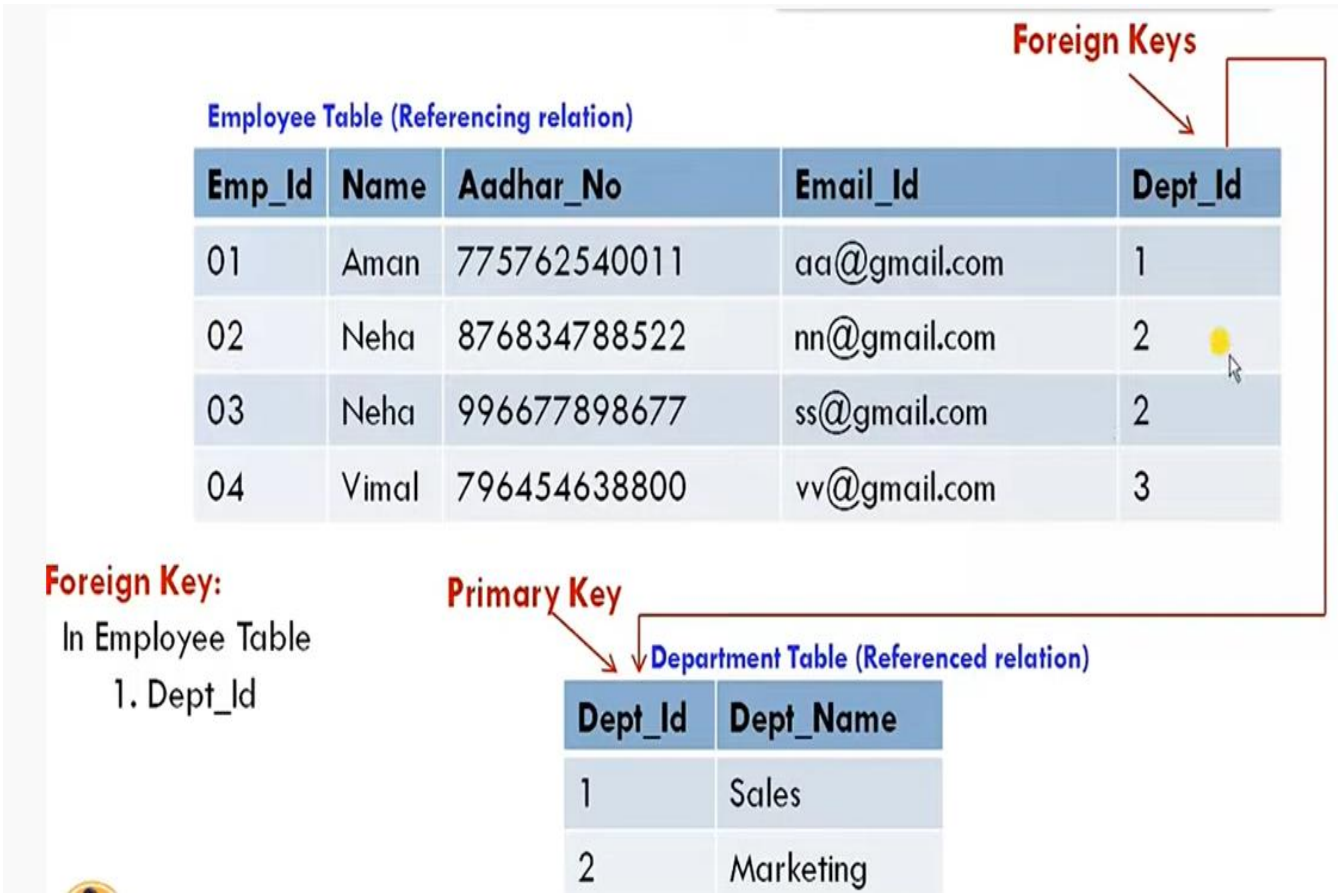
- Foreign keys are the column of the table which is used to **point to the primary key of another table**.
- A foreign key is the one that is **used to link two tables together** via the primary key. It means the columns of one table points to the primary key attribute of the other table.

In a company, every employee works in a **specific department**, and employee and department are two different entities. So we can't store the information of the department 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.

Now in the EMPLOYEE table, Department_Id is the foreign key, and both the tables are related





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

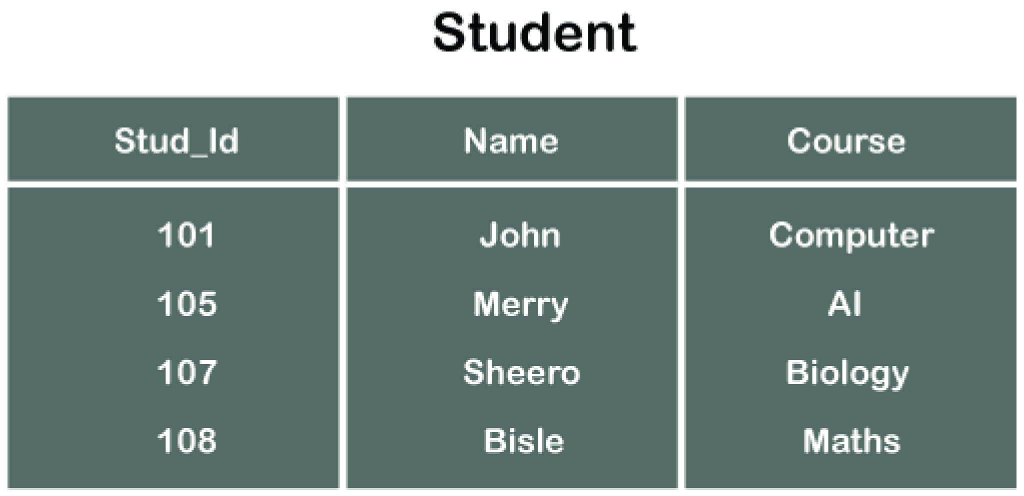
1. Dept_Id

Primary Key

Department Table (Referenced relation)

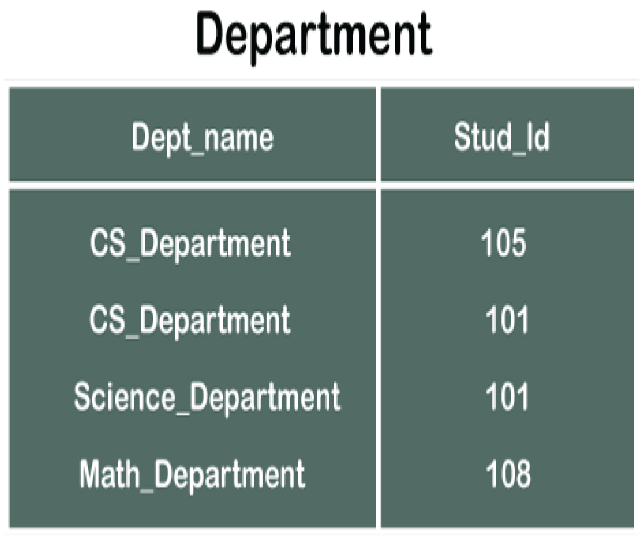
Dept_Id	Dept_Name
1	Sales
2	Marketing

- 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

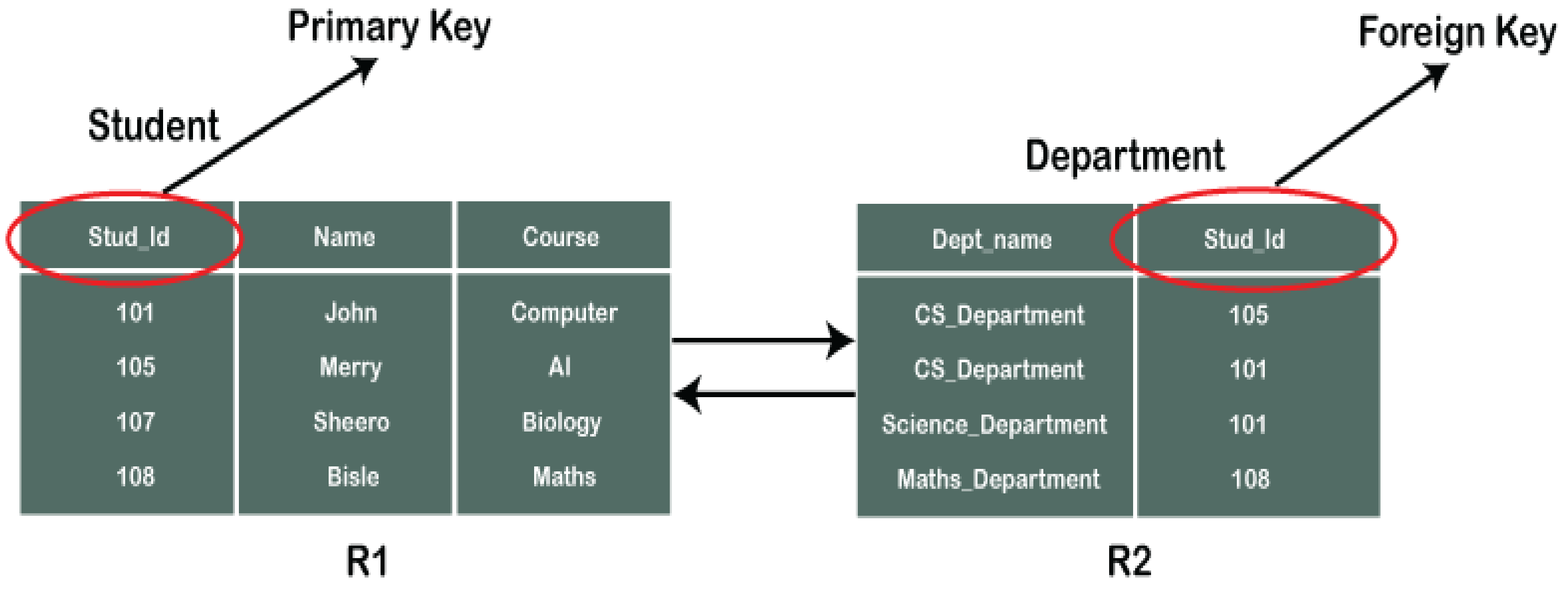


Department

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.



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

Dept_name	Stud_Id
CS_Department	105
CS_Department	101
Science_Department	101
Maths_Department	108

3. 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.

A super key is a set of one or more attributes (columns) that can uniquely identify a tuple (row) in a relation (table).

It is a broader concept and can include attributes that are not strictly necessary for uniqueness.

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

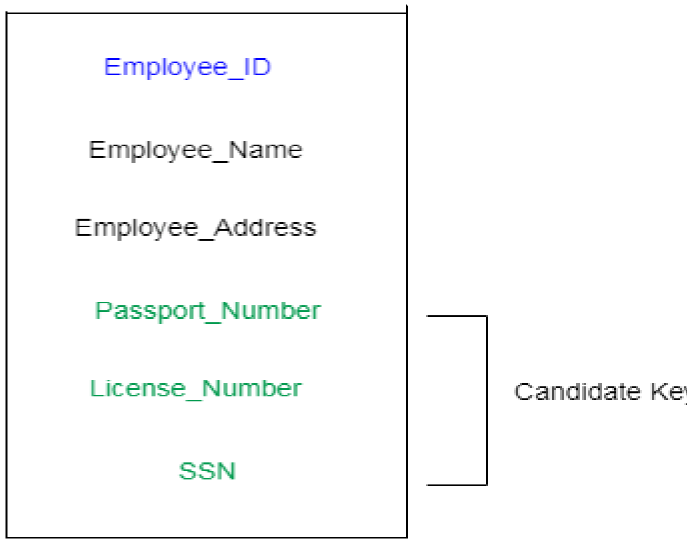
4. Candidate key

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.

A candidate key is a minimal super key, meaning it is a super key with the property that if any attribute is removed from the key, it will no longer uniquely identify a tuple.

Candidate keys are the smallest set of attributes that can uniquely identify a tuple in a relation.

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 candidate 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

5. Alternate Key

A table can have multiple choices for a primary key; however, it can choose only one. So, all the keys which did not become the primary Key are called alternate keys.

E.g. In student table we can take **aadhar no.** as alternate key

Homework

Difference between different keys in database with example

Quiz Time

1. An Entity set that does not have sufficient attributes to form a primary key is a ---------?
2. In relational model relations are termed as --------?
3. In relational model Cardinality is termed as ------?
4. Set of All entities having same attribute is called as -----??
5. The set of Values which specifies which values are to be assigned to individual entities are considered as-----??

Answers

1. An Entity set that does not have sufficient attributes to form a primary key is a Weak Entity Set
2. In relational model relations are termed as Tables?
3. In relational model Cardinality is termed as Number of tuples?
4. Set of All entities having same attribute is called as Entity ??
5. The set of Values which specifies which values are to be assigned to individual entities are considered as Domain of Values??

Relationship Types

A relationship type **R** among n entity types **E1,E2,….En** defines a **set of associations** or a relationship set **among entities** from these entity types

OR

The **association** between **two different entities** is called as **relationship.**

For example: An employee works at a department ,a student enrols for a course .So here, works at and enrolls are called relationships.

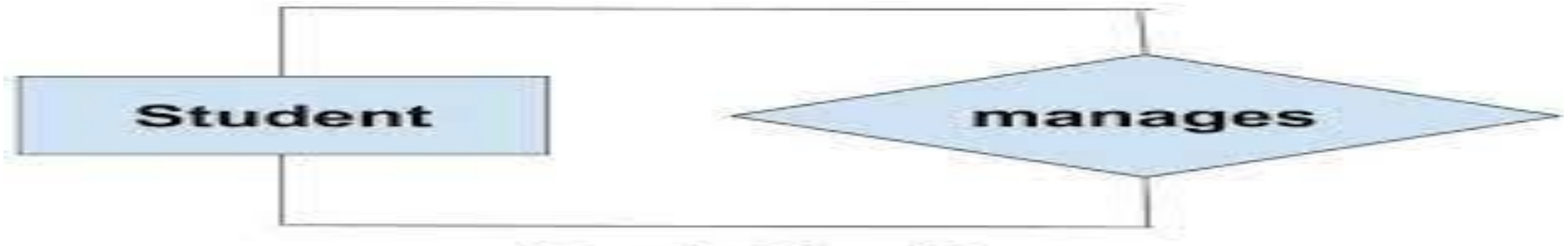
Degree of Relationship:

1. Unary Relationship
2. Binary Relationship
3. Ternary Relationship
4. Quaternary Relationship(N-ary)

Relationship Types(cont..)

Degree of Relationship: The **degree of relationship** refers to **number of entities** participated in the relation. Based on the **number of linked entity types**, we have 4 types of degrees of relationships.

1. **Unary Relationship (Degree1)-** In a relation **only one entity set** is participating then such type of relationship is known as a unary relationship.

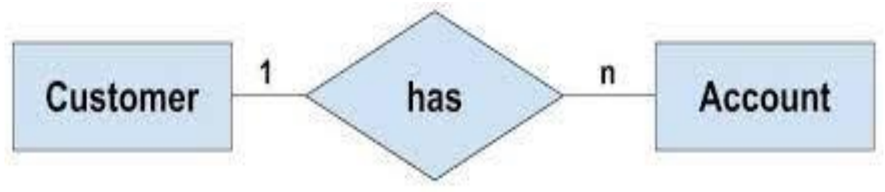


Unary Relationship
(degree 1)

Relationship Types(Cont..)

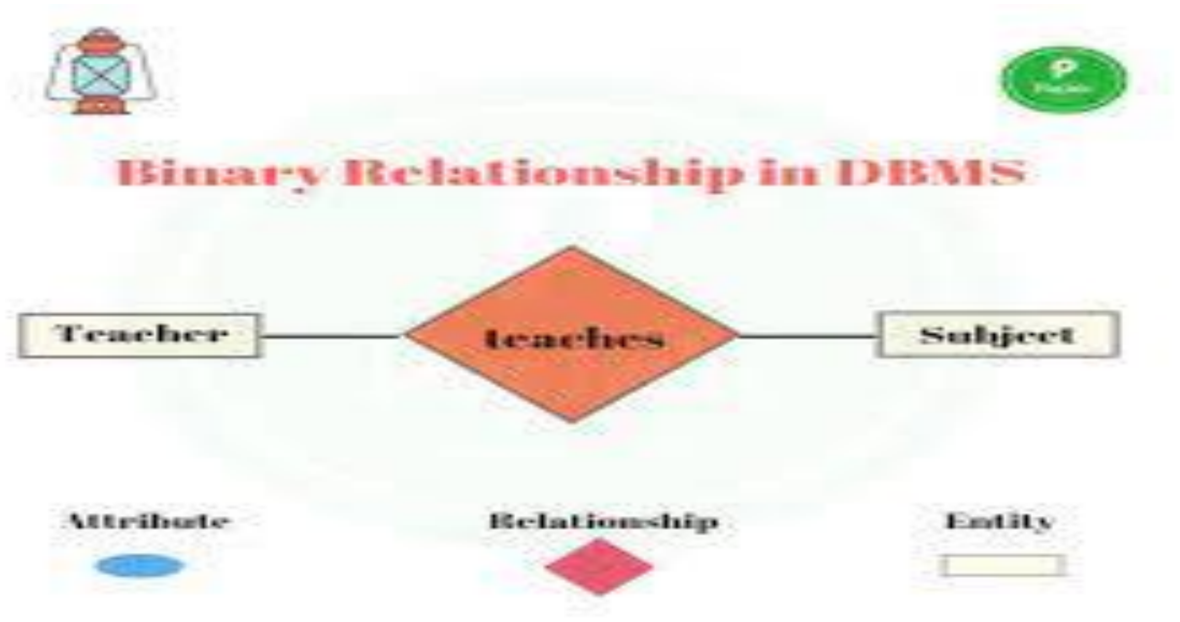
2. Binary Relationship(Degree 2)

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** Or in other words, in a relation when two entity sets are participating then such type of relationship is known as a binary relationship.



Binary Relationship
(degree 2)

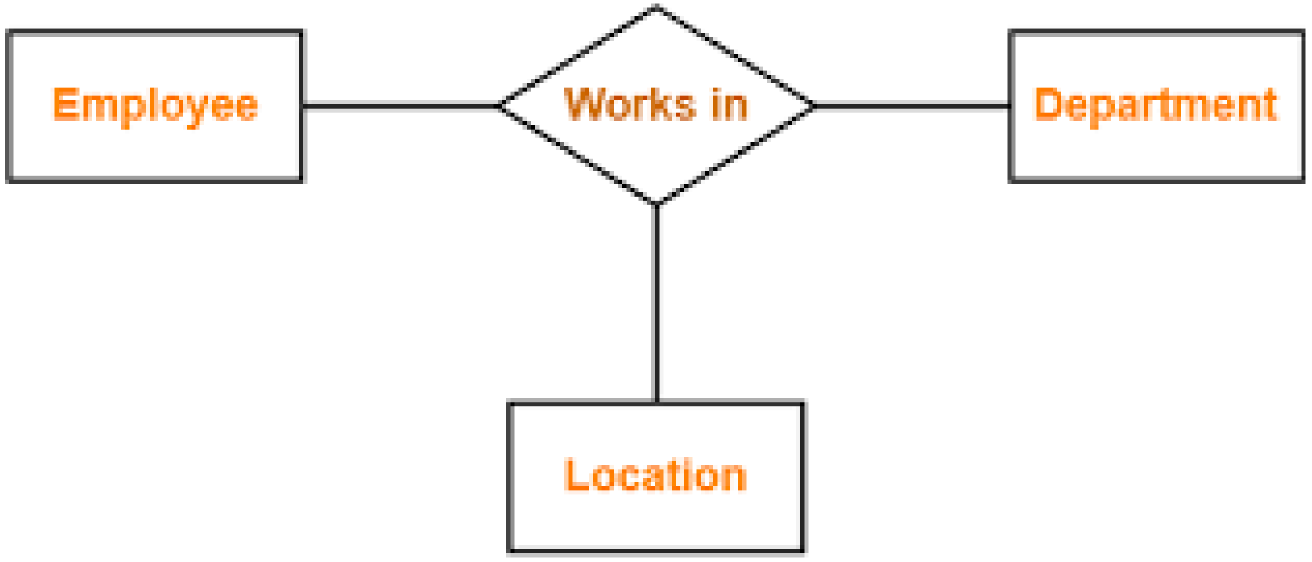
Relationship Types(cont..)



Relationship Types(cont..)

3. Ternary Relationship(Degree 3)

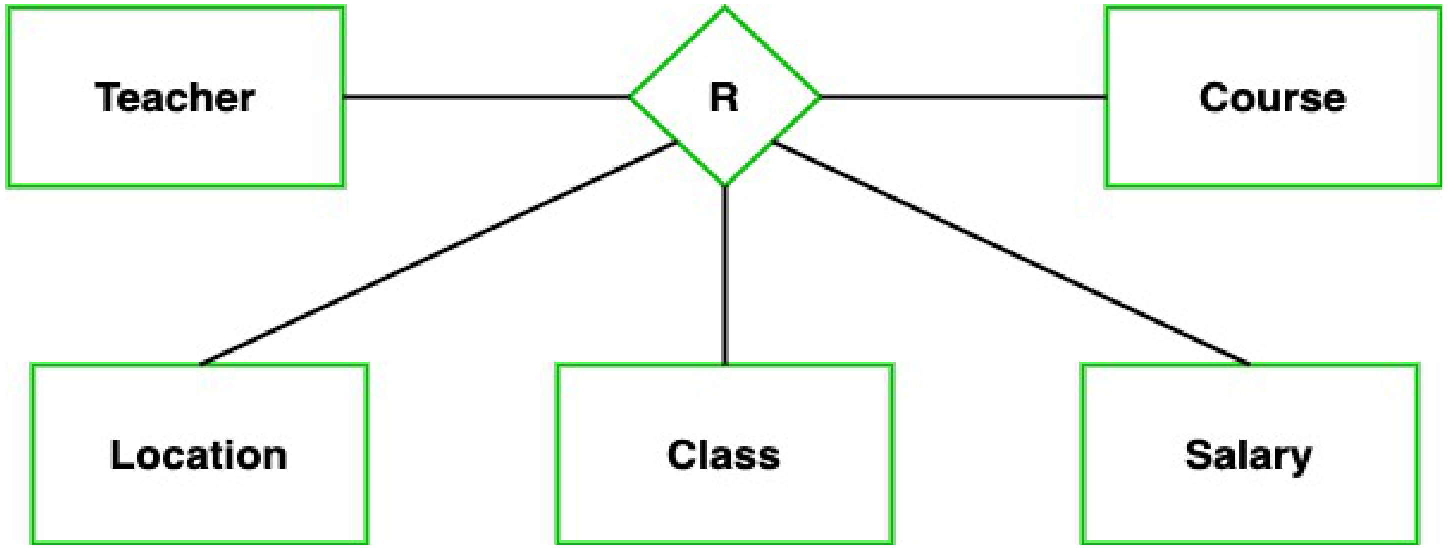
In the Ternary relationship, there are **three types of entity associates.** So, we can say that a Ternary relationship exists when there are three types of entity and we call them **a degree of relationship is 3.**



Ternary Relationship Set

Relationship Types(cont..)

4. Quaternary Relationship - 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.



Extended Entity-Relationship (EE-R) Model

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

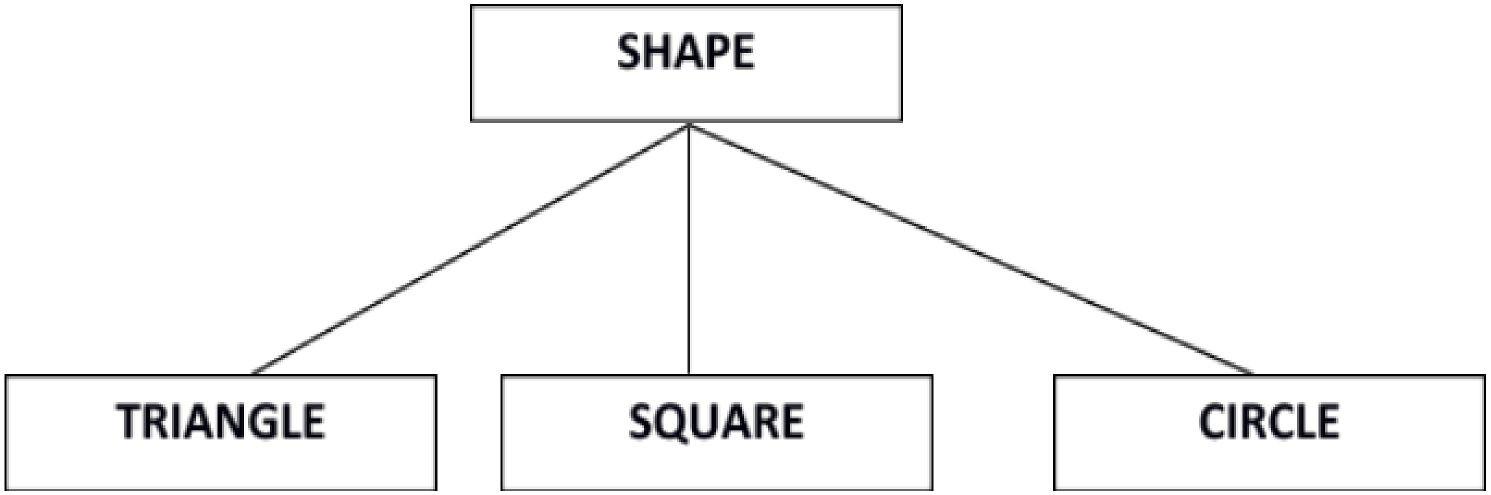
In addition to ER model concepts EE-R includes –

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

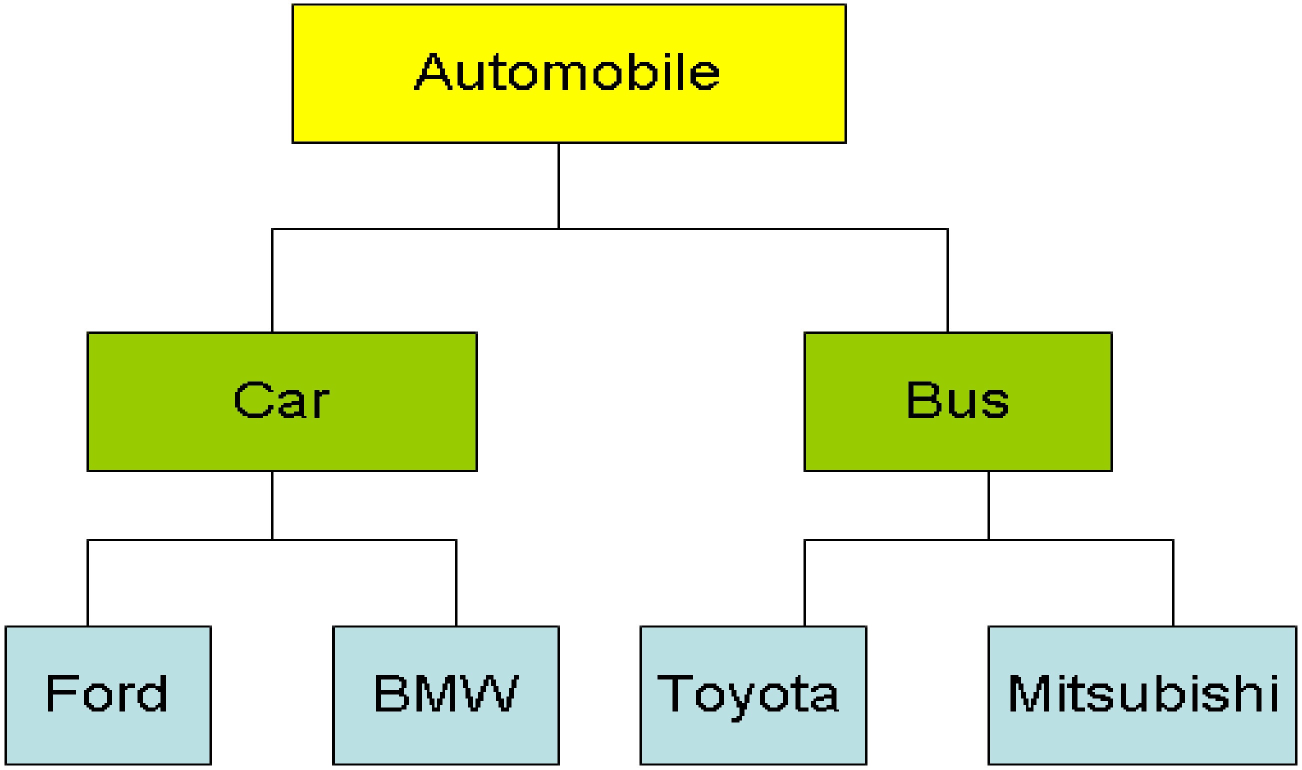
These concepts are used to create EE-R diagrams.

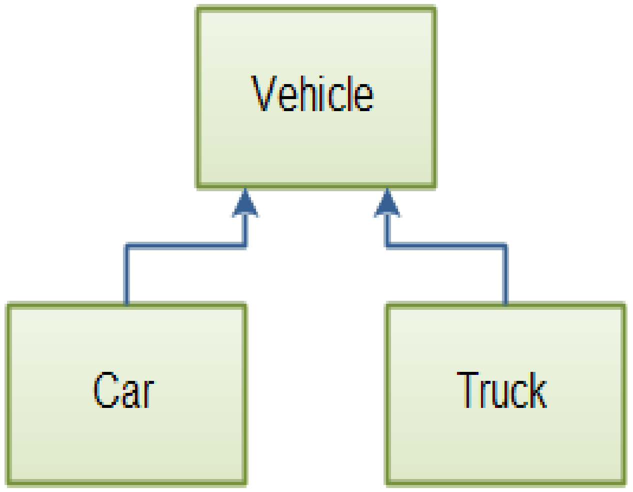
Subclasses and Super class

- 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



- 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.

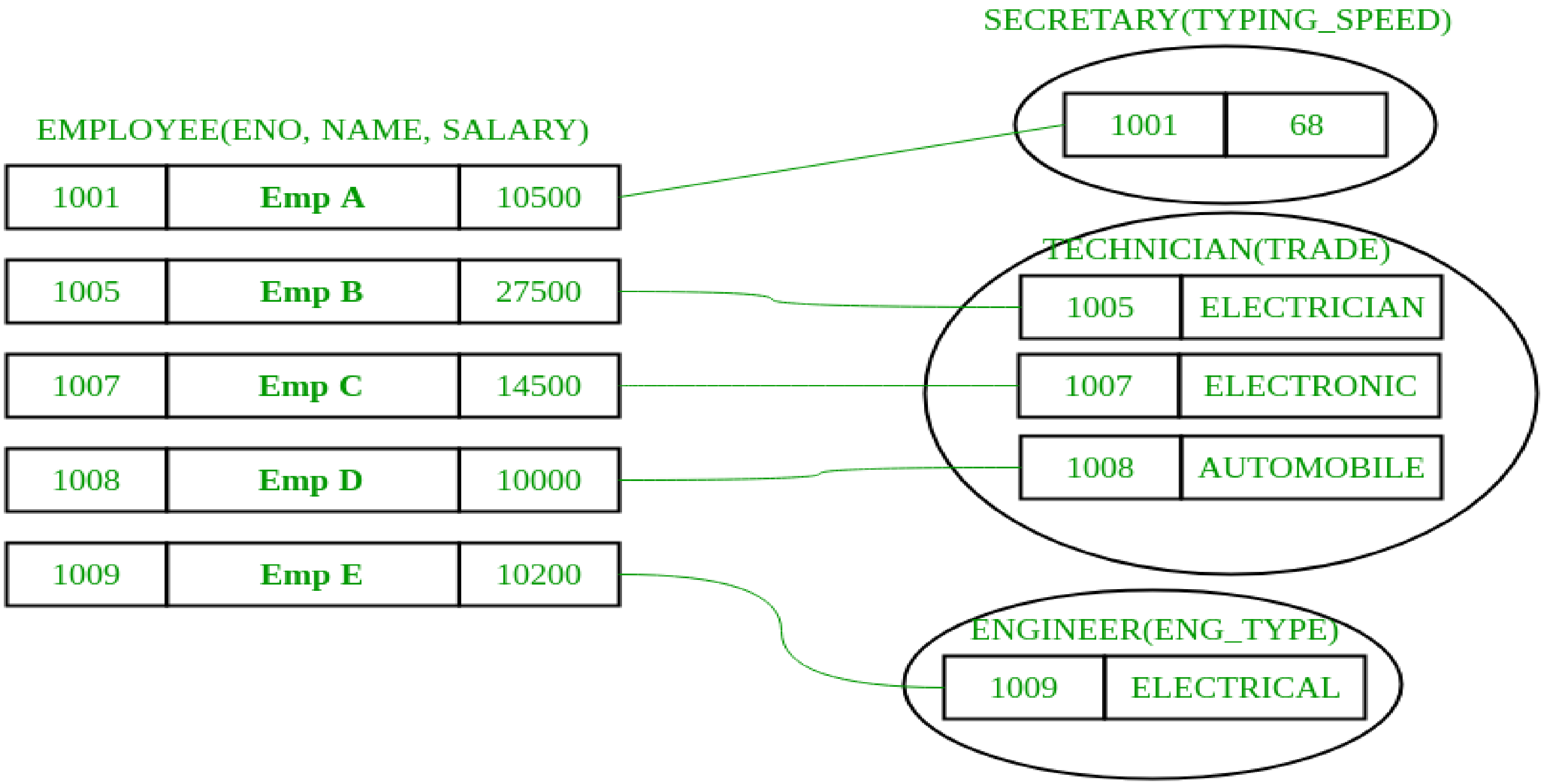



Vehicle
Car
Truck

- 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.



Generalization, Specialization and Aggregation in ER model are **used for data abstraction in which abstraction mechanism is used to hide details of a set of objects**

Specialization

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

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

- 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:

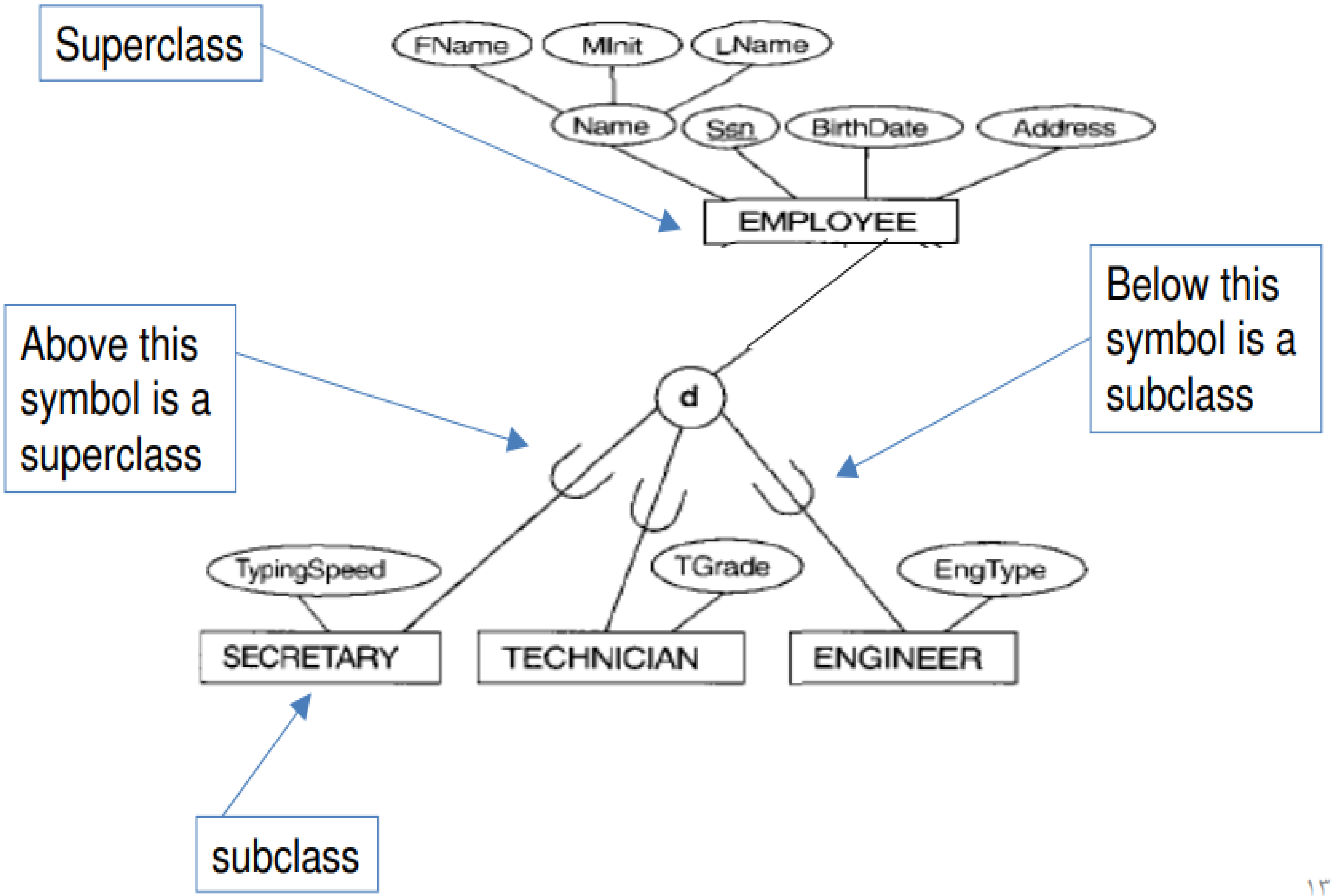
1) {SalariedEmployee,HourlyEmployee}
   - Classified based on paying method.

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

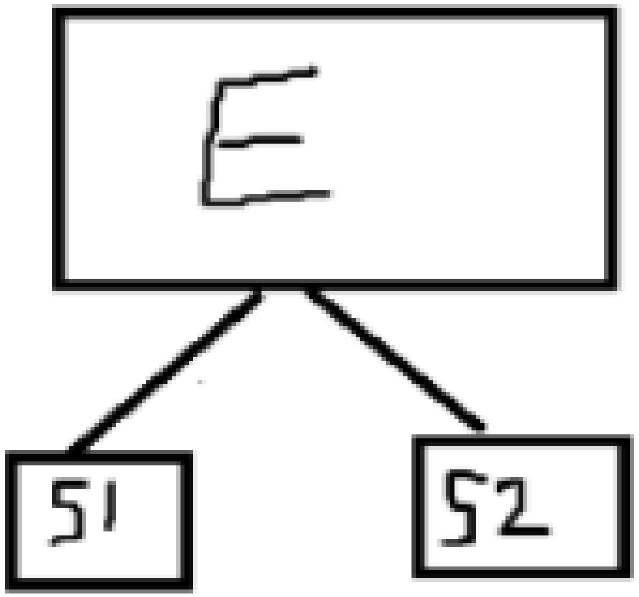
How specialization is represented in EER: –

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
d
Below this symbol is a subclass
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.

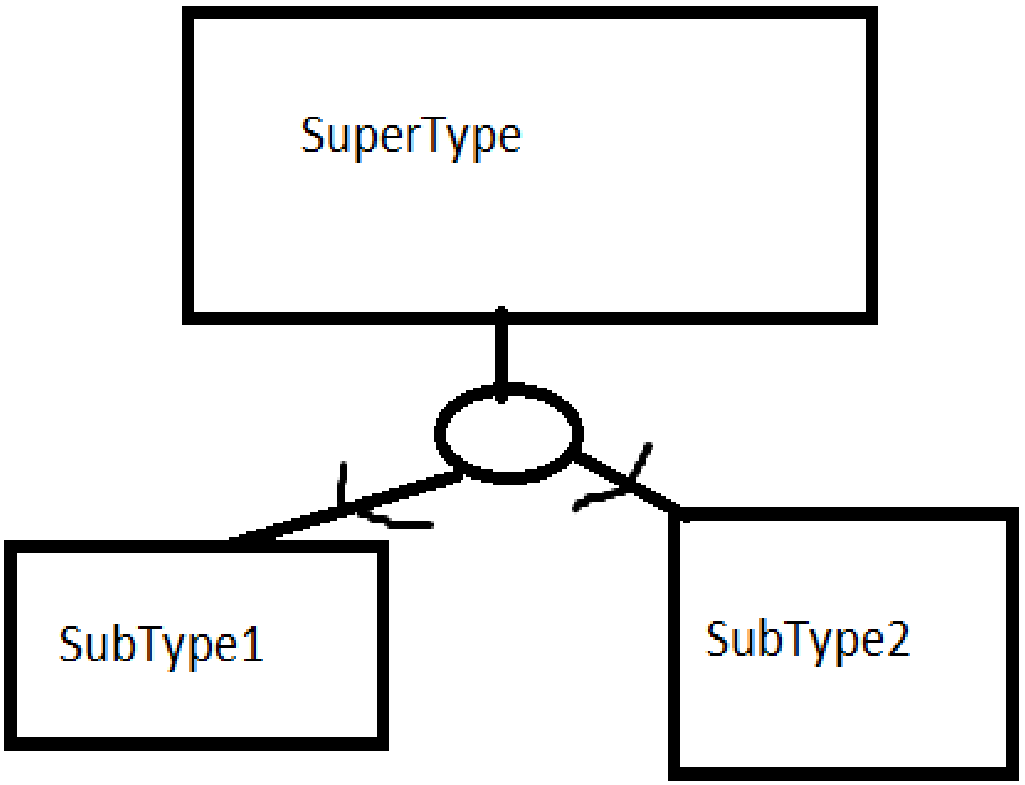
Totally Seperated.

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.



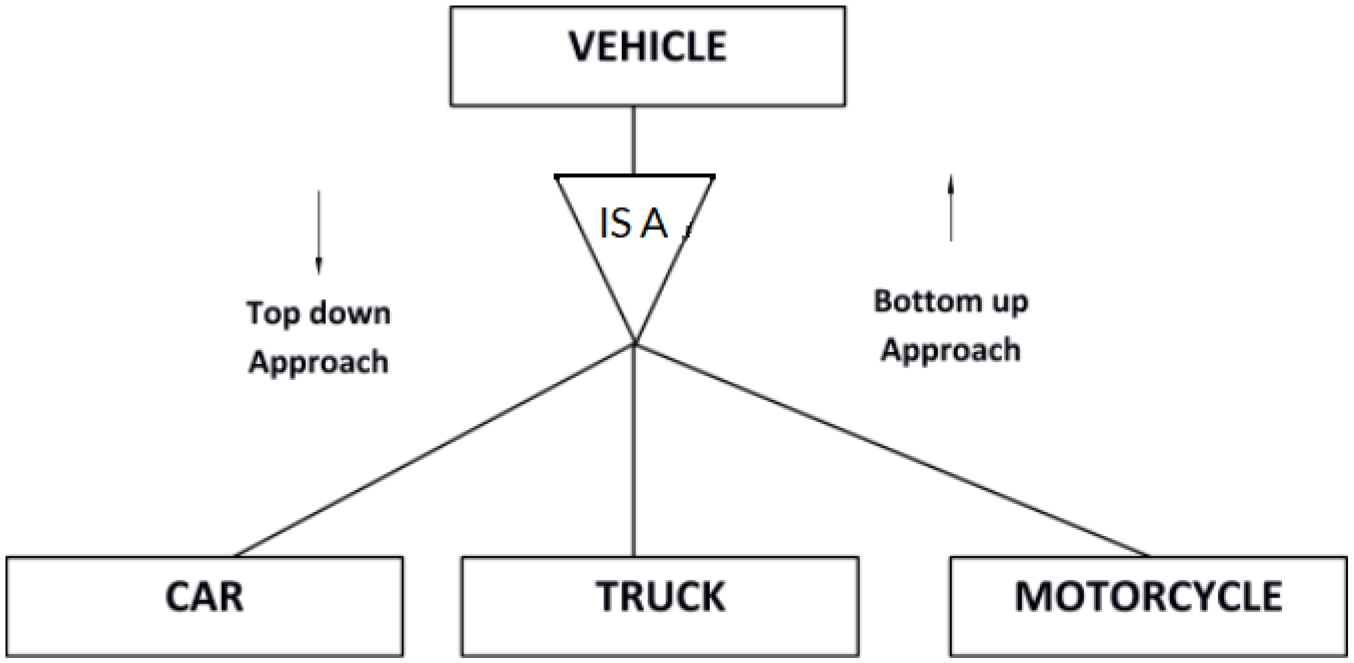
In summary, specialization allows us to: –

1. Define subclasses of entity types.
2. Define specific attributes for subclasses.
3. Define specific relationships between subclasses and other entities or subclasses.
4. Specialization is a Top-Down approach

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).

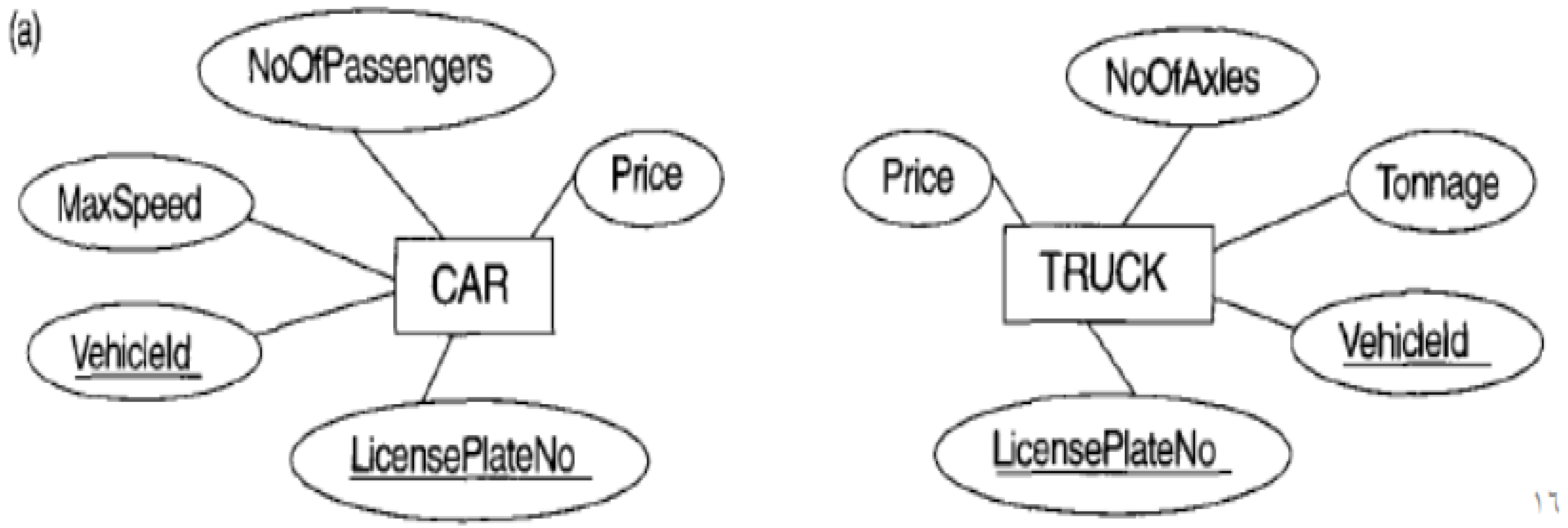
Generalization is a Bottom up process i.e. consider we have 3 sub entities Car, Truck and Motorcycle. Now these three entities can be generalized into one super class named as Vehicle



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

- Car
- Truck

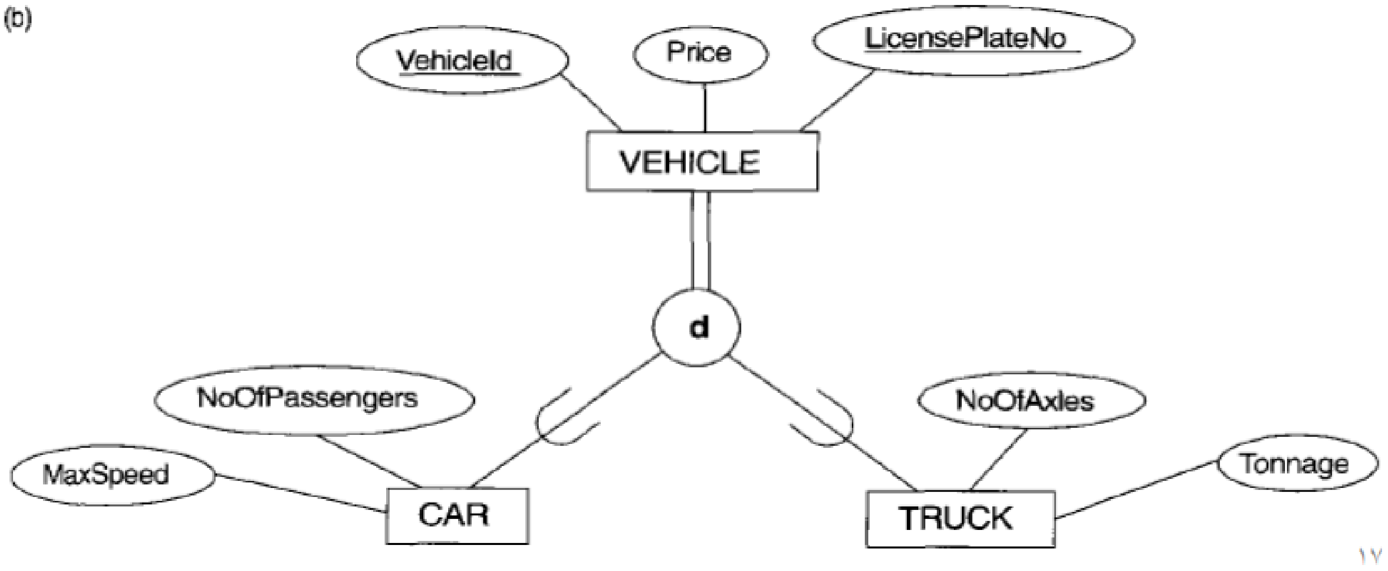
How can we generalize this?


(a)
NoOfPassengers
MaxSpeed
Price
CAR
VehicleId
LicensePlateNo
NoOfAxles
Price
Tonnage
TRUCK
VehicleId
LicensePlateNo
١٦

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

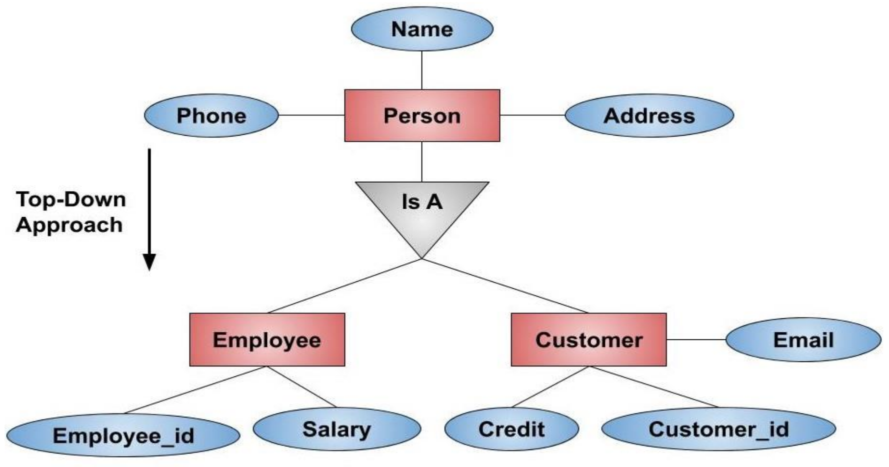
- An Vehicle must either be car or truck-total participation

(b)

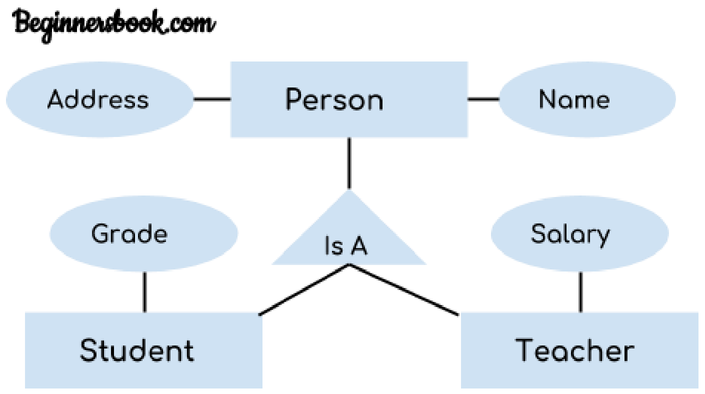


- Generalization is the **process of extracting common properties** from a set of entities and create a generalized entity from it.

- 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

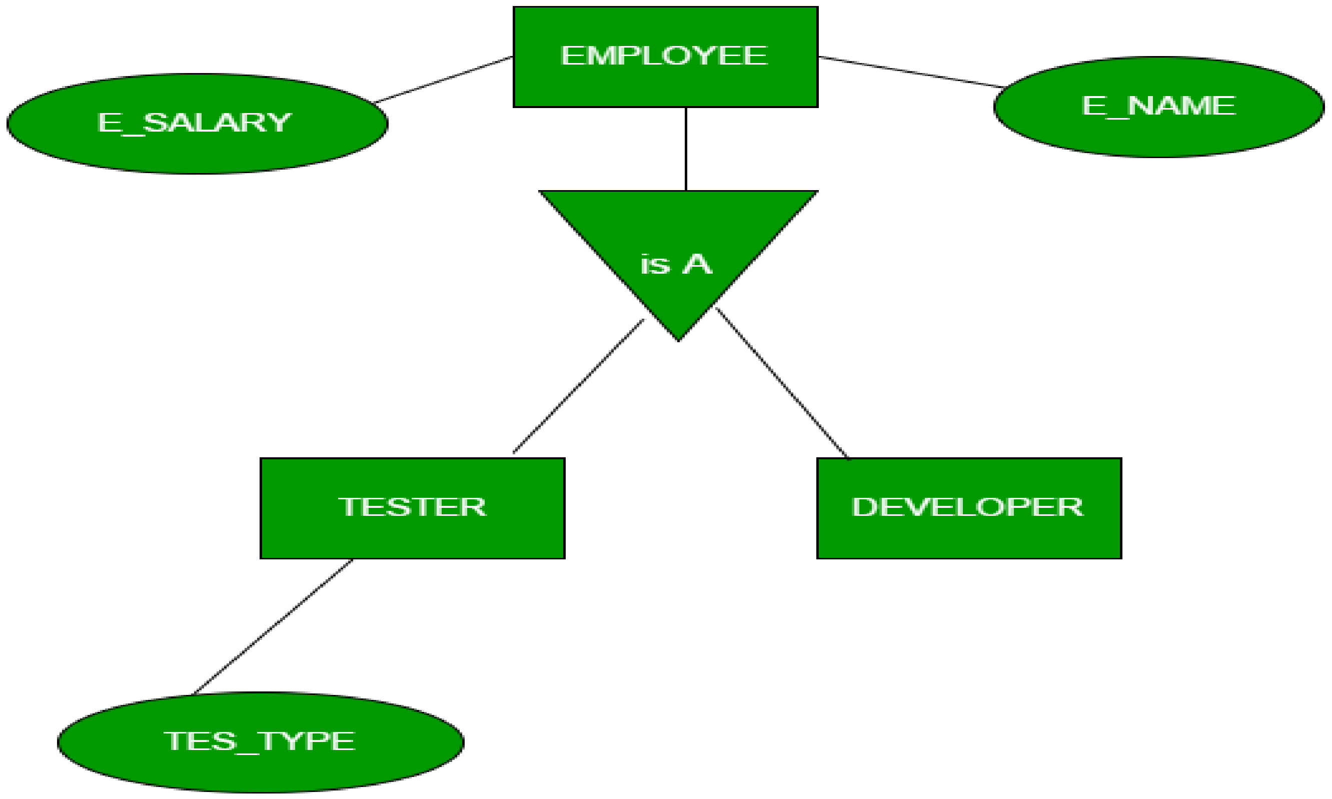


Generalization

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).

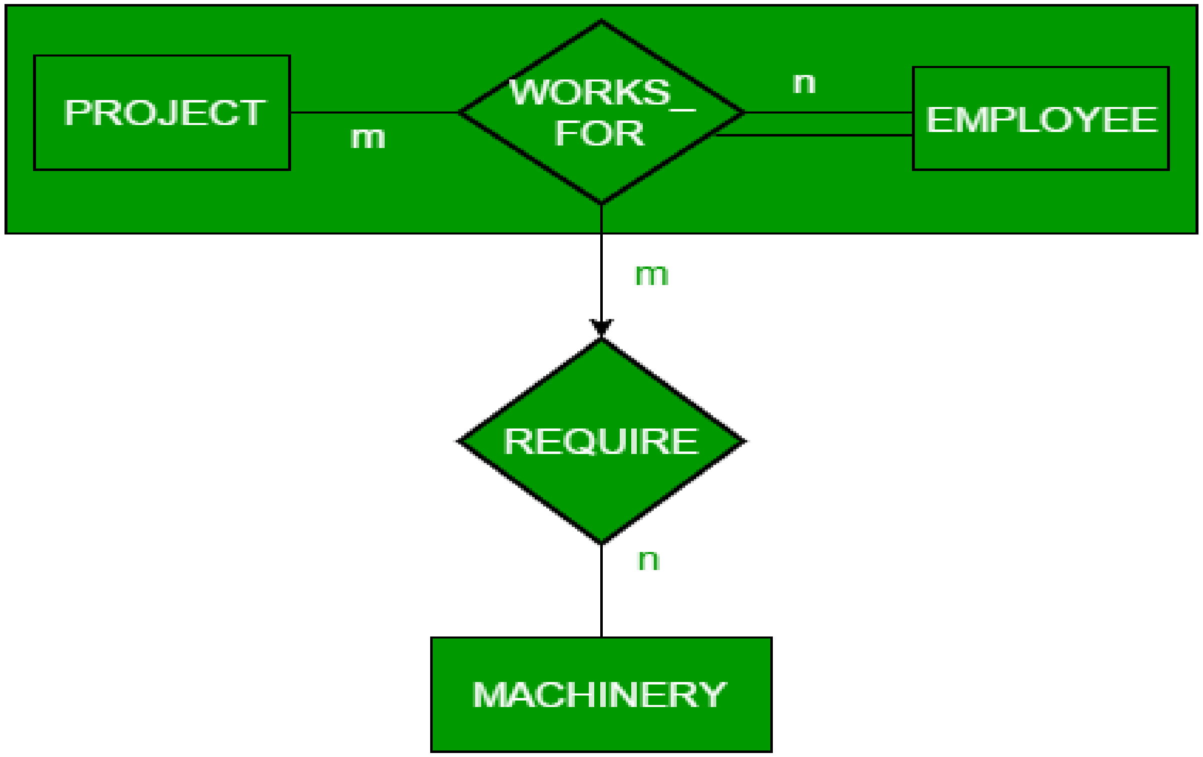


Specialization

Aggregation

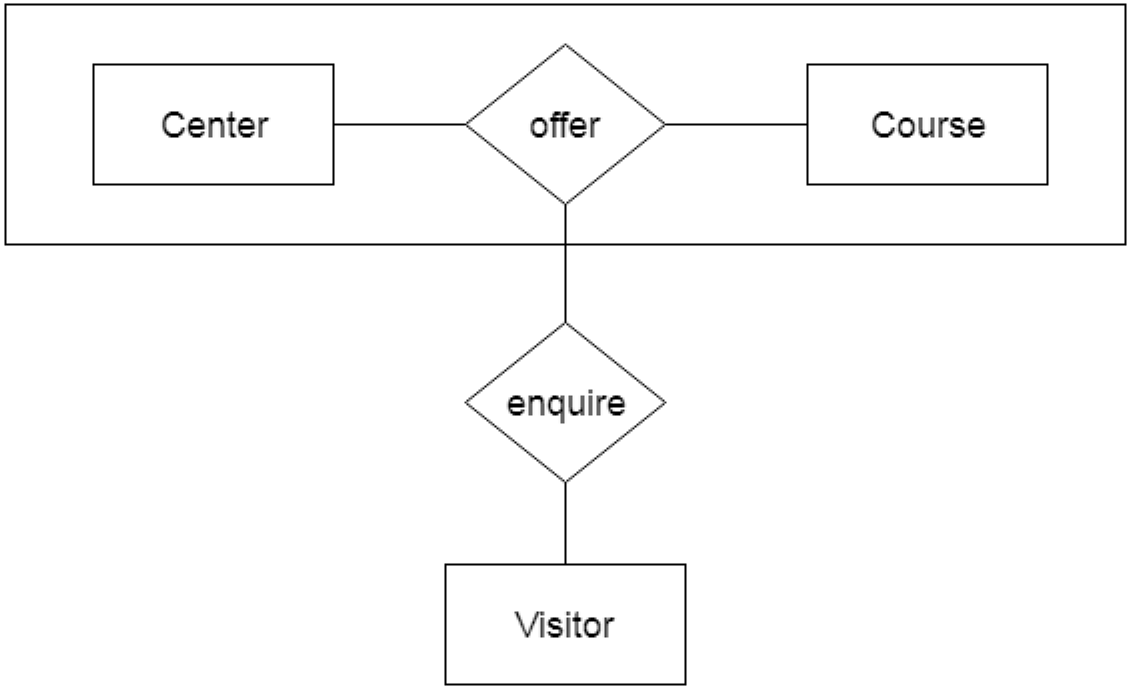
- An ER diagram is not capable of representing relationship between an entity and a relationship which may be required in some scenarios.
- In those cases, a relationship with its corresponding entities is aggregated into a higher level entity.
- Aggregation is an abstraction through which we can represent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

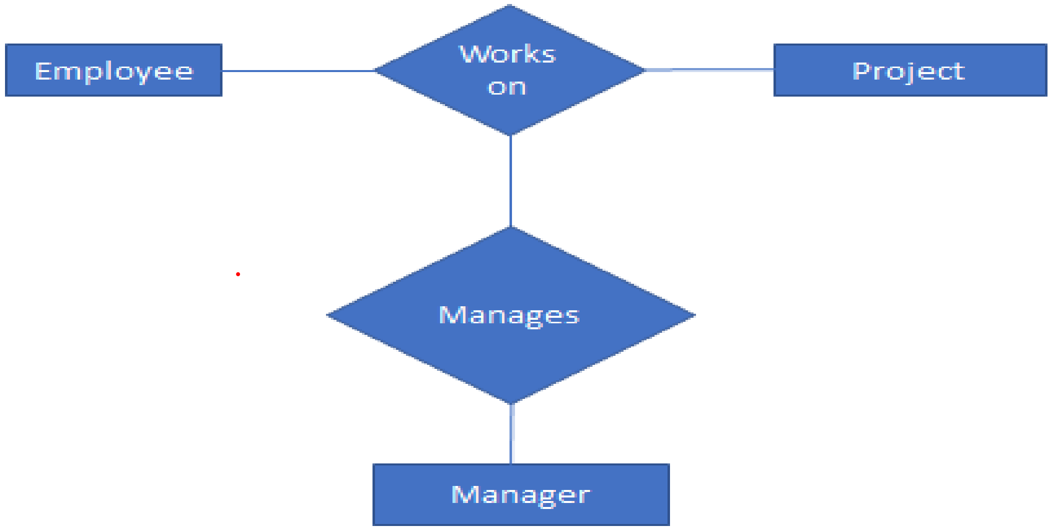


Aggregation

- **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”.



Quiz Time

1. The generalization process is similar to the ----------------

A. Top-Bottom
B. Top-Up
C. Bottom-Up
D. Up-Bottom

2. When two or more lower-level entities share some attributes, they lead to a ___. This is often referred to as a bottom-up approach.

A. Lower-level entity
B. Higher-level entity
C. Middle-level entity
D. Center-level entity

3. As a general rule, entities at higher levels can combine with entities at lower levels to form a ___ level entity.

1.Lower
2.Higher
3.Middle
4.Central

HW Questions

1. Construct an E-R diagram forCompany Database.
2. Construct an E-R diagram for a hospital with a set of patients and a set of medical doctors. Associate with each patient a log of the various tests and examinations conducted.

Quiz Time

4. There are more ___ in generalization.
1.Subclasses
2.Superclasses
3.Both a and b
4.None of the above

5. Specialization is a –
1.Bottom-Up
2.Top-Up
3.Top-Down
4.Bottom-Top

6. Several lower-level entities can be decomposed into one higher-level entity in ___.
1.Generalization
2.Specialization
3.Both A. and B.
4.None of the above

Thank you…