# 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

# ER Modeling Constructs


The basics of Entity-Relationship modeling

1. Entities
2. Attributes
3. Relationships

# 1. Entities


- 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.**


- **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 Set

- **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

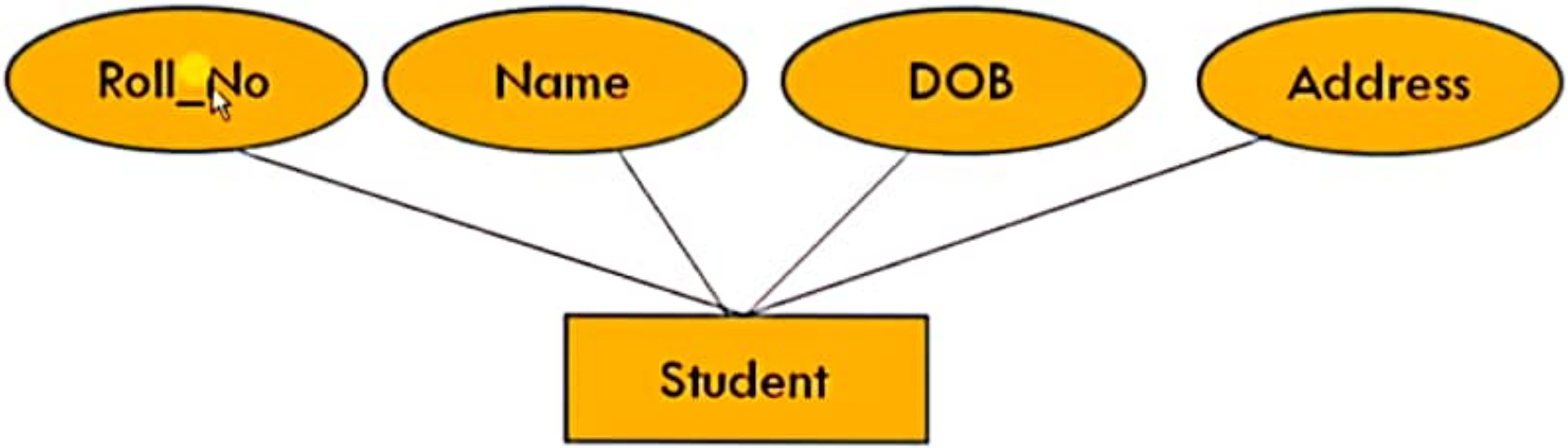
# 2. Attributes


- 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 ne 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

# Attributes Types

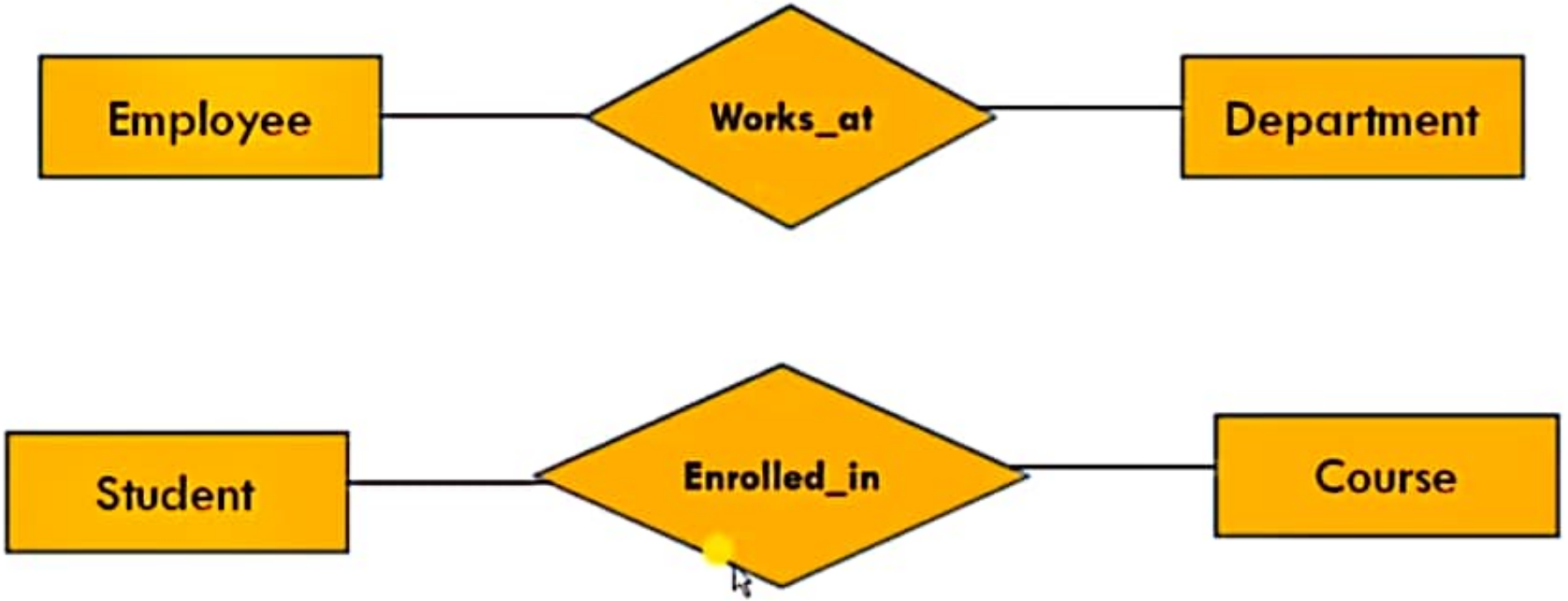

- Simple and composite attributes
- Single-valued and multi-valued attributes
- Stored and Derived attributes

# 3. Relationships


- 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.

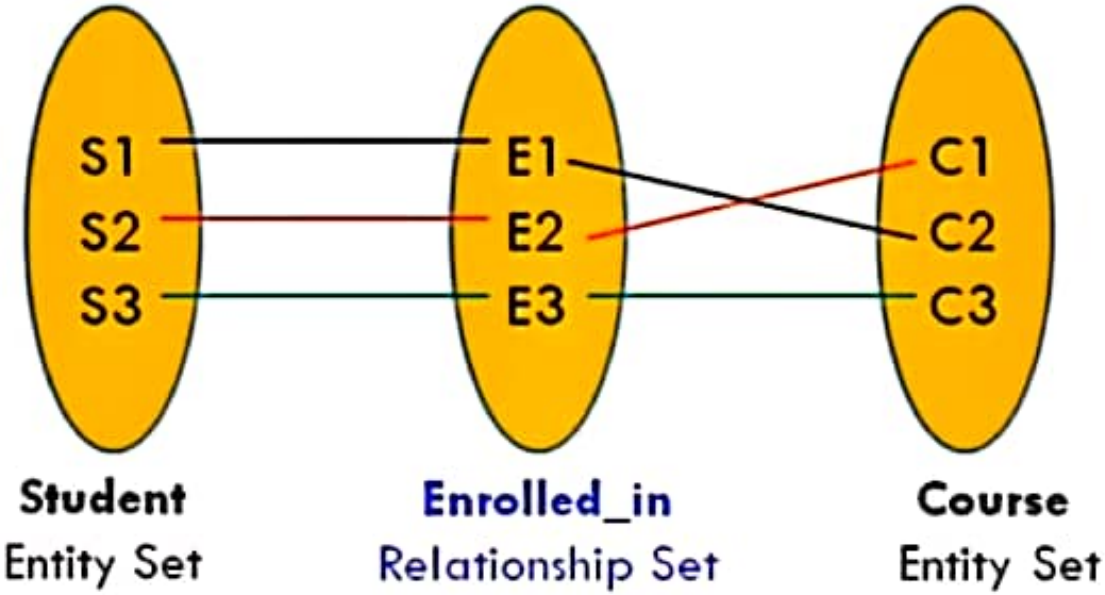
# Example



# Relationship Set

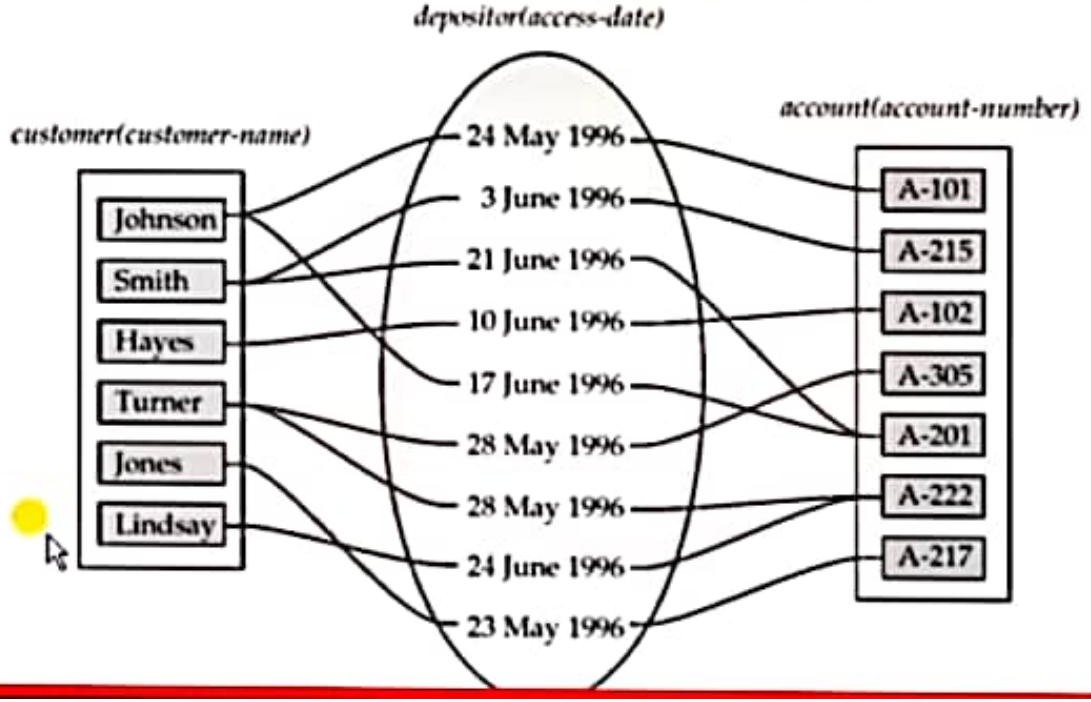

- A set of relationships of similar type is called **relationship Set**
- The following relationship set Enrolls (E1, E2, E3) depicts:

S1 is enrolled in C2, S2 is enrolled in C1 and S3 is enrolled in C3



# 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.

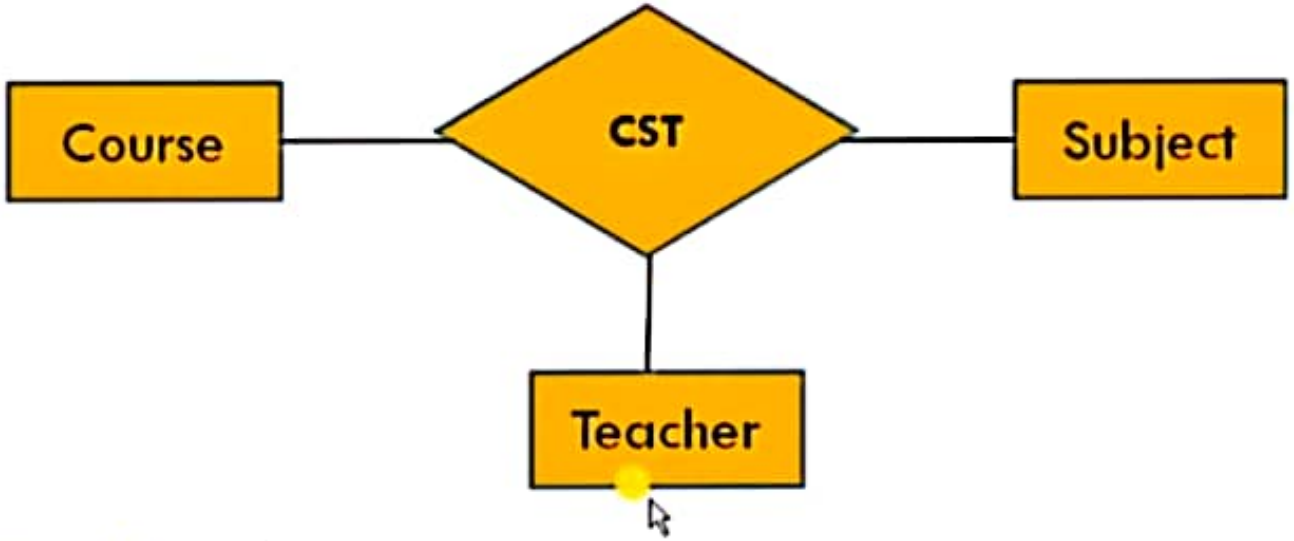
  - For example, Student is enrolled in Course.

# 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.




- **n - ary Relationship: (degree=n)**

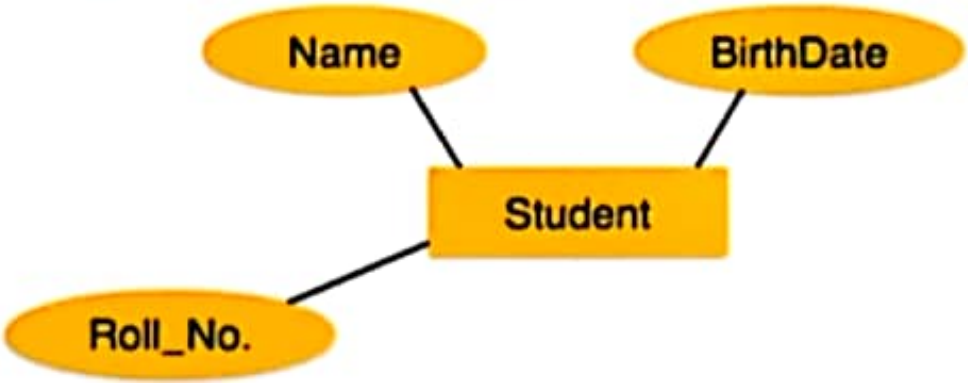
  When there are n entities set participating in a relation, the relationship is called as n-ary relationship.

# Mapping Cardinalities


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

# Attributes in ER Model

- An **Entity** is a **"thing" or "object"** in the real world that is distinguishable from other objects.
  - For eg: Student, Teacher, Class, Course, Employee, Customer, account
- An **Entity** is represented by a **set of attributes**
- **Attribute** is used to describe the **property of an entity**
  - **Example:** a Student entity may have Roll_No, Name, BirthDate, Age as attribute
- There exists a **domain** or **range** of values that can be assigned to **attributes.**
  - For example, a student's name cannot be a numeric value. It has to be alphabetic. A student's age cannot be negative, etc.




# Attributes Types

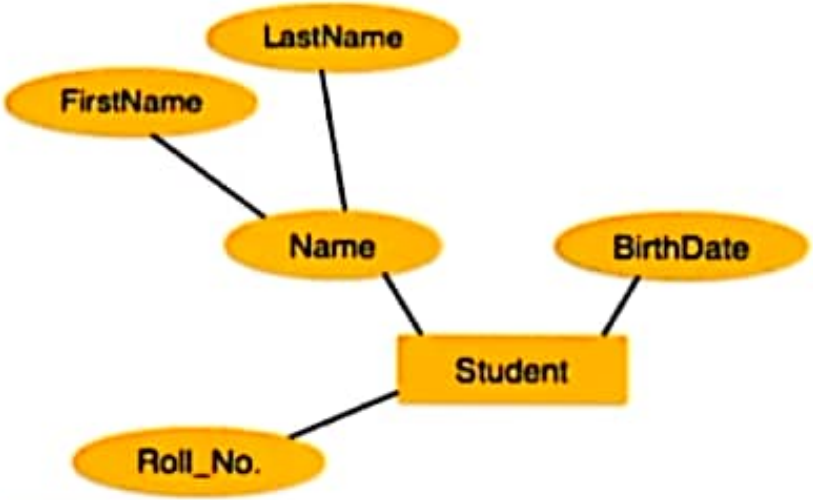
- **Simple** and **Composite attributes**
- **Single-valued** and **Multi-valued attributes**
- **Stored** and **Derived attributes**
- **Key Attribute**

# 1. Simple attributes

- **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 attributes

- **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**



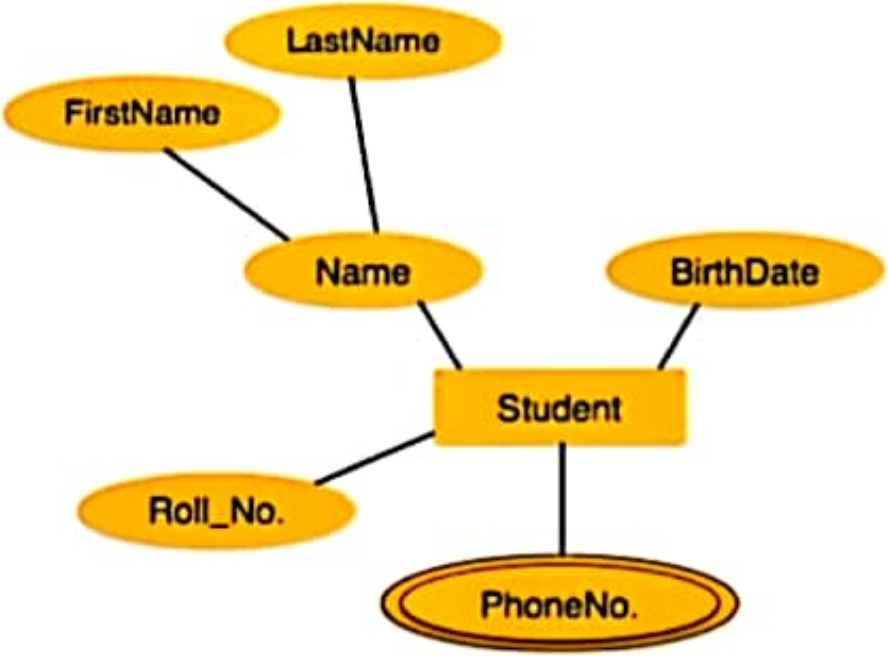

# 3. Single-valued attribute

- **Single-value attributes** contain single value.
- For example:
  - Social_Security_Number, Aadhar_card_no, Roll_no

# 4. Multi-valued attribute

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


- For example: a person can have more than one phone number, email_address, etc.



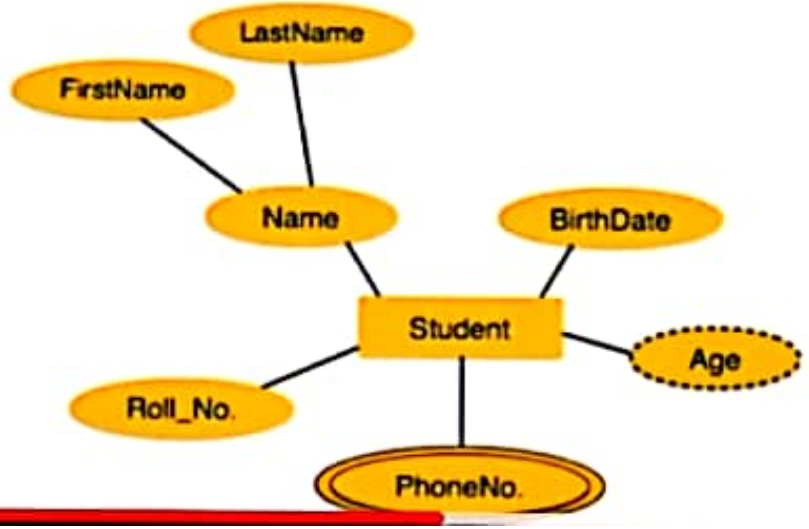

# 5. Stored attribute

- **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

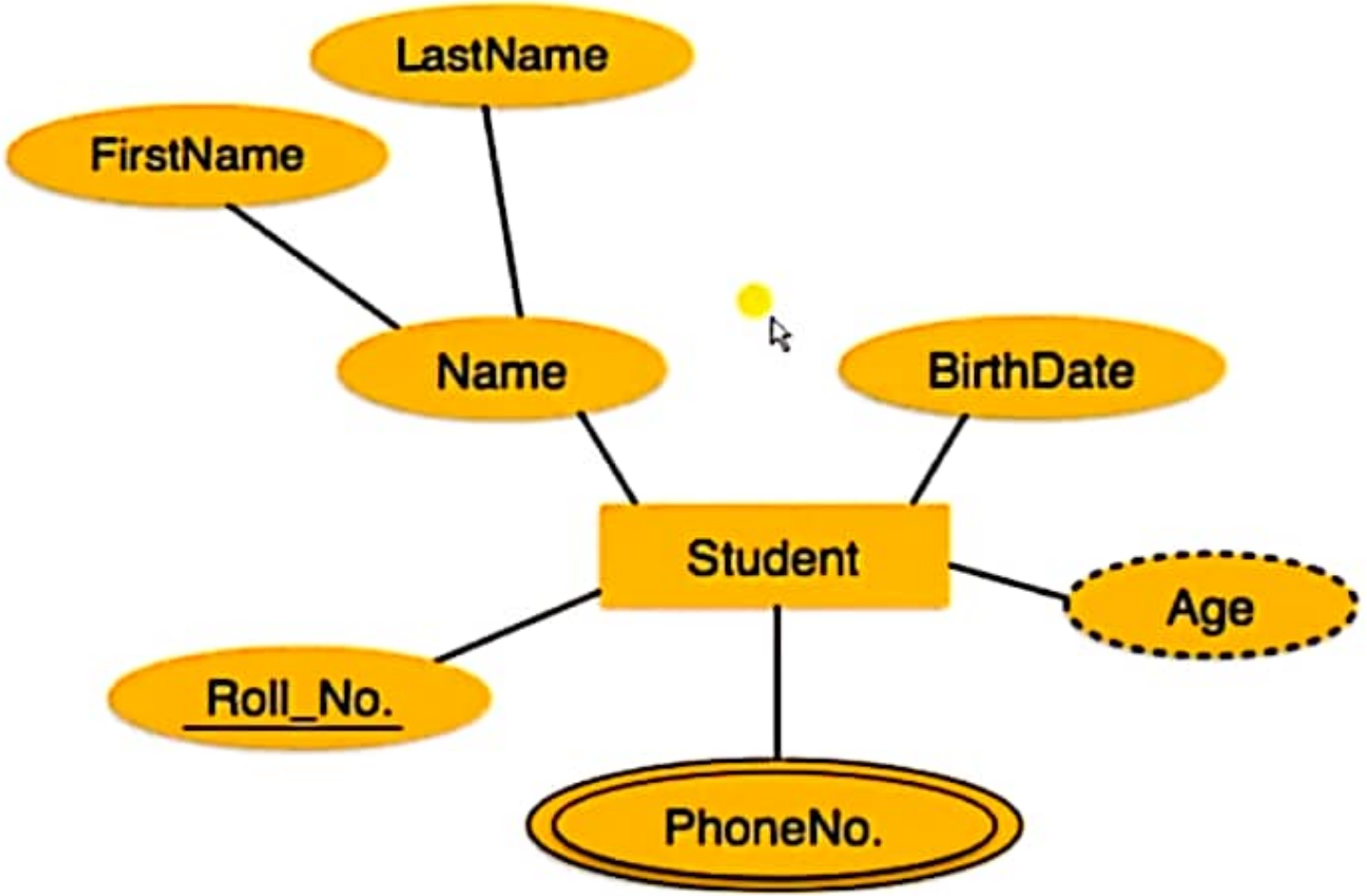
# 6. Derived attribute

- **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.




**E-R diagram** of **Student** entity type with its attributes can be represented as:



# Other possible combinations

These attribute types can come together in a way like:

- simple single-valued attributes
- simple multi-valued attributes
- composite single-valued attributes
- composite multi-valued attributes

# Relationships

- A **Relationship** is an **association** among entities
- For example:
  - an employee **works_at** a department

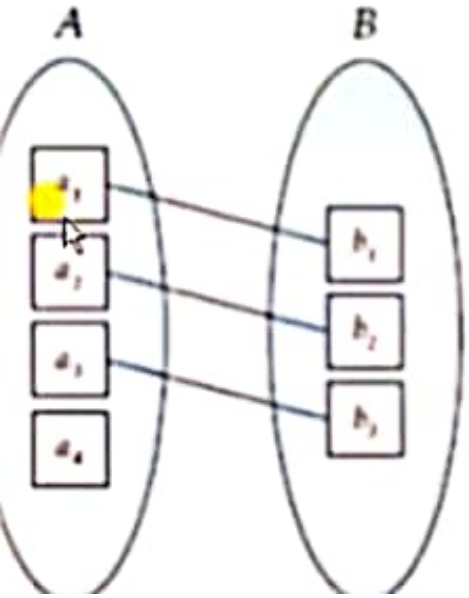
Here, **Works_at** is called **relationships.**

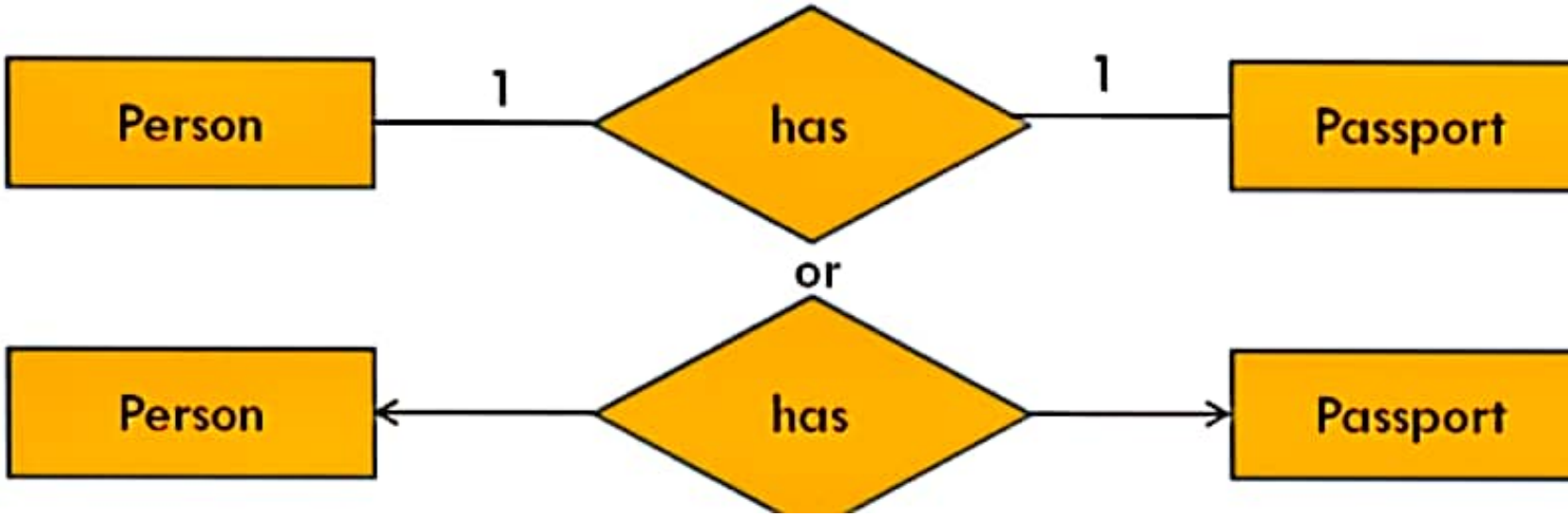
# Mapping Cardinalities (Cardinality Ratio

- ***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 (1-1) Relationship

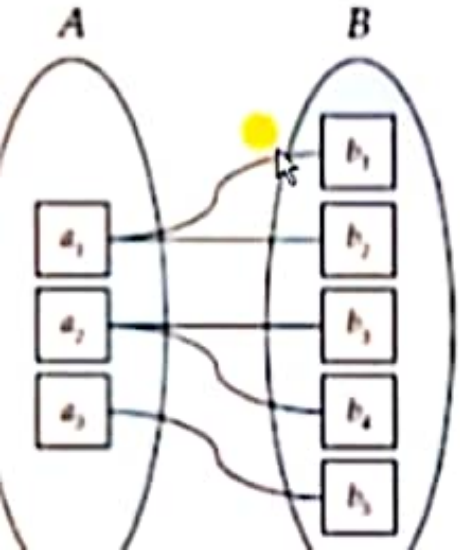
- One entity from entity set A can be associated with *at most* one entity of entity set B and vice versa
  - For example, a person has only one passport and a passport is given to one person.

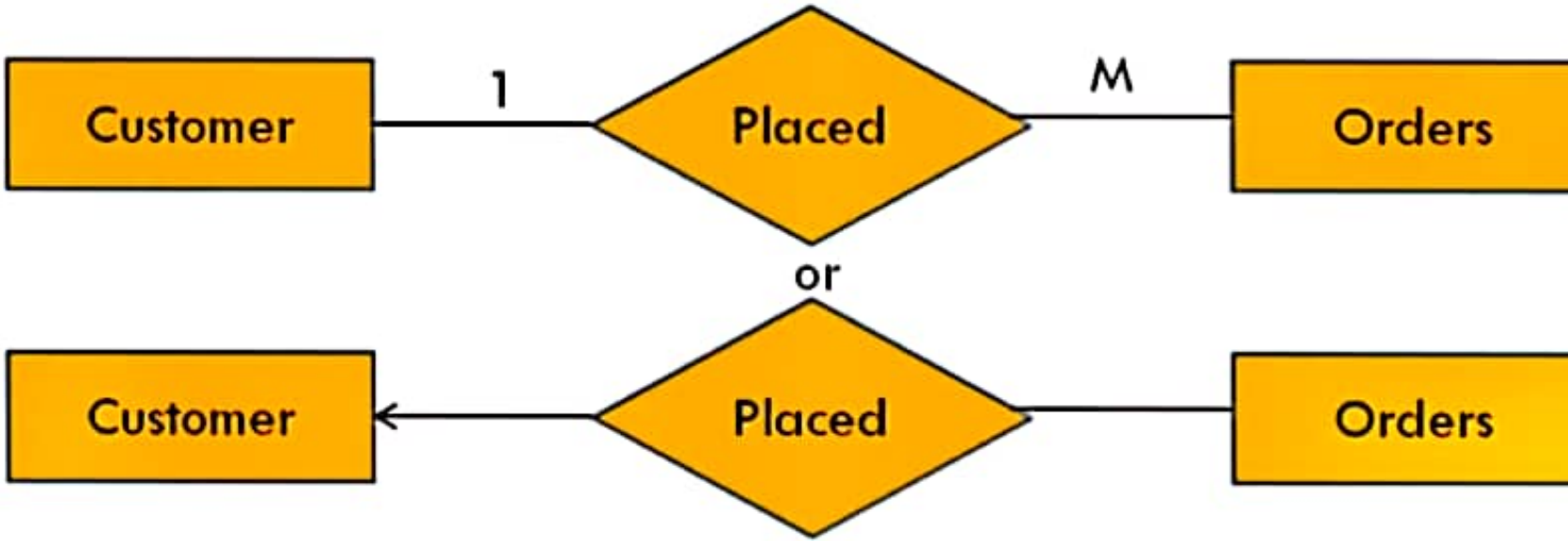


# One-to-Many (1-M) Relationship

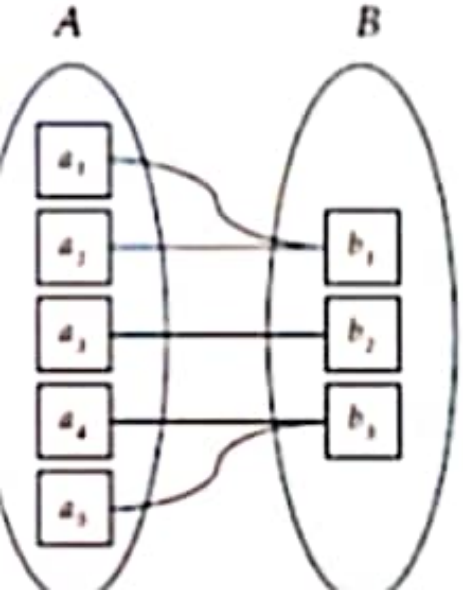
- One entity from entity set A can be associated with more than one entities of entity set B however an entity from entity set B, can be associated with at most one entity
  - For example – a customer can place many orders but a order cannot be placed by many customers.

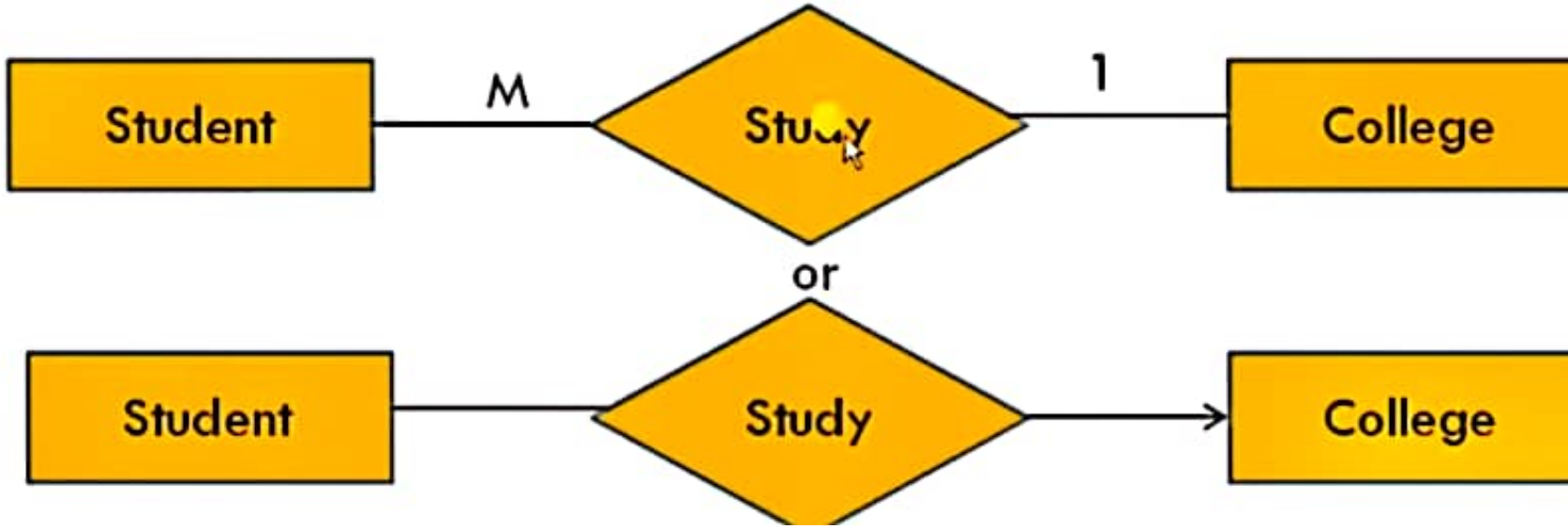


# Many-to-One (M-1) Relationship

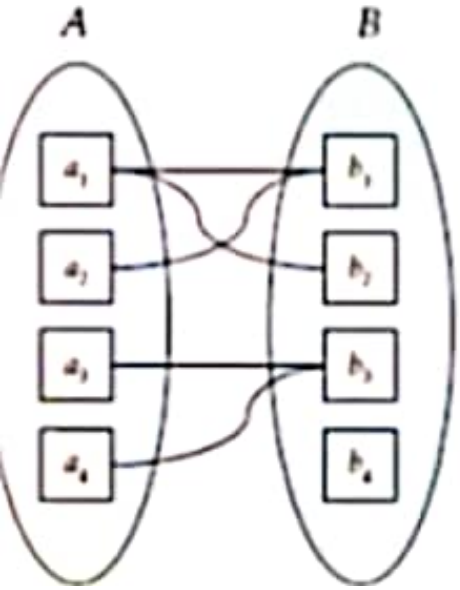
- More than one entities from entity set A can be associated with *at most* one entity of entity set B, however an entity from entity set B can be associated with more than one entity from entity set A
  - For example – many students can study in a single college but a student cannot study in many colleges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

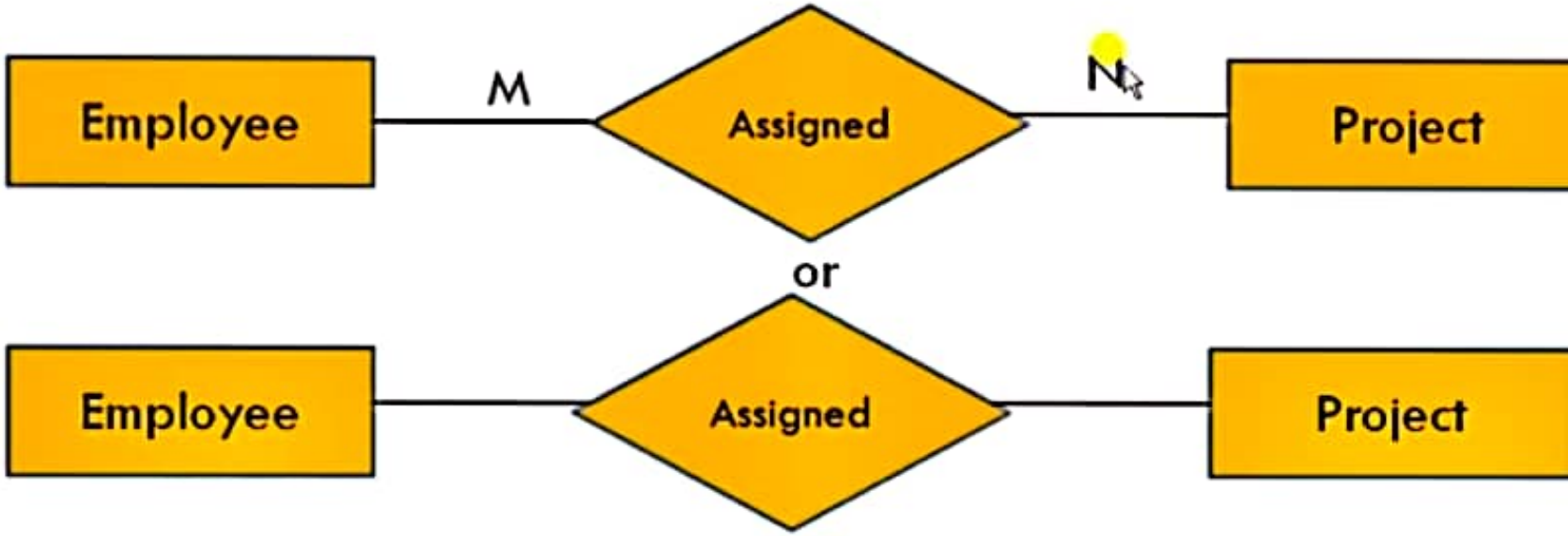


# Many-to-Many (M-N) Relationship

- 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**

- 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**


- 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

**Participation Constraint** is applied on the ***entity*** participating in the relationship set.

- **Total Participation**
- **Partial Participation**

# 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

# Keys in DBMS

- ***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 |

# Keys in DBMS

- ***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.**

# 1. Super Key

- A **super key** is a combination of all possible attributes that can uniquely identify the rows (or tuple) in the given relation.
  - 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 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 |

**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, Dep_Id}
10. {Emp_Id,Name, Aaadhar_No, Email_Id, Dept_Id}, etc....

# 2. 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

| 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}

# 3. 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.

| 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 |

# 4.Alternate Keys

- 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.**

| 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 |

- **Alternate Keys**
  1. {Aadhar_No}
  2. {Email_Id}

# 5. Foreign keys

- 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 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 |

oreign Key:

In Employee Table

1. Dept_Id

Primary Key

Department Table (Referenced relation)

| Dept_Id | Dept_Name |
|---|---|
| 1 | Sales |
| 2 | Marketing |
| 3 | HR |

# 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.

# 6. 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}

# Entity

- An **Entity** is a **"thing" or "object"** in the real world that is distinguishable from other objects. In ER Diagram, an **entity** is represented using rectangles.
  - Consider an example of an Organisation- Employee, Manager, Department, Product can be taken as entities in an Organisation.


- Two types of **Entities** in ER Diagram:

1. **Strong entity**
2. **Weak entity**

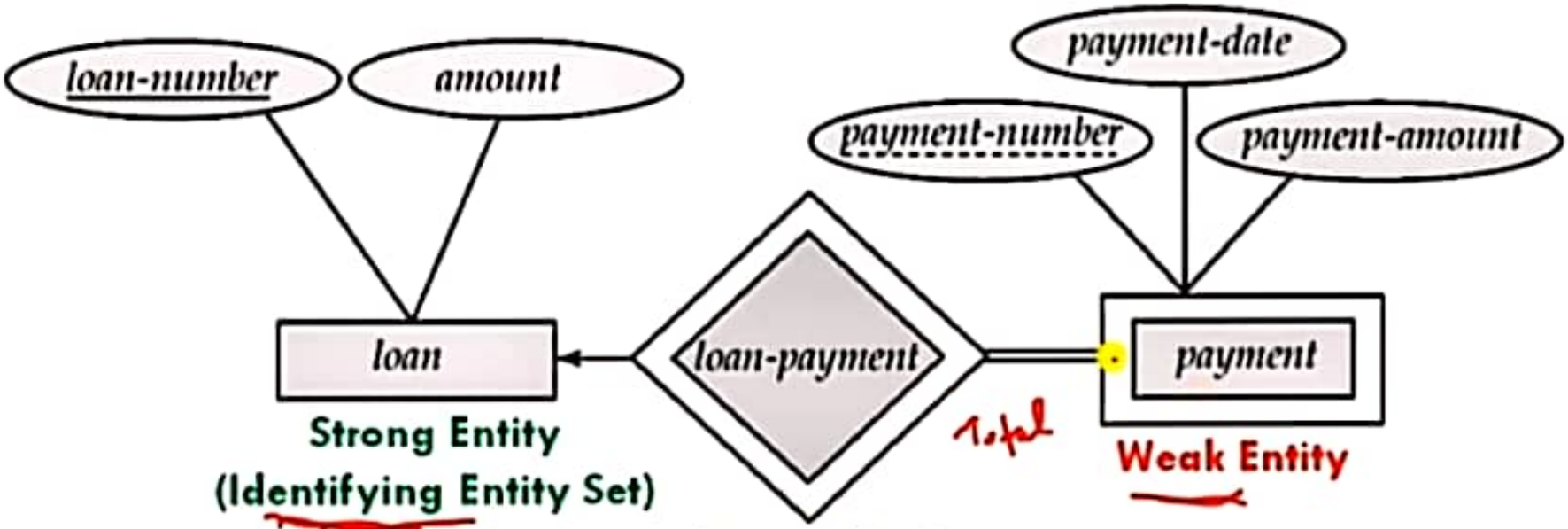
# Strong Entity (Strong Entity Set)

- 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*

# Weak Entity (Weak Entity Set)

- 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:

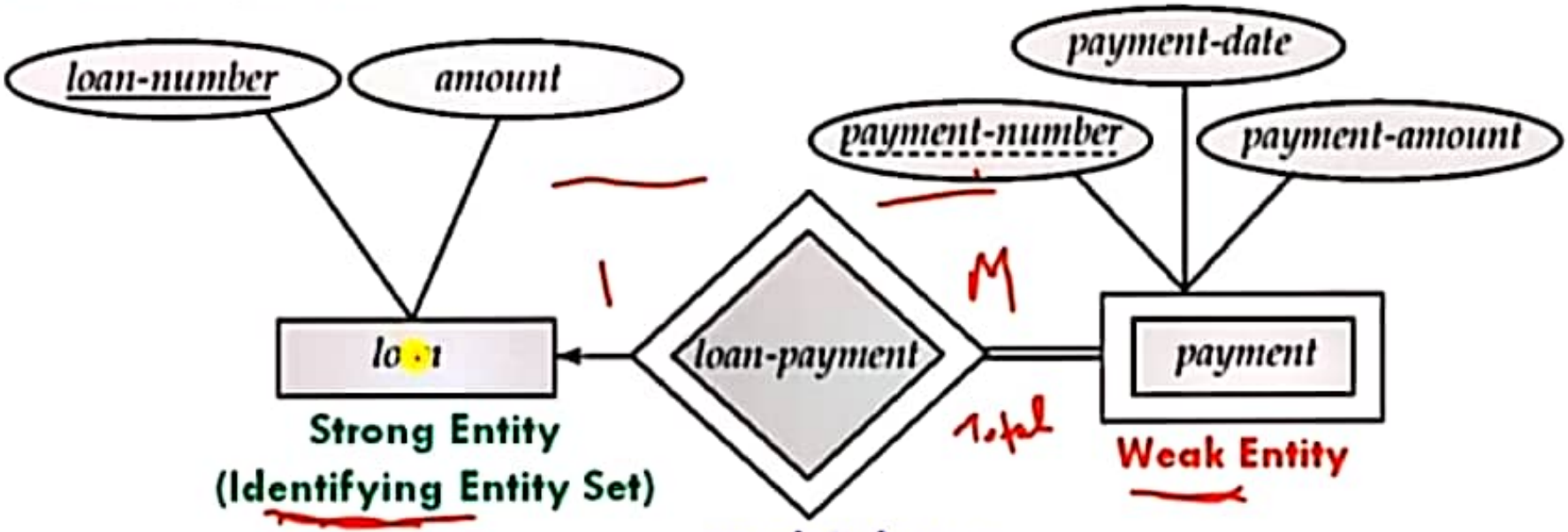
# Example 1




| Loan_no | Amount |
|---|---|
| L1 | 1,00,000 |
| L2 | 2,00,000 |
| L3 | 3,00,000 |

| Payment_no | Payment_date | Payment_amount |
|---|---|---|
| 1 | 05-06-2020 | 5000 |
| 1 | 08-07-2020 | 10000 |
| 1 | 10-08-2020 | 15000 |
| 2 | 05-07-2020 | 5000 |
| 2 | 08-08-2020 | 10000 |
| 2 | 10-09-2020 | 15000 |

# Example 1



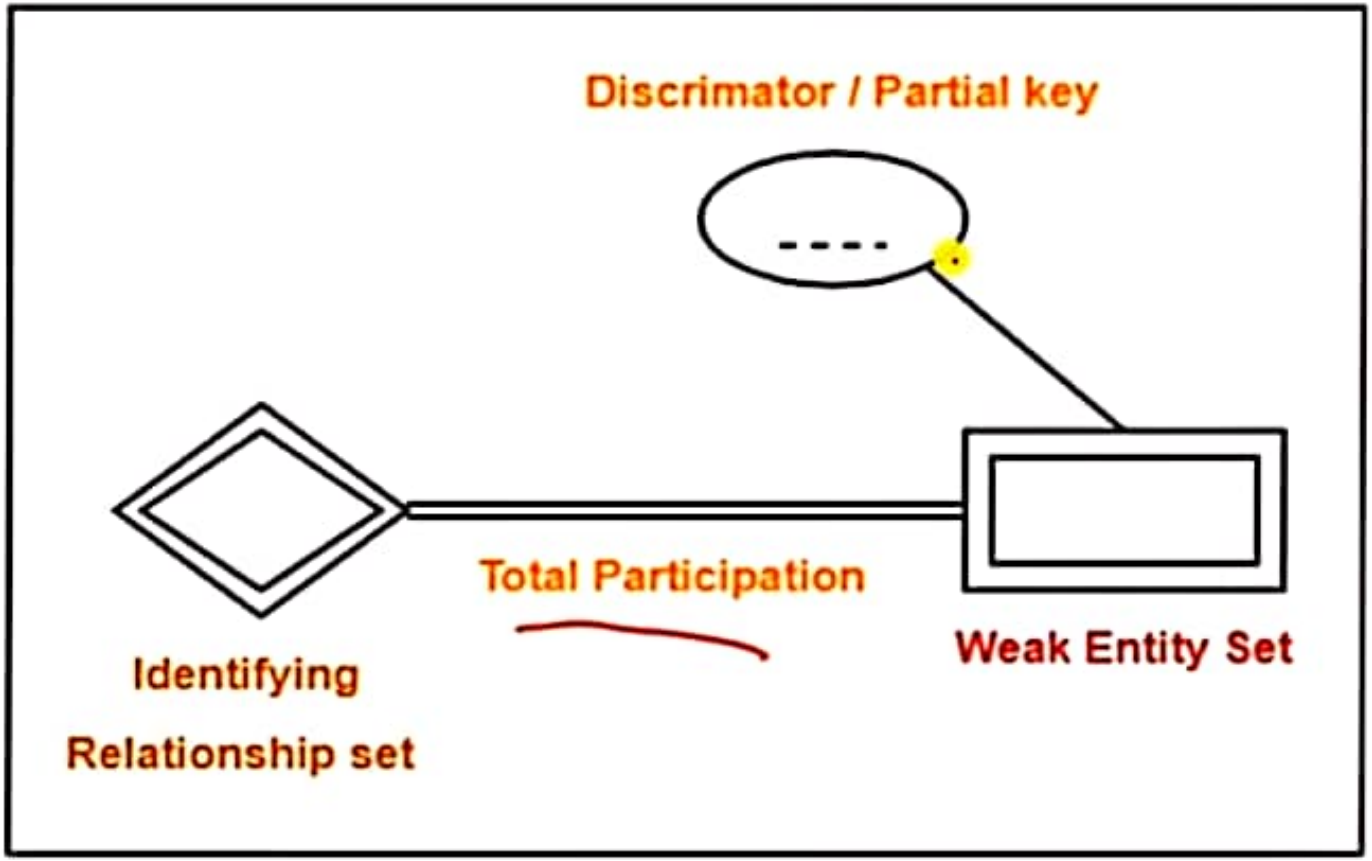

| Loan_no | Amount |
|---|---|
| L1 | 1,00,000 |
| L2 | 2,00,000 |
| L3 | 3,00,000 |

| Payment_no | Payment_date | Payment_amount |
|---|---|---|
| 1 | 05-06-2020 | 5000 |
| 1 | 08-07-2020 | 10000 |
| 1 | 10-08-2020 | 15000 |
| 2 | 05-07-2020 | 5000 |
| 2 | 08-08-2020 | 10000 |
| 2 | 10-09-2020 | 15000 |

# Weak Entity Set

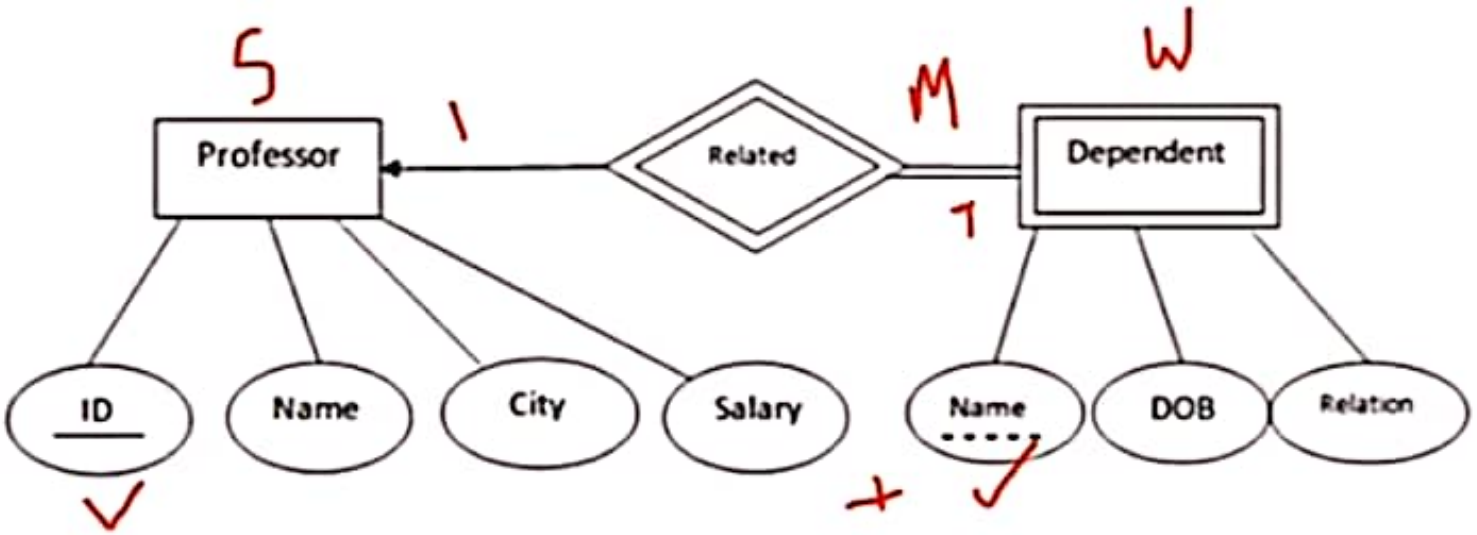
- The **existence of a weak entity set depends** on the existence of a **strong entity set** (called ***identifying entity set)***
- *The **relationship** associating the weak entity set with the strong (or identifying) entity set is called **identifying relationsh***
  - Identifying relationship depicted using a **double diamond**
- The **participation of Weak entity set** from the identifying relation set is always **Total**
- The Identifying relationship is **One-to-Many Relationship** from the ***identifying entity set to the weak entity set***
  - *1-M relationship from **Loan to Payment***
- The ***discriminator (or partial key)*** of a weak entity set is the set of attributes that distinguishes among all the entitie of a weak entity set.
  - The discriminator of a weak entity set us underline with a **dashed line**
- The **primary key of a weak entity set** is formed by the **primary key of the strong entity set** on which the weak entity set is existence dependent, **plus** the **weak entity set's discriminator**.
  - **Primary Key** of Weak Entity **Payment { loan_no, Payment_no }**

# Representation



# Example 2

- The **Professor** is a **Strong Entity**, whereas **Dependent** is a **Weak Entity.**




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

| S.NO | 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 |

# Extended Entity Relationship (ER) Features

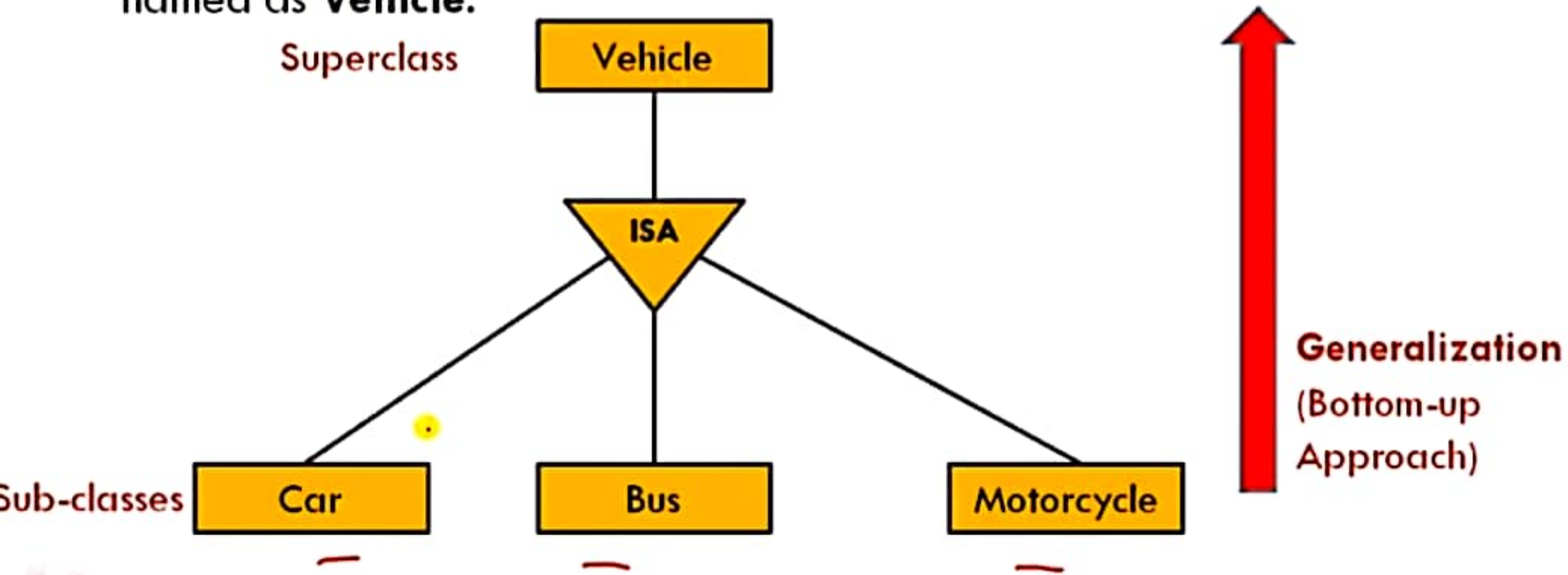
- As the complexity of data increased in the late 1980s, it became more and more difficult to use the traditional ER Model for database modelling. Hence some improvements or enhancements were made to the existing ER Model to make it able to handle the complex applications better.
- Hence, as part of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

# Generalization

- **Generalization** is the process of extracting common properties from a set of entities and create a generalized entity from it.
- **Generalization is a "bottom-up approach"** in which two or more entities can be combined to form a higher level entity if they have some attributes in common.
  - subclasses are combined to make a superclass.
- **Generalization is used** to emphasize the similarities among lower-level entity set and to hide differences in the schema

# Example: 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.**

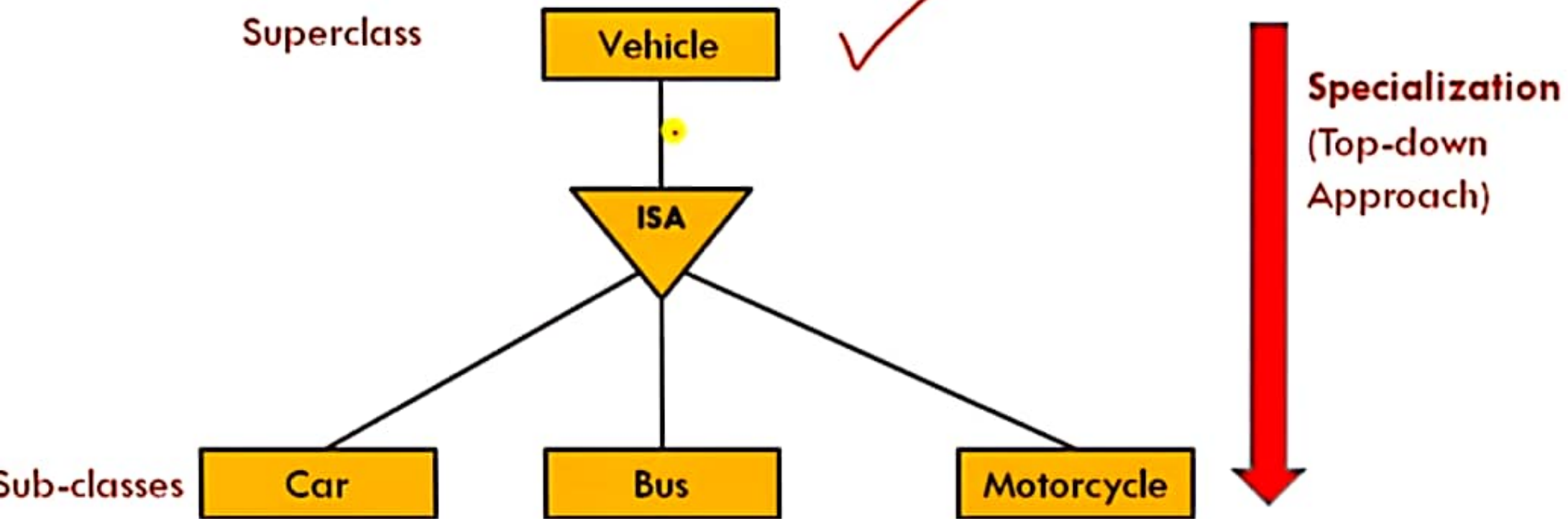


# Specialization

- 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**

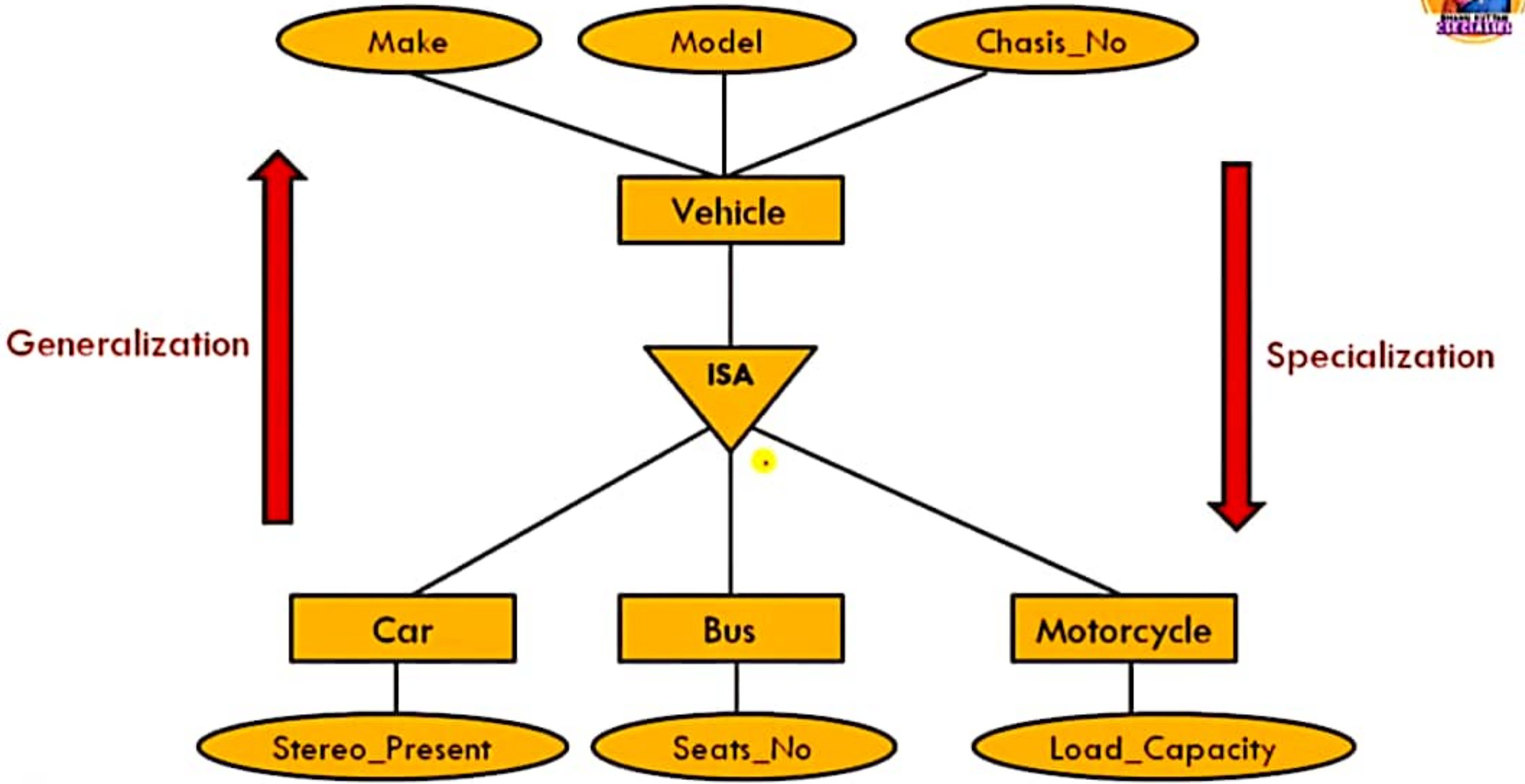
# Example: Specialization

- **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.



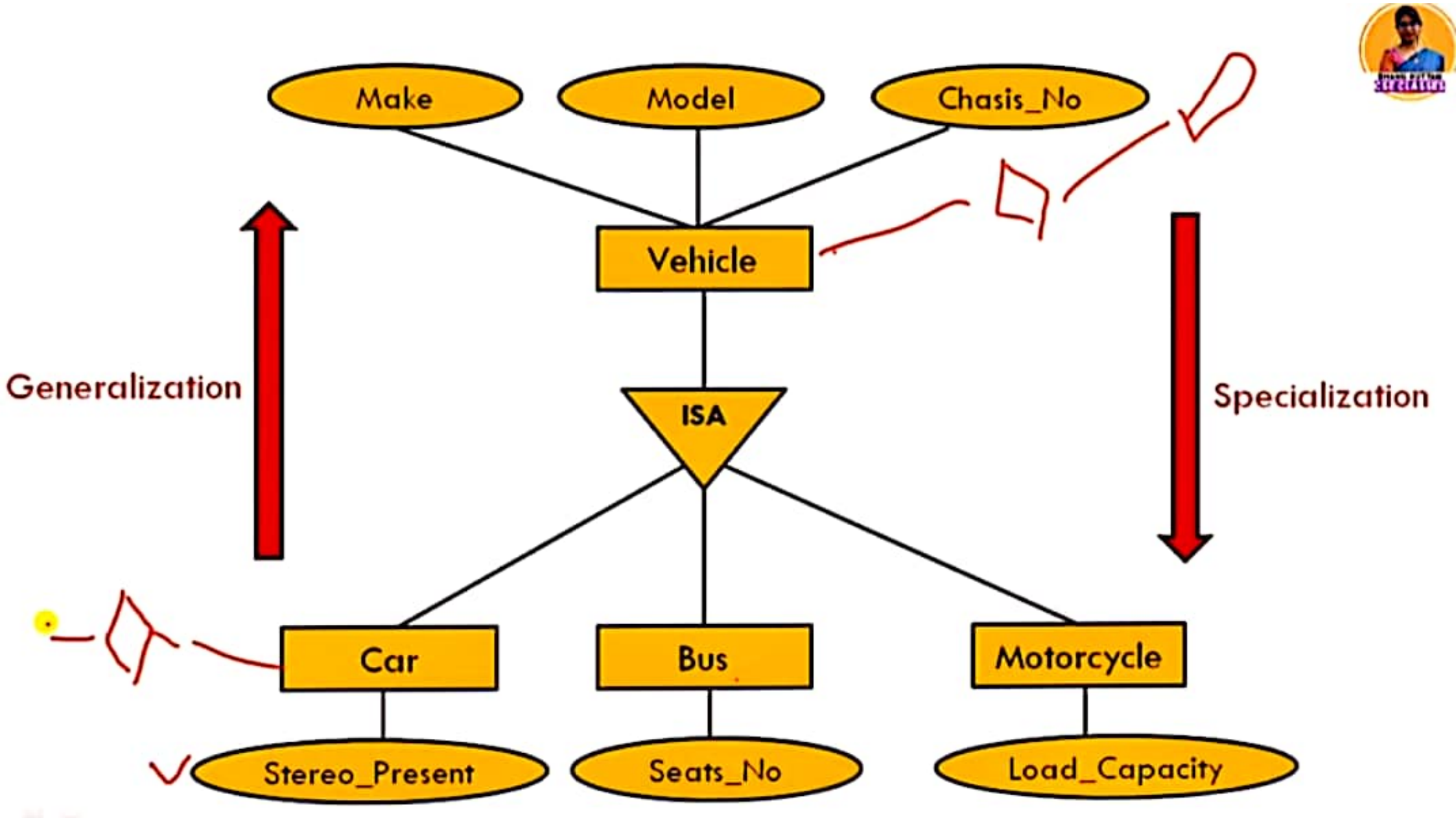
# Inheritance

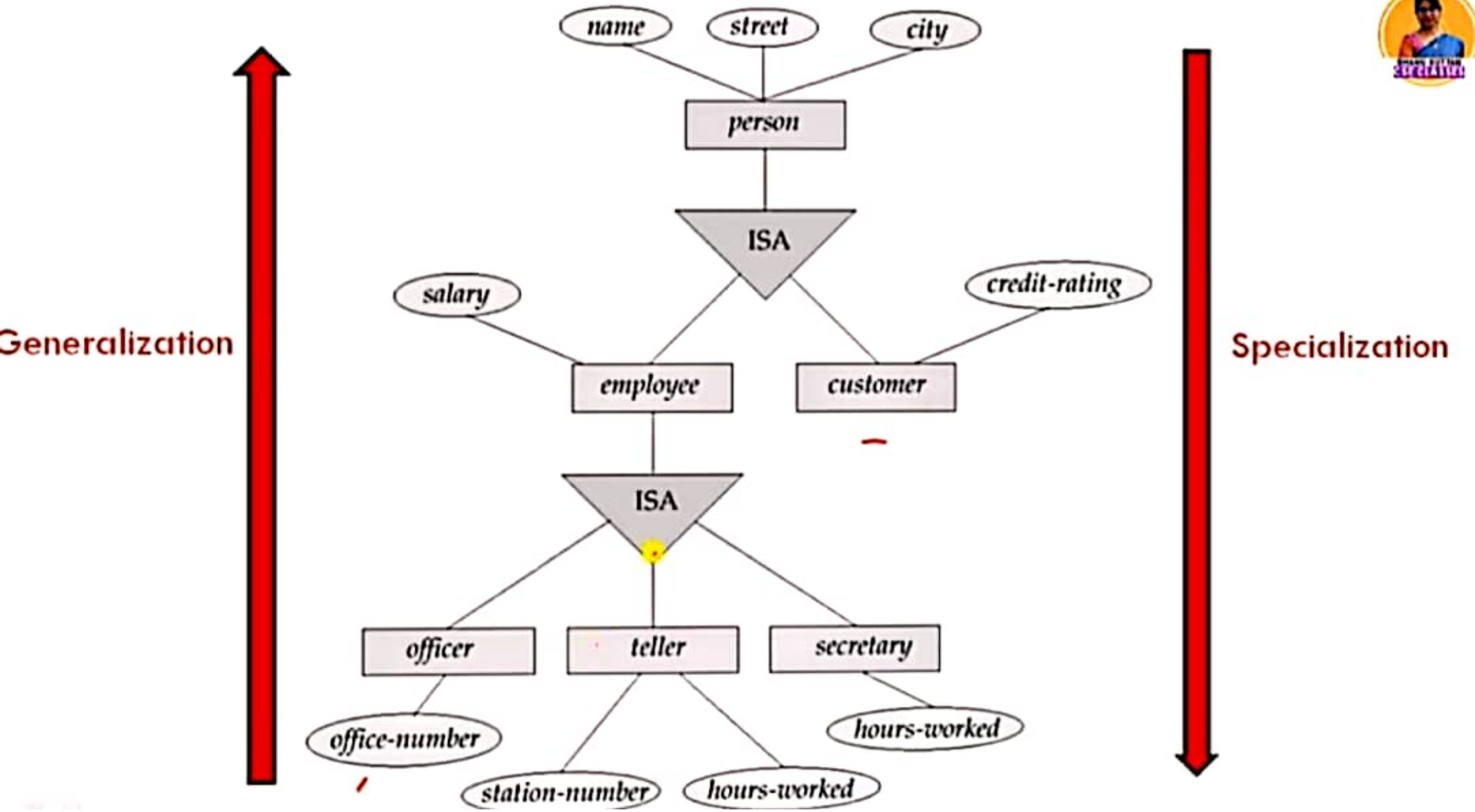
- **Inheritance** is an important feature of generalization and specialization.
- **Attribute inheritance allows lower level entities to *inherit the attributes* of higher level entities.**
  - **For example,** Consider relations Car and Bus inheriting the attributes of Vehicle. Thus, Car is described by attributes of super-class Vehicle as well as its own attributes.
- This also extends to **Participation Inheritance** in which ***relationships involving higher-level entity-sets are also inherited by lower-level entity-sets.***
  - A lower-level entity-set can participate in its own relationship-sets, too


Make
Model
Chasis_No
Vehicle
Generalization
ISA
Specialization
Car
Bus
Motorcycle
Stereo_Present
Seats_No
Load_Capacity

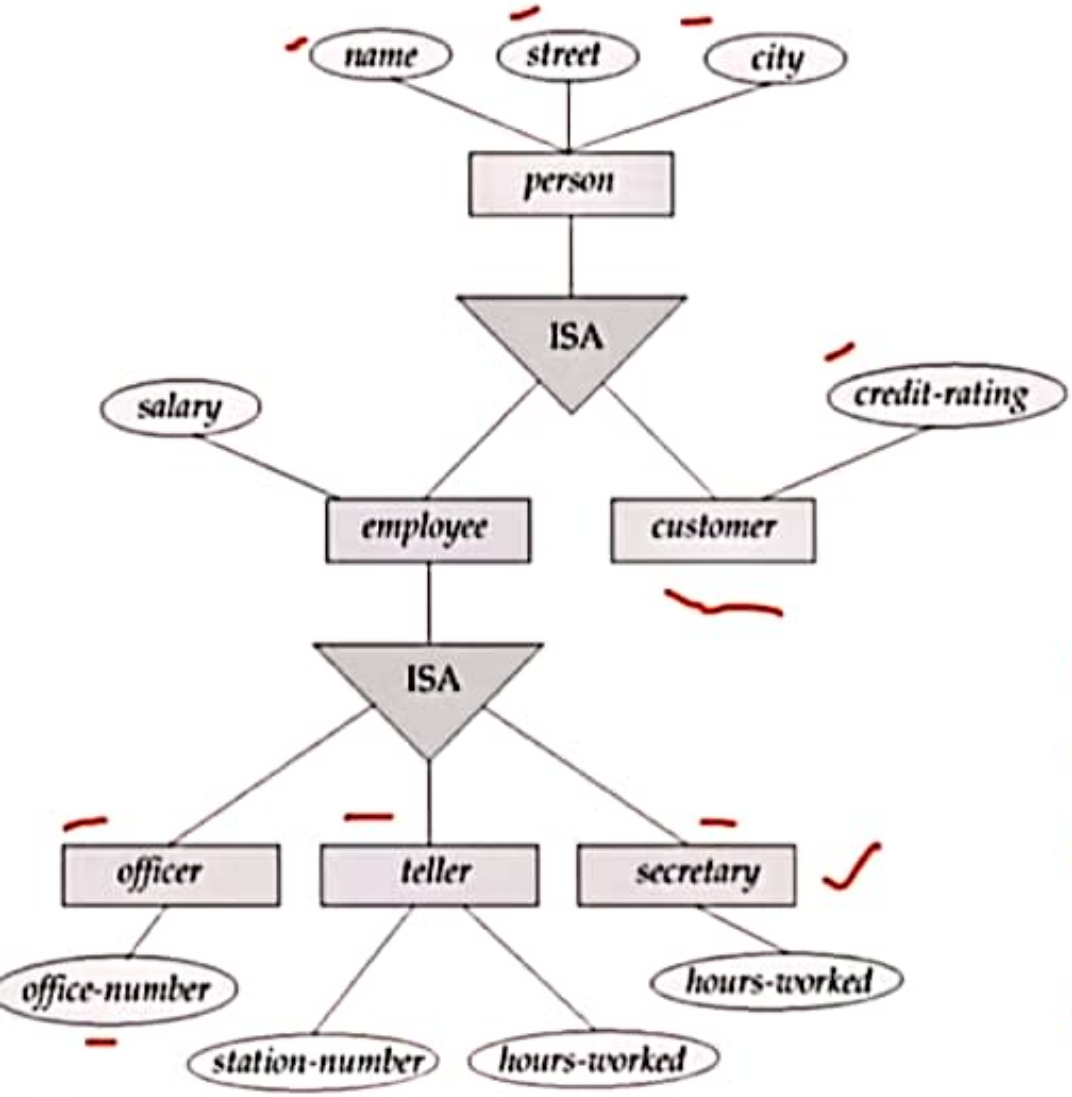
# Inheritance

- **Inheritance** is an important feature of generalization and specialization.
- **Attribute inheritance allows lower level entities to *inherit the attributes* of higher level entities.**
  - **For example,** Consider relations Car and Bus inheriting the attributes of Vehicle. Thus, Car is described by attributes of super-class Vehicle as well as its own attributes.
- This also extends to **Participation Inheritance** in which ***relationships involving higher-level entity-sets are also inherited by lower-level entity-sets.***
  - A lower-level entity-set can participate in its own relationship-sets, too


Make
Model
Chasis_No
Vehicle
ISA
Generalization
Specialization
Car
Bus
Motorcycle
Stereo_Present
Seats_No
Load_Capacity


Generalization
Specialization
name
street
city
person
ISA
salary
credit-rating
employee
customer
ISA
officer
teller
secretary
office-number
station-number
hours-worked
hours-worked

# How Schema or Tables can be formed?




Four tables can be formed:

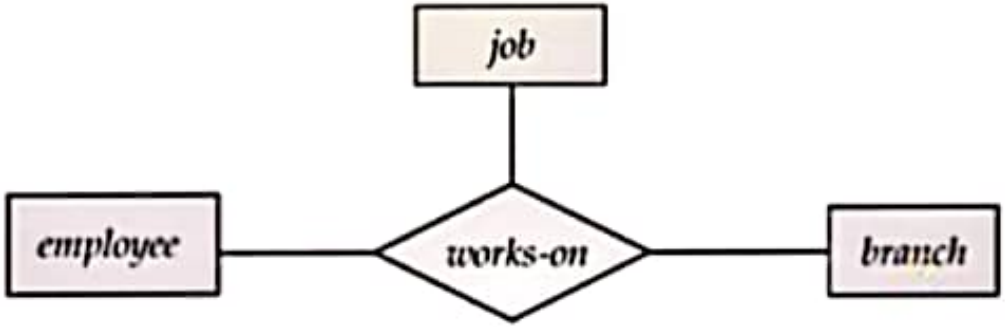
1. **customer** (name, street, city, credit_rating)
2. **officer** (name, street, city, salary, office_number)
3. **teller** (name, street, city, salary, station_number, hours_worked)
4. **secretary** (name, street, city, salary, hours_worked

# 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**

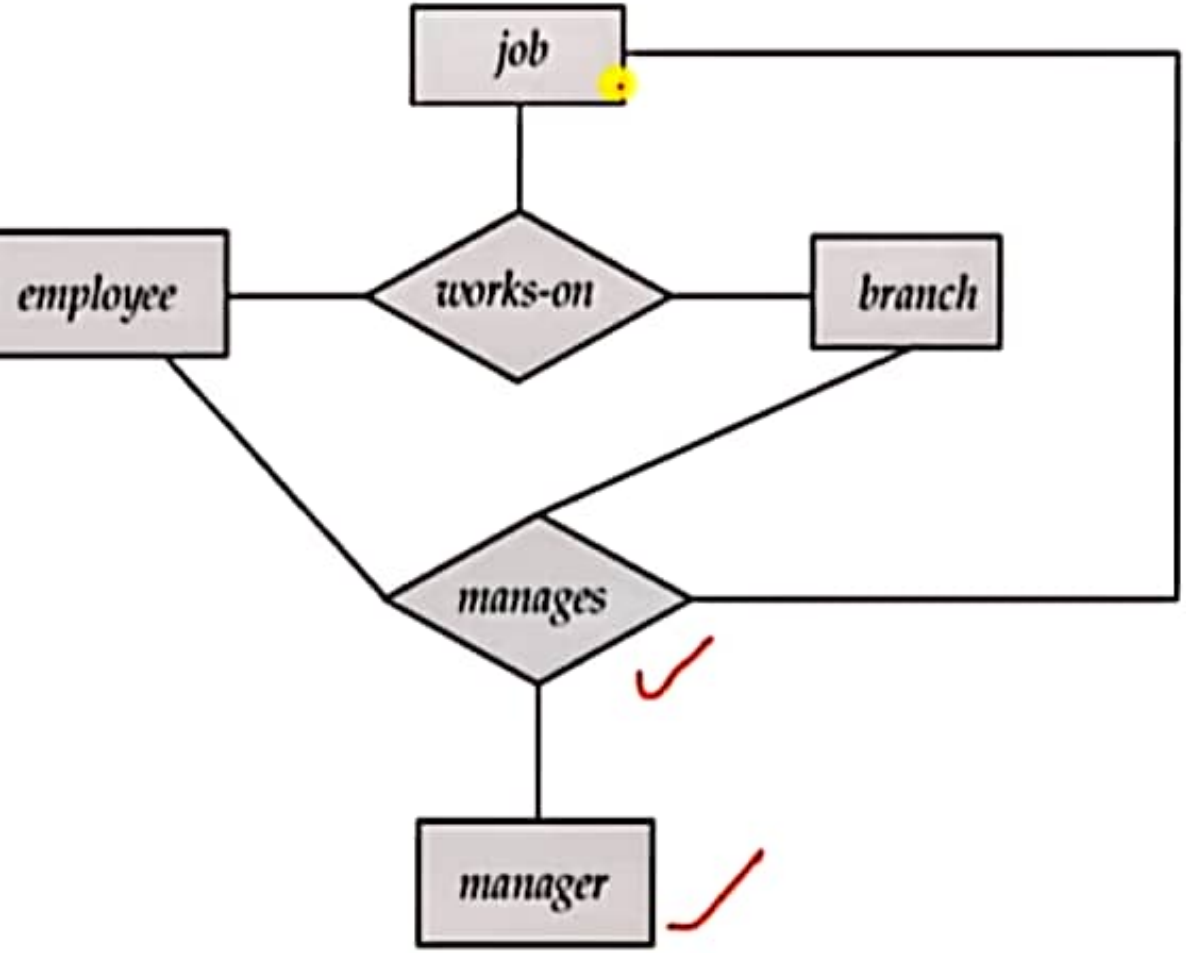
# Example: (Relationship of Relations)

- 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: (Redundant Relationship)




ER Diagram with Redundant Relationship

- Relationship sets ***works-on*** and ***manages*** represent overlapping (redundant) information
  - Every *manages* relationship corresponds to a *works-on* relationshi
  - However, some *works-on* relationships may not correspond to any *manages* relationships
    - So we can't discard the *works-* relationship

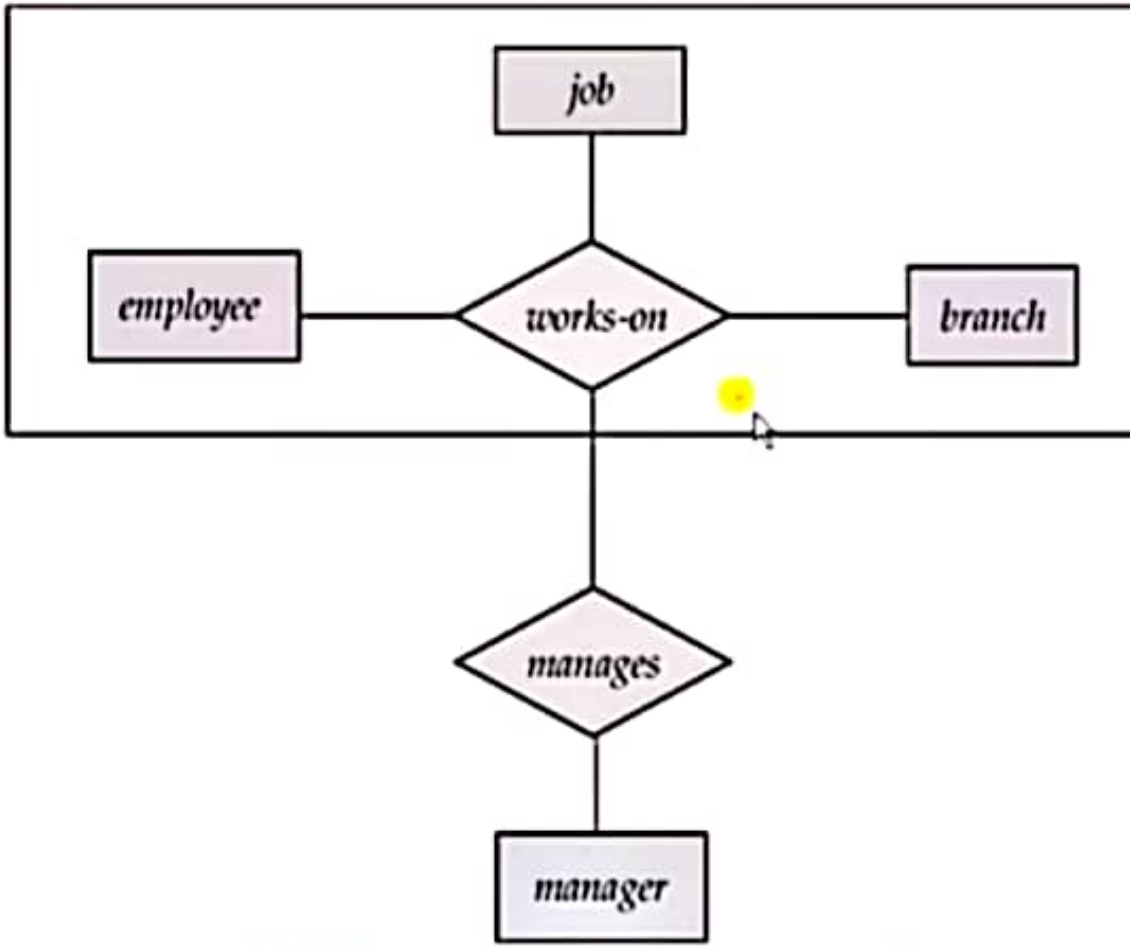
# Aggregation

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

# Example: (Aggregation)

With **Aggregation** (without introducing redundancy) the ER diagram can be represented as:

- An employee works on a particular job at a particular branch
- An employee, branch, job combination may have an associated manager

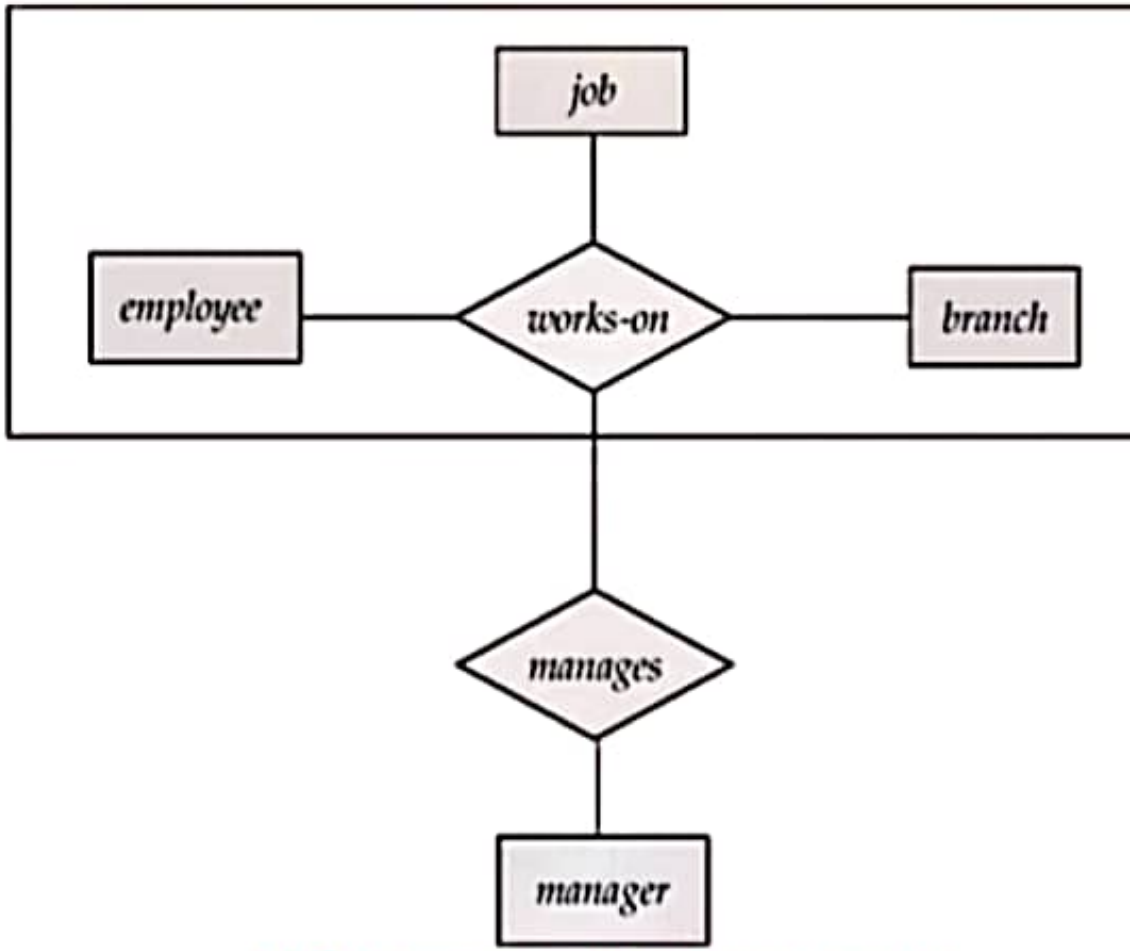



ER Diagram with Aggregation

# Example: (Aggregation)

With **Aggregation** (without introducing redundancy) the ER diagram can be represented as:

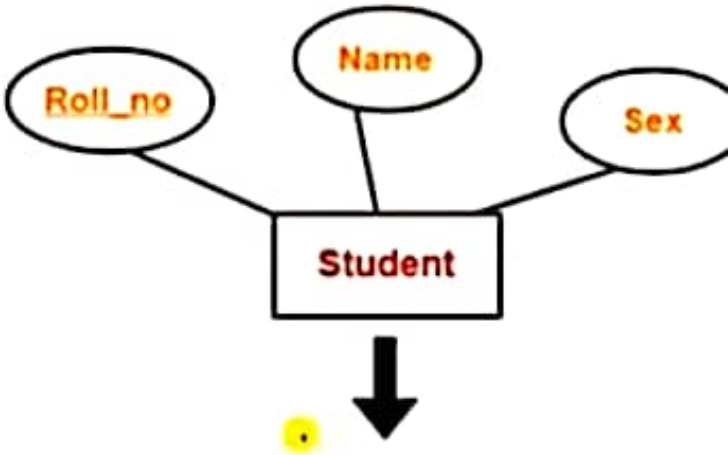
- An employee works on a particular job at a particular branch
- An employee, branch, job combination may have an associated manager




ER Diagram with Aggregation

# Rule1: Strong Entity Set With Only Simple Attributes

- A strong entity set with only ***simple attributes*** will require only **one table** in relational model.
  - Attributes of the table will be the attributes of the entity set.
  - The primary key of the table will be the key attribute of the entity set.

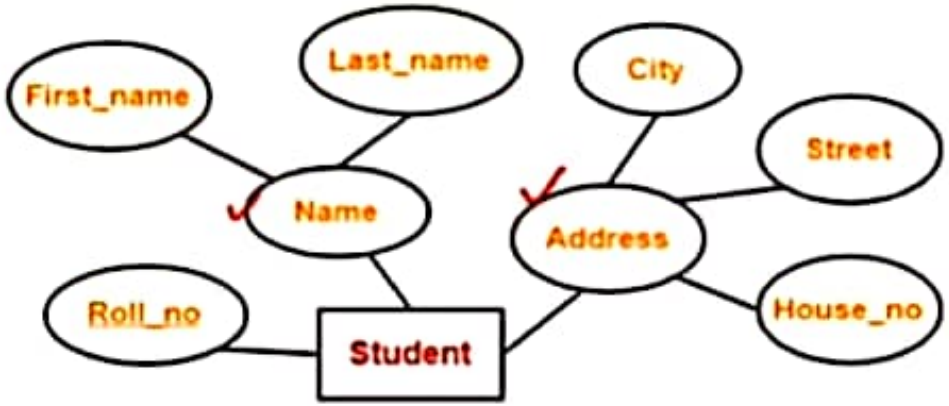



| Roll_no | Name | Sex |
|---|---|---|
| | | |

Schema : Student ( Roll_no , Name , Sex )

# Rule 2: Strong Entity Set With Composite Attributes

- A strong entity set with any number of ***composite attributes*** will require **only one table** in relational model.
- While conversion, simple attributes of the composite attributes are taken into account and not the composite attribute itself.



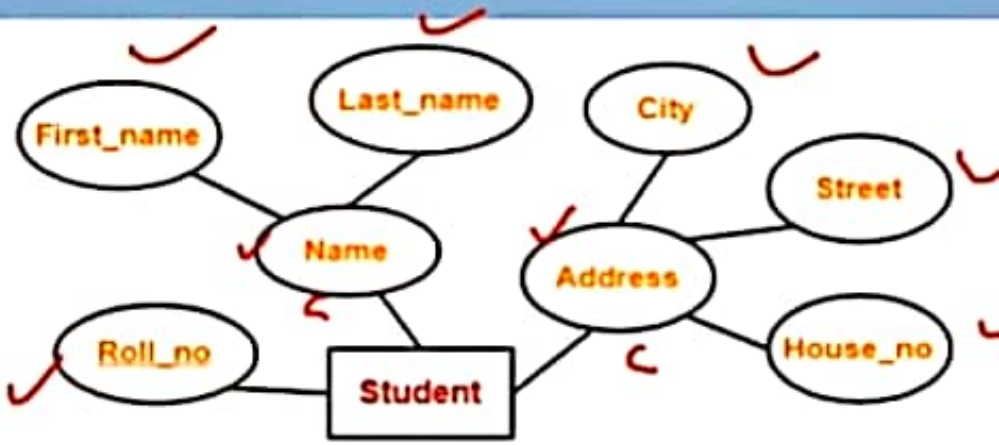

1

| Roll_no | First_name | Last_name | House_no | Street | City |
|---|---|---|---|---|---|
| | | | | | |

Schema : Student ( Roll_no , First_name , Last_name , House_no , Street , City )

# Rule 2: Strong Entity Set With Composite Attributes

- A strong entity set with any number of ***composite attributes*** will require **only one table** in relational model.
- While conversion, simple attributes of the composite attributes are taken into account and not the composite attribute itself.

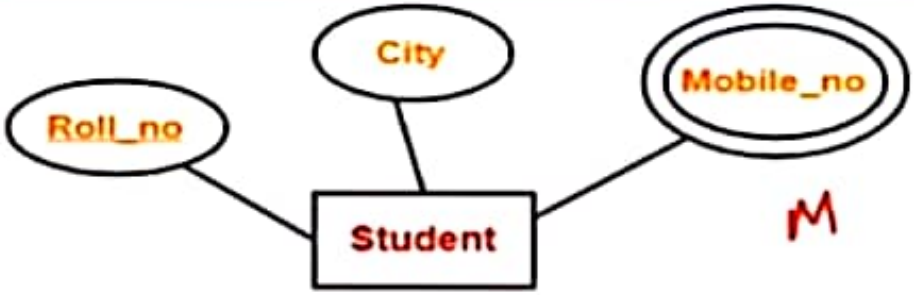



| Roll_no | First_name | Last_name | House_no | Street | City |
|---|---|---|---|---|---|
| | | | | | |

Schema : Student ( Roll_no , First_name , Last_name , House_no , Street , City )

# Rule 3: Strong Entity Set With Multi Valued Attributes

- A strong entity set with any number of ***multi-valued attributes*** will require **two tables** in relational model.
  - One table will contain all the simple attributes with the primary key.
  - Other table will contain the primary key and all the multi valued attributes.



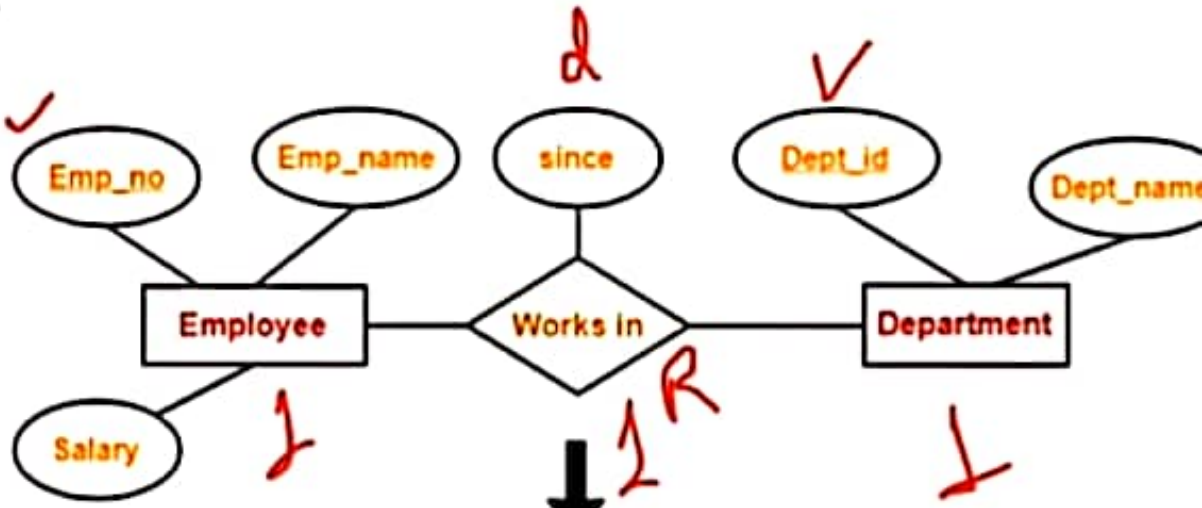

| Roll_no | City |
|---|---|
| | |

M

| Roll_no | Mobile_no |
|---|---|
| | |

# Rule 4: Translating Relationship Set into a Table

- A ***relationship set*** will require **one table** in the relational model.
- Attributes of the table are:
  - Primary key attributes of the participating entity sets
  - Its own descriptive attributes if any.
- Set of non-descriptive attributes will be the primary key
- For given ER diagram, three tables will be required in relational model-
  - One table for the entity set "Employee"
  - One table for the entity set "Department"
  - One table for the relationship set "Works in"



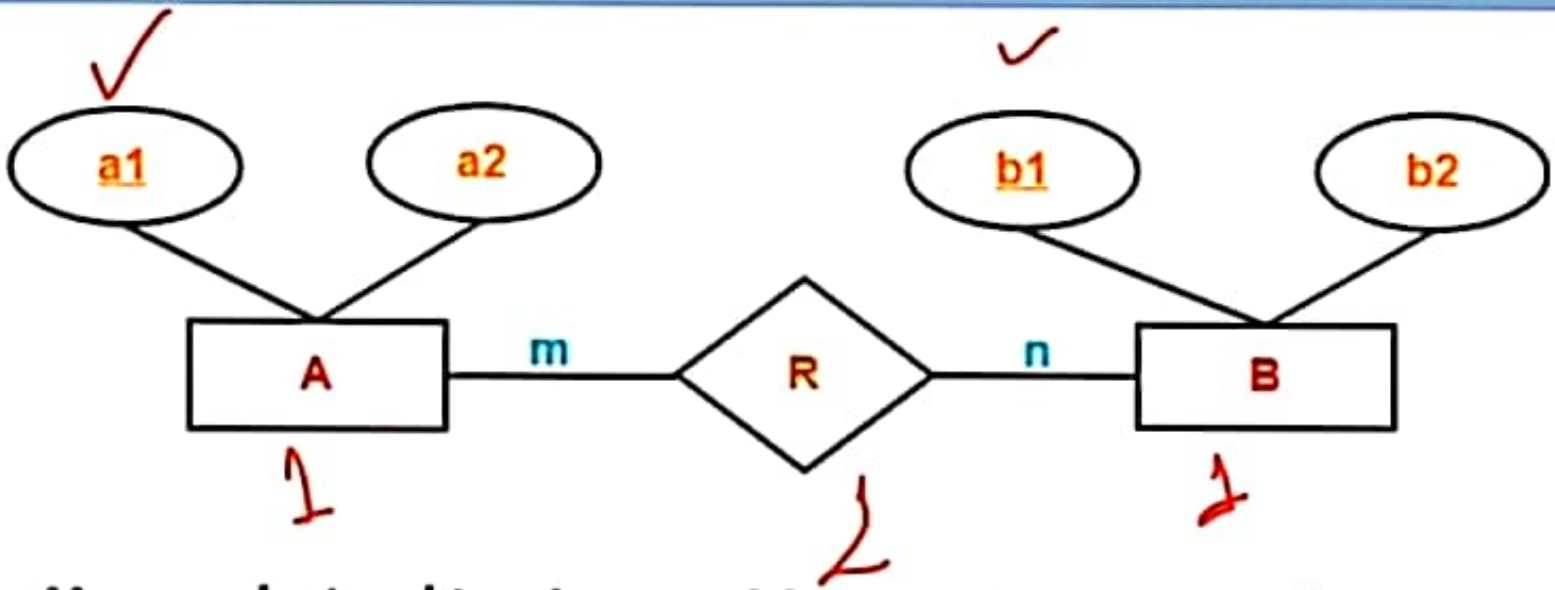

| Emp_no | Dept_id | since |
|---|---|---|
| | | |

Schema : Works in ( Emp_no , Dept_id , since )

# Rule 5: For Binary Relationships With Cardinality Ratios

- The following four cases are possible-
  - **Case-1:** Binary relationship with cardinality ratio m:n
  - **Case-2:** Binary relationship with cardinality ratio 1:n
  - **Case-3:** Binary relationship with cardinality ratio m:1
  - **Case-4:** Binary relationship with cardinality ratio 1:1

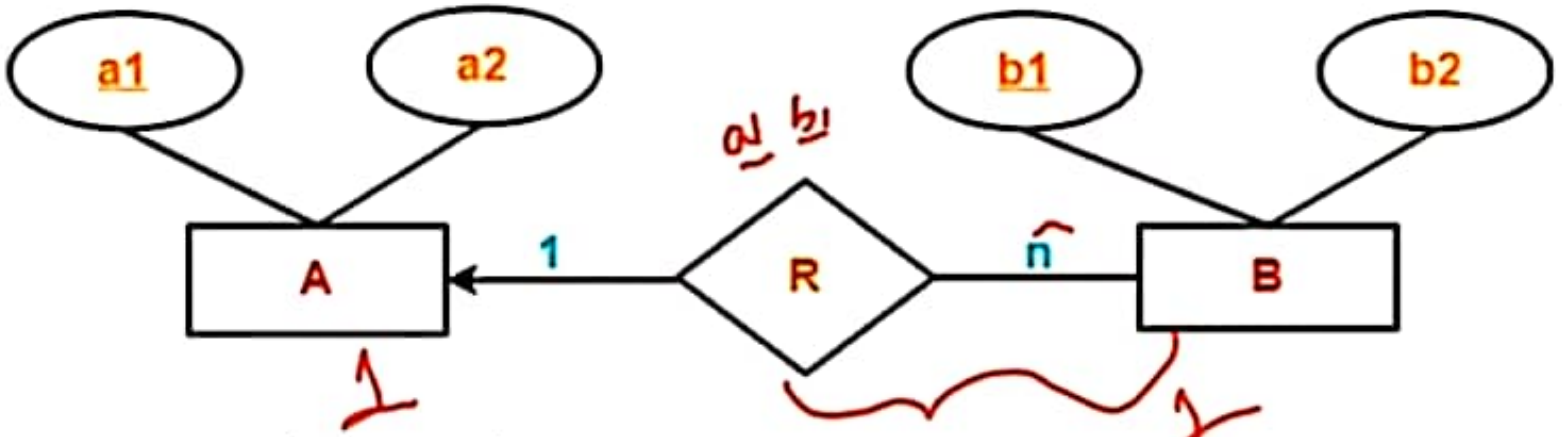
# Case-1: For Binary Relationship with Cardinality Ratio m:n




In ***Many-to-Many relationship***, **three tables** will be required-

1. **A ( a1 , a2 )**
2. **R ( a1 , b1 )**
3. **B ( b1 , b2 )**

# Case-2: For Binary Relationship with Cardinality Ratio 1:n

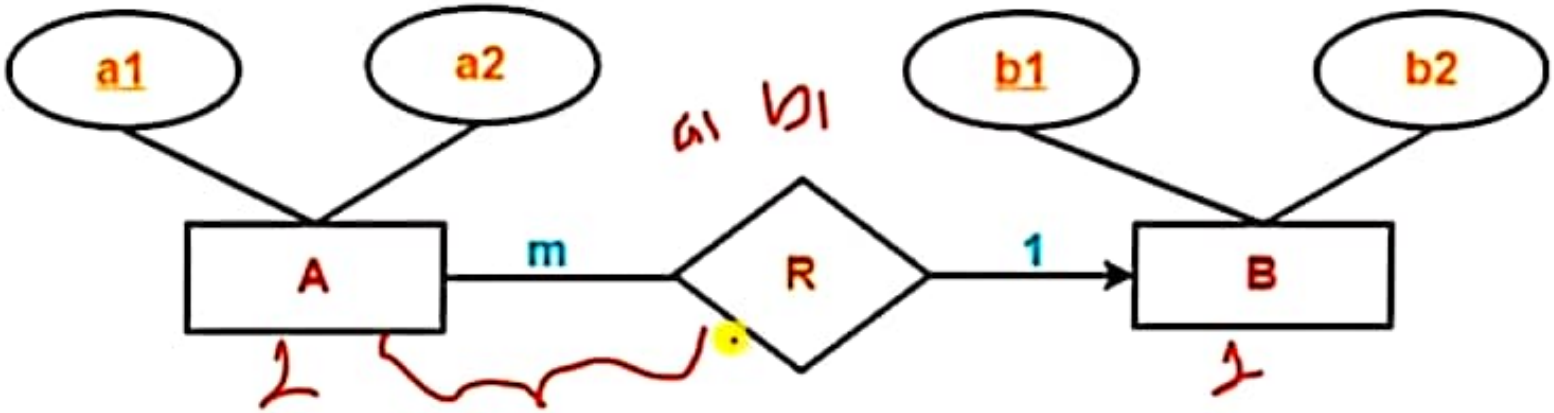



In ***One-to-Many relationship***, **two tables** will be required-

1. **A ( a1 , a2 )**
2. **BR ( b1 , b2, a1 )**

**NOTE** - Here, combined table will be drawn for the entity set B and relationship set R.

# Case-3: For Binary Relationship with Cardinality Ratio m:1

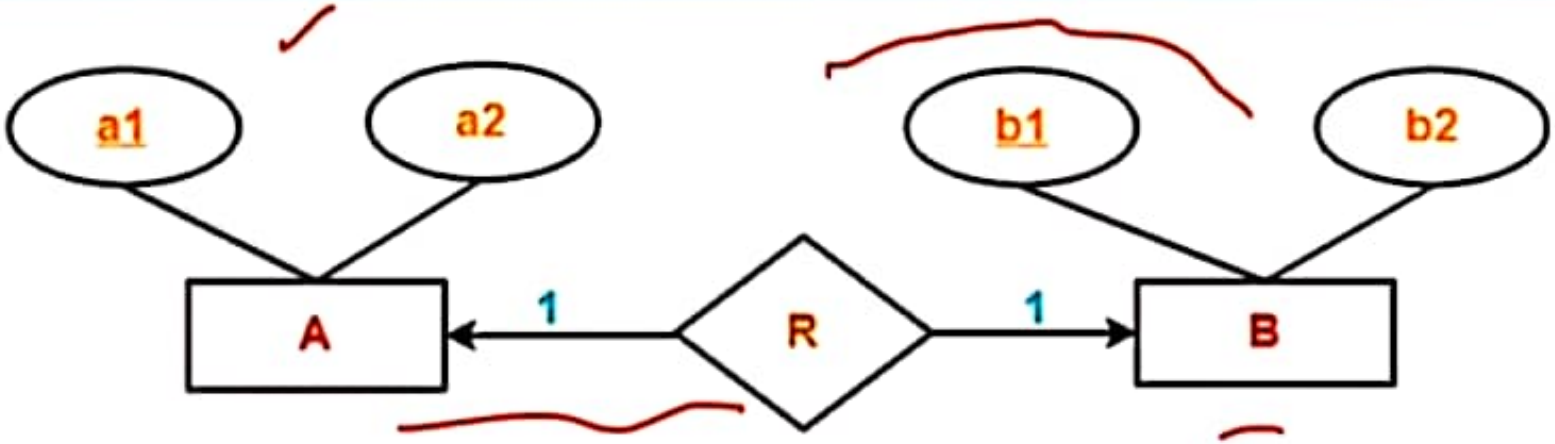



In ***Many-to-One relationship***, **two tables** will be required-

1. **AR ( a1 , a2 , b1 )**
2. **B ( b1 , b2 )**

**NOTE** - Here, combined table will be drawn for the entity set A and relationship set R.

# Case-4: For Binary Relationship with Cardinality Ratio 1:1




In ***One-to-One relationship***, **two tables** will be required. Either combine 'R' with 'A' or 'B'

**Way-01:**

1. AR ( a1 , a2 , b1 )
2. B ( b1 , b2 )

**Way-02:**

1. A ( a1 , a2 )
2. BR ( b1 , b2, a1 )

# Thumb Rules to Remember:

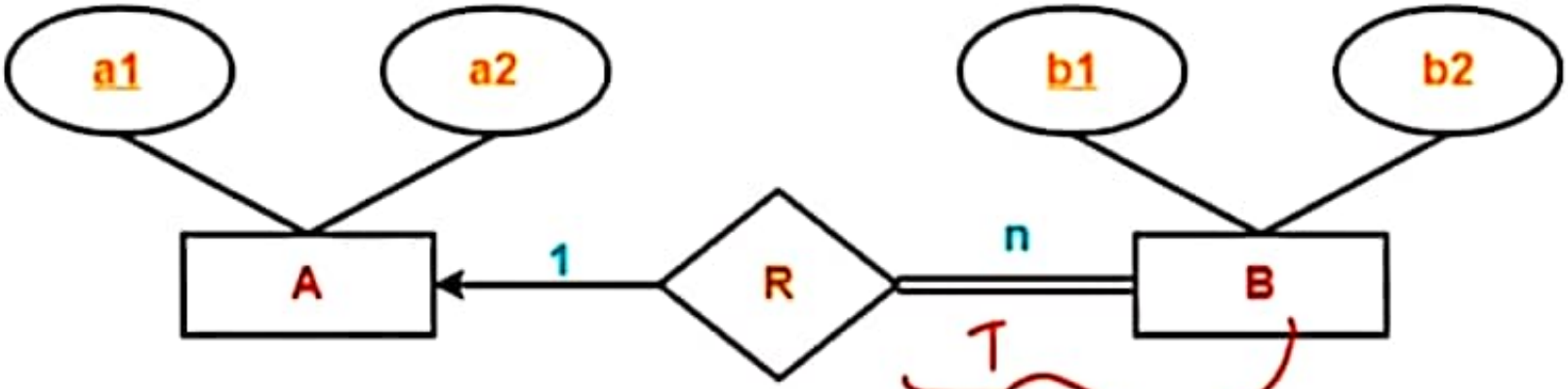
## (For determining minimum number of tables)

- **While determining the minimum number of tables required for binary relationships with given cardinality ratios,** following thumb rules must be kept in mind-
  - For binary relationship with cardinality ration **m : n** ,
    - *Separate and individual tables* will be drawn for each entity set and relationship (i.e. **three tables** will be required).
  - For binary relationship with cardinality ratio either **m : 1 or 1 : n** ,
    - Always remember "many side will consume the relationship" i.e. a combined table will be drawn for many side entity set and relationship set (i.e. **Two tables** will be required).
  - For binary relationship with cardinality ratio **1 : 1** ,
    - **Two tables** will be required. You can combine the relationship set with any one of the entity sets.

# Rule-6: For Binary Relationship with Both Cardinality Constraints and Participation Constraints

- Cardinality constraints will be implemented as discussed in Rule-5.
- Because of the *total participation constraint*, foreign key acquires **NOT NULL** constraint i.e. now foreign key can not be null.
- Two Cases:
  - **Case-1**: For Binary Relationship with Cardinality Constraint and Total Participation Constraint from One Side
  - **Case-2**: For Binary Relationship with Cardinality Constraint and Total Participation Constraint from Both Sides

## Case-1: For Binary Relationship with Cardinality Constraint and Total Participation Constraint from One Side



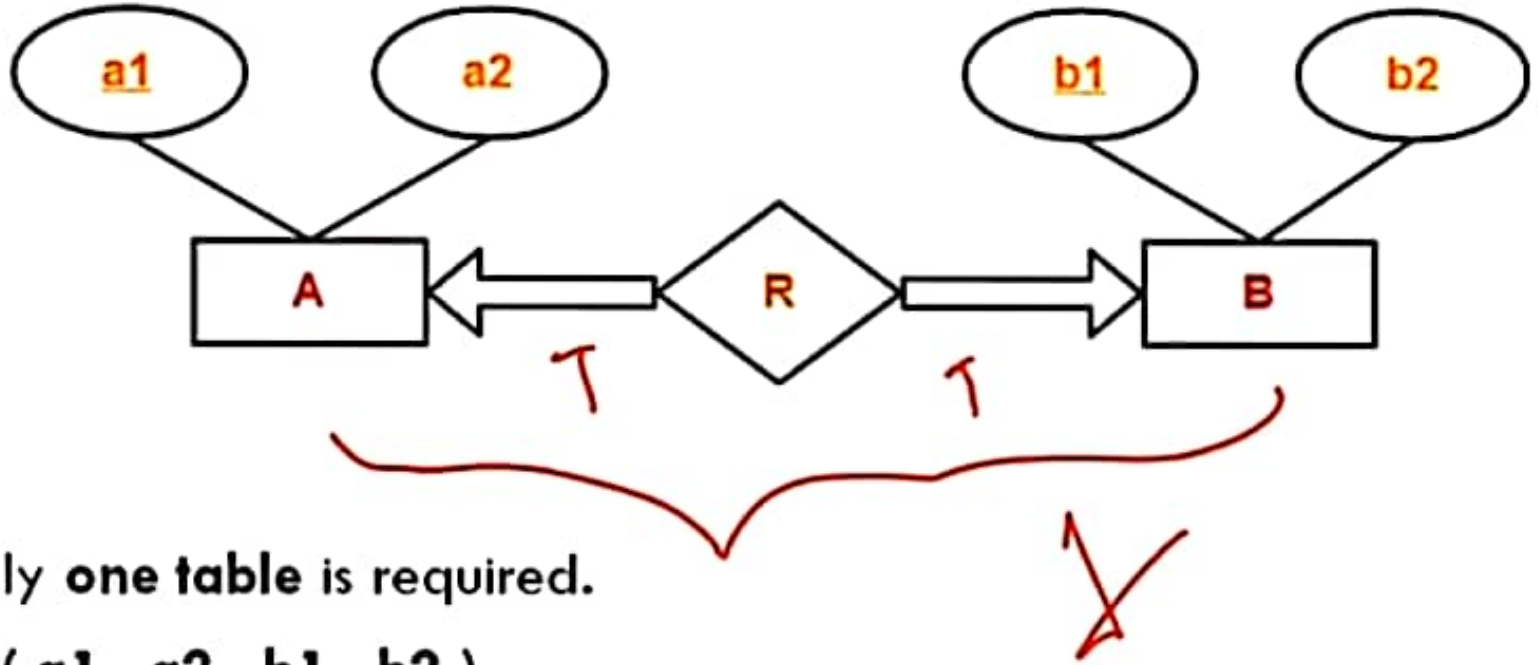

- Because cardinality ratio = 1 : n , we will combine the entity set B and relationship set R.
- Then, **two tables** will be required-

1. **A ( a1 , a2 )**
2. **BR ( b1 , b2 , a1 )**

Because of total participation, ***foreign key a1*** has acquired ***NOT NULL constraint***, so it can't be null now.

## Case-2: For Binary Relationship with Cardinality Constraint and Total Participation Constraint from Both Sides

- If there is a key constraint from both the sides of an entity set with total participation, then that binary relationship is represented using **only single table.**

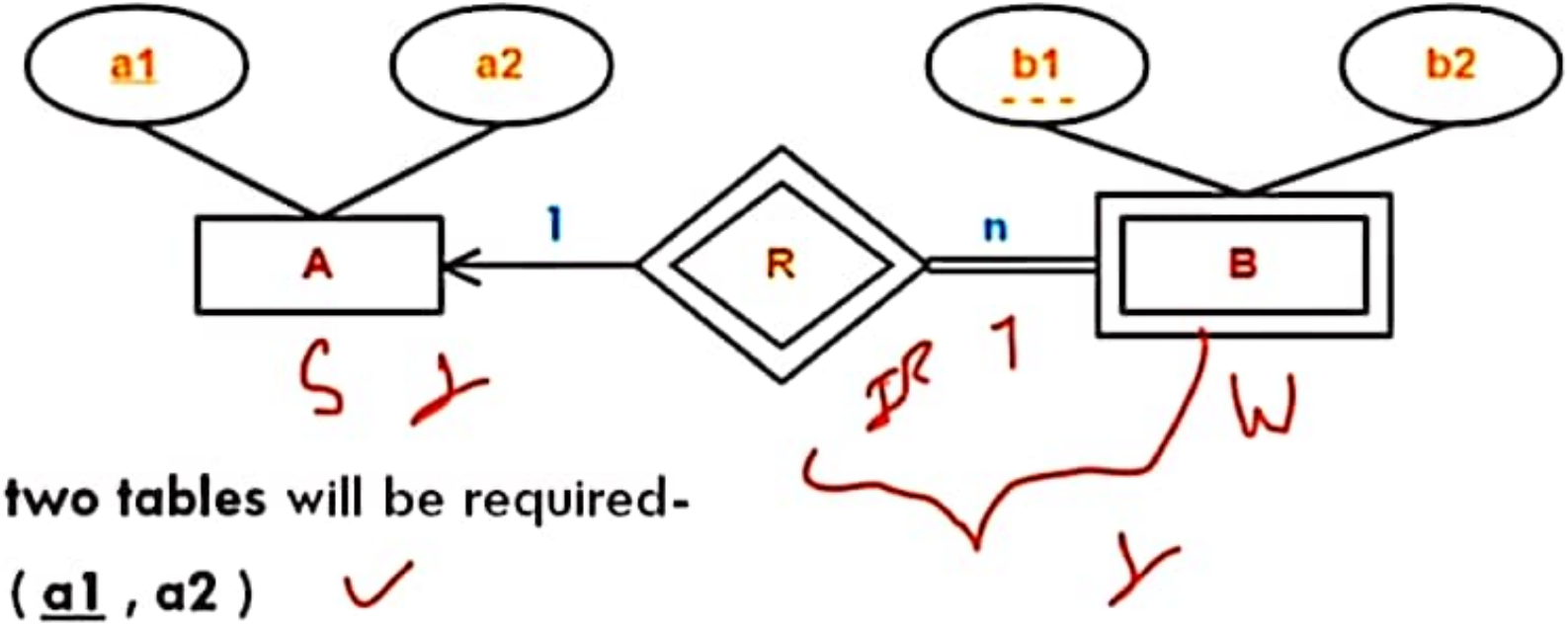



- Here, Only **one table** is required.

1. **ARB ( a1 , a2 , b1 , b2 )**

# Rule-7: For Binary Relationship with Weak Entity Set

- Weak entity set always appears in association with identifying relationship with **total** participation constraint and there is always **1: n** relationship from identifying entity set to weak entity set.

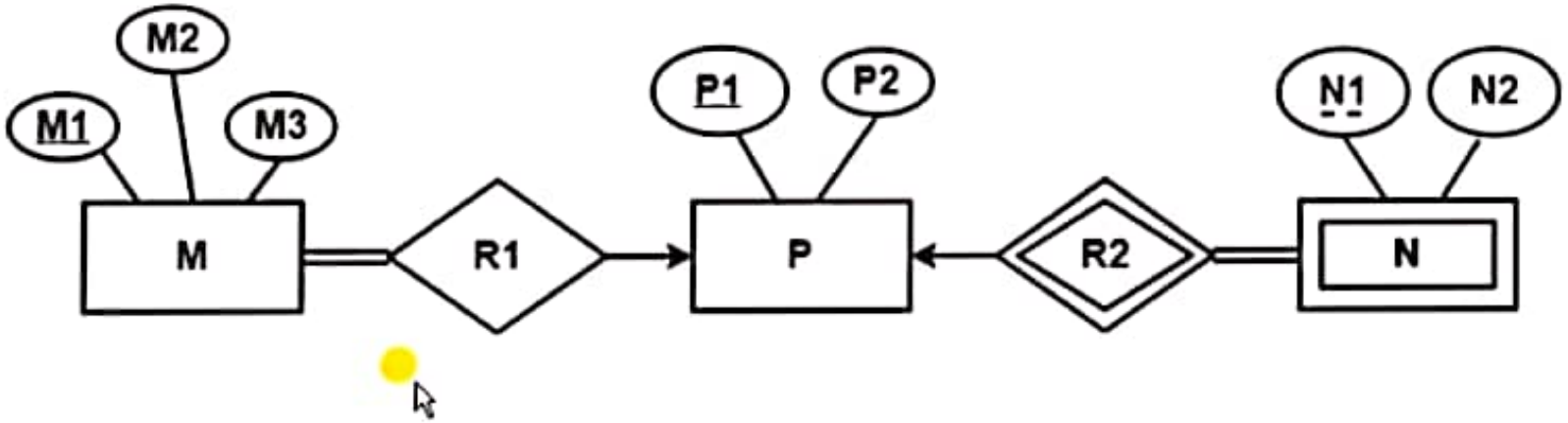



- Here, **two tables** will be required-

1. **A ( a1 , a2 )**
2. **BR ( a1 , b1 , b2 )**

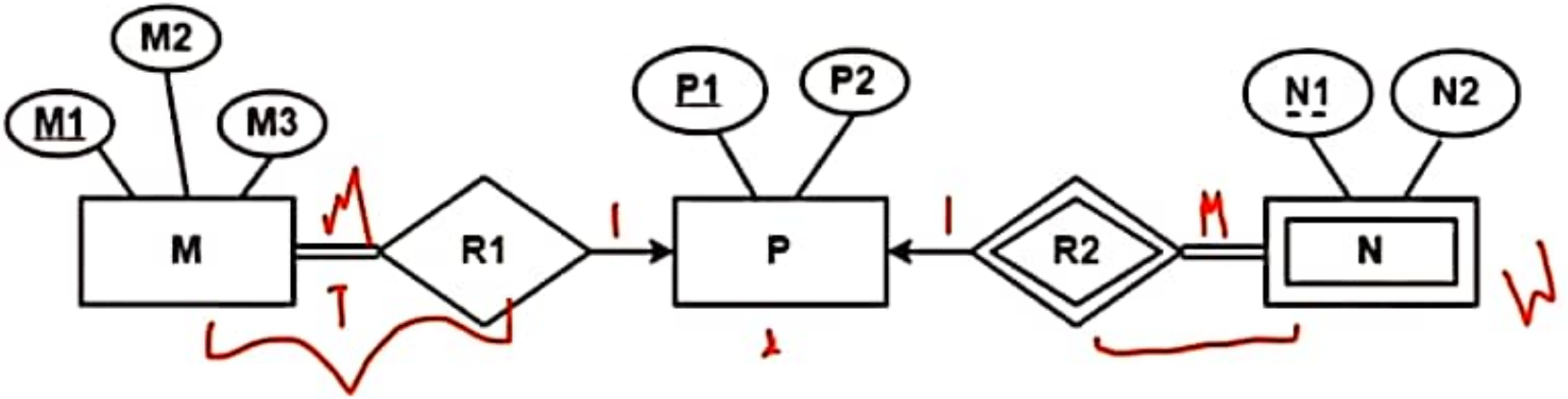
# Question 1:


- Find the minimum number of tables required for the following ER diagram in relational model



# Question 1:

- Find the minimum number of tables required for the following ER diagram in relational model



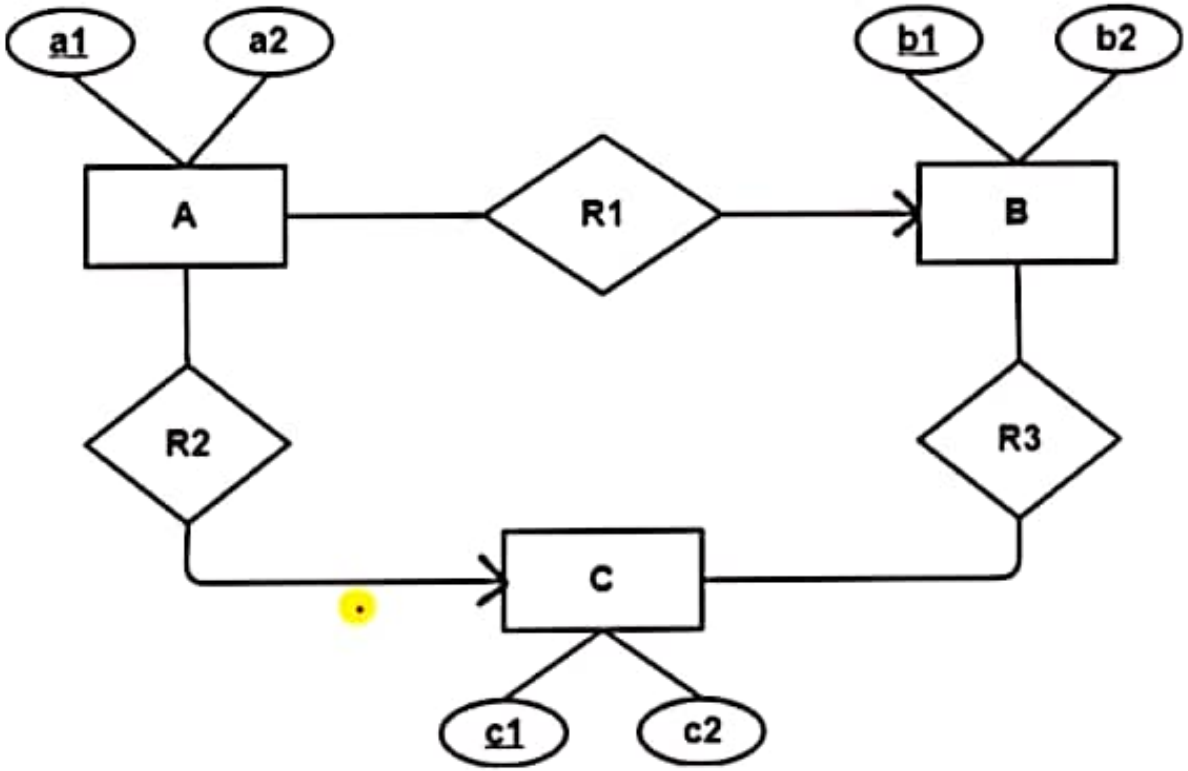

**Solution-**

Minimum 3 tables will be required-

1. MR1 (M1 , M2 , M3 , P1)
2. P (P1 , P2)
3. NR2 (P1 , N1 , N2)

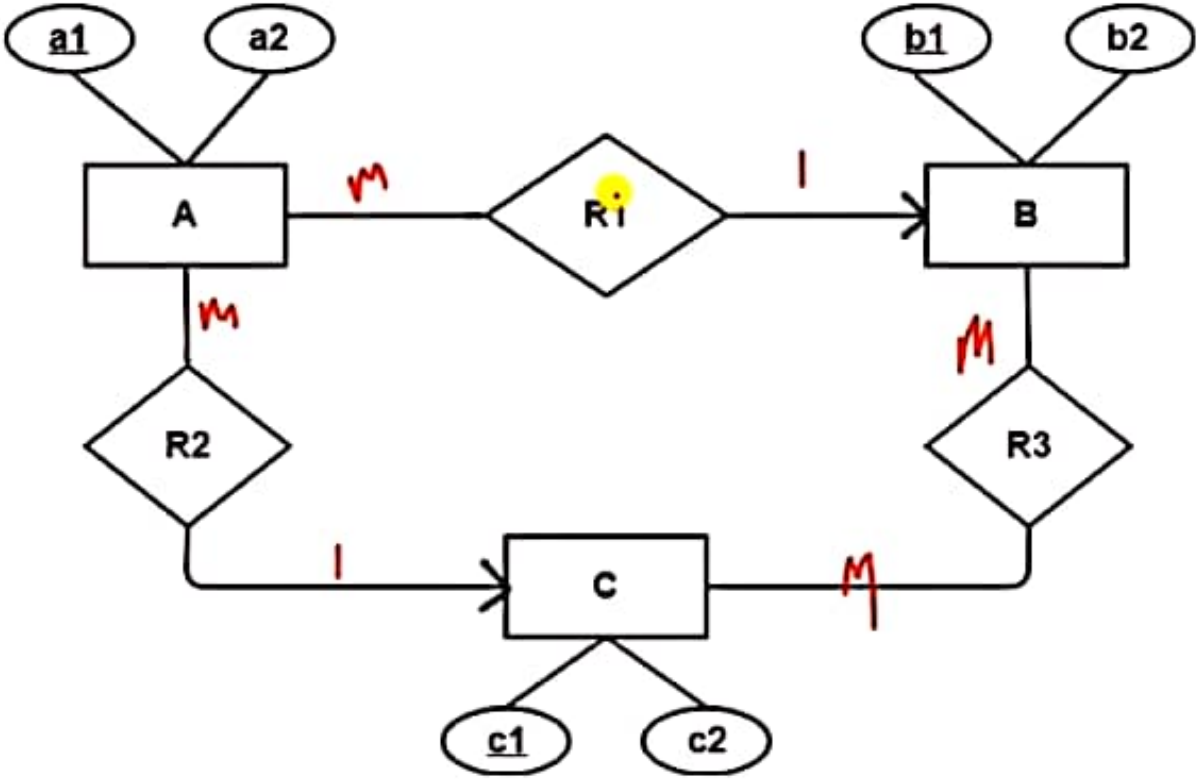
# Question 2:


- Find the minimum number of tables required for the following ER diagram in relational model



# Question 2:

- Find the minimum number of tables required for the following ER diagram in relational model



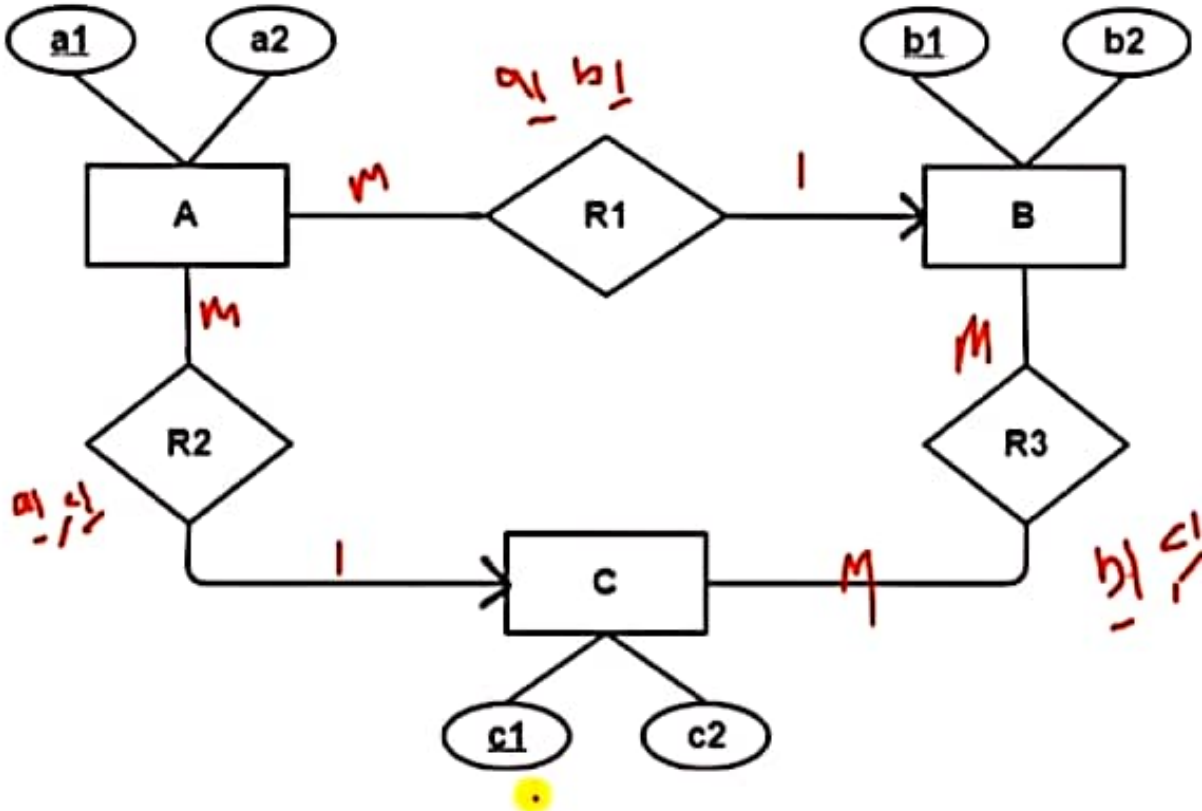

**Solution-**

Minimum 4 tables will be required:

1. AR1R2 (a1 , a2 , b1 , c1)
2. B (b1 , b2)
3. C (c1 , c2)
4. R3 (b1 , c1)

# Question 2:

- Find the minimum number of tables required for the following ER diagram in relational model



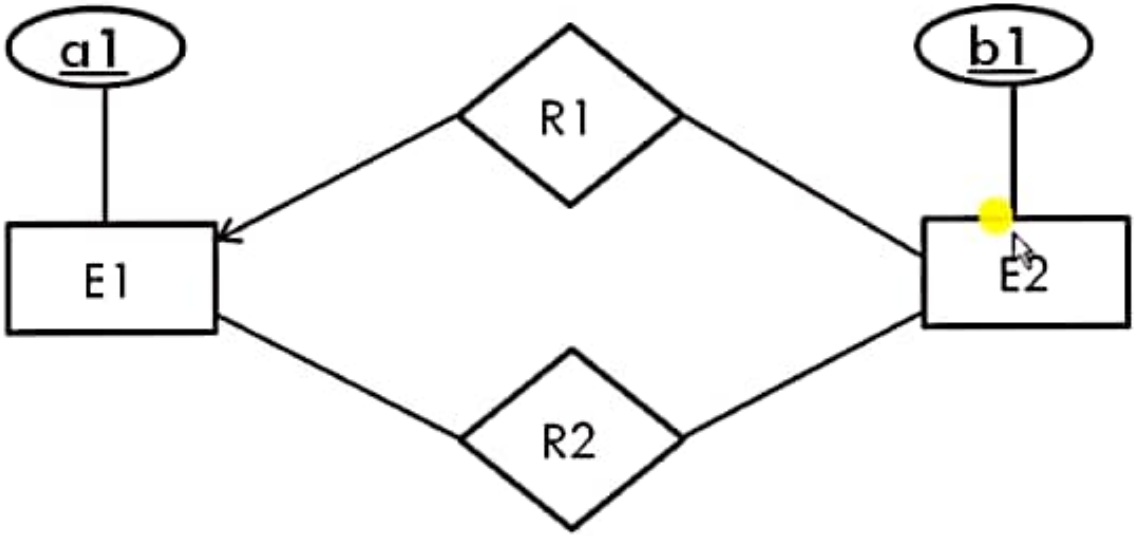

**Solution-**

Minimum 4 tables will be required:

1. AR1R2 (a1 , a2 , b1 , c1)
2. B (b1 , b2)
3. C (c1 , c2)
4. R3 (b1 , c1)

# Question 3:

- Find the minimum number of tables required for the following ER diagram in relational model



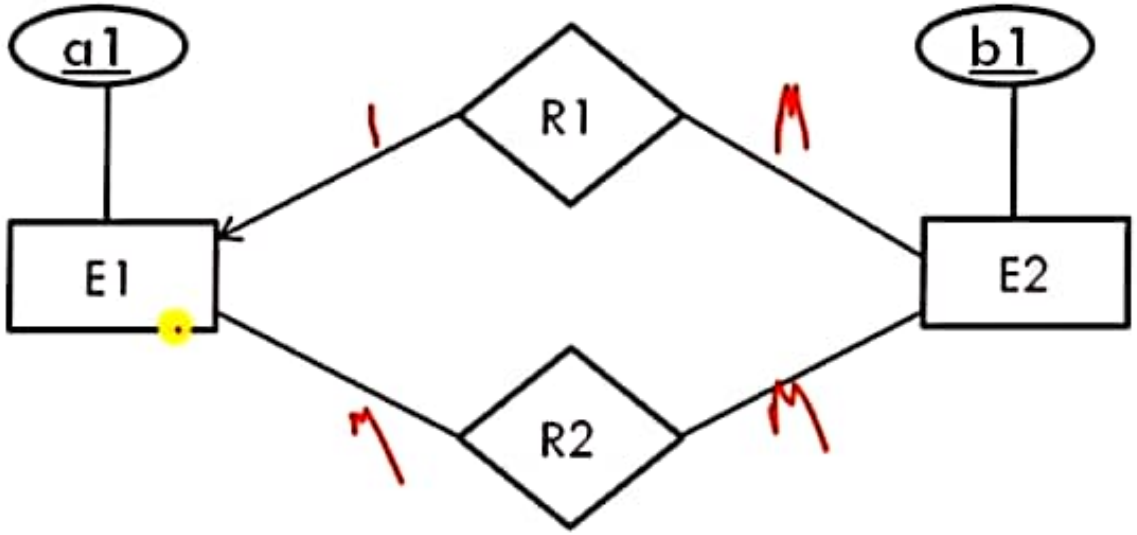
# Question 3:


- Find the minimum number of tables required for the following ER diagram in relational model

- **Solution:**

  Three tables will be formed

  1. E1(a1)
  2. E2R1 (b1, a1)
  3. R2 (a1, b1)



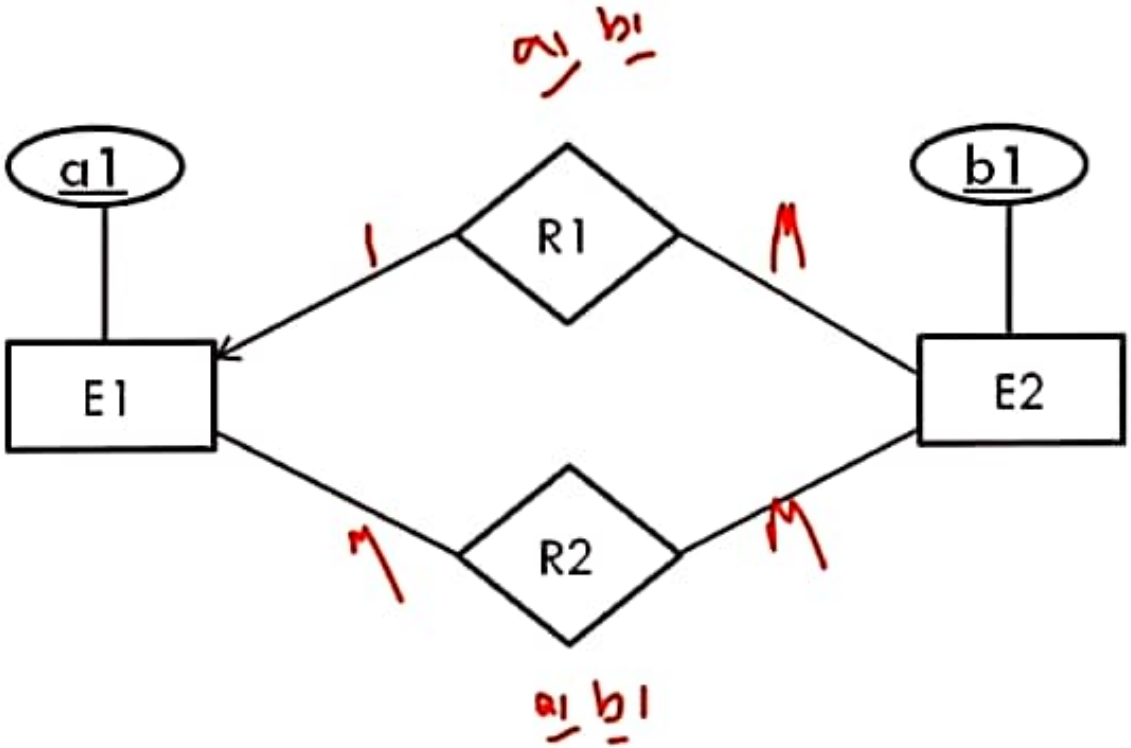
# Question 3:

- Find the minimum number of tables required for the following ER diagram in relational model

- **Solution:**

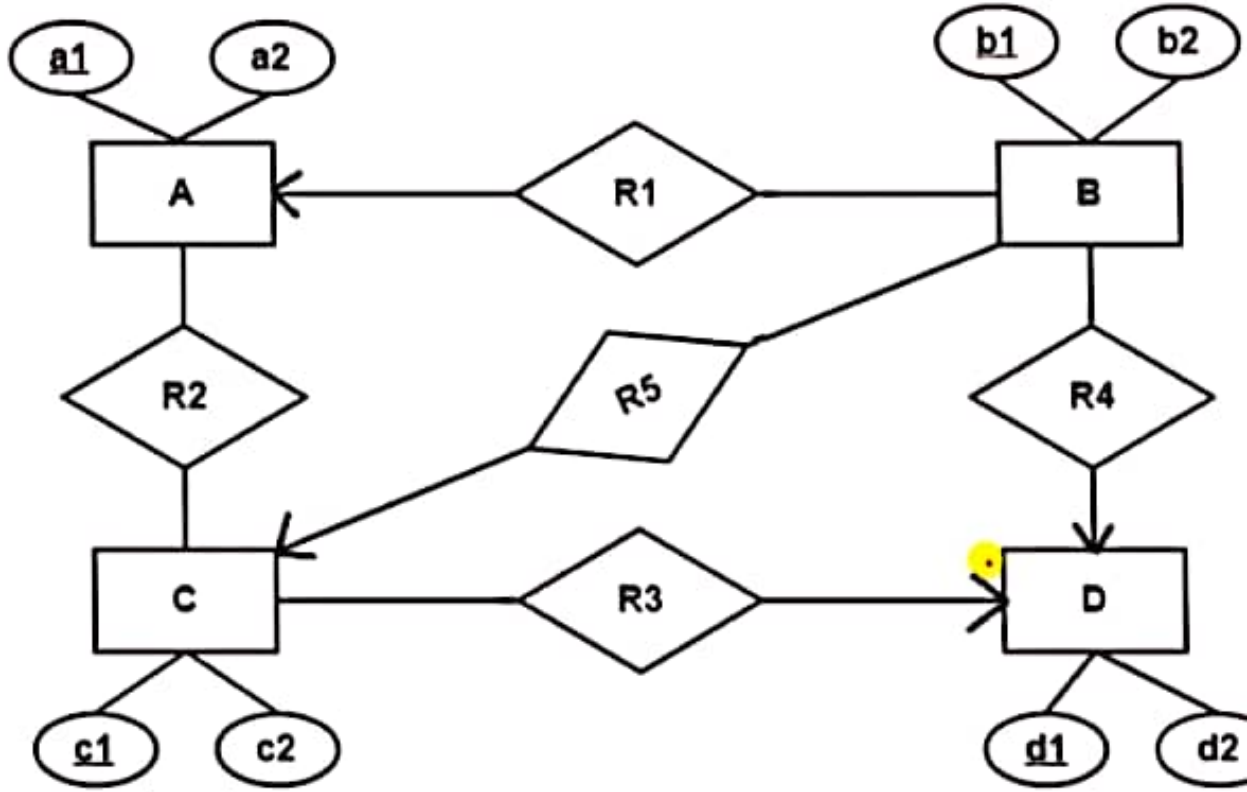
  Three tables will be formed

  1. E1(a1)
  2. E2R1 (b1, a1)
  3. R2 (a1, b1)



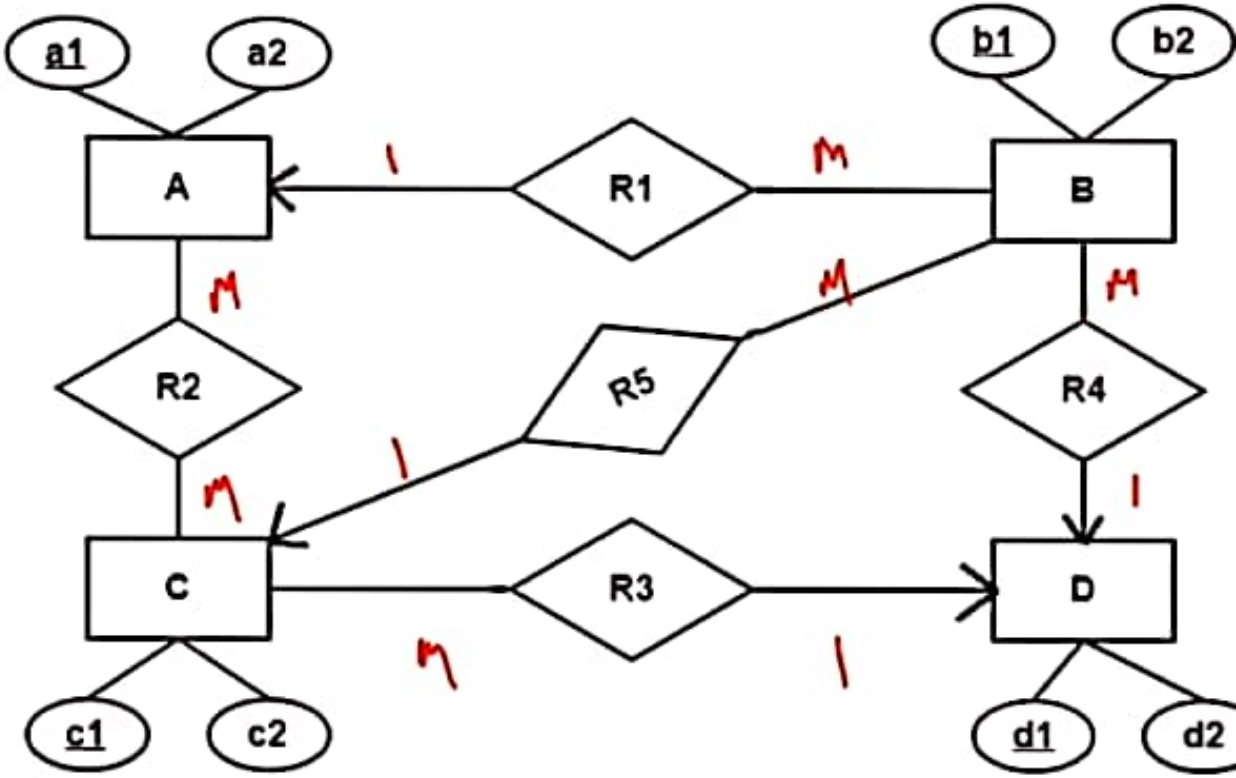
# Question 4:

- Find the minimum number of tables required for the following ER diagram in relational model



# Question 4:


- Find the minimum number of tables required for the following ER diagram in relationa model



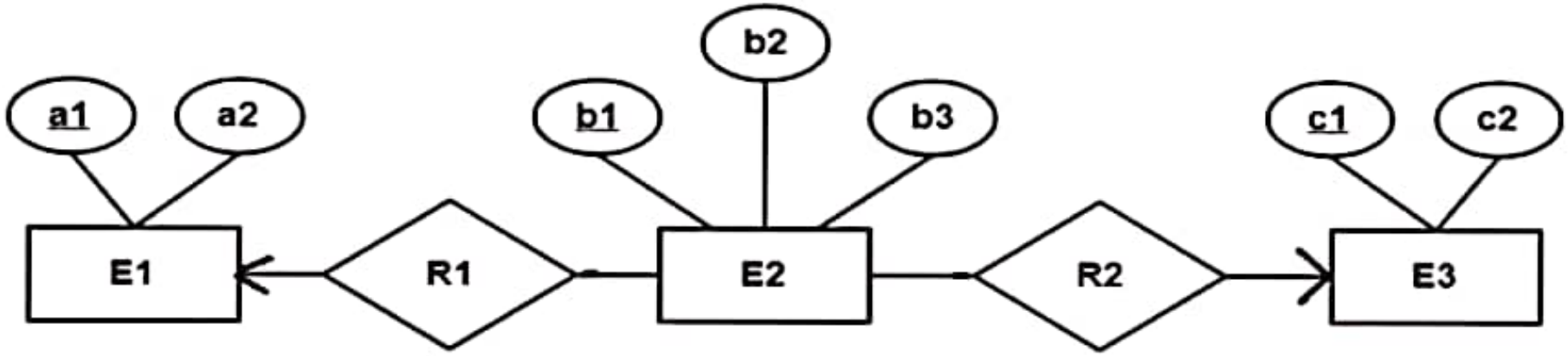

**Solution-**

Minimum 5 tables will be required:

1. BR1R4R5 (b1 , b2 , a1 , c1 , d1)
2. A (a1 , a2)
3. R2 (a1 , c1)
4. CR3 (c1 , c2 , d1)
5. D (d1 , d2)

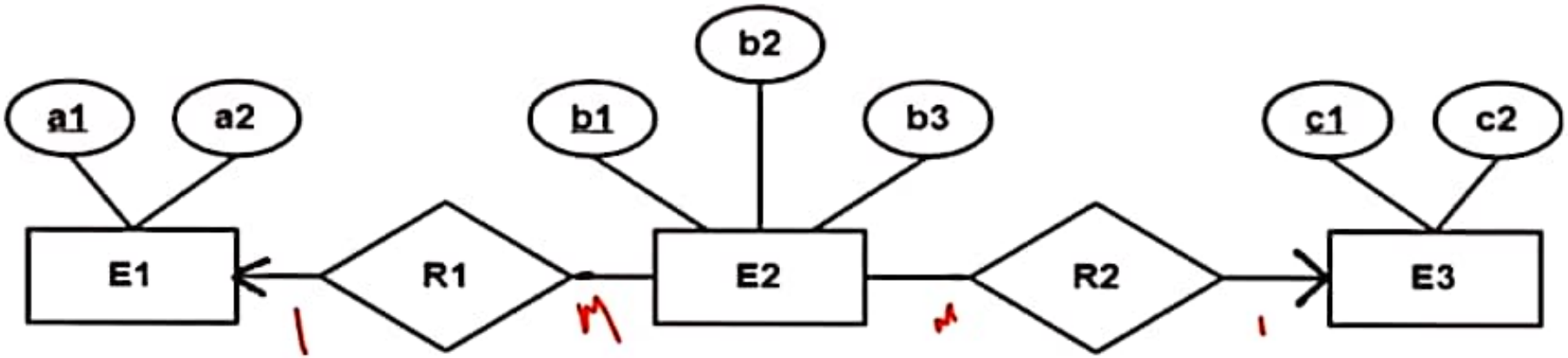
# Question 5:


- Find the minimum number of tables required for the following ER diagram in relational model



# Question 5:


- Find the minimum number of tables required for the following ER diagram in relational model



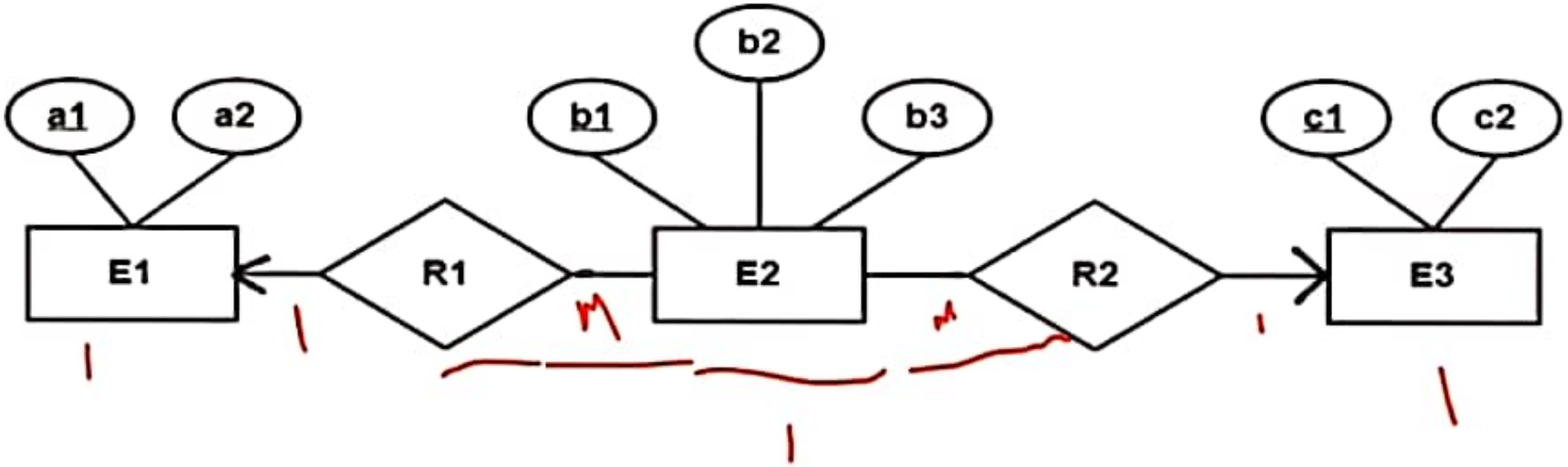

**Solution-**

Minimum 3 tables will be required:

1. E1 (a1 , a2)
2. E2R1R2 (b1 , b2 , b3, a1 , c1)
3. E3 (c1 , c2)

# Question 5:

- Find the minimum number of tables required for the following ER diagram in relational model



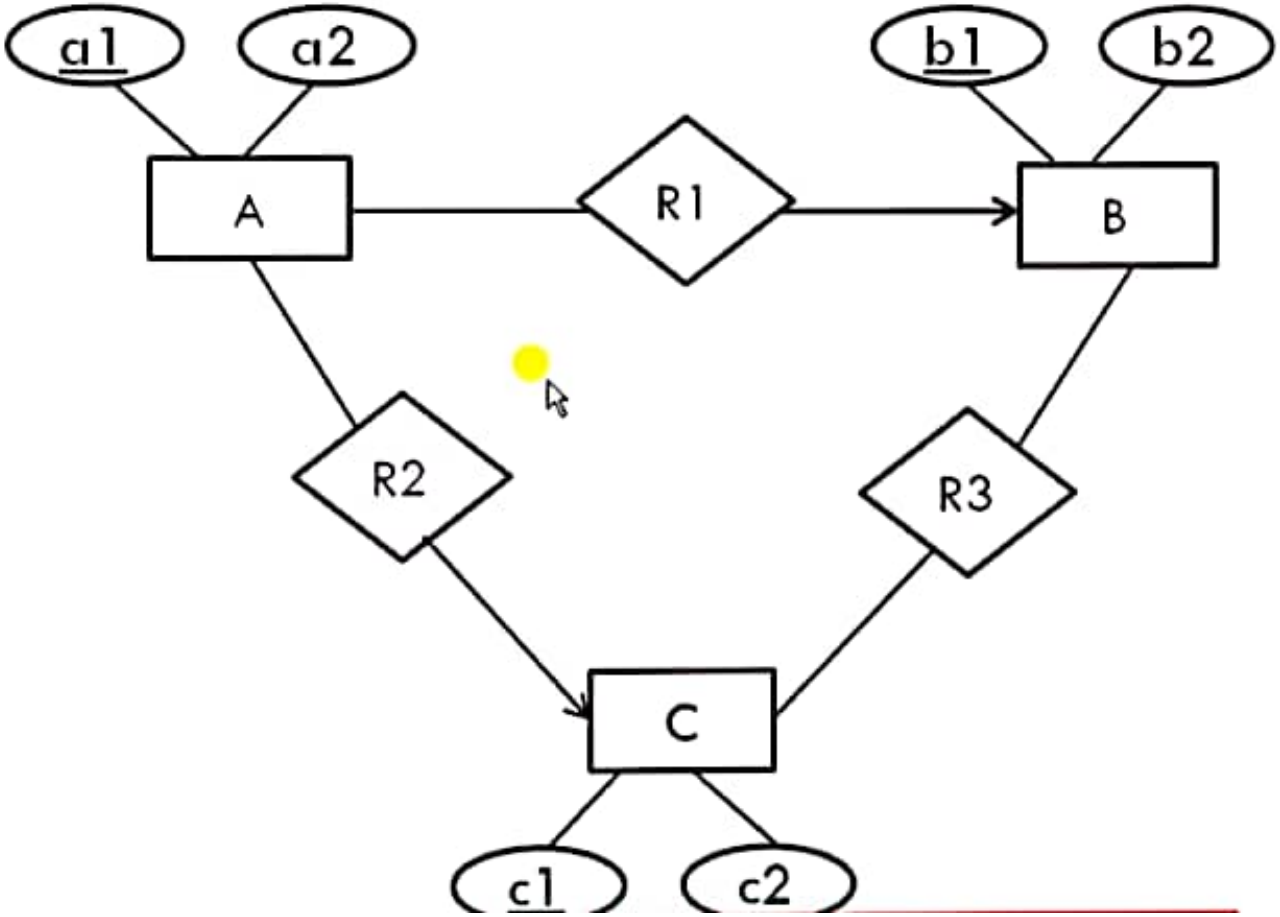

**Solution-**

Minimum 3 tables will be required:

1. E1 (a1 , a2)
2. E2R1R2 (b1 , b2 , b3, a1 , c1)
3. E3 (c1 , c2)

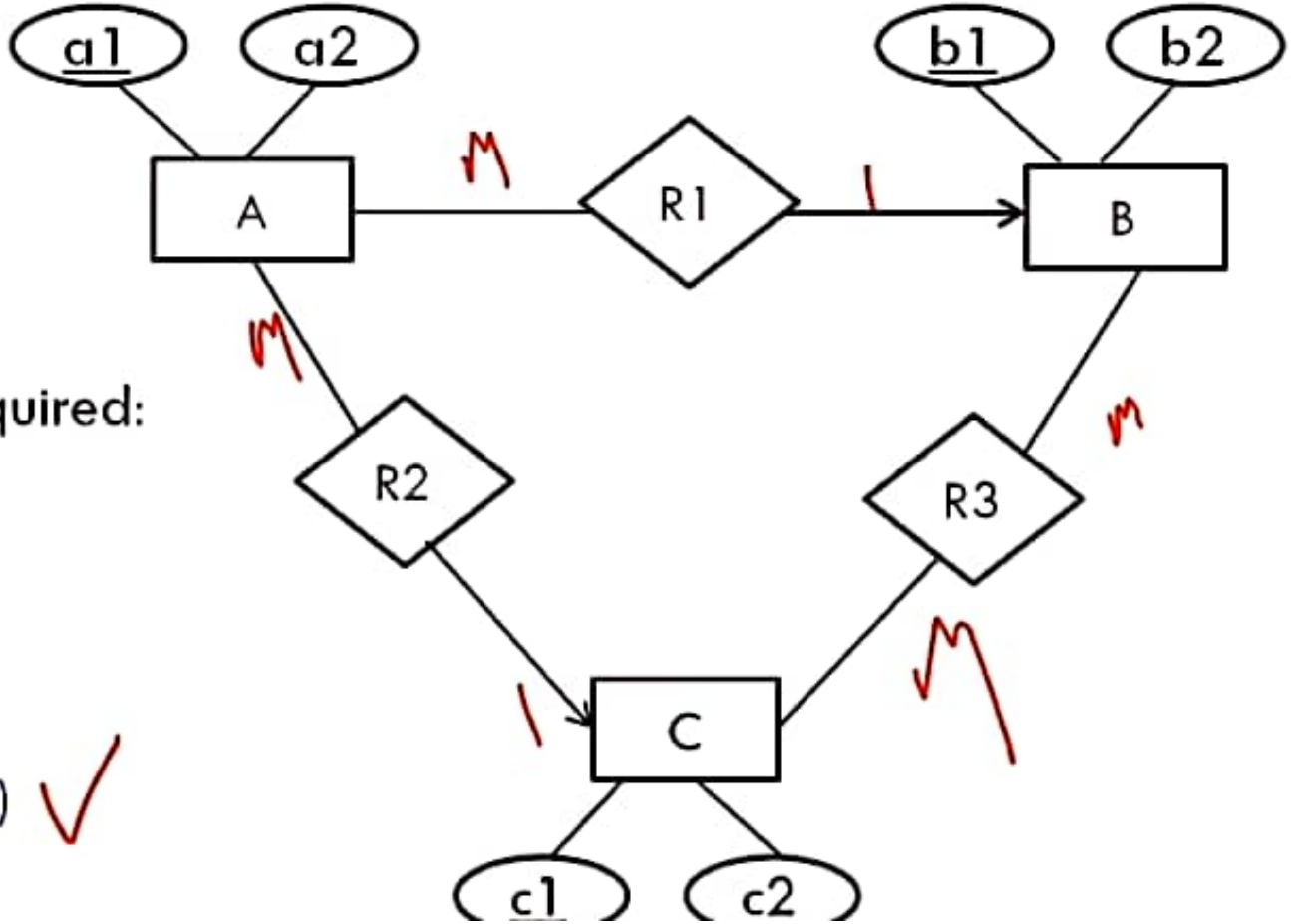
# Question 6:

- Find the minimum number of tables required for the following ER diagram in relational model



# Question 6:

- Find the minimum number of tables required for the following ER diagram in relational model




**Solution-**

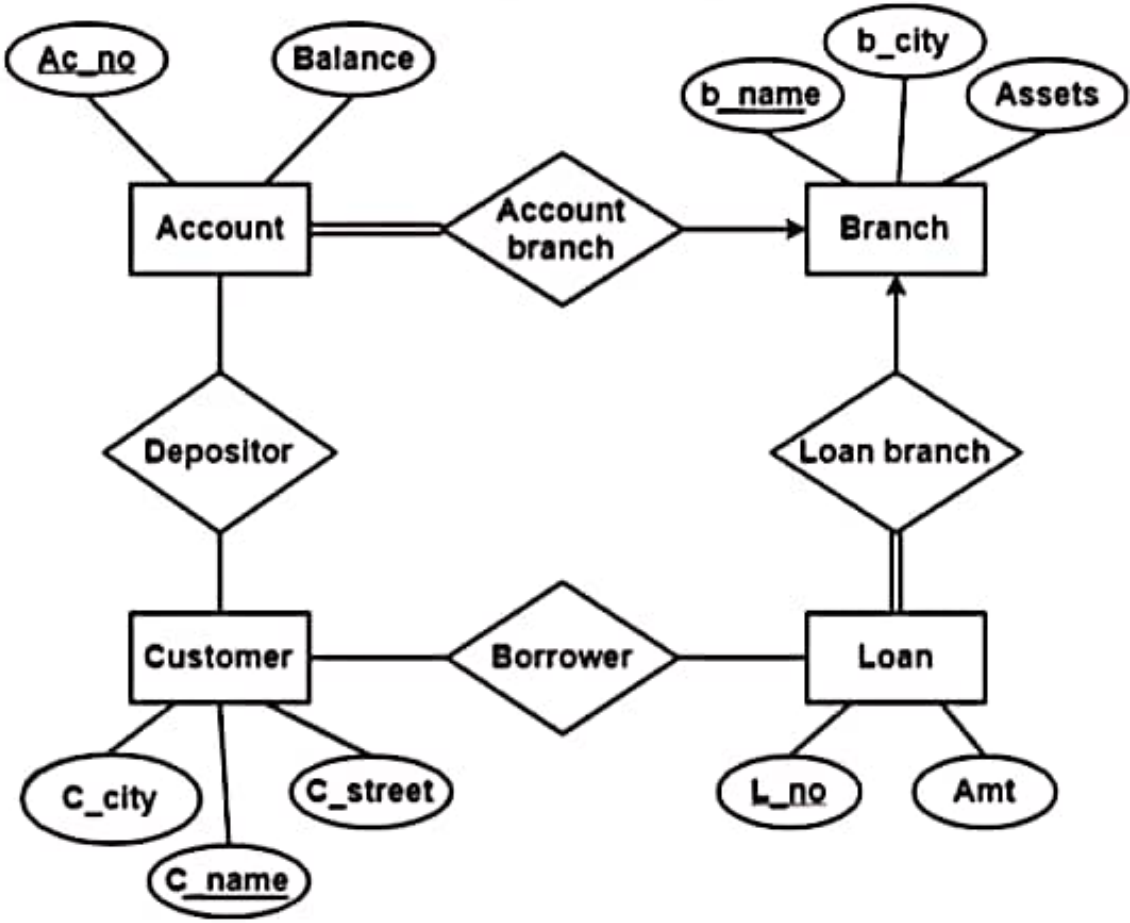
Minimum 4 tables will be required:

1. B (b1 , b2)
2. C (c1 , c2)
3. R3(b1, c1)
4. AR1R2 ( a1 , a2, b1, c1 )

F F

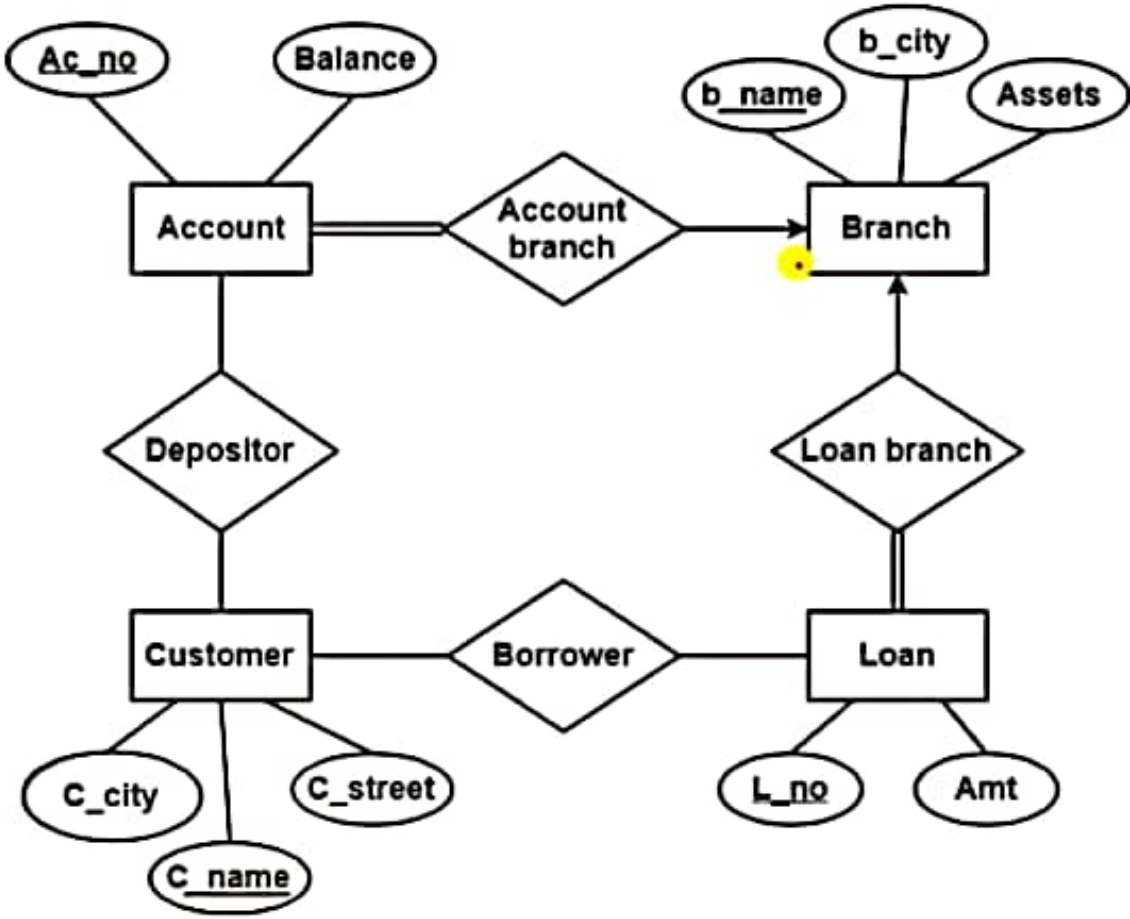
# Question 1:

- Find the minimum number of tables required for the following ER diagram in relational model



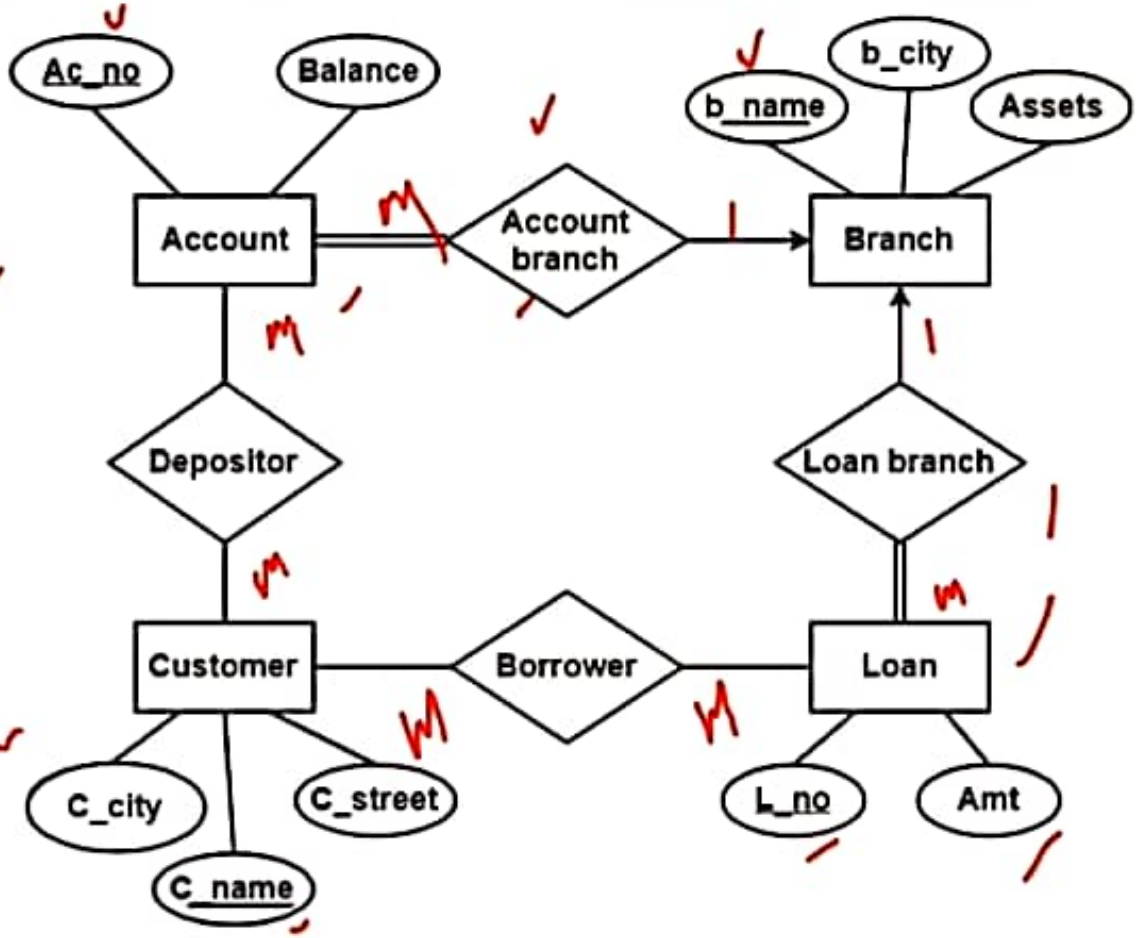
# Question 1:

- Find the minimum number of tables required for the following ER diagram in relational model



# Question 1:


- Find the minimum number of tables required for the following ER diagram in relational model

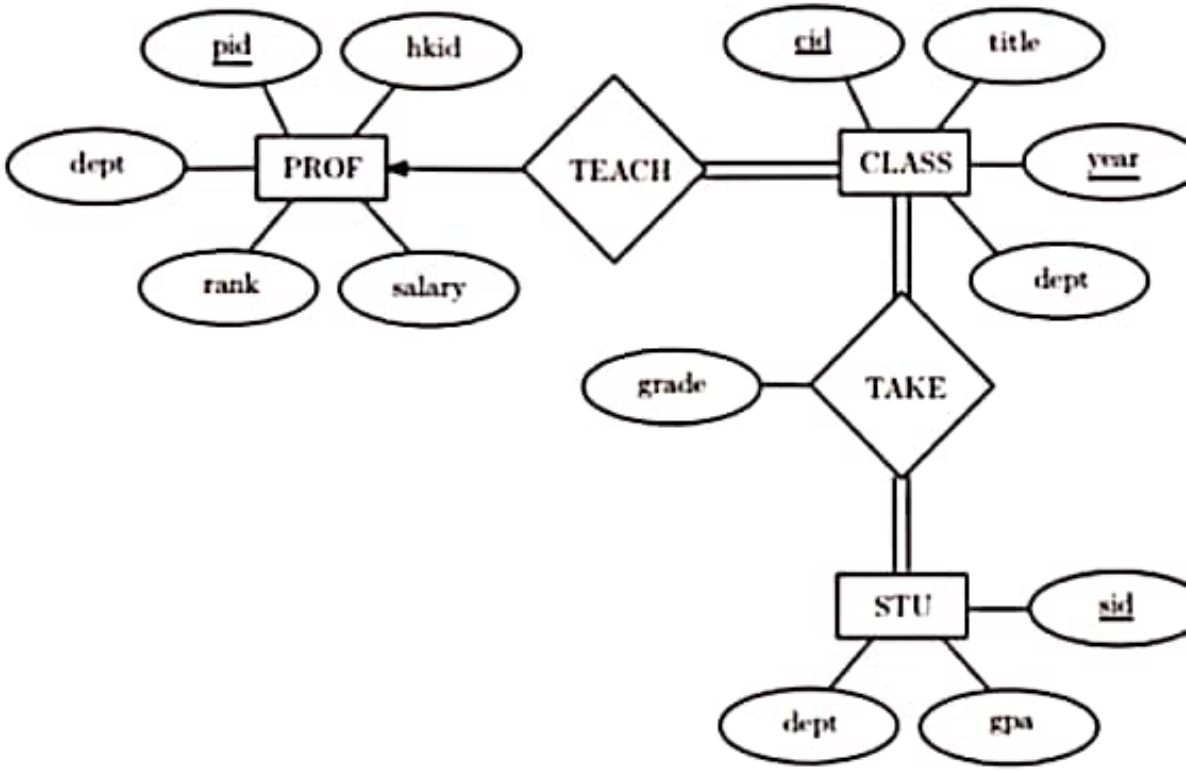



**Solution-**

Minimum 6 tables will be required:

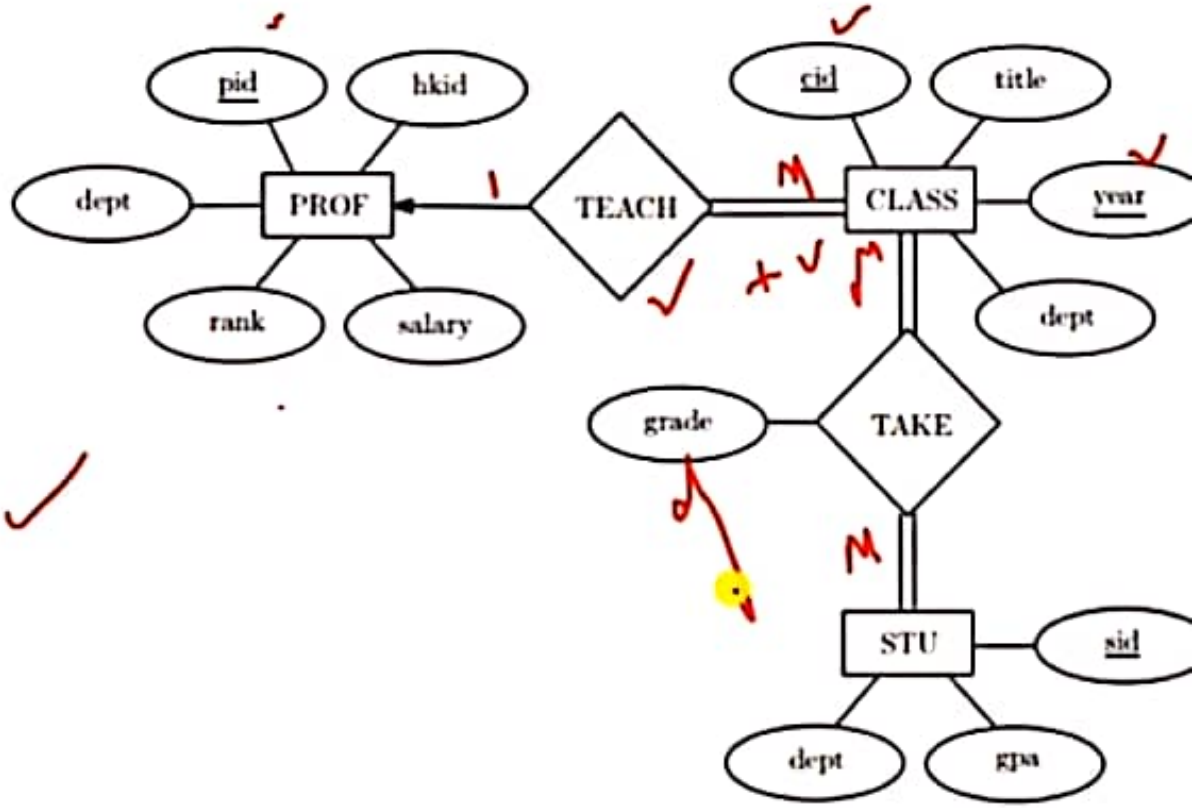
1. Account (Ac_no , Balance , b_name)
2. Branch (b_name , b_city , Assets)
3. Loan (L_no , Amt , b_name)
4. Borrower (C_name , L_no)
5. Customer (C_name , C_street , C_city)
6. Depositor (C_name , Ac_no)

- Find the minimum number of tables required for the following ER diagram in relational model



# Question 2:

- Find the minimum number of tables required for the following ER diagram in relational model

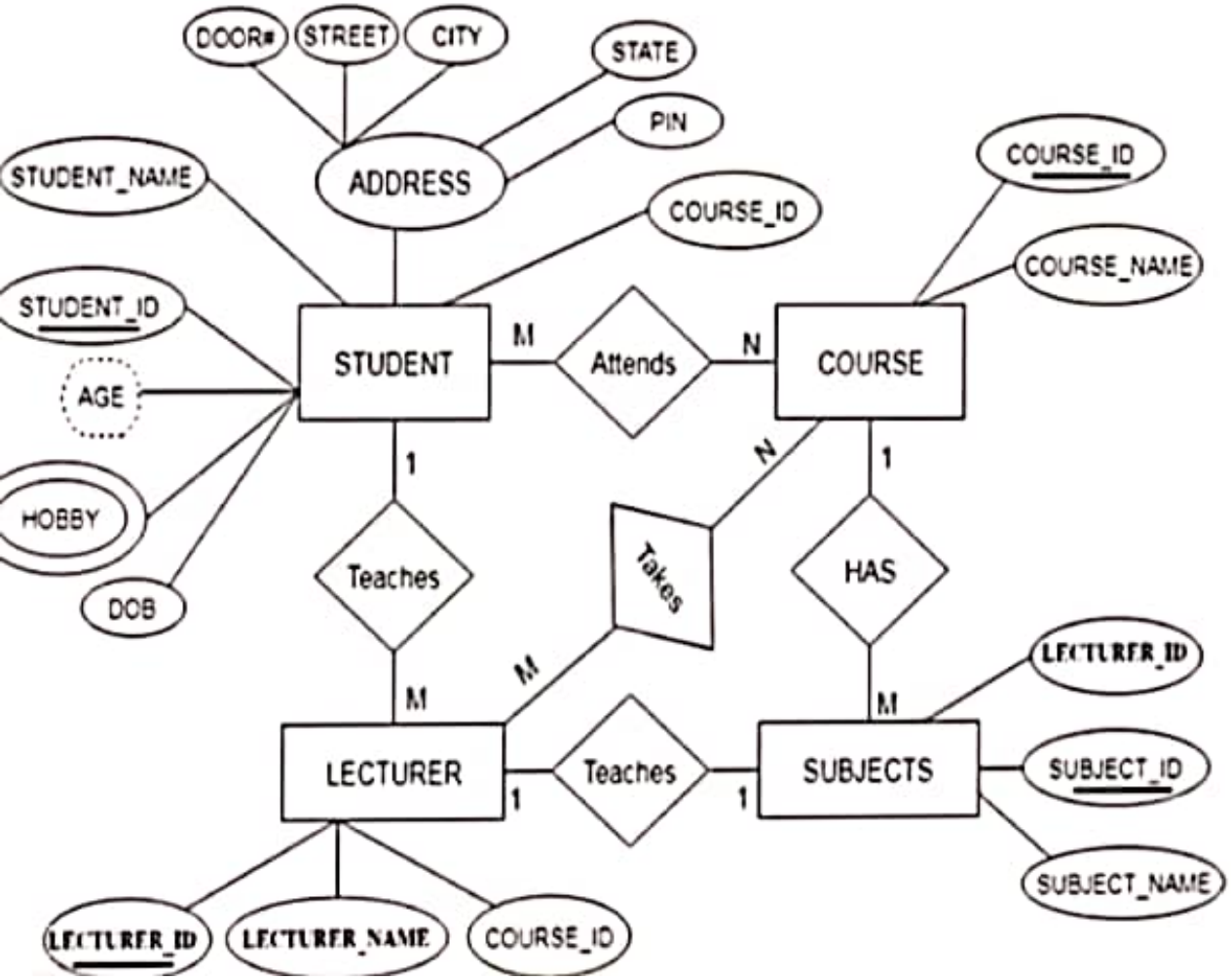



**Solution-**

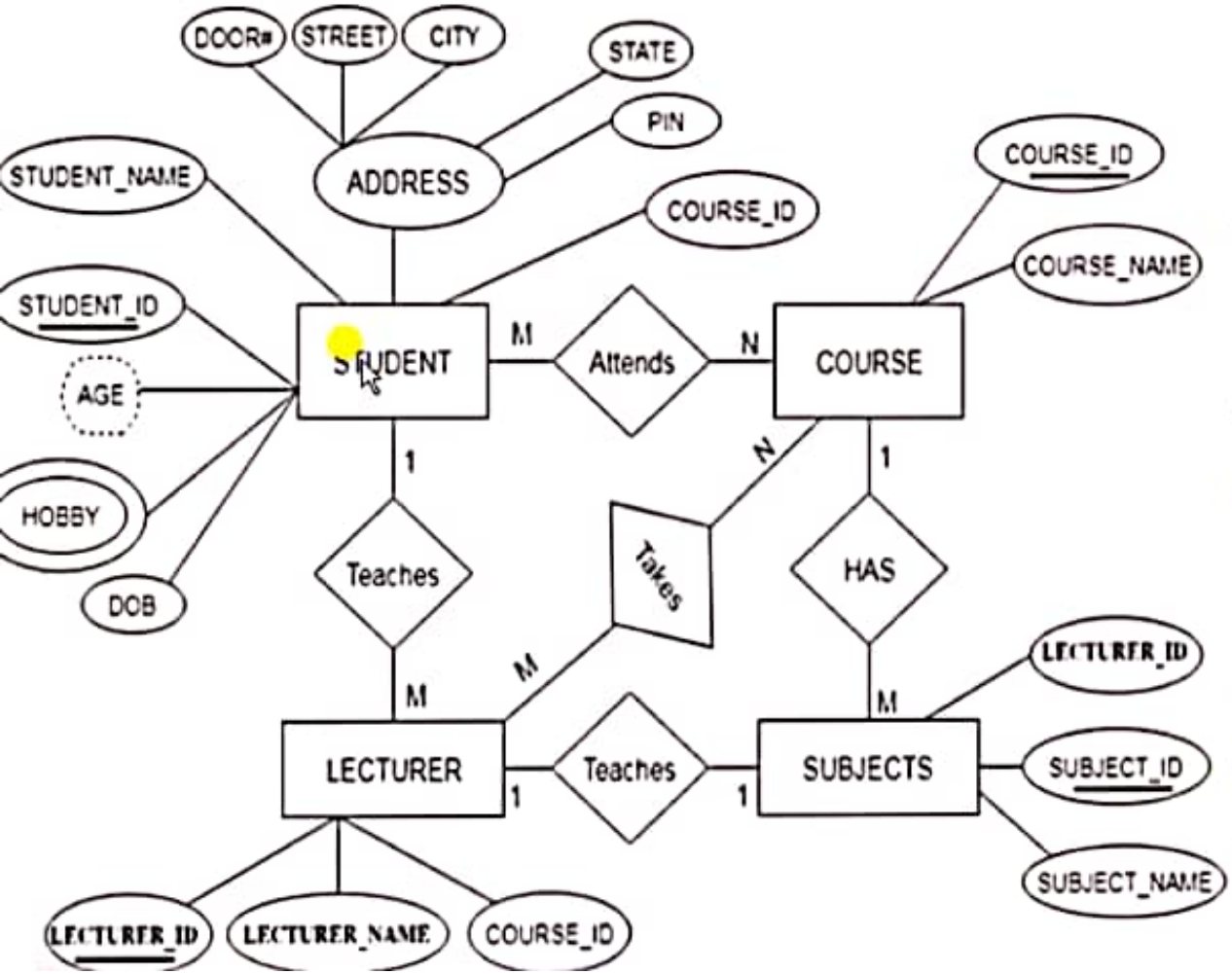
Applying the rules, minimum 4 tables will be required:

1. PROF (pid, hkid, dept, rank, salary)
2. CLASS (cid, year, title, dept, pid)
3. STU (sid, dept, gpa)
4. TAKE (cid, year, sid, grade)

# Question 3:


DOOR#
STREET
CITY
STATE
PIN
ADDRESS
STUDENT_NAME
COURSE_ID
STUDENT_ID
AGE
HOBBY
DOB
STUDENT
M
Attends
N
COURSE
COURSE_ID
COURSE_NAME
1
Teaches
M
N
Takes
M
1
HAS
M
LECTURER_ID
LECTURER
1
Teaches
1
SUBJECTS
SUBJECT_ID
SUBJECT_NAME
LECTURER_ID
LECTURER_NAME
COURSE_ID

# Question 3:

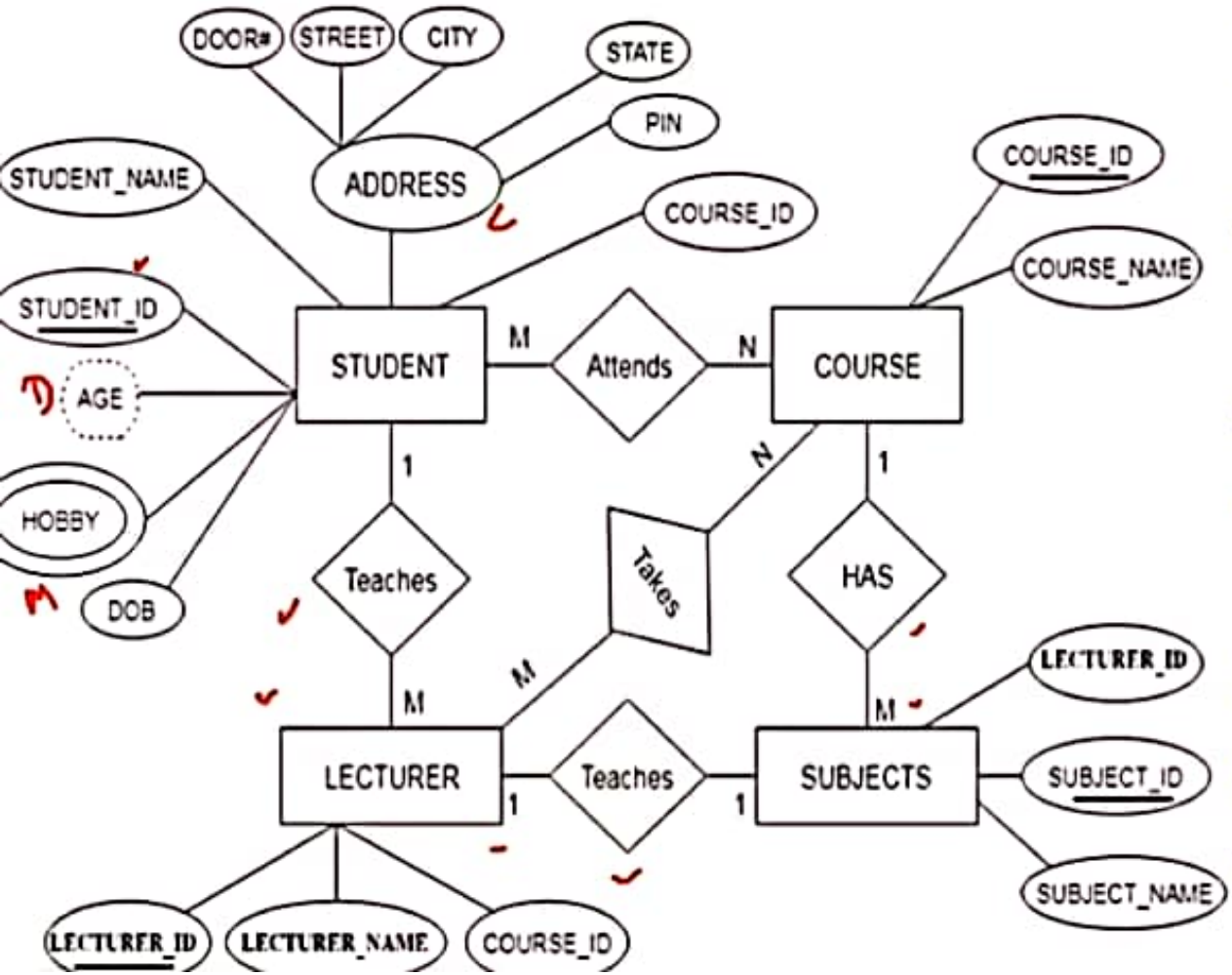



**Solution**:

**Minimum 7 tables will be required:**

1. **Student** (Student_Id, Student_Name, DOB, Door#, Street, City, State, Pin, Course_Id)
2. **Lecturer** (Lecturer_Id, Lecturer_Name, Course_Id, Student_Id, Subject_Id)
3. **Course** (Course_Id, Course_Name)
4. **Subjects** (Subject_Id, Subject_Name, Lecturer_Id, Course_Id)
5. **Hobby** (Subject_Id, Hobby)
6. **Attends** (Subject_Id, Course_Id)
7. **Takes** (Lecturer_Id, Course_Id)

# Question 3:

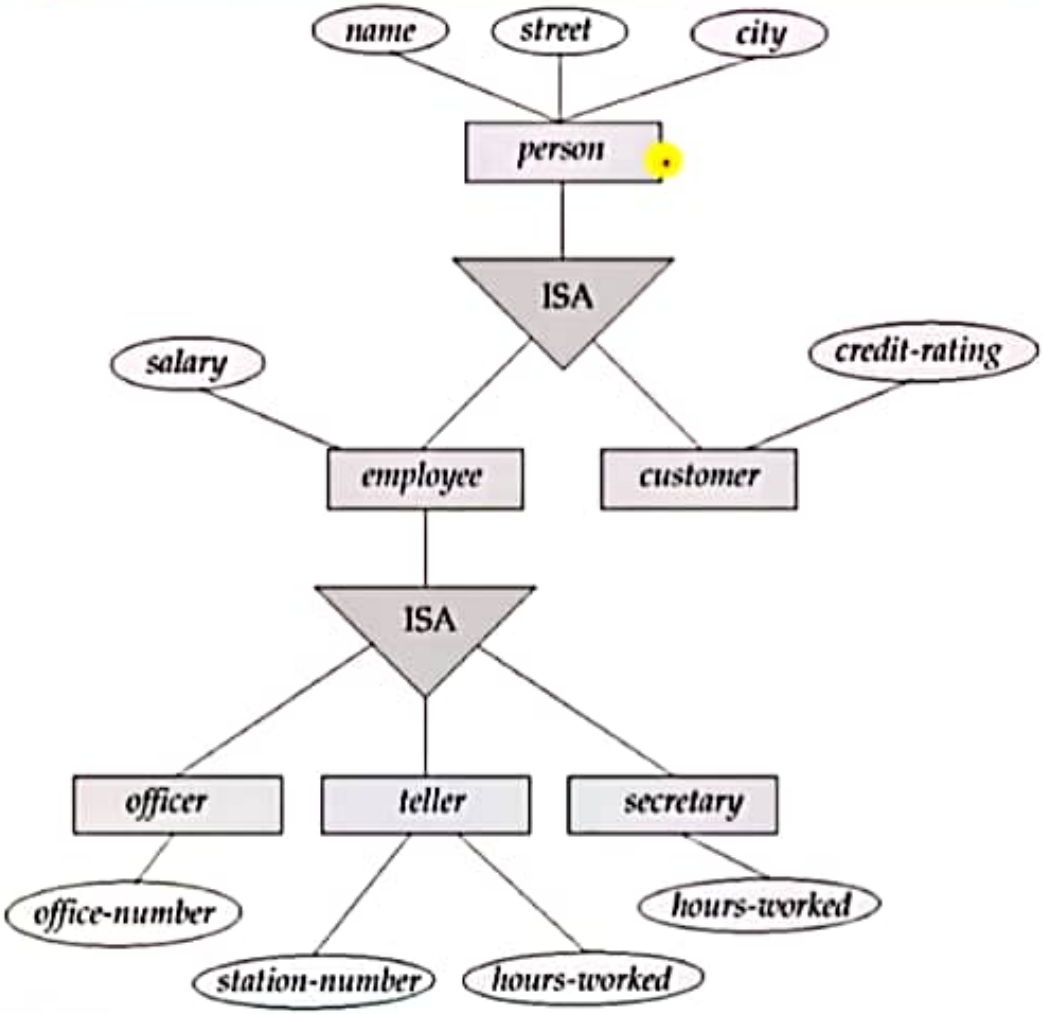



**Solution**:

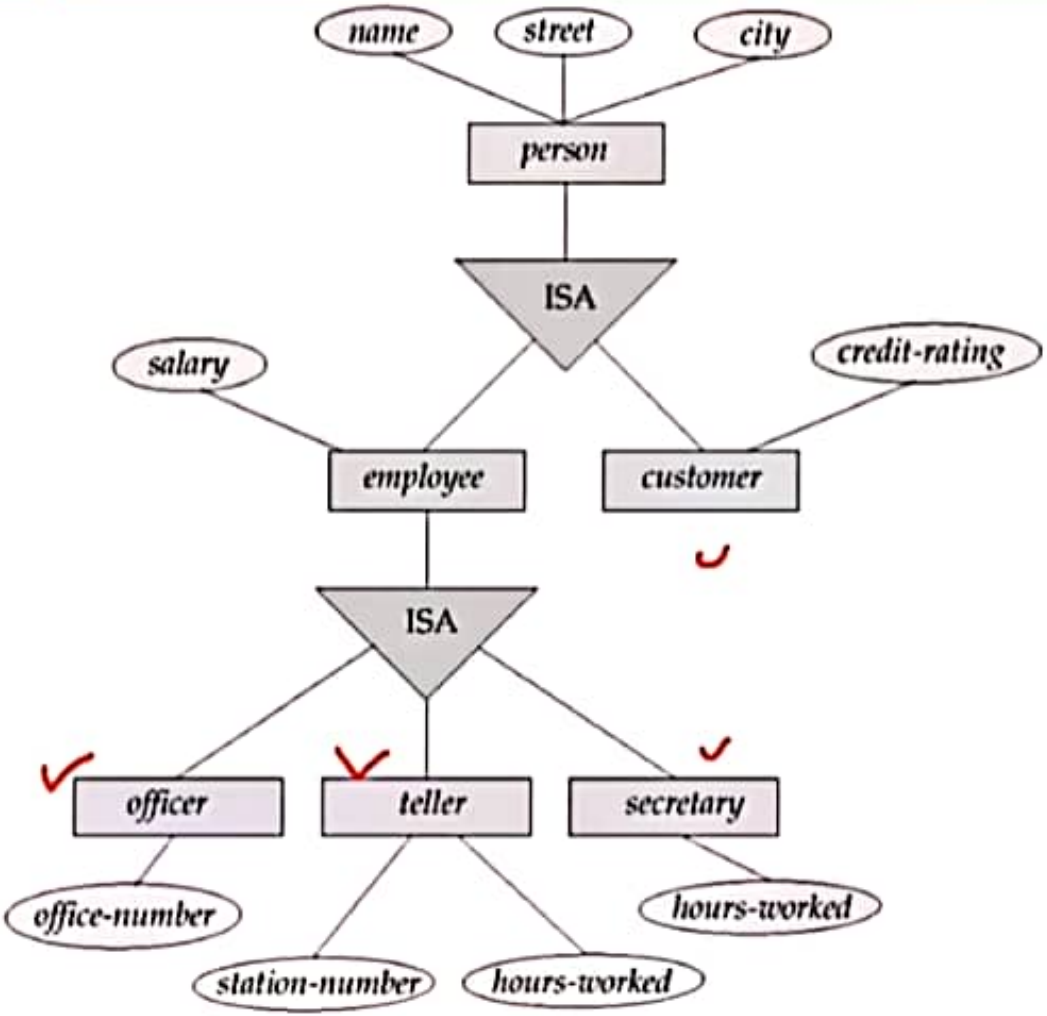
**Minimum 7 tables will be required:**

1. **Student** (Student_Id, Student_Name, DOB, Door#, Street, City, State, Pin, Course_Id)
2. **Lecturer** (Lecturer_Id, Lecturer_Name, Course_Id, Student_Id, Subject_Id)
3. **Course** (Course_Id, Course_Name)
4. **Subjects** (Subject_Id, Subject_Name, Lecturer_Id, Course_Id)
5. **Hobby** (Subject_Id, Hobby)
6. **Attends** (Subject_Id, Course_Id)
7. **Takes** (Lecturer_Id, Course_Id)

# Question 4:


name
street
city
person
ISA
salary
credit-rating
employee
customer
ISA
officer
teller
secretary
office-number
station-number
hours-worked
hours-worked

# Question 4:

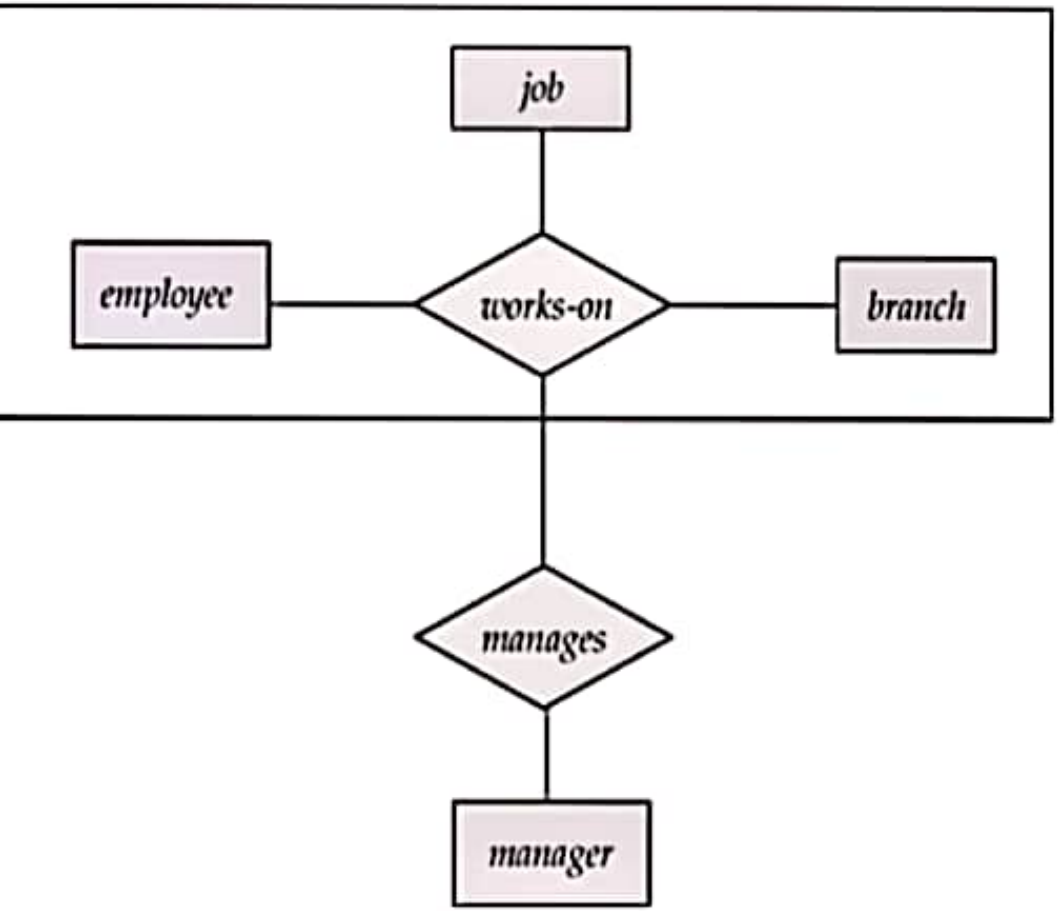



**Solution-**

Minimum 4 tables will be required:

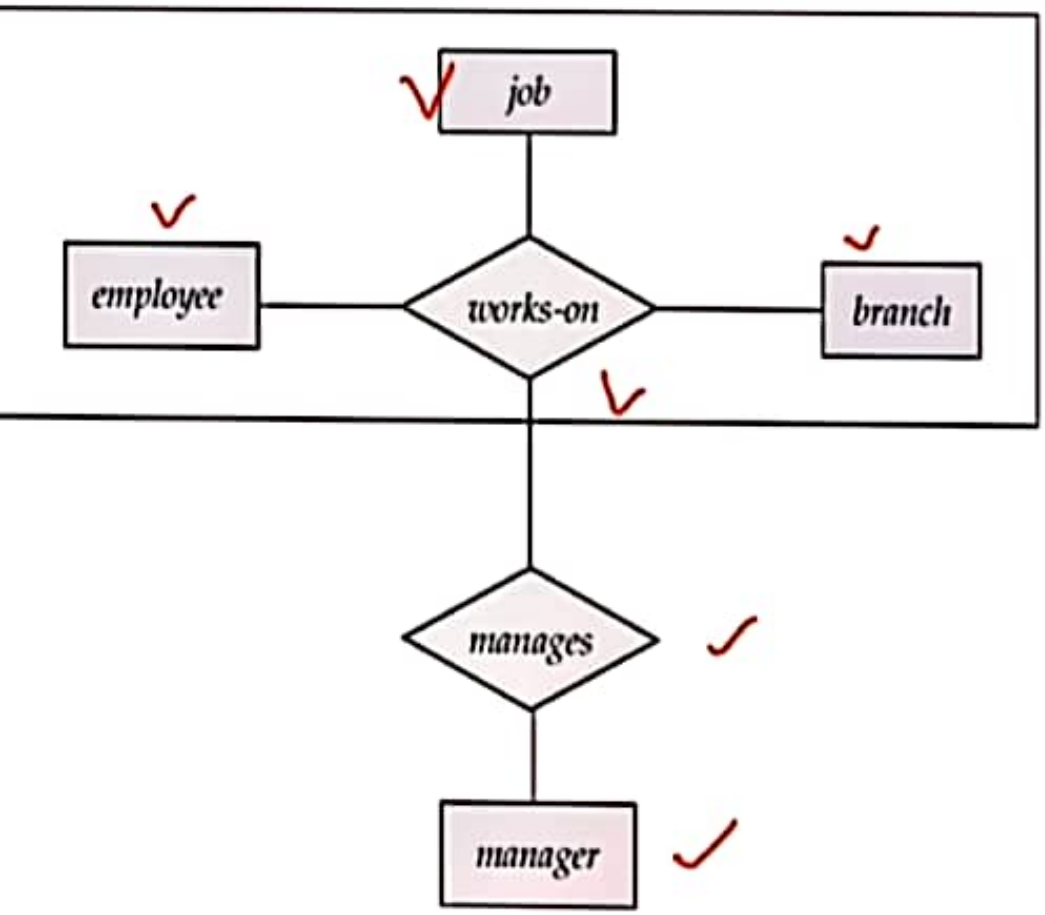
1. **customer** (name, street, city, credit_rating)
2. **officer** (name, street, city, salary, office_number)
3. **teller** (name, street, city, salary, station_number, hours_worked)
4. **secretary** (name, street, city, salary, hours_worked)

# Question 5:




ER Diagram with Aggregation

# Question 5:




ER Diagram with Aggregation

**Solution:**

Minimum six tables will be required

*Job schemas:*

1. employee
2. job
3. Branch
4. works_on

*Manager schemas:*

5. manager
6. manages

# Steps to draw an ER Diagram


1. Identify the **entities**
2. Identify the **attributes**
3. Identify the **primary key**
4. Identify the **relationships and participation constraints**

# Symbols Used in E-R Notation

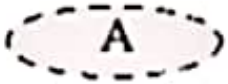

Entity Set


Attribute


Weak Entity Set

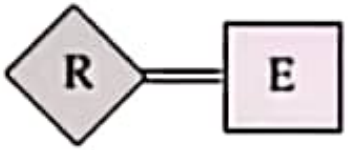

Multivalued Attribute


Relationship Set




Derived Attribute


Identifying Relationship Set for Weak Entity Set

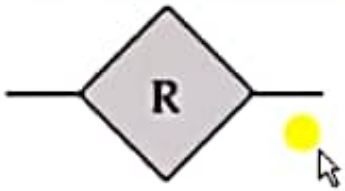



Total Participation of Entity Set in Relationship


Primary Key


Discriminating Attribute of Weak Entity Set

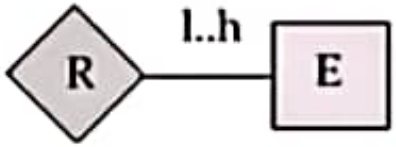
# Symbols Used in E-R Notation



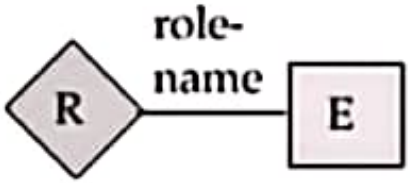

Many to Many Relationship


Many to One Relationship


One to One Relationship




Cardinality Limits




Role Indicator

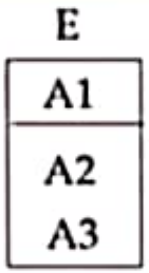

ISA (Specialization or Generalization)


Total Generalization


Disjoint Generalization

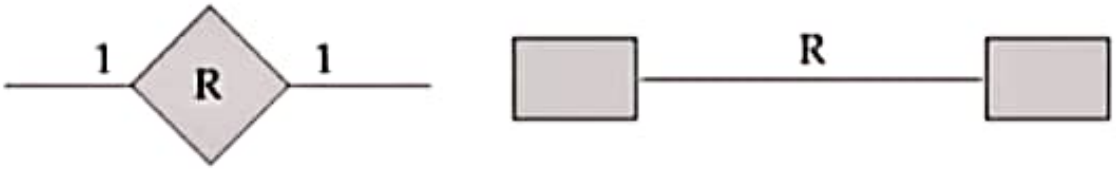
# Alternative E-R Notations

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




Many to Many Relationship

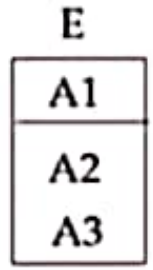

One to One Relationship



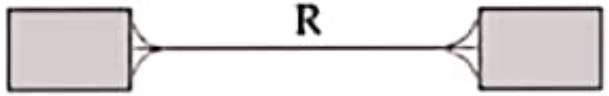

Many to One Relationship

# Alternative E-R Notations

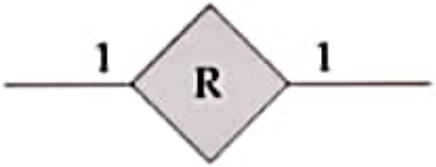
Entity set E with attributes A1, A2, A3 and primary key A1

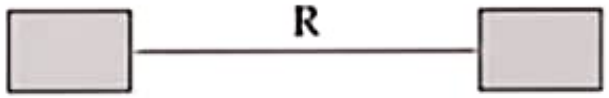



Many to Many Relationship

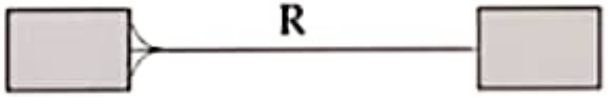




One to One Relationship




Many to One Relationship

# Requirements for a Bank Database:

- 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.

# 1. Identifying the ENTITIES


- 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.

# 1. Identifying the ENTITIES

**Banking Database consists of 6 entities:**

1. Branch
2. Customer
3. Employee
4. Account
5. Loan
6. Payment (it is a weak entity depends on Loan entity)

# 2. Identify the ATTRIBUTES


- 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.

# 2. Identify the ATTRIBUTES


1. **branch**: branch-name, branch-city, assets
2. **customer**: customer-id, customer-name, customer-street, customer-city
3. **employee**: employee-id, employee-name, telephone-number, start-date, dependent-name, employment-length
4. **account**: account-number, balance
5. **loan**: loan-number, amount
6. **payment**: payment-number, payment-date, payment-amount

# 3. Identify the PRIMARY KEY

| Entity | Primary Key |
|---|---|
| branch | branch-name |
| customer | customer-id |
| employee | employee-id |
| account | account-number |
| loan | loan-number |
| payment | payment-number |

# 4. Identify the RELATIONSHIPS

- 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.

# 4. Identify RELATIONSHIP & PARTICIPATION CONTR


- A **customer** can have multiple **accounts** and an account can be held by multiple customers. Cardinality will be M : N

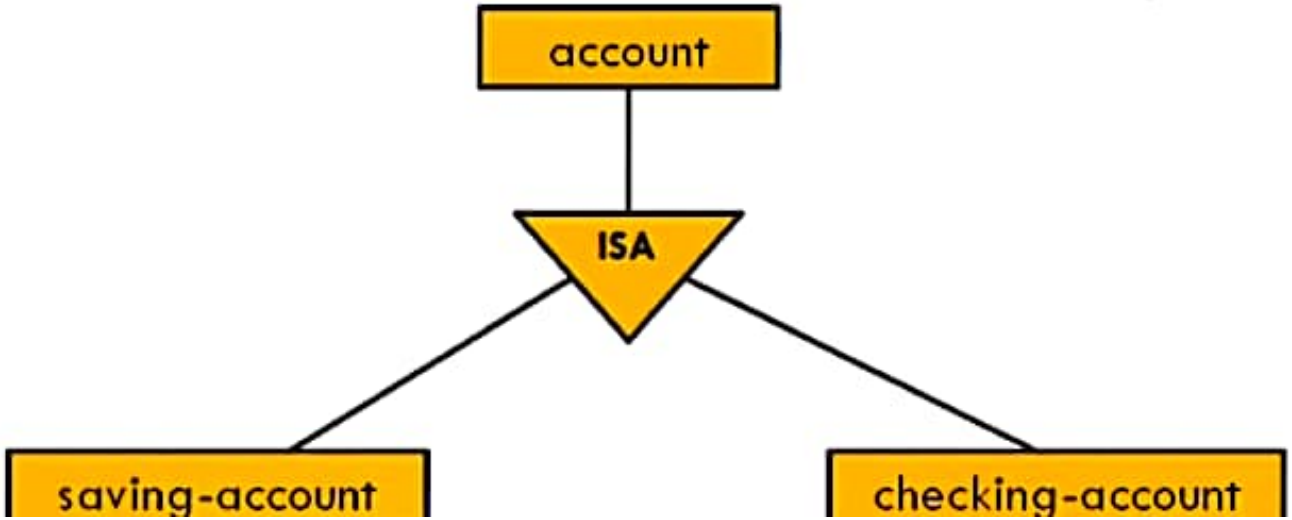

- **Customers** are allowed to take **loans** from the bank, a loan can be held by one or more customers. Cardinality will be M : N

- Each **loan** is paid back with the multiple **payments** having payment number. Cardinality will be 1 : N


- A **customer** can be an **employee** of the bank. Cardinality will be M : N

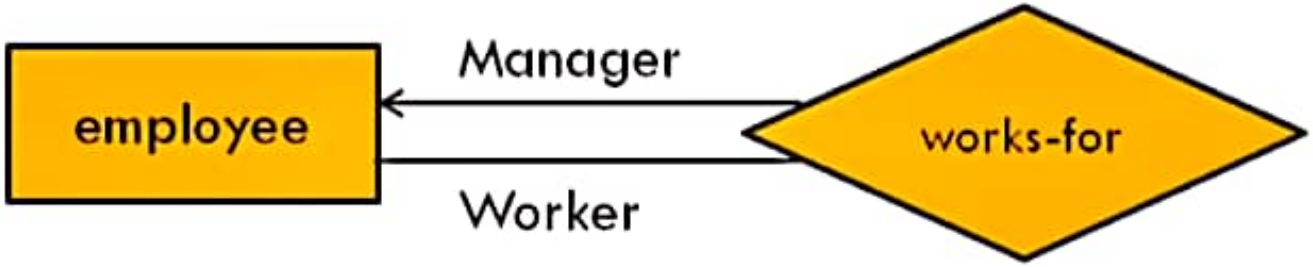
- A **branch** can have multiple **accounts.** Cardinality will be 1 : N


- An **account** can be divided into two accounts: saving account and checking account



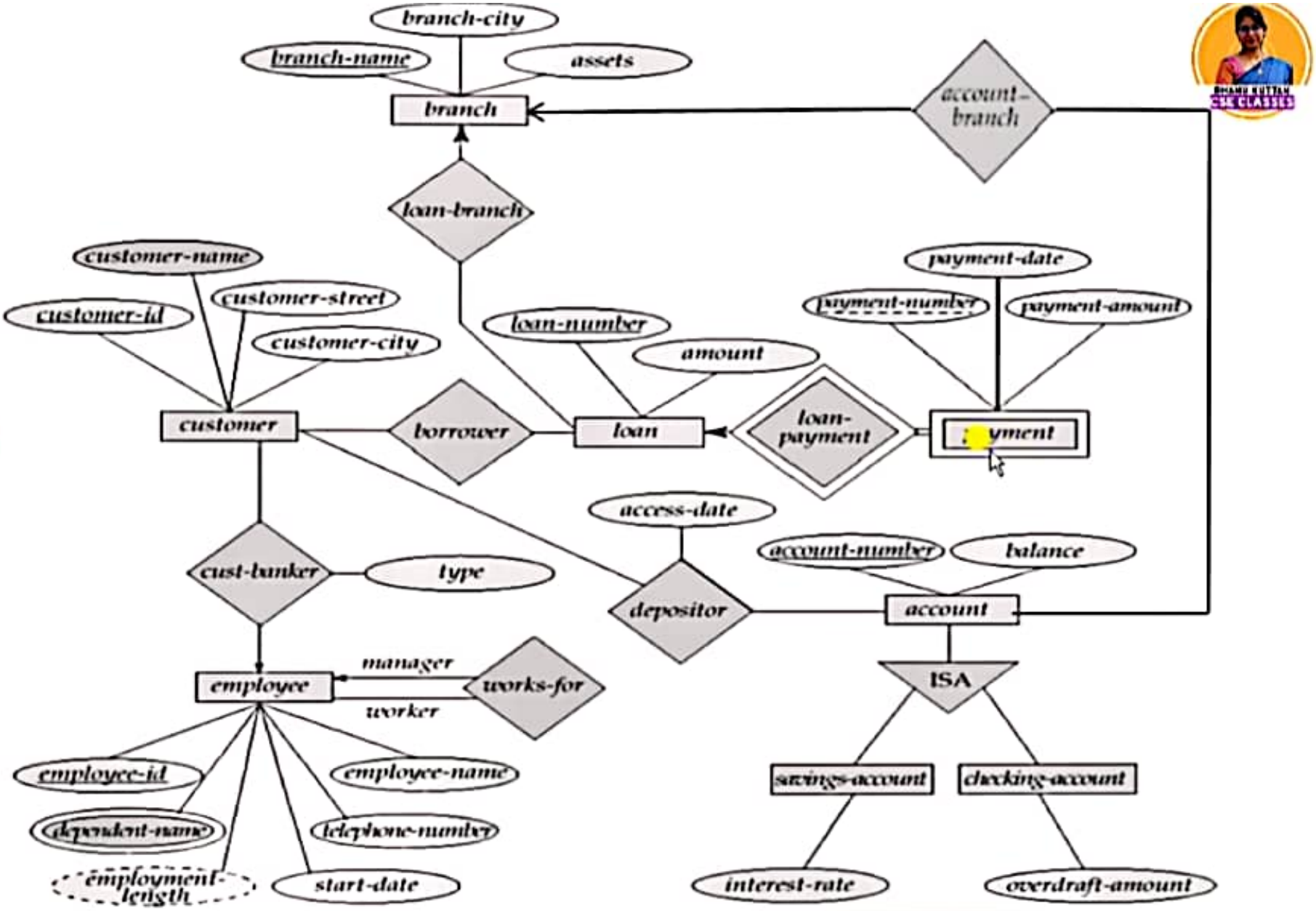
- A **branch** of bank give **loans** to the customers. Cardinality will be 1 : N


- **Employees** can be a **manger** or a **worker** of that particular branch. (Recursive Relationship exist)



# 5. Construct ER Diagram

## ER Diagram Of Banking Enterprise



## 5. Construct ER Diagram

**ER Diagram Of Banking Enterprise**