

# Chapter-2

## Entity– Relationship Data Model

Dr. Jyoti Wadmare

# Contents

- The Entity-Relationship (ER) Model: Entity types: Weak and strong entity sets,
- Entity sets
- Types of Attributes
- Keys
- Relationship constraints: Cardinality and Participation
- Extended Entity-Relationship (EER) Model: Generalization, Specialization and Aggregation

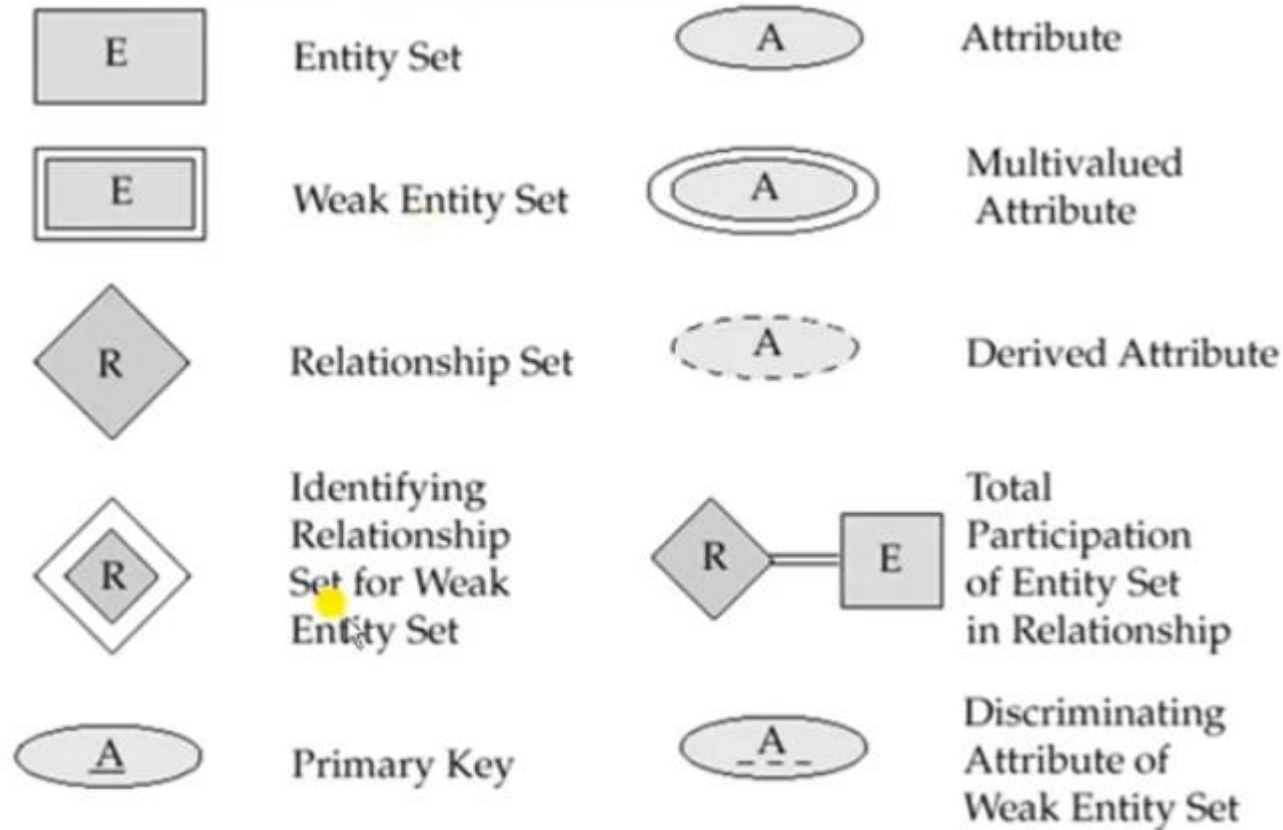
# Entity Relationship model

- ER model was introduced by Peter Chenn in 1976
- The **ER model** defines the **conceptual (or logical) view of a database**.
- It is used for **designing database**.
- It works around **real-world entities** and the **relationships among them**
- A **database schema** in the ER Model can be represented pictorially as **ER diagrams**
- An ER diagram maps well into a relational schema.

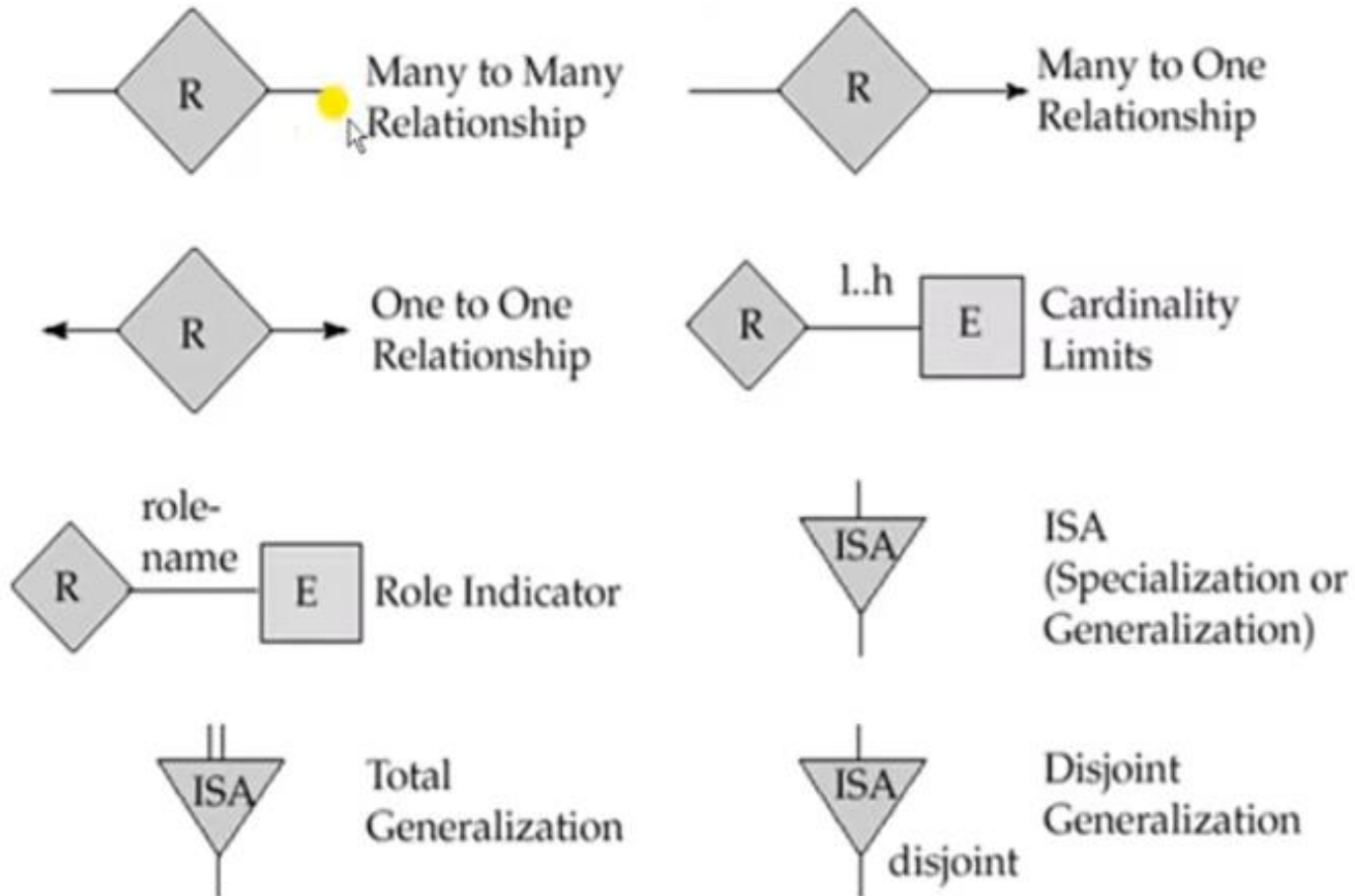
# Why ER model is useful?

- An **ER Model maps well to the relational model** i.e. the constructs used in ER Model can be easily transformed into relational tables
- An ER Model can be **used by the database designer to communicate the database design to the user**
- An ER Model can be **used as a design plan by the database developer to implement** a data model in specific DBMS software

# Symbols used in ER diagram

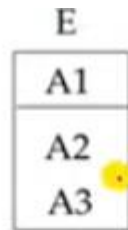


# Symbols used in ER diagram

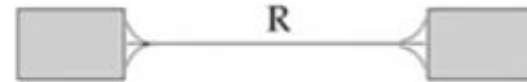


# Symbols used in ER diagram

Entity set E with  
attributes A1, A2, A3  
and primary key A1



Many to Many  
Relationship



One to One  
Relationship



Many to One  
Relationship



# ER Modeling Constructs

The basics of Entity-Relationship modeling

1. **E**ntities
2. **A**tttributes
3. **R**elationships



# Entity

- An **Entity** is a “**thing**” or “**object**” in the real world that is distinguishable from other objects.
- Entity can be any thing that has an independent existence and about which we collect data. It is also known as **entity type**.
  - Example: *In a School database, the students, teachers, classes, courses, or projects can be taken as an entity*
- Entities are represented by means of **rectangles**.

Student

Teacher

Course

- **Entities** have **attributes** that give them their identity
  - Example: Students have *roll. no., names and addresses*
- **Entities** becomes **table** in relational model

# Entity Set

- An **Entity Set** is a collection of similar type of entities, that share same attributes.

- Example: a *Students* set may contain all the students of a school;  
a *Teachers* set may contain all the teachers of a school from all faculties.



- **Entity sets** need not be **disjoint**

- Example: the entity set Employee (all employees of a bank) and the entity set Customer (all customers of the bank) may have members in common

# Attributes

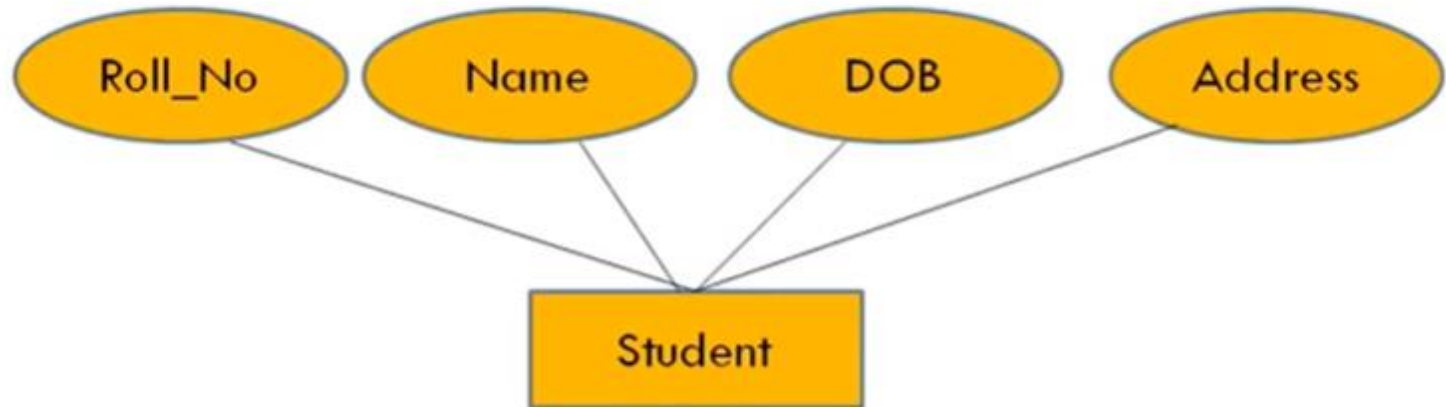
- An **entity** is represented by a **set of attributes**
- **Attributes** are used to describe the **property** of an **entity**
  - **Example:** a Student entity may have Roll\_No, Name, DOB, Age, Address, Mobile\_No as attribute
- For each attribute there is a set of permitted values, called **domain** (or **range**) of that attribute
  - **Example:** a *student's name* cannot be a numeric value. It has to be alphabetic. A *student's age* cannot be negative, etc. A *student roll\_no* can be numeric between some range like (0-10000)
- Attributes are represented by **Ellipse**



# Example

## Schema (Entity type):

Student (Roll\_No, Name, DOB, Address)



## Entity 1:

101, Rahul Sharma, 12-10-2005, Delhi

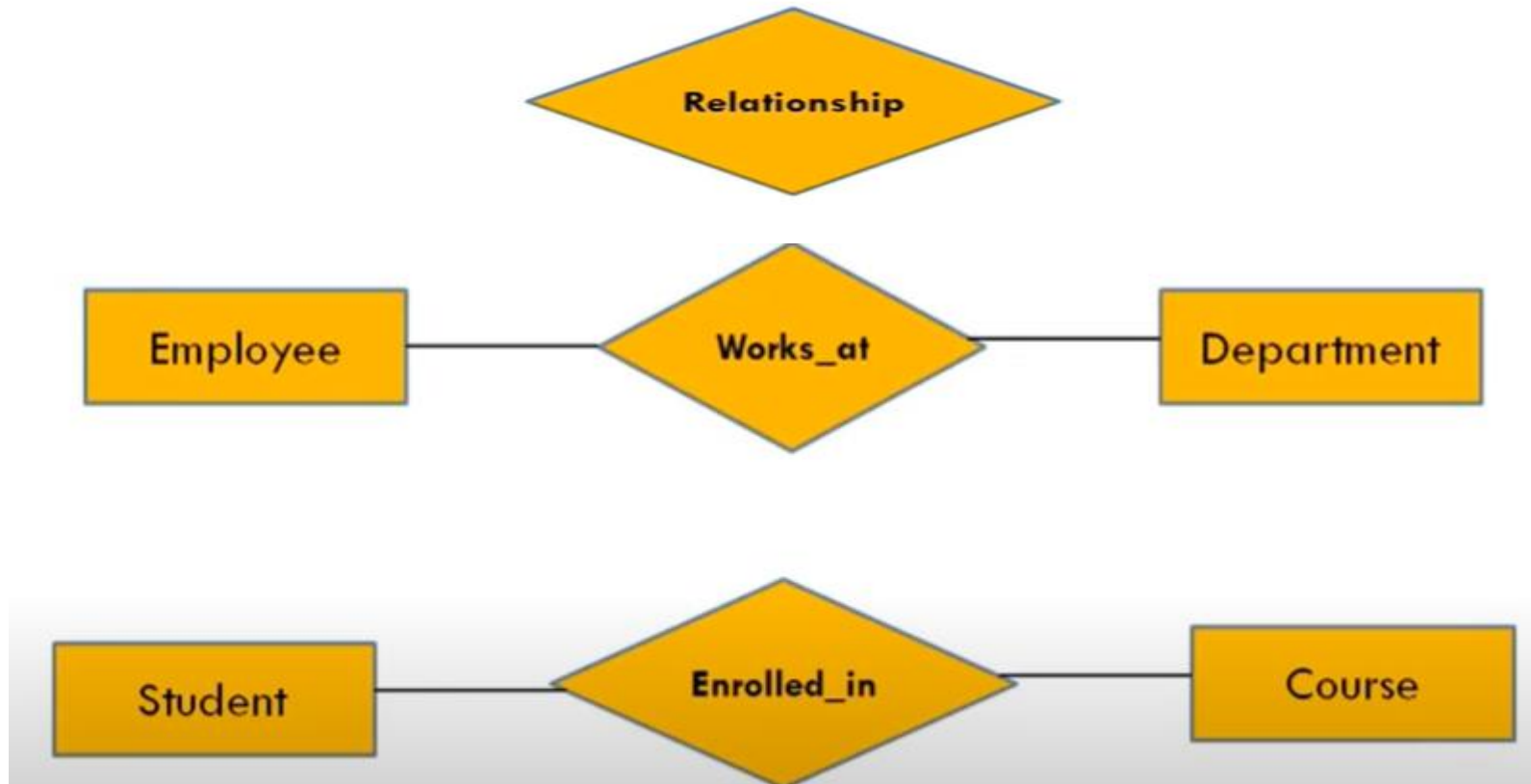
## Entity 2:

102, Seema Singh, 15-12-2007, Jaipur

# Relationship

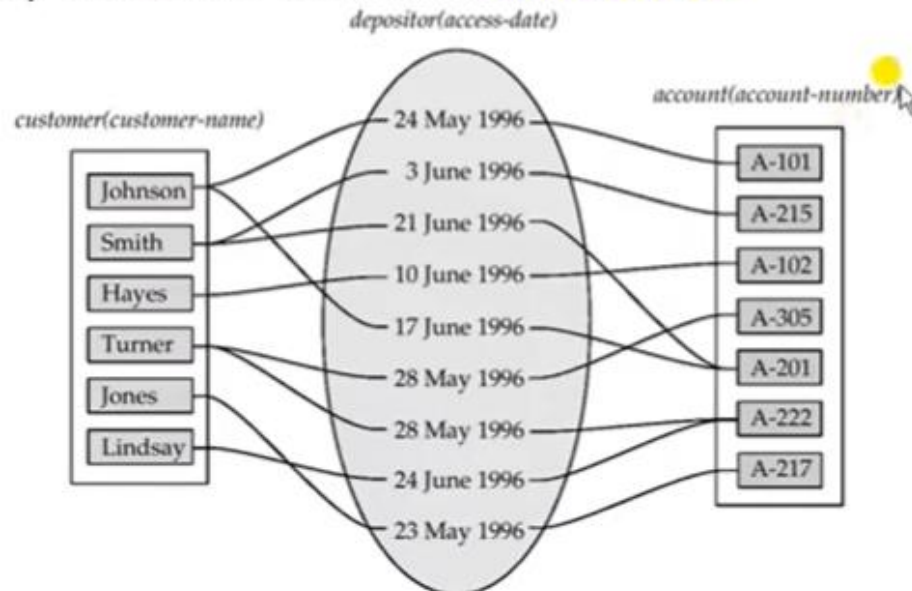
- A **Relationship** is an **association** among entities
  - For example:
    - an employee **works\_at** a department
    - a student **enrolls** in a course.
- Here, **Works\_at** and **Enrolls** are called **relationships**.
- Relationships are represented by **diamond-shaped** box.

Eg.



# Relationship: Descriptive Attribute

- Like entities, a relationship too can have **attributes**. These attributes are called **descriptive attributes**
- For instance, the *depositor* relationship set between entity sets *customer* and *account* may have the attribute *access-date*



# Degree of Relationship

- The number of different entity sets **participating in a relationship** set is called as **degree of a relationship set**
  - Unary
  - Binary
  - Ternary
  - N-ary

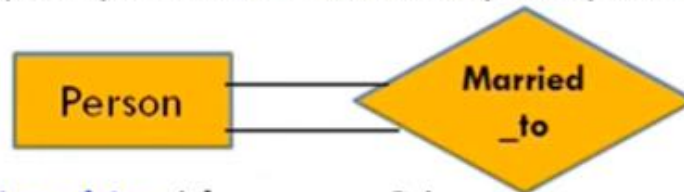


# Degree of Relationship

- **Unary Relationship: (degree = 1)**

A **unary relationship** is **only one** entity participate in a relationship, the relationship is called as unary relationship.

- For example, one person is married to only one person.



- **Binary Relationship: (degree = 2 )**

A **binary relationship** is when **two** entities participate in a relationship, and is the most common relationship degree.

- For example, Student is enrolled in Course.



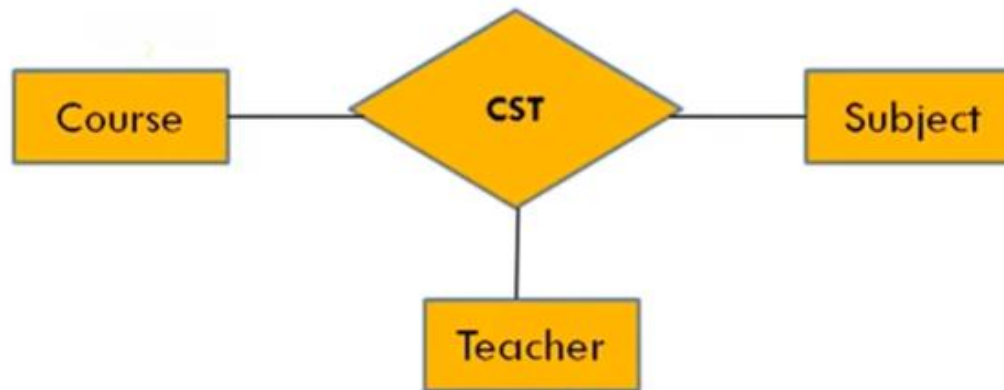


# Degree of Relationship

- **Ternary Relationship: (degree=3)**

A **ternary relationship** is when **three entities** participate in the relationship.

- For example, The University might need to record which teachers taught which subjects in which courses.



Eg. DBMS subject taught by a teacher at different department  
Like computer dept, AIDS dept etc.

# Mapping cardinalities

- ▣ One-to-One
- ▣ One-to-Many
- ▣ Many-to-One
- ▣ Many-to-Many

# Types of attributes

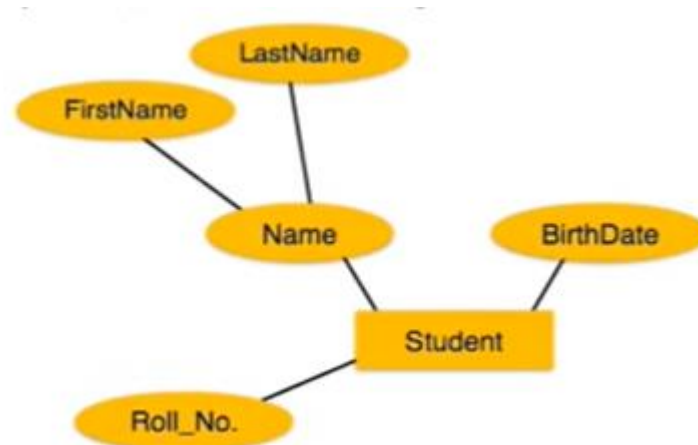
- ▣ **Simple** and **Composite** attributes
- ▣ **Single-valued** and **Multi-valued** attributes
- ▣ **Stored** and **Derived** attributes
- ▣ **Key Attribute**

# 1. Simple Attribute

- **Simple attributes** are atomic values, which cannot be divided further.
- For example:
  - ▣ a student's mobile number is an atomic value of 10 digits.
  - ▣ a Birth Date is an atomic value of date-month-year

## 2. Composite Attribute


- **Composite attributes** are made of more than one simple attribute.
- A composite attribute is divided in a tree like structure.
- For example:
  - A student's complete **name** may have **first\_name** and **last\_name**.
  - An **address** may have **street**, **city**, **state**, **country** and **pin code**

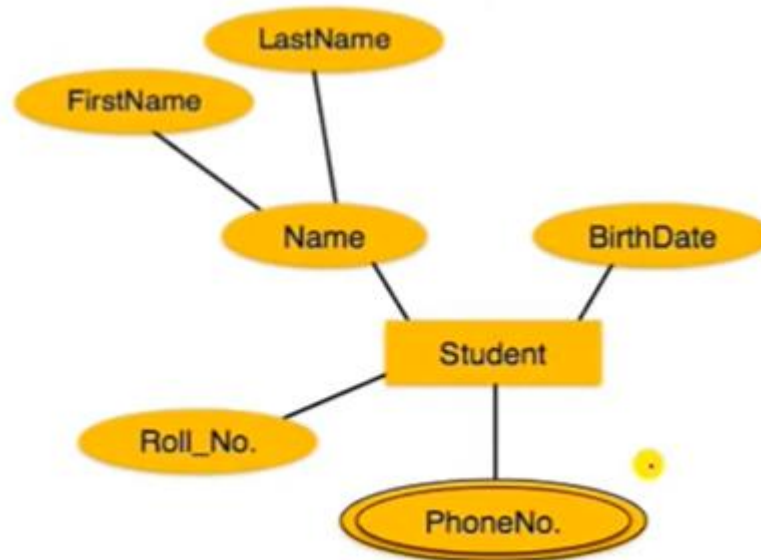


### 3. Single valued attributes

- **Single-value attributes** contain single value.
- For example:
  - ▣ Social\_Security\_Number, Aadhar\_card\_no, Roll\_no

## 4. Multivalued Attribute

- ❑ **Multi-valued attributes** may contain more than one values.
- ❑ Represented by **double-ellipse** 



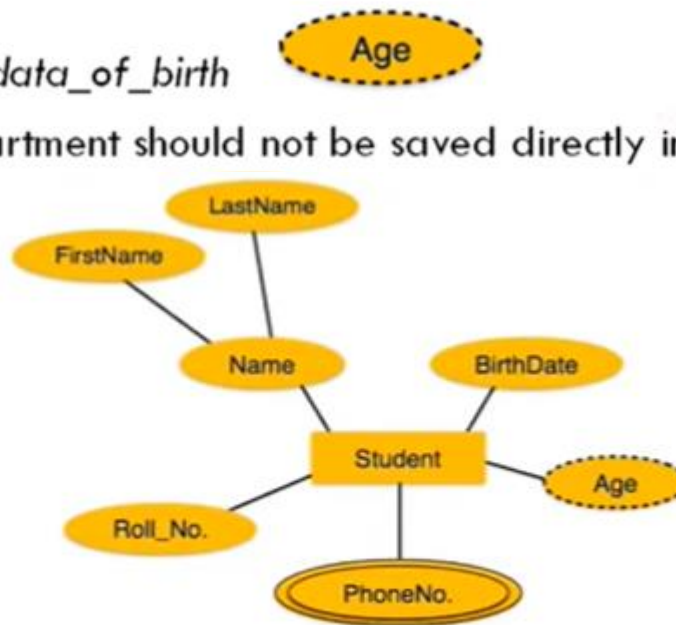
# 5. Stored Attributes

- **Stored attributes** are physically stored in the database
- Mostly all attributes are stored in database except few ones
- For example: Roll\_no, Name, birth\_date, phone\_no



# 6. Derived Attribute

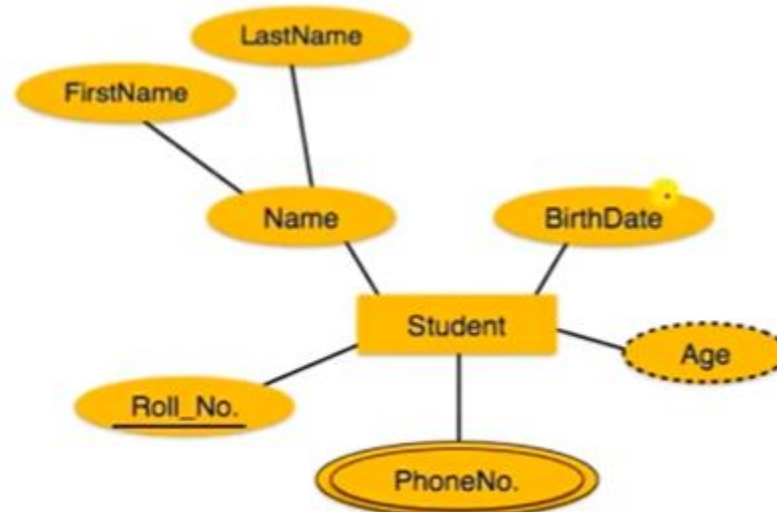
- **Derived attributes** are the attributes that do not exist in the physical database, but their values are derived from other attributes present in the database.
- **Derived** attributes are depicted by **dashed ellipse**.
- For example:
  - **age** can be derived from *data\_of\_birth*
  - **average\_salary** in a department should not be saved directly in the database, instead it can be derived



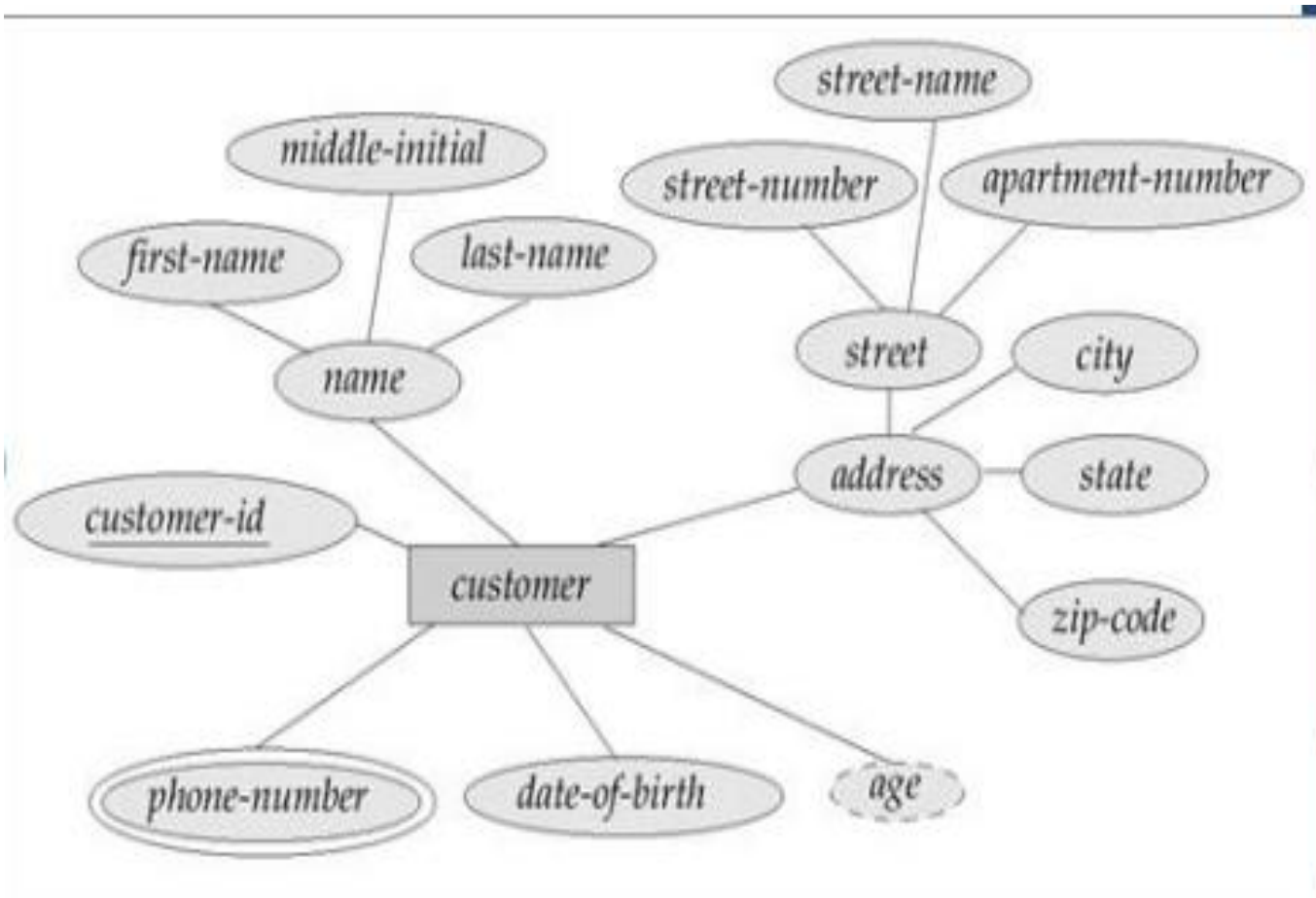
# 7. Key Attribute (Primary key)

- **Key Attribute:** The attribute which **uniquely identifies each entity** in the entity set is called **key attribute**
- It represents a primary key
- Key attribute is represented by an ellipse with underlying lines
- For example: Roll\_No will be unique for each student.

Roll\_No.



# Example of Customer entity With Composite, Multivalued, and Derived Attributes

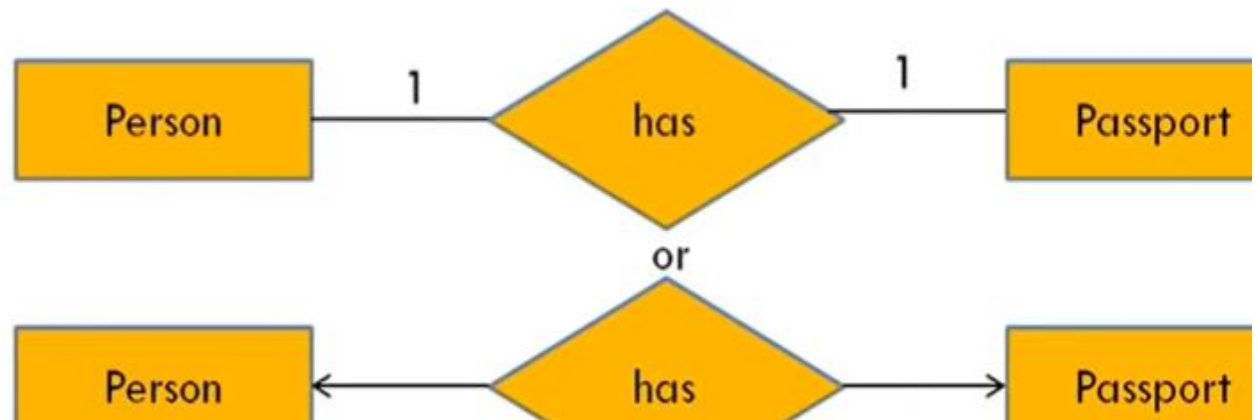
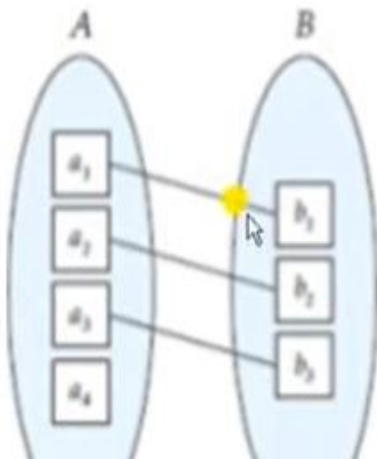


# Mapping cardinalities

- ▣ **Cardinality** defines the number of entity of an entity set participates in a relationship set.
- ▣ Most useful in describing *binary relationship*
- ▣ Cardinality can be of different types or there are four types of relationships:
  - **One-to-One (1-1)**
  - **One-to-Many (1-M)**
  - **Many-to-One (M-1)**
  - **Many-to-Many (M-N)**

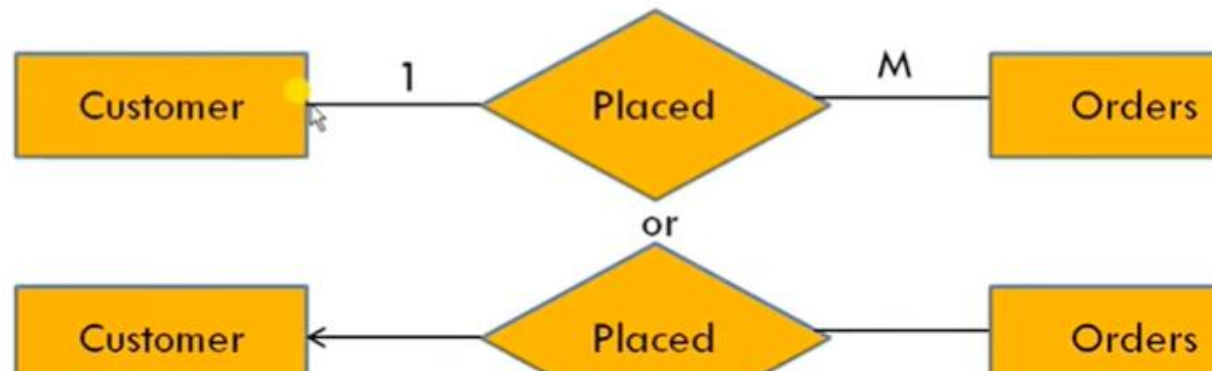
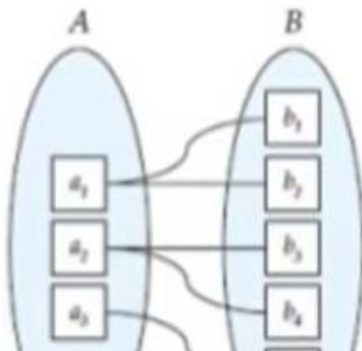
# One to one Relationship

- One entity from entity set A can be associated with *at most* one entity of entity set B and vice versa
  - For example, a person has only one passport and a passport is given to one person.



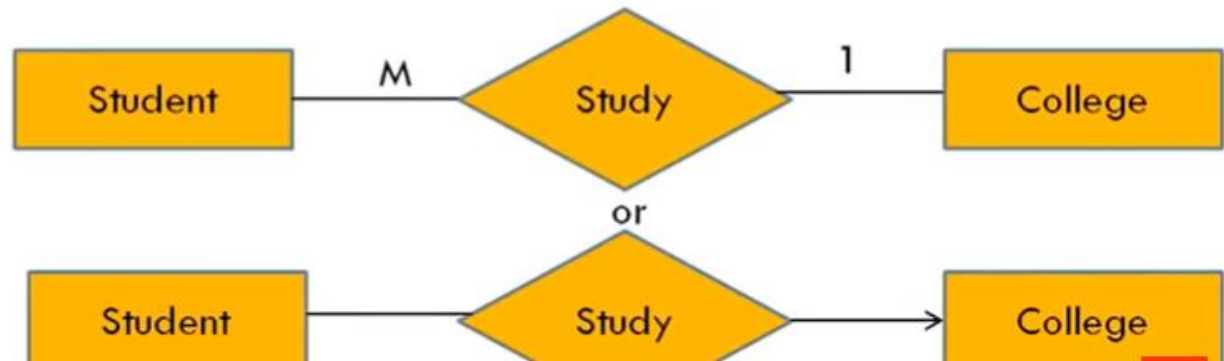
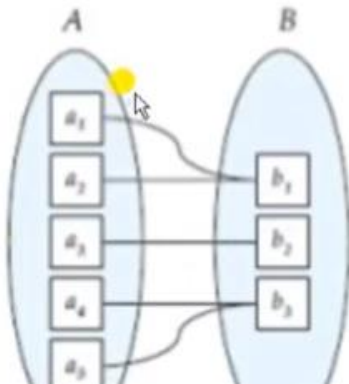
# One to many Relationship

- One entity from entity set A can be associated with more than one entities of entity set B however an entity from entity set B, can be associated with at most one entity
- For example – a customer can place many orders but a order cannot be placed by many customers.



# Many to one Relationship

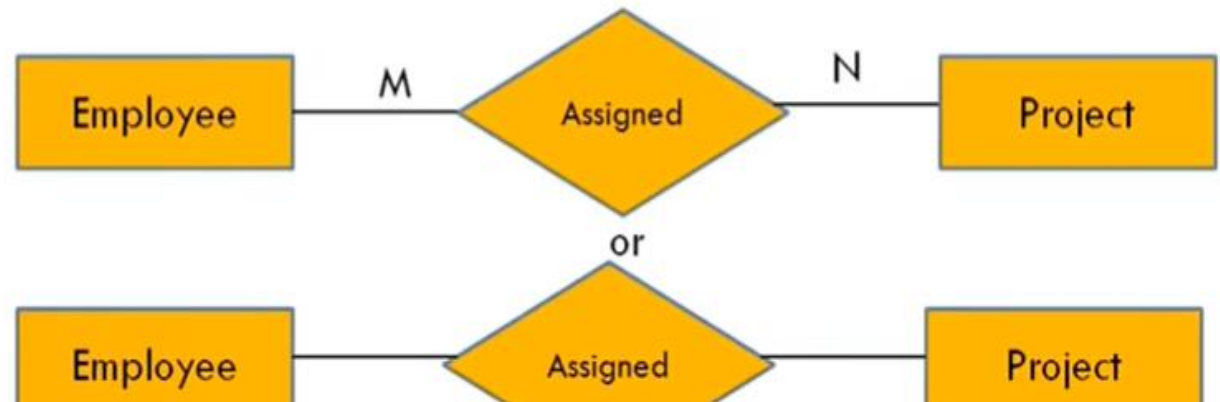
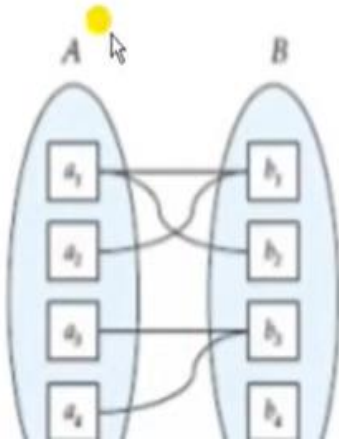
- More than one entities from entity set A can be associated with *at most* one entity of entity set B, however an entity from entity set B can be associated with more than one entity from entity set A
  - For example – many students can study in a single college but a student cannot study in many colleges at the same time.





# Many to many Relationship

- One entity from A can be associated with more than one entity from B and vice versa.
  - For example, an Employee can be assigned to many Projects and a Project can have many Employee.





# How to choose relationship

- The appropriate mapping cardinality for a particular relationship set depends on the real world situation being modeled
- Consider **Borrower** relationship set in a Bank :
  - ▣ If in a particular bank, a loan can belong to only one customer and a customer can have only one loans then the relationship set from customer to loan is **1:1**



- ▣ If a loan can belong to only one customer and a customer can have several loans then the relationship set from customer to loan is **1:M**



Eg. One customer has applied for personal loan, educational loan, home loan etc

# How to choose relationship

- If a loan can belong to several customers and a customer can have only one loan then the relationship set from customer to loan is **M:1**



Eg. Joint loan – two or three customers jointly taken one loan ( start-up, business etc)

- If a loan can belong to several customers (as loans can be taken jointly by several business partners) then the relationship set is **M:N**



# Participation constraints

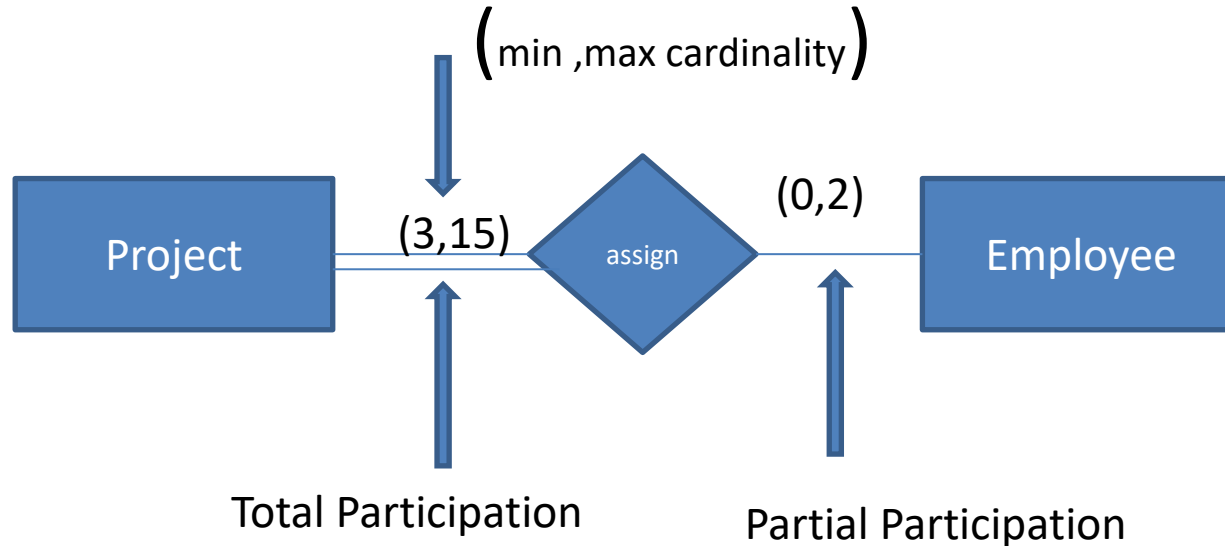
- ▣ **Total Participation** – Each entity is involved in the relationship. Total participation is represented by **double lines**.
  - E.g. participation of *loan* in *borrower* is total
    - every loan must have a customer associated to it via borrower
- ▣ **Partial participation** – Not all entities are involved in the relationship. Partial participation is represented by **single lines**.
  - participation of *customer* in *borrower* is partial
    - A customer may have no loans



Not necessary each customer has taken loan

Each loan is associated with the customer

# Example Participation Constraints




Min-3-> 3 Employee Works on one project  
Max15-> 15 employee can work on one project

Min-0-> Not necessary each Employee Works on project  
Max2-> one employee can work on maximum 2 projects

# Entity Types: Strong Entity

- The **strong entity** always have a **primary key**.
- Its existence is not dependent on any other entity i.e. it is independent of other entity.
- A set of strong entities is known as **strong entity set**.
- Strong Entity is represented by a *single rectangle*



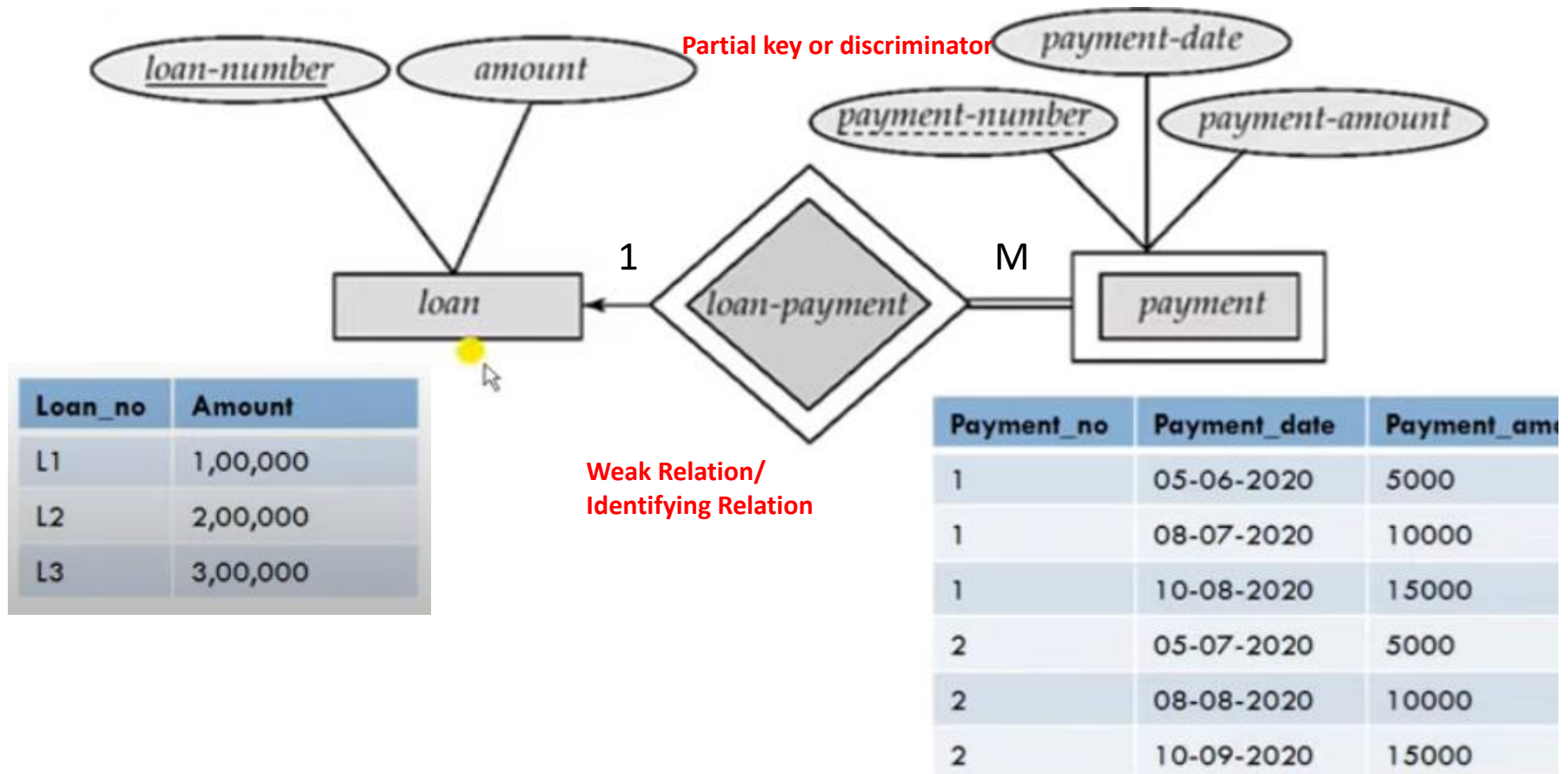
Strong Entity

# Weak Entity

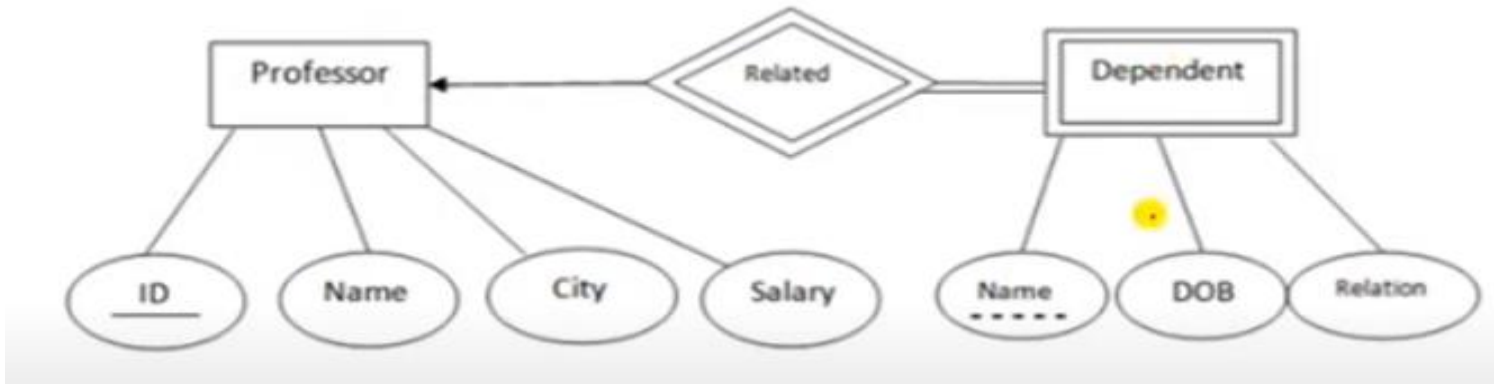
- The **weak entity** does not have a sufficient attributes to form a primary key i.e. **Weak entity do not have a primary key.**
- A **weak entity** is **dependent on a strong entity** to ensure the its existence.
- A set of weak entities is known as *weak entity set*.
- Weak Entity is represented by double rectangle:



# Example: Strong and weak entity



# Another Example



**ID** is the **Primary key** (represented with a line) and **Name** in **Dependent** entity is called **Partial Key** (represented with a dotted line).



# Difference between Strong entity and weak entity

	STRONG ENTITY	WEAK ENTITY
1.	Strong entity always has primary key	While weak entity has partial key or discriminator key
2.	Strong entity is not dependent of any other entity	Weak entity is depends on strong entity
3.	Strong entity is represented by single rectangle	Weak entity is represented by double rectangle
4.	Two strong entity's relationship is represented by single diamond	While the relation between one strong and one weak entity is represented by double diamond
5.	Strong entity have either total participation or not	While weak entity always has total participation

# Exercise of ER Diagram

- The bank is organized into branches. Each branch is located in a particular city and is identified by a unique name. The bank monitors the assets of each branch.
- Bank customers are identified by their customer id values. The bank stores each customer's name and the street and city where the customer lives. Customers may have accounts and can take out loans. A customer may be associated with a particular bank employee, who may act as a loan officer or personal banker for that customer.
- Bank employees are identified by their employee id values. The bank administration stores the name and telephone number of each employee, the names of the employee's dependents, and the employee id number of the employee's manager. The bank also keeps the track of the employee's start date and, thus, length of employment.
- The bank offers two types of accounts - savings and checking accounts. Accounts can be held by more than one customer, and a customer can have more than one account. Each account is assigned a unique account number. The bank maintains a record of account's balance and the most recent date on which the account was accessed by each customer holding the account. In addition, each savings account has an interest rate and overdrafts are recorded for each checking account.
- A loan originates at a particular branch and can be held by one or more customers. A loan is identified by a unique loan number. For each loan, the bank keeps track of the loan amount and the loan payments. Although a loan payment number does not uniquely identify a particular payment among those for all the bank's loans, a payment number does identify a particular payment for a specific loan. The **date** and **amount** are recorded for each payment.

# Exercise of ER Diagram

The bank is organized into **branches**. Each branch is located in a particular **city** and is identified by a unique **name**. The bank monitors the **assets** of each branch.

Bank **customers** are identified by their **customer id** values. The bank stores each customer's **name** and the **street** and **city** where the customer lives. Customers may have **accounts** and can take out **loans**. A customer may be associated with a particular bank **employee**, who may act as a loan officer or personal banker for that customer.

Bank **employees** are identified by their **employee id** values. The bank administration stores the **name** and **telephone number** of each employee, the names of the employee's **dependents**, and the employee id number of the employee's manager. The bank also keeps the track of the employee's **start date** and, thus, **length of employment**.

- The bank offers two types of accounts - savings and checking accounts. **Accounts** can be held by more than one **customer**, and a customer can have more than one account. Each account is assigned a unique **account number**. The bank maintains a record of account's **balance** and the **most recent date** on which the account was accessed by each customer holding the account. In addition, each savings account has an interest rate and overdrafts are recorded for each checking account.

A **loan** originates at a particular **branch** and can be held by one or more **customers**. A loan is identified by a unique **loan number**. For each loan, the bank keeps track of the loan **amount** and the **loan payments**. Although a loan **payment number** does not uniquely identify a particular payment among those for all the bank's loans, a payment number does identify a particular payment for a specific loan. The **date** and **amount** are recorded for each payment.



# KEYS

- **Definition:** A **key** is an **attribute** or **set of attributes** that **uniquely identifies** any record (or tuple) from the table.
- **Purpose:**
  - **Key** 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.**

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

# Types of Keys

1. Super Key
2. Candidate Key
3. Primary Key
4. Alternate Key
5. Foreign Key
6. Composite Key

# Super key

- A **super key** is a combination of all possible attributes that can uniquely identify the rows (or tuple) in the given relation. **It is kind of super set of keys.**
- Super key is a superset of a candidate key.
- A table can have many super keys
- A super key may have additional attribute that are not needed for unique identity

# Super key example

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

## Super Keys:

1. {Emp\_Id}
  2. {Aadhar\_No}
  3. {Email\_Id}
  4. {Emp\_Id, Aadhar\_No}
  5. {Aadhar\_No, Email\_Id}
  6. {Emp\_Id, Email\_Id}
  7. {Emp\_Id, Aadhar\_No, Email\_Id}
  8. {Emp\_Id, Name}
  9. {Emp\_Id, Name, Dept\_Id}
- Etc.

# Candidate key

- A candidate key is an attribute or set of an attribute which can uniquely identify a tuple.
- A candidate key is a **minimal super key**; or a Super key with no redundant attributes.
  - It is called a minimal super key because we select a candidate key from a set of super key such that selected candidate key is the minimum attribute required to uniquely identify the table


Candidate keys are defined as distinct set of attributes from which **primary key can be selected**

Candidate keys are not allowed to have NULL values



# Candidate key example

## Candidate Keys



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


### □ Candidate Keys

1. {Emp\_Id}
2. {Aadhar\_No}
3. {Email\_Id}

# Primary key

- A **primary key** is one of the candidate key chosen by the database designer to uniquely identify the tuple in the relation
  - ▣ The value of primary key can never be NULL.
  - ▣ The value of primary key must always be unique (not duplicate).
  - ▣ The values of primary key can never be changed i.e. no updation is possible.
  - ▣ The value of primary key must be assigned when inserting a record.
  - ▣ A relation is allowed to have only one primary key.

**Primary Key**



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

- **Primary Key**  
1. {Emp\_Id}

# Alternate Key

- Out of all candidate keys, only one gets selected as primary key, remaining keys are known as alternate keys.
- In the Employee table
  - **Emp\_Id** is best suited for the **primary key**.
  - Rest of the attributes like **Aadhar\_No**, **Email\_Id** are considered as a **alternate keys**.

# Foreign Key

- A **Foreign Key** is:
  - A **key** used to link two tables together.
  - An **attribute (or set of attributes)** in one table that refers to the **Primary Key** in another table.
- The **purpose** of the **foreign key** is
  - to ensure (or maintain) **referential integrity** of the data.

# Foreign key example

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 Keys

Primary Key

Department Table (Referenced relation)

Dept_Id	Dept_Name
1	Sales
2	Marketing
3	HR

Foreign Key:

In Employee Table  
1. Dept\_Id

# Foreign key

- ❑ Foreign key references the primary key of the table.
- ❑ Foreign key can take only those values which are present in the primary key of the referenced relation.
- ❑ Foreign key may have a name other than that of a primary key.
- ❑ Foreign key can take the NULL value.
- ❑ There is no restriction on a foreign key to be unique.
- ❑ In fact, foreign key is not unique most of the time.
- ❑ Referenced relation may also be called as the master table or primary table.
- ❑ Referencing relation may also be called as the foreign table.

# Composite key

A key that has more than one attributes is known as composite key. It is also known as compound key.

Cust_Id	Order_Id	Product_Code	Product_Count
C01	001	P111	5
C02	012	P111	8
C02	012	P222	6
C01	001	P333	9

**Composite Key:**

{Cust\_Id, Product\_Code}



# Extended ER Model Features

- As the complexity of data increased in the late 1980s, it became more and more difficult to use the traditional ER Model for database modelling. Hence some improvements or enhancements were made to the existing ER Model to make it able to handle the complex applications better.
- Hence, as part of the **Extended ER Model**, along with other improvements, three new concepts were added to the existing ER Model:

1. **Generalization**
2. **Specialization**
3. **Aggregation**

- It is important for newer applications of database technology, such as databases for engineering design and manufacturing (CAD/CAM),
- telecommunications,
- complex software systems, and
- Geographic Information Systems (GIS), among many other applications

These types of databases have more complex requirements than do the more traditional applications.

This led to the development of additional **semantic data modeling concepts**



# Extended ER Model Features

Many of these concepts were also developed independently in related areas of computer science, such as

**knowledge representation area** of artificial intelligence and

**object modeling area** in software engineering

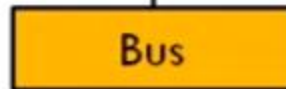
**ER model** can be enhanced to include these concepts, leading to the **Enhanced ER (EER) model**.

# Generalization

- Consider we have 3 sub entities Car, Bus and Motorcycle. Now these three entities can be generalized into one higher-level entity (or super class) named as **Vehicle**.

Superclass

High level entity



Sub-classes

Car

Bus

Motorcycle

Low level entities



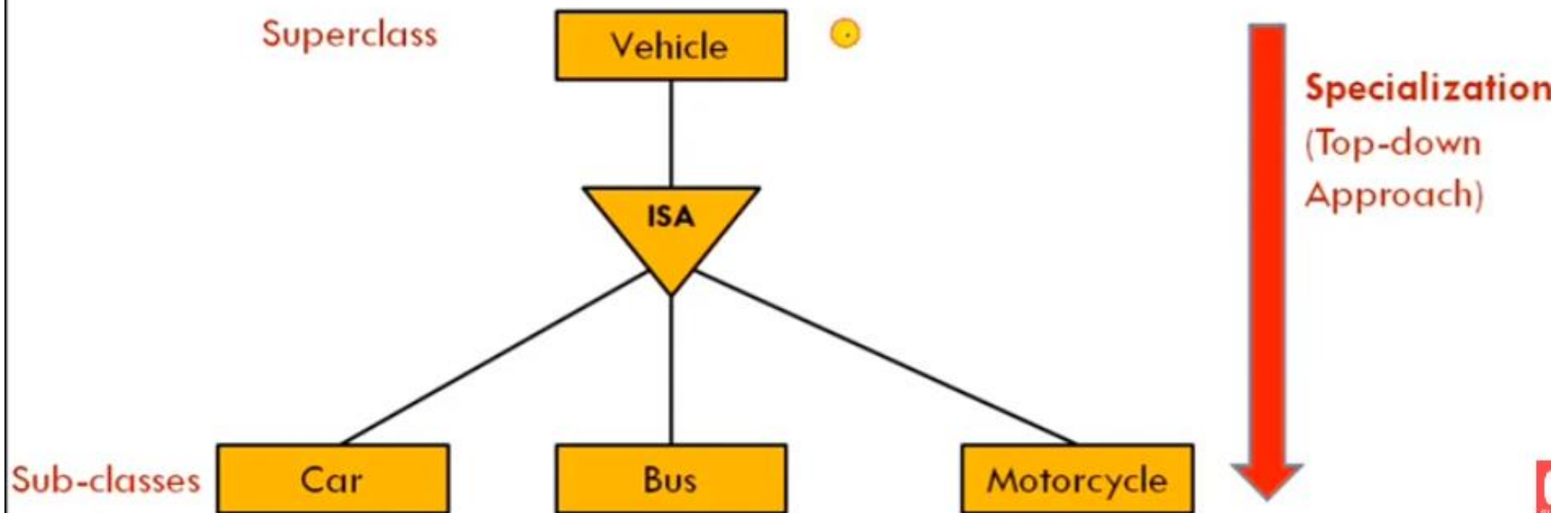
**Generalization**  
(Bottom-up Approach)

# Specialization

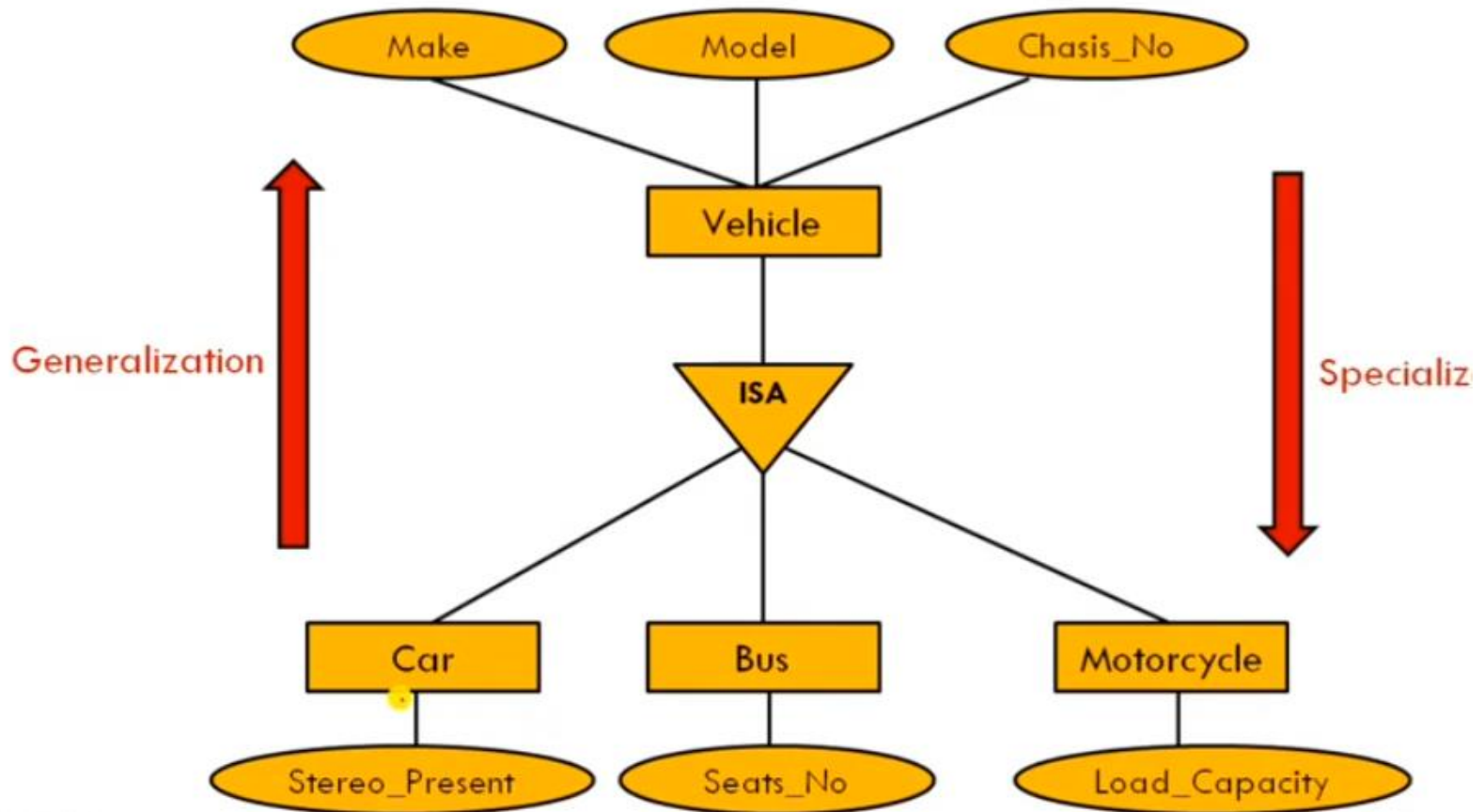
- Specialization is **opposite** of Generalization
- In **Specialization**, an entity is broken down into sub-entities based on their characteristics.
- **Specialization is a “Top-down approach”** where higher level entity is specialized into two or more lower level entities.
- **Specialization is used** to identify the subset of an entity set that shares some distinguishing characteristics.
- **Specialization** can be repeatedly applied to refine the design of schema
- depicted by triangle component labeled **ISA**

# Example

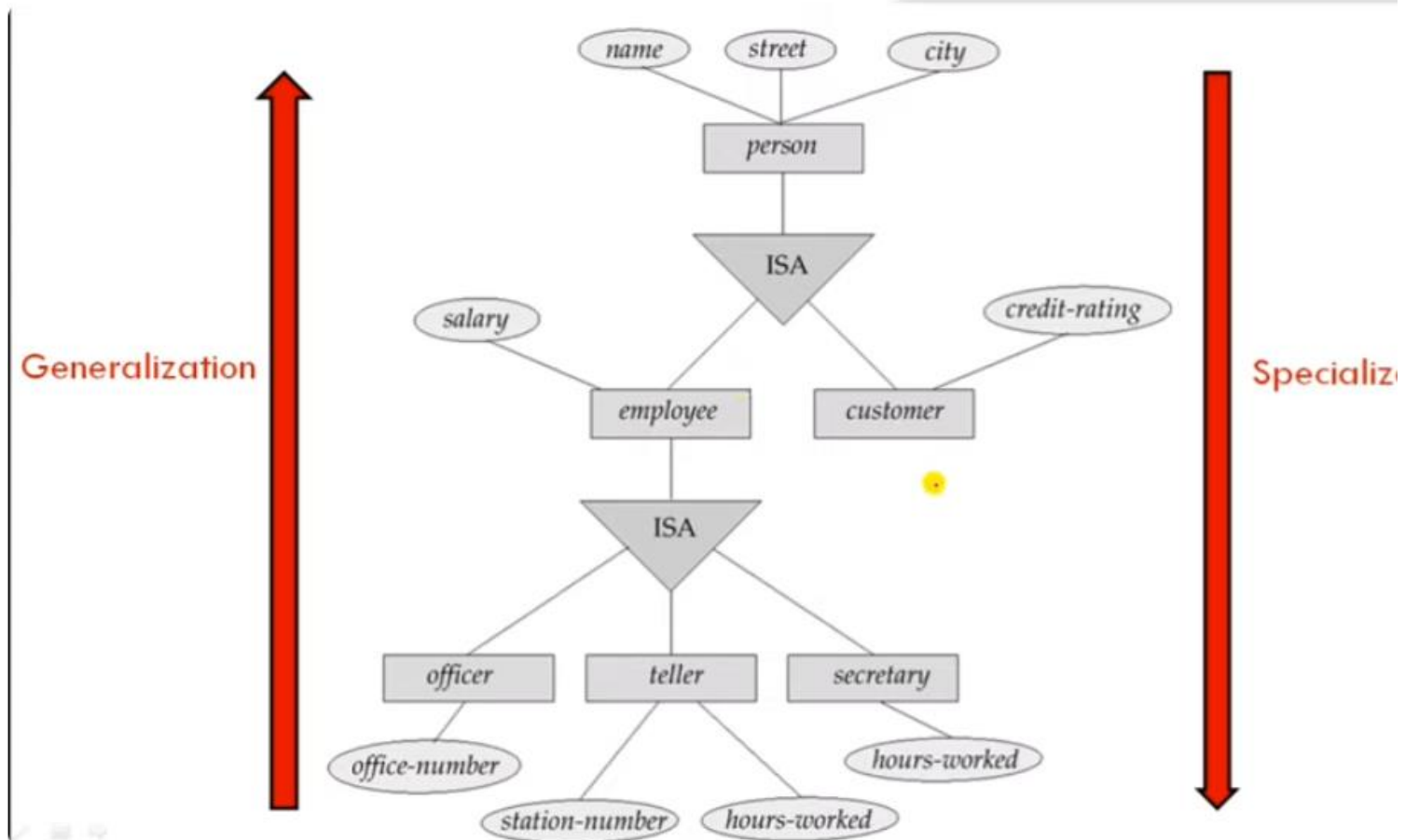
- **Vehicle** entity can be a Car, Truck or Motorcycle.
  - Normally, the superclass is defined first, the subclass and its related attributes are defined next, and relationship set are then added.



# Example: Attribute inheritance



# How schema and Tables can be formed



Four tables can be formed:

1. **customer** (name, street, city, credit\_rating)
2. **officer** (name, street, city, salary, office\_number)
3. **teller** (name, street, city, salary, station\_number, hours\_worked)
4. **secretary** (name, street, city, salary, hours\_worked)

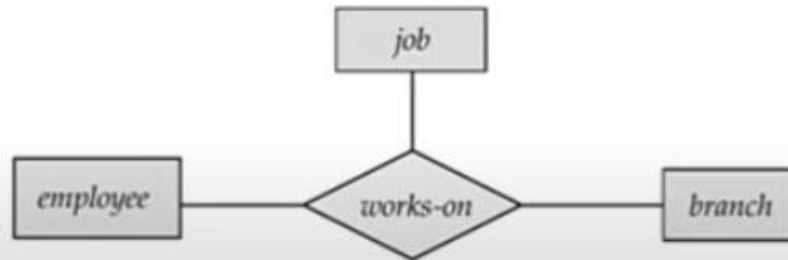
All leaf Nodes →

# Aggregation

- **Aggregation** is used when we need to express a **relationship among relationships**
- **Aggregation** is an **abstraction** through which **relationships are treated as higher level entities**
- **Aggregation** is a process when a **relationship between two entities is considered as a single entity** and again this single entity has a **relationship with another entity**

# Example ( Relationship of relation)

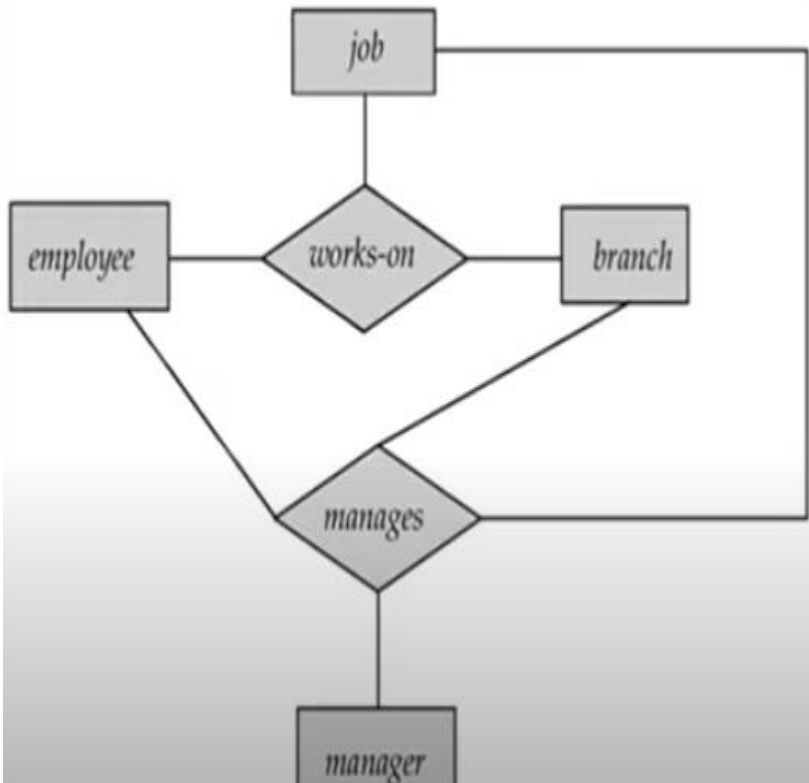
- Basic E-R model can't represent relationships involving other relationships
- Consider a ternary relationship **works\_on** between **Employee**, **Branch** and **Job**.
  - Am **employee works on** a particular **job** at a particular **branch**



- Suppose we want to assign a **manager** for jobs performed by an employee at a branch (i.e. want to assign managers to each employee, job, branch combination)
  - Need a separate manager entity-set
  - Relationship between each manager, employee, branch, and job entity

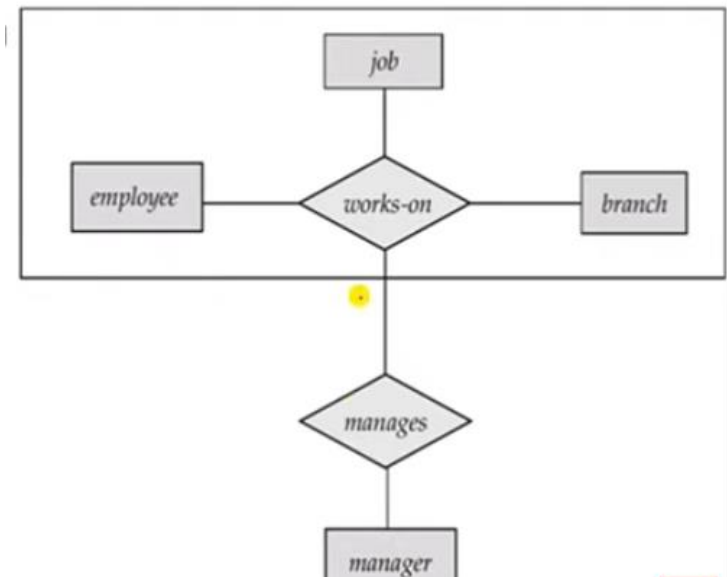


# Example: Aggregation



ER Diagram with Redundant Relationship

- Eliminate this redundancy via **aggregation**
  - Treat relationship as an abstract entity
  - Allows relationships between relationships
  - Abstraction of relationship into new entity

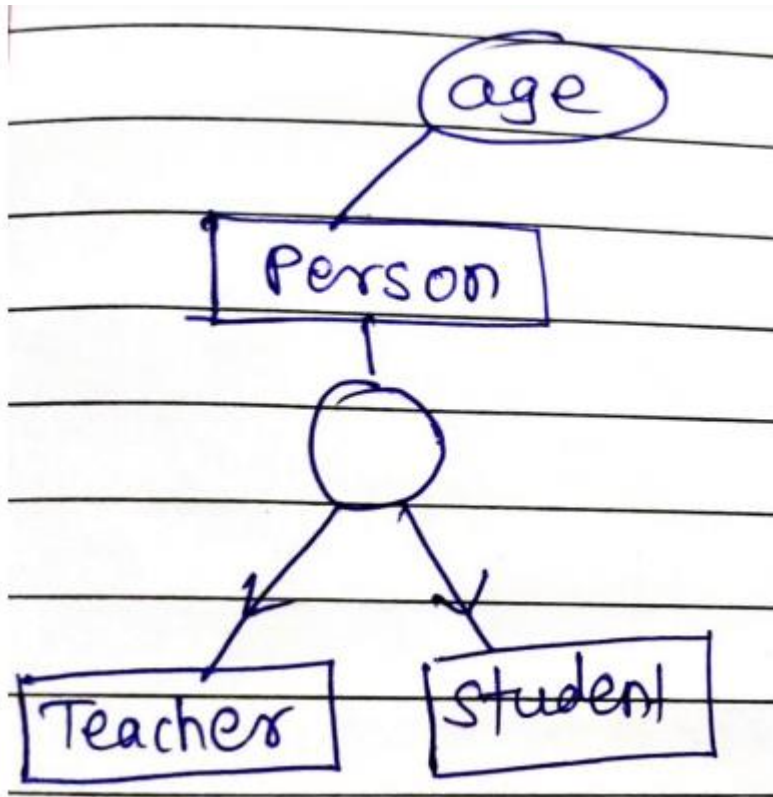


ER Diagram with Aggregation

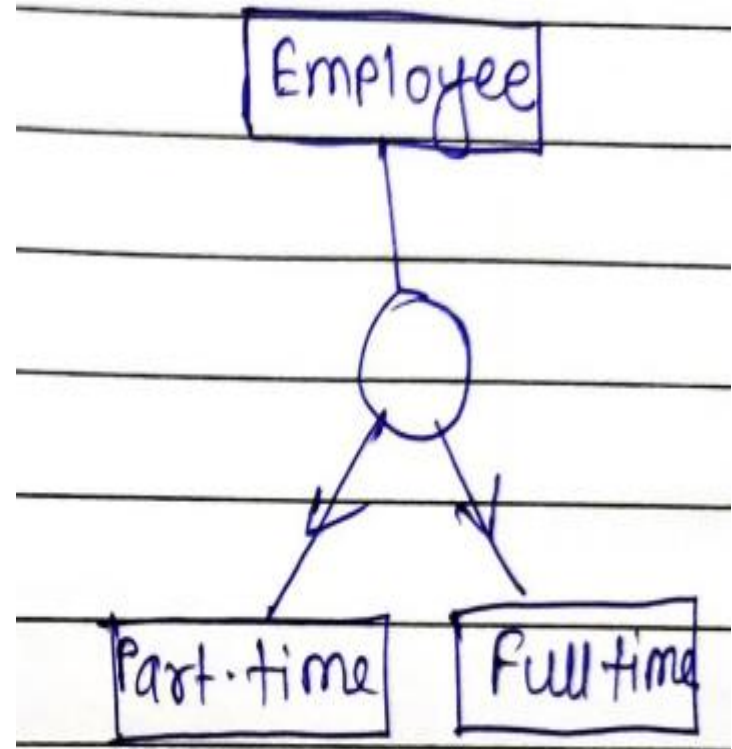
# Constraints on Generalization and Specialization

- Membership constraints
  - Attribute defined
  - User defined
- Disjointness constraints
  - Disjoint
  - Overlapping
- Completeness constraints
  - Total G/S
  - Partial G/S

Attribute defined

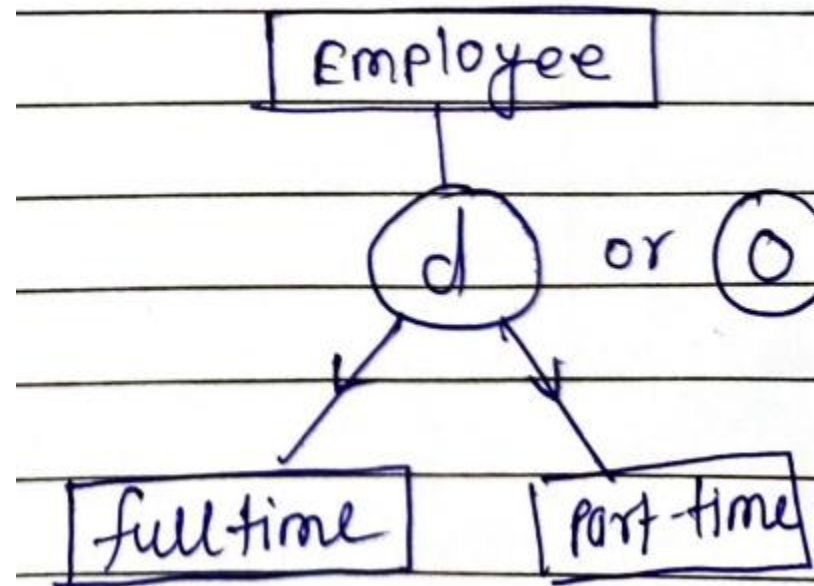


User defined



Disjoint

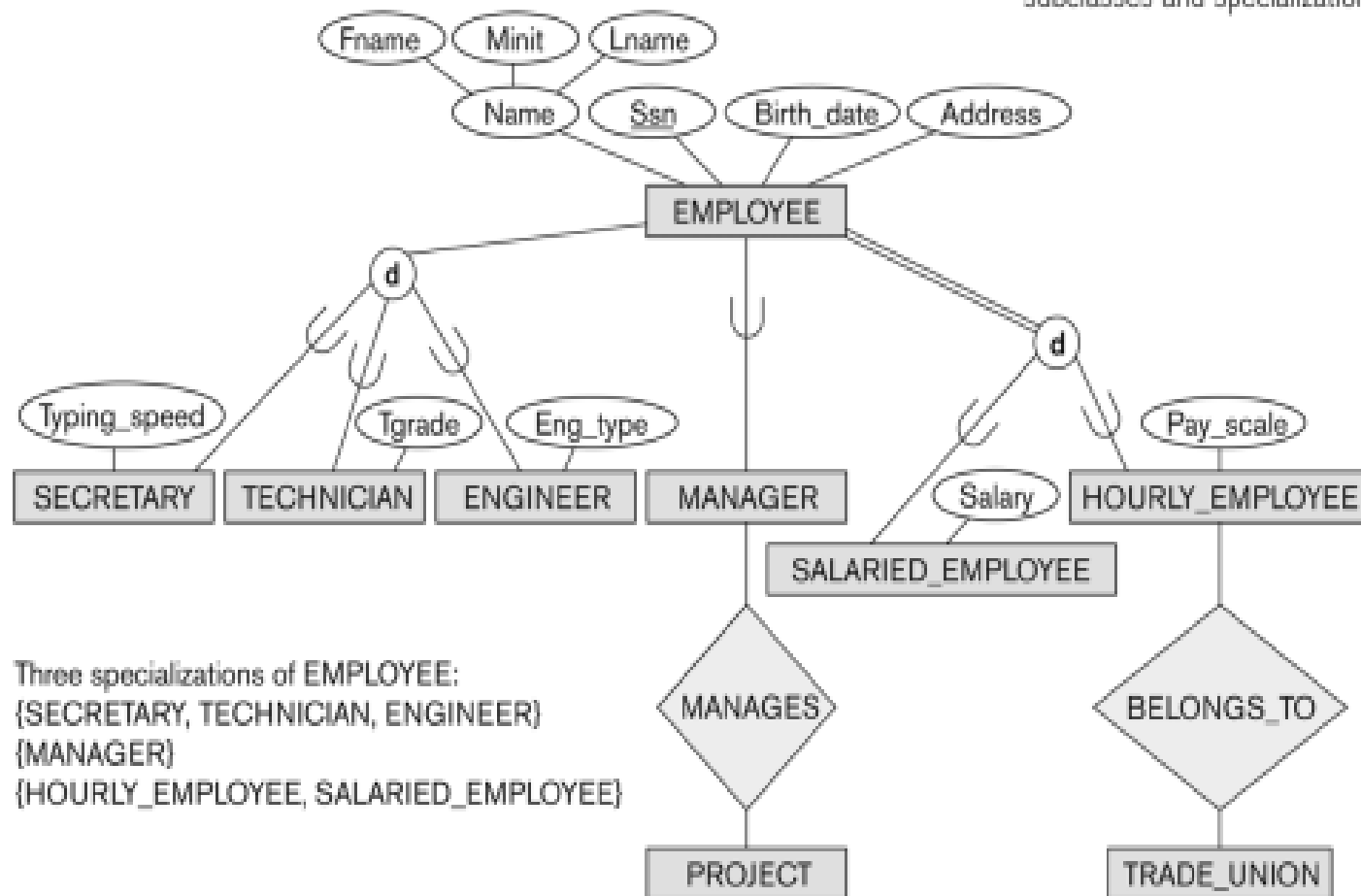
Overlapping



# Example

**Figure 4.1**

EER diagram notation to represent subclasses and specialization.



# Example

