# ER DIAGRAM

- Instructor

  Dr. Sanam Shahla Rizvi

  PhD in Information and Communication from Ajou University, Korea

# Entity-Relationship Model

- **Entity-Relationship (ER) model** is a popular conceptual data model.
- This model is used in the design of database applications
- The model describes data to be stored and the constraints over the data.
- E-R model views the real world as a collection of **entities** and **relationships** among entities.

# OUTLINE

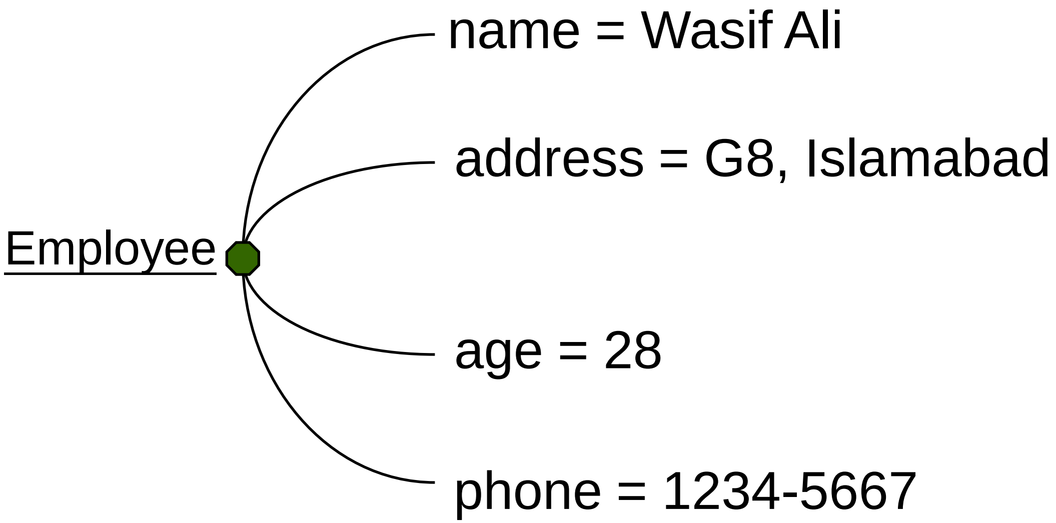
- Entity
- Relationship
  - Binary relationship
- Weak Entity/Strong Entity
- Class Hierarchy
- Relationship
  - Non-Binary relationship

# Entities and attributes

- An **entity** is an object in the real world that is distinguishable from other objects
  - E.g.,
    - A classroom
    - A teacher
    - The address of the teacher

# ENTITIES AND ATTRIBUTES

- An **entity** is described using a set of **attributes** whose values are used to distinguish one entity from another of same type



Employee

name = Wasif Ali

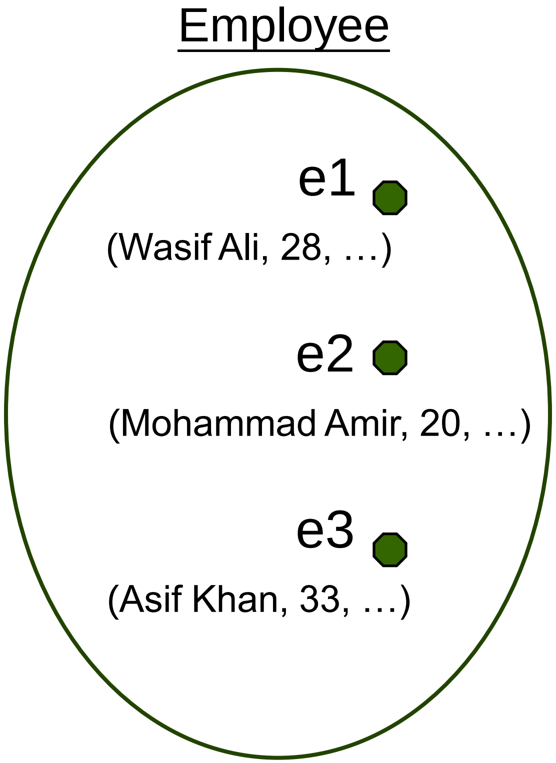
address = G8, Islamabad

age = 28

phone = 1234-5667

# Entities and attributes

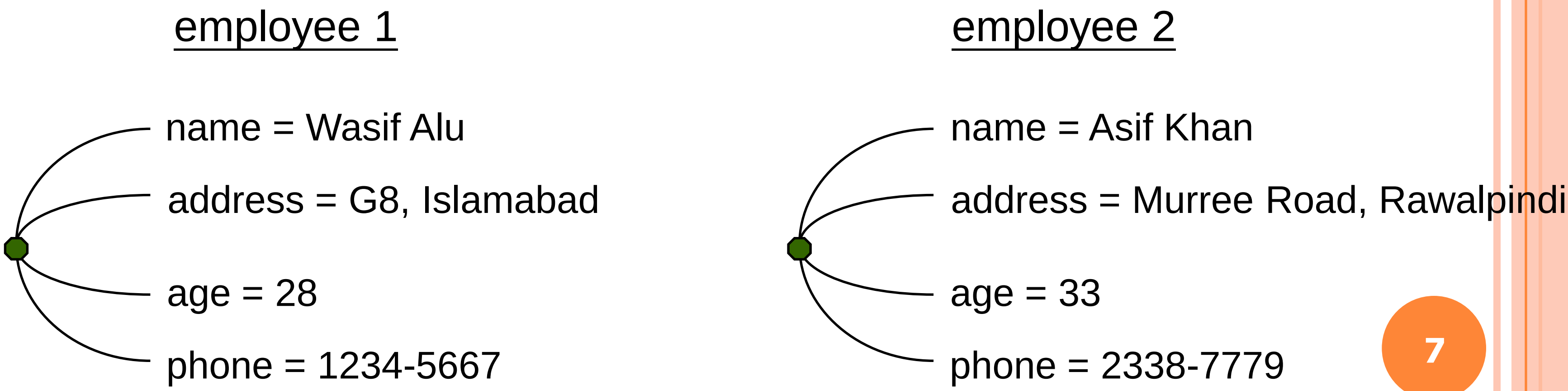
- An **entity set** is a collection of entities of the same type



# ENTITIES AND ATTRIBUTES

- All entries in a given entity set have the same attributes (the values may be different).

employee = (name, address, age, phone)



employee 1

name = Wasif Alu

address = G8, Islamabad

age = 28

phone = 1234-5667

employee 2

name = Asif Khan

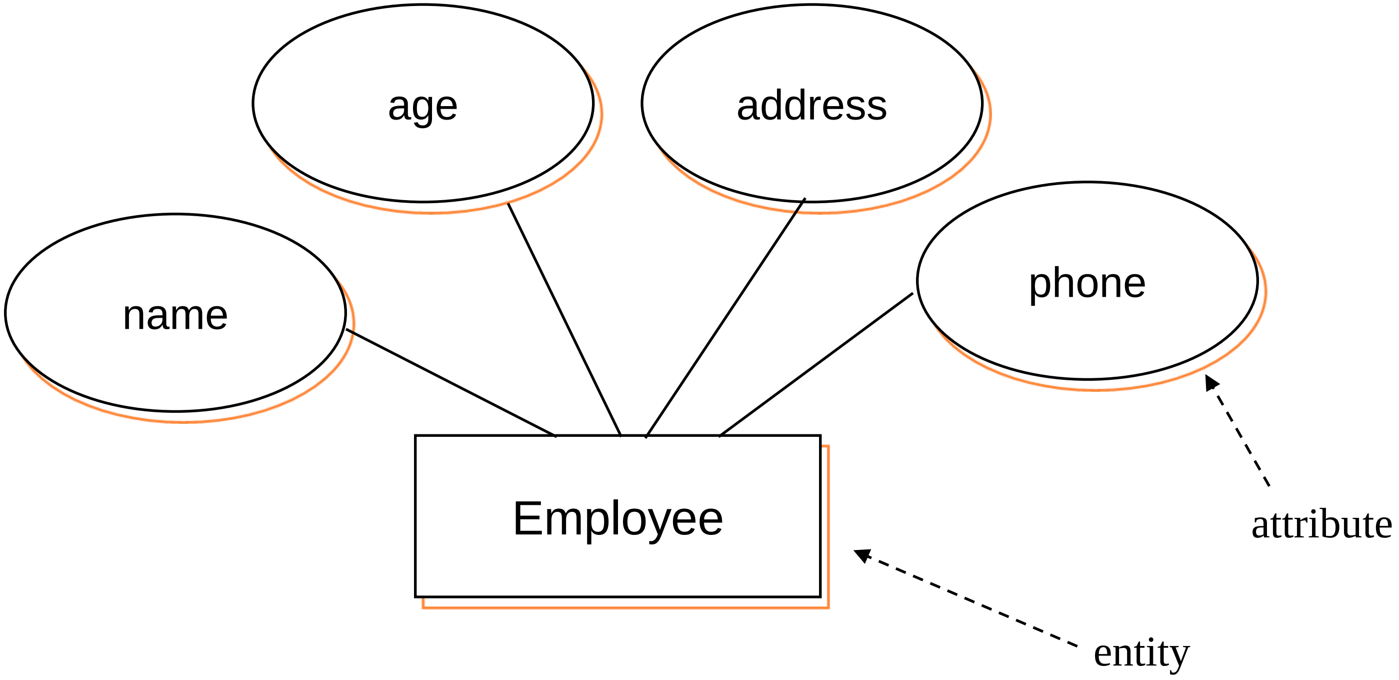
address = Murree Road, Rawalpindi

age = 33

phone = 2338-7779

# ER Diagram

- The ER model can be presented graphically by an ER diagram

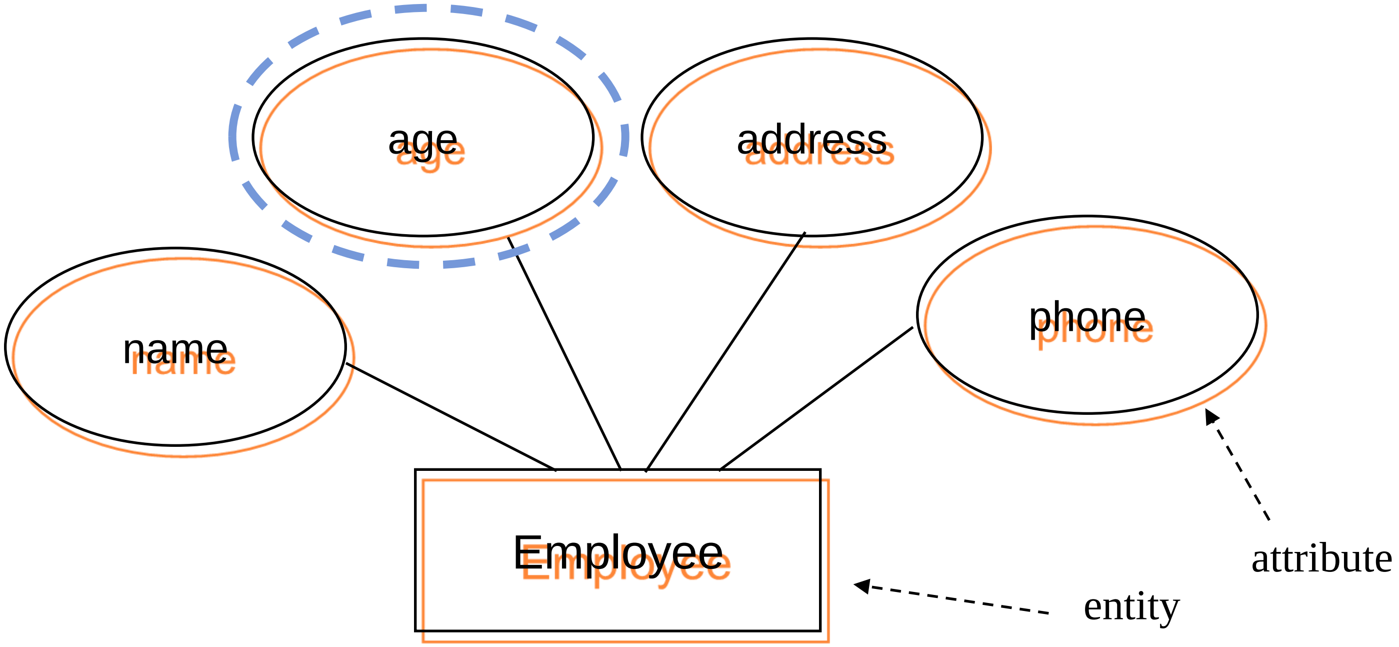


# DIFFERENT ATTRIBUTE TYPES

- Simple attribute
- Composite attribute
- Multi-valued attribute
- Derived attribute

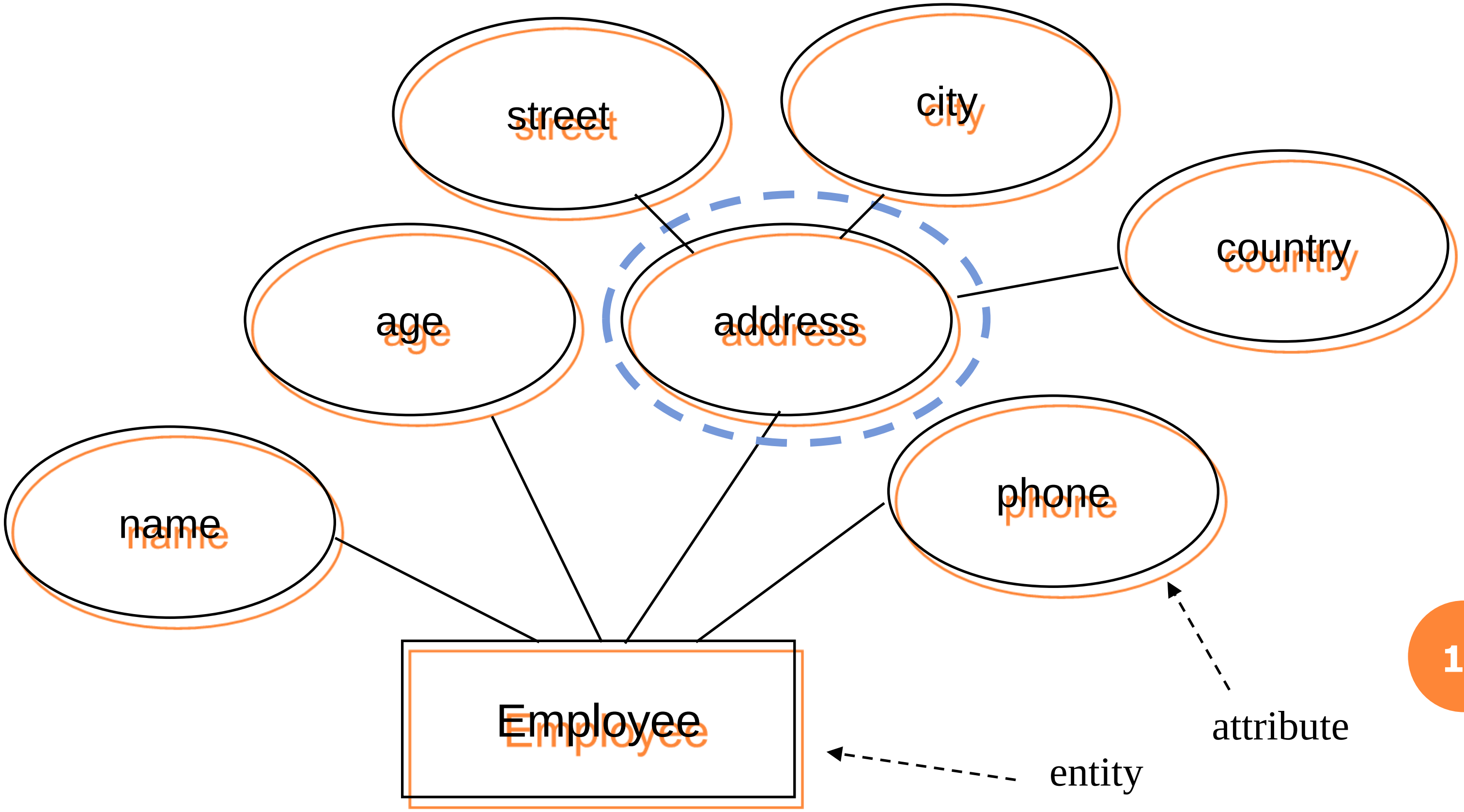
# Simple attribute

- **Simple attribute**
  - contains a single value



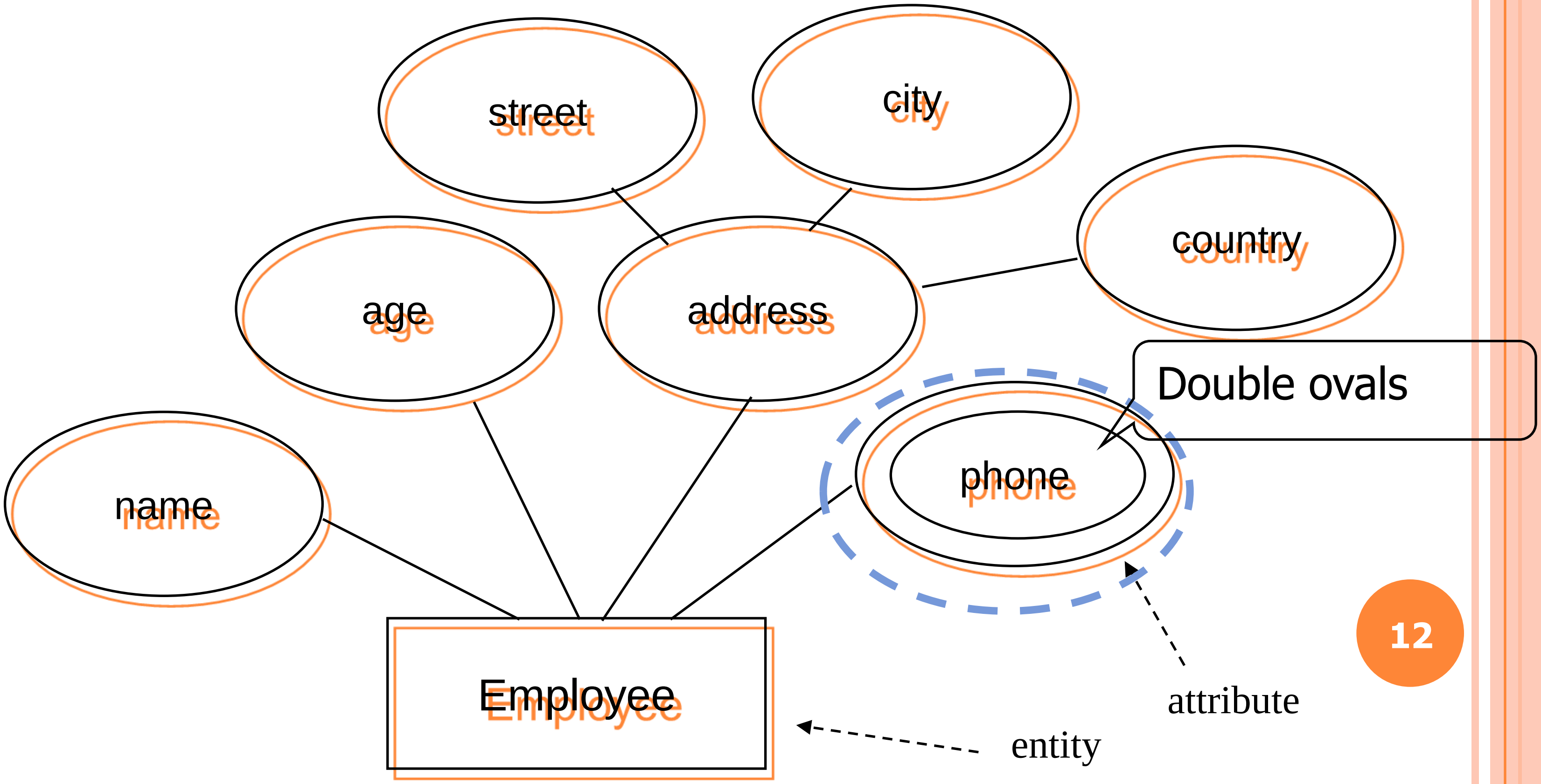
# Composite attribute

- **Composite attribute**
  - Contains several components



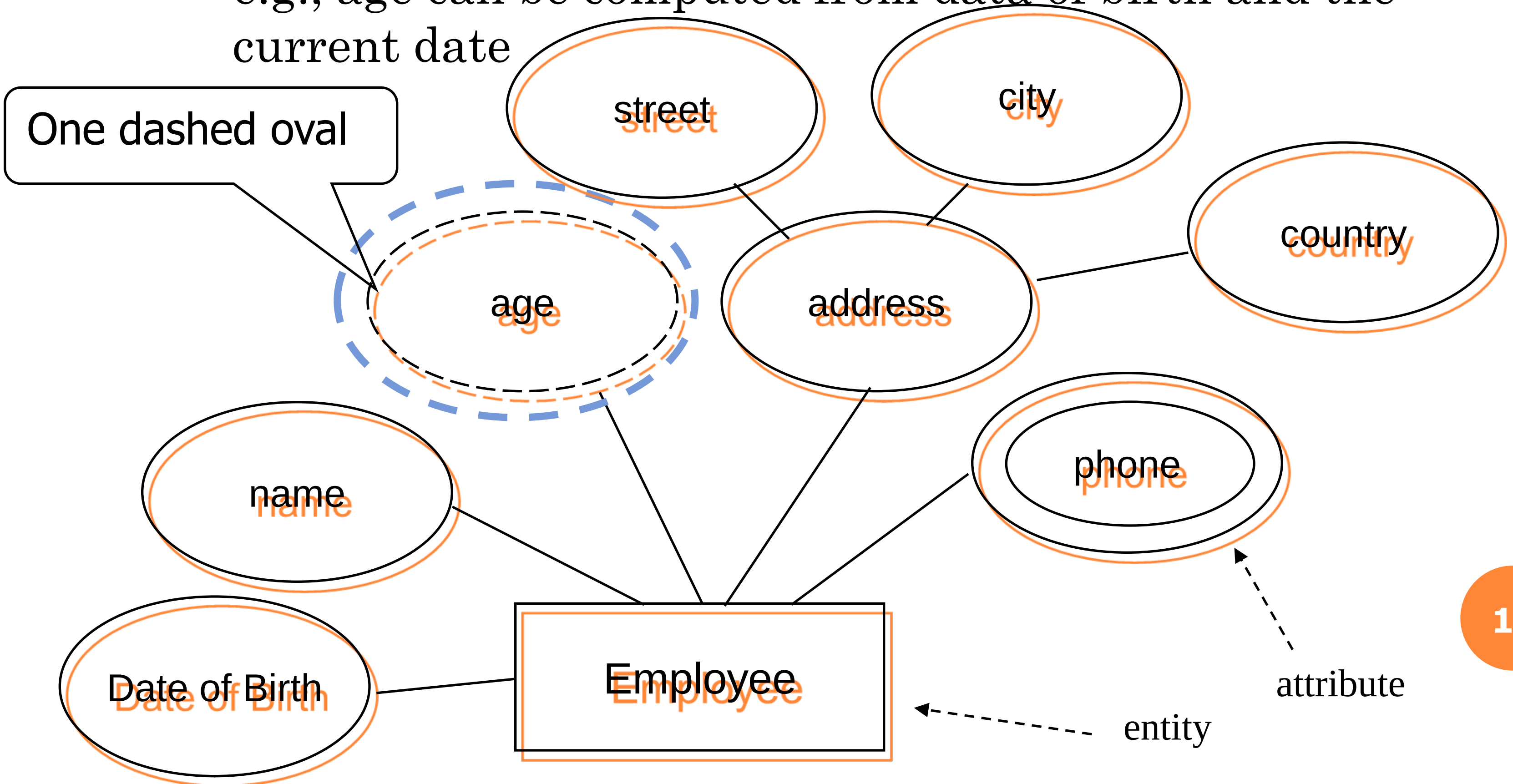
# MULTI-VALUED ATTRIBUTE

- **Multi-valued attribute**
  - Contains more than one value



# Derived attribute

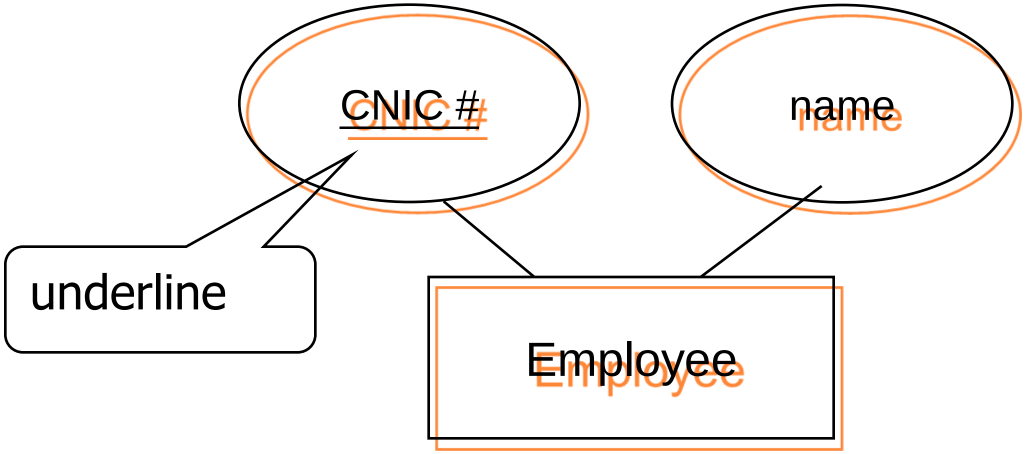
- **Derived attribute**
  - Computed from other attributes
  - e.g., age can be computed from data of birth and the current date



One dashed oval

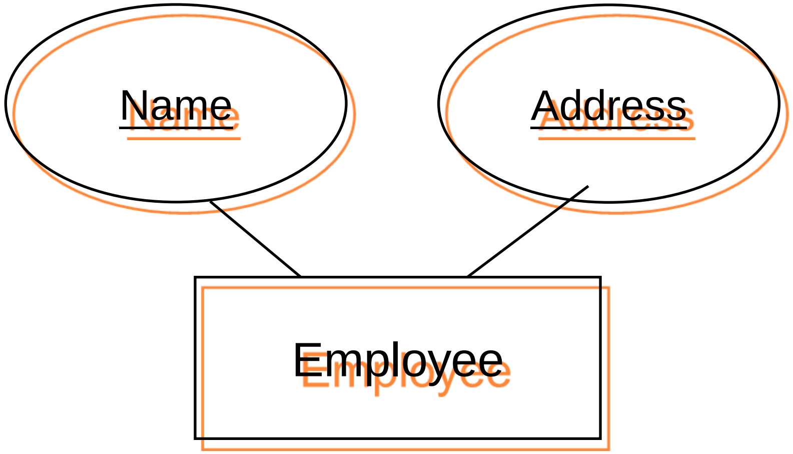
# KEY ATTRIBUTES

- **Key**
  - A set of attributes that can **uniquely** identity an entity
  - E.g., Identity card number



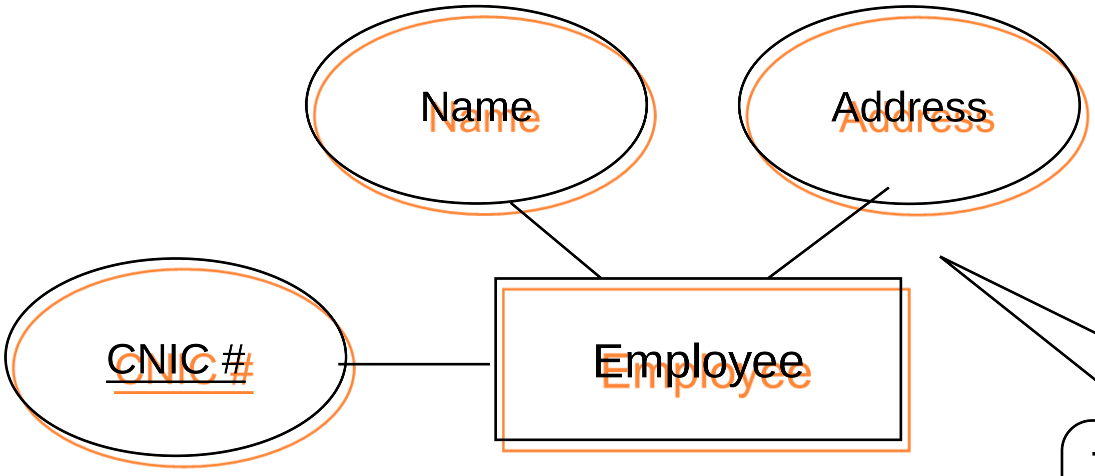
# Key attributes

- **Composite Key**
  - Two or more attributes are used to serve as a key
  - E.g., Name or Address alone cannot uniquely identify an employee, but together they can



# KEY ATTRIBUTES

- An entity may have more than one key
- E.g., {CNIC #} and {Name, Address} both are two keys



Is {CNIC #, Name, Address} a key?

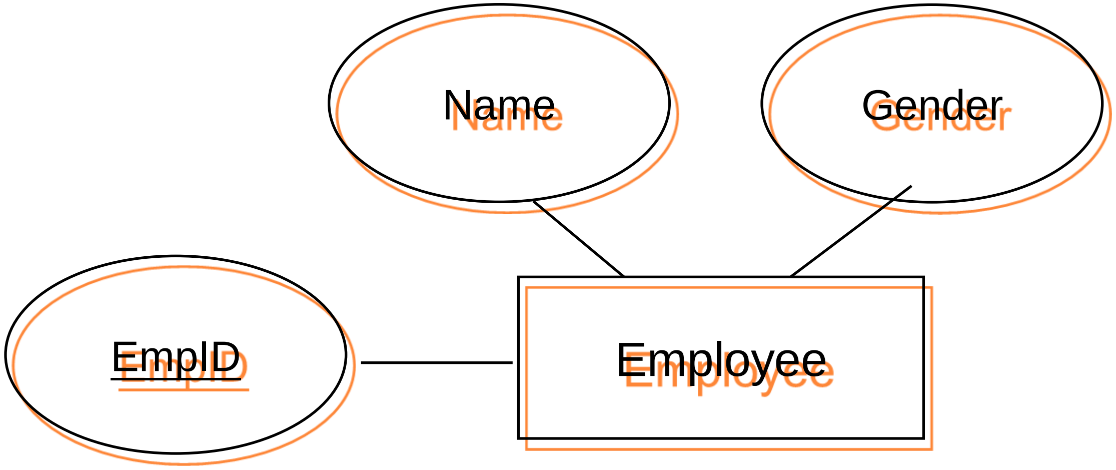
# Key attributes

- A **minimal** set of attributes that uniquely identifies an entity is called a **candidate key**.
- E.g., {CNIC #} and {Name, Address} both are two candidate keys

- If there are many candidate keys, we should choose one candidate key as the **primary key**.

# Key attributes

- Sometimes, artificial keys can be created
- E.g., if there is no CNIC # stored in Employee, we can create a new attribute called “EmpID”



# OUTLINE

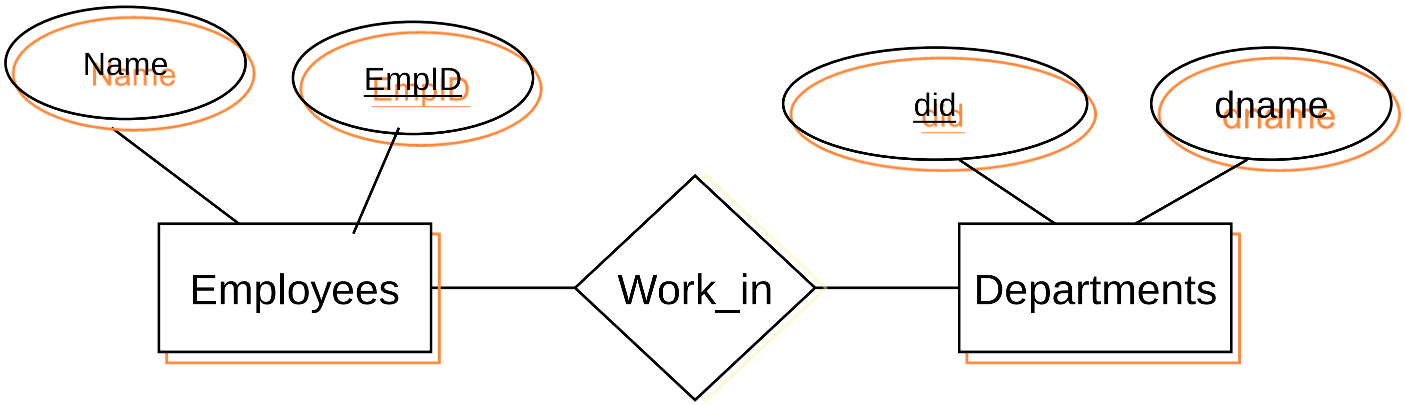
- Entity
- Relationship
  - Binary relationship
- Weak Entity/Strong Entity
- Class Hierarchy
- Relationship
  - Non-Binary relationship

# Relationship

- A relationship is an association among several entities
- The **degree** refers to the number of entity sets that participate in a relationship set.
- Relationship sets that involve two entity sets are **binary** (or degree two).
- Relationships among more than two entity sets are rare.

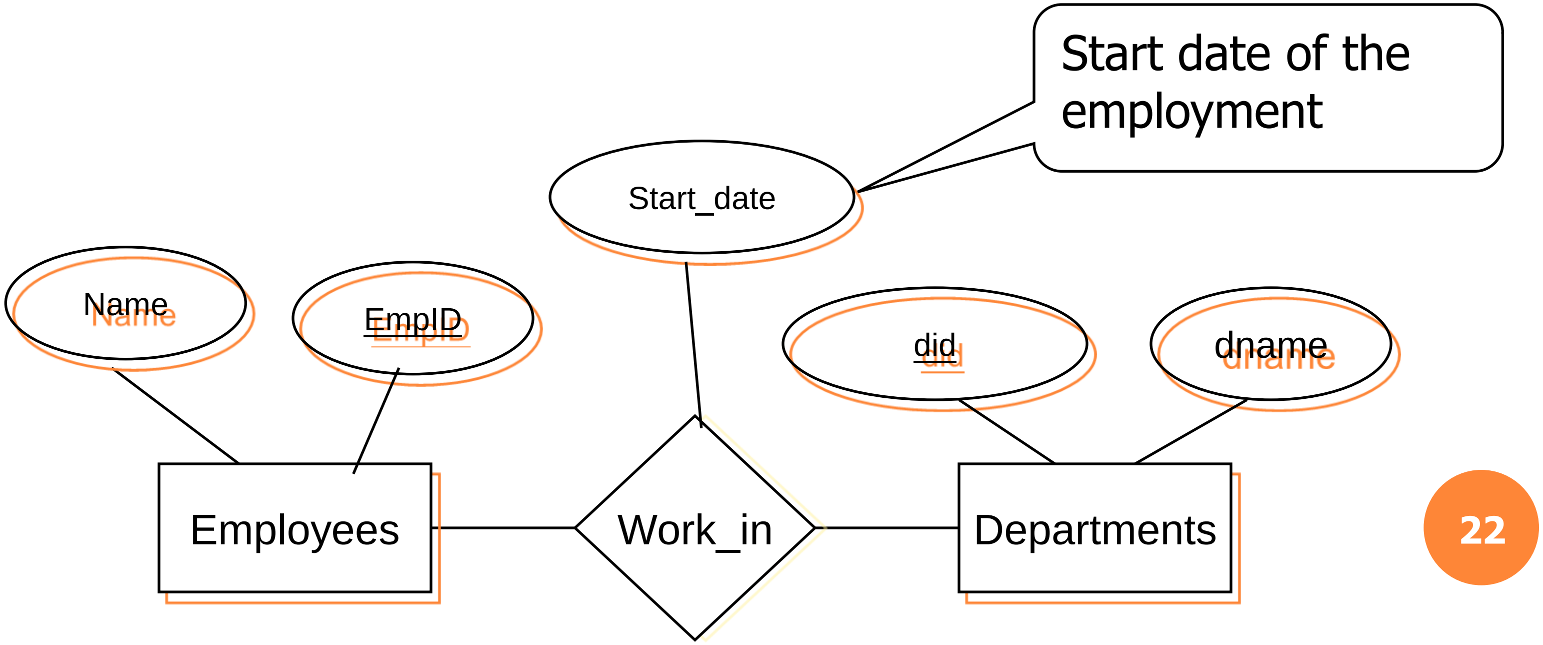
# Binary Relationship

- Employees work in departments
- "Work_in" is a relationship between Employees and Departments



# Binary Relationship

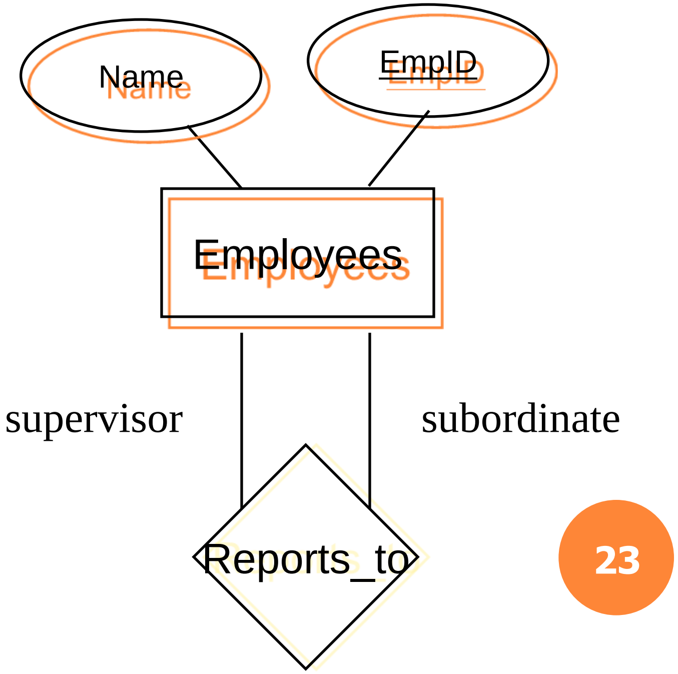
- A relationship can also have attributes which are used to describe the record information about the **relationship** (instead of the information of each individual entity).



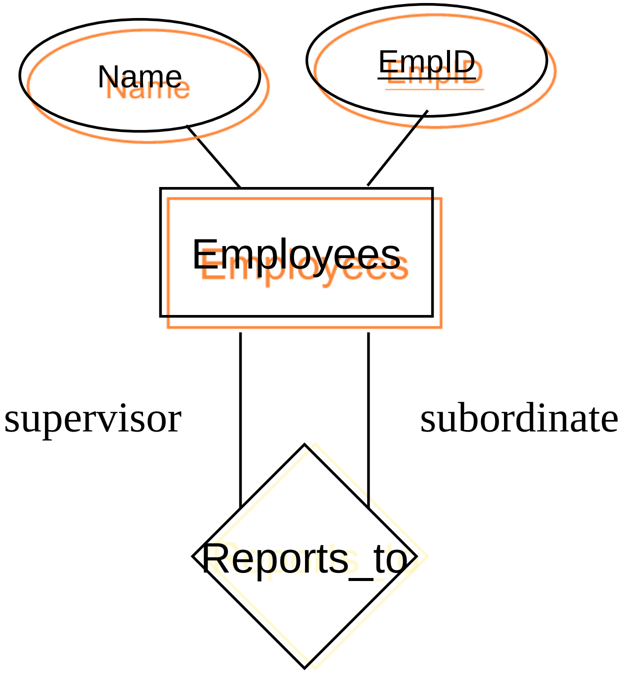
Start date of the employment

# Recursive Relationship

- **Recursive Relationship**
  - Entity sets of a relationship need not be distinct
  - Sometimes, a relationship might involve two entities in the same entity set
  - E.g., Employees related to employees



# Recursive Relationship



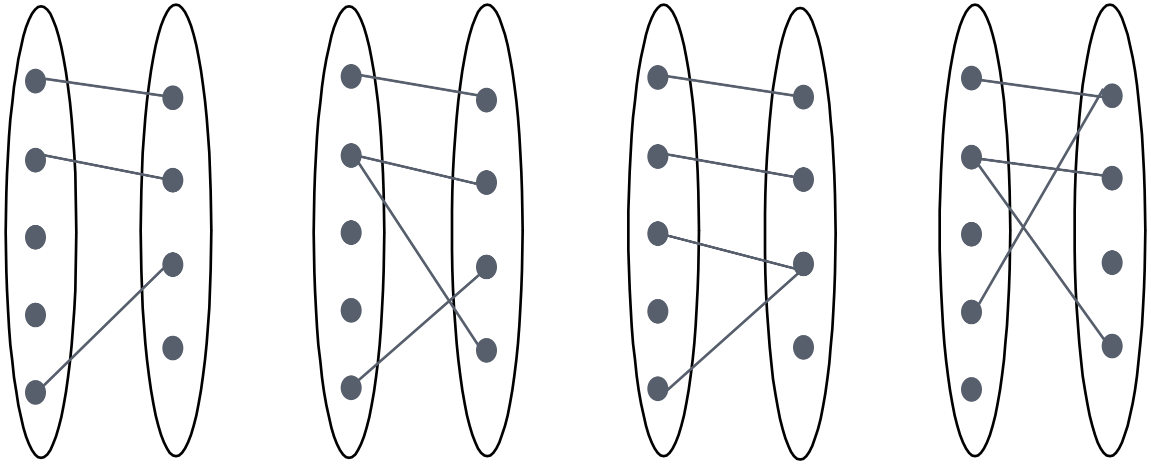
- Since employees report to other employees
  - Every relationship in “Reports_To” is of form (emp1, emp2) where both emp1 and emp2 are entities in employees.
  - However, they play different roles.
    - emp1 reports to emp2, which is reflect in the role indicators supervisor and subordinate in the diagram

# CONSTRAINTS

- The model describes data to be stored and the **constraints** over the data.
- **Cardinality Ratio** specifies the number of relationship instances in which an entity can participate.

# Constraints

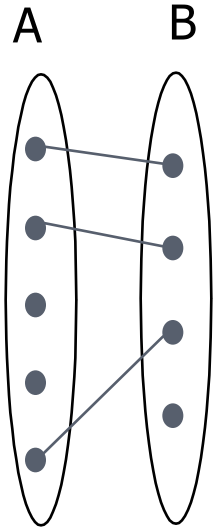
- The mapping of a binary relationship can be classified into the following cases:



1-to-1 1-to Many Many-to-1 Many-to-Many

# One-to-one relationship

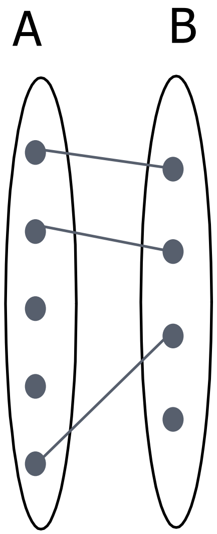
- One-to-one (1-to-1) relationship
  - An entity in A is related to at most one entity in B
  - An entity in B is related to at most one entity in A.



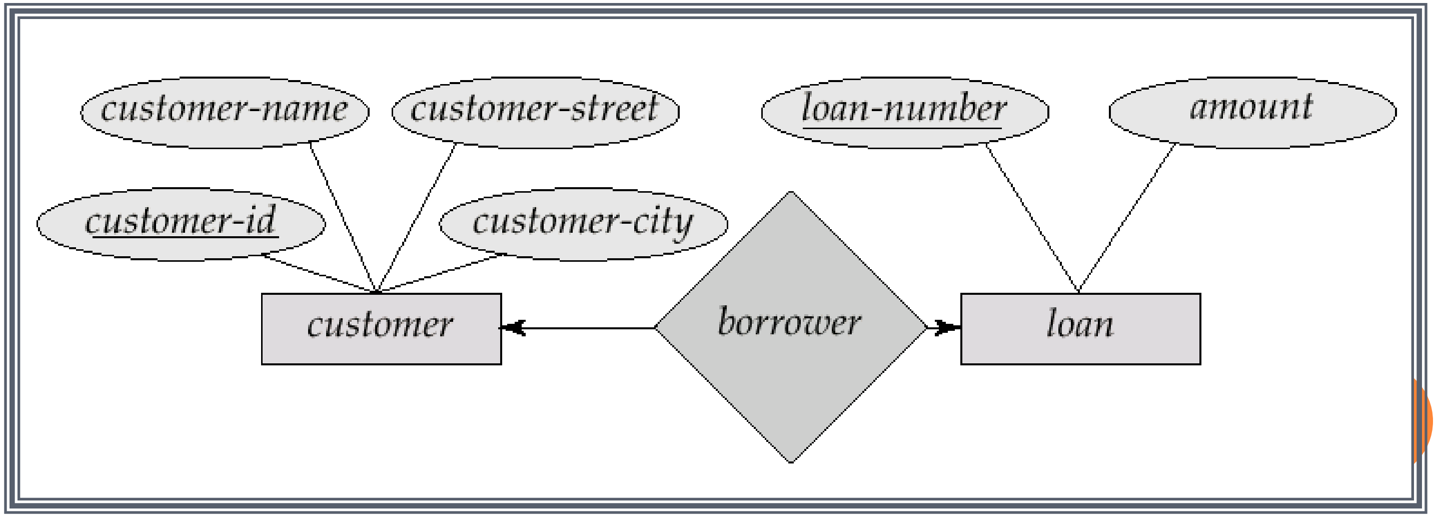
1-to-1

# ONE-TO-ONE RELATIONSHIP

- A customer is associated with at most one loan via the relationship borrower
- A loan is associated with at most one customer via borrower

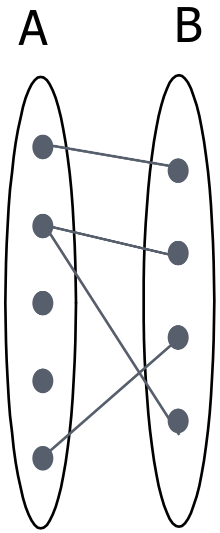


1-to-1



# One-to-many relationship

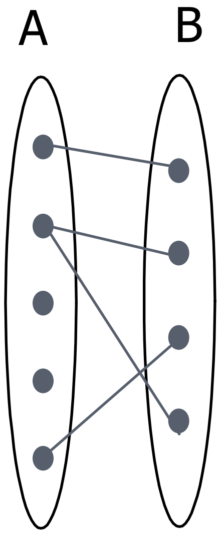
- One-to-many (1-to-Many) relationship
  - An entity in B can be associated with at most one entity in A



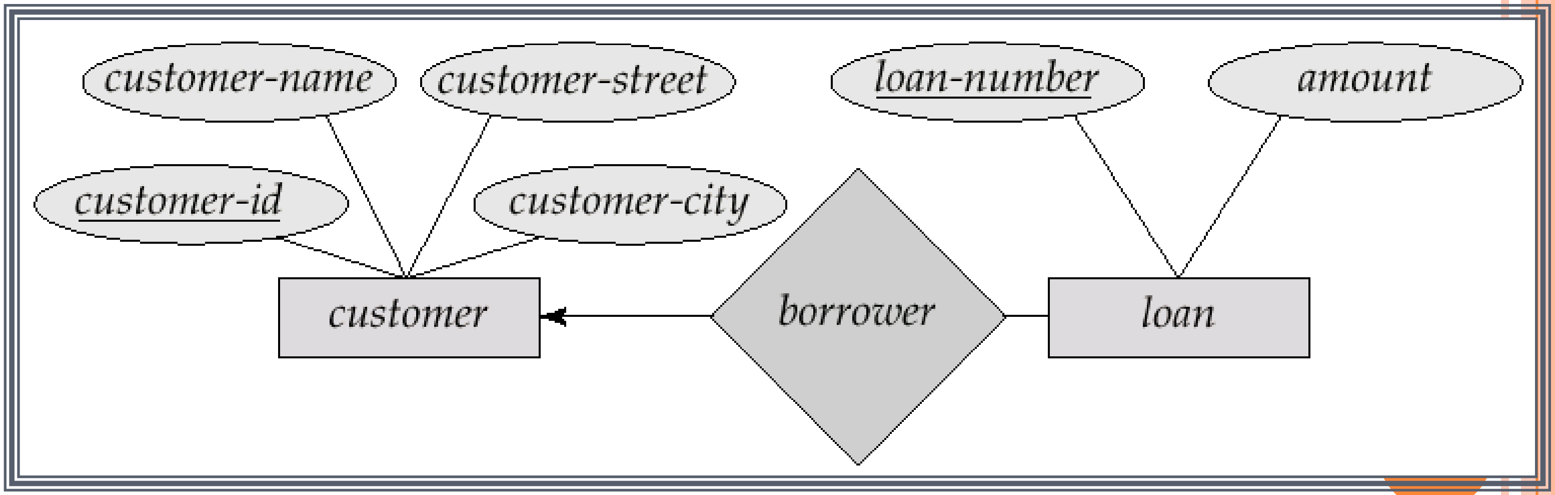
1-to Many

# ONE-TO-MANY RELATIONSHIP

- A loan is associated with at most one customer via borrower
- A customer is associated with several (including 0) loans via borrower

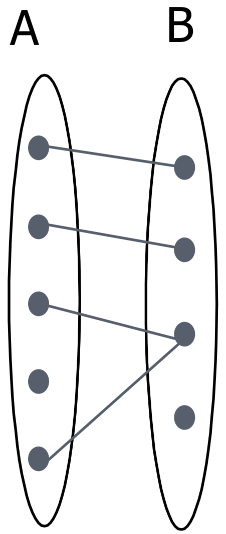


1-to Many



# Many-to-one relationship

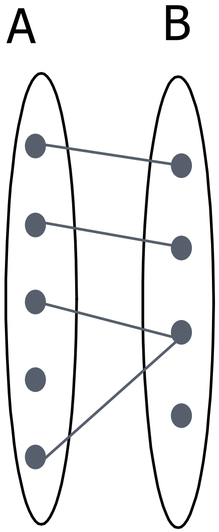
- Many-to-one (Many-to-1) relationship
  - Similar to 1-to-many



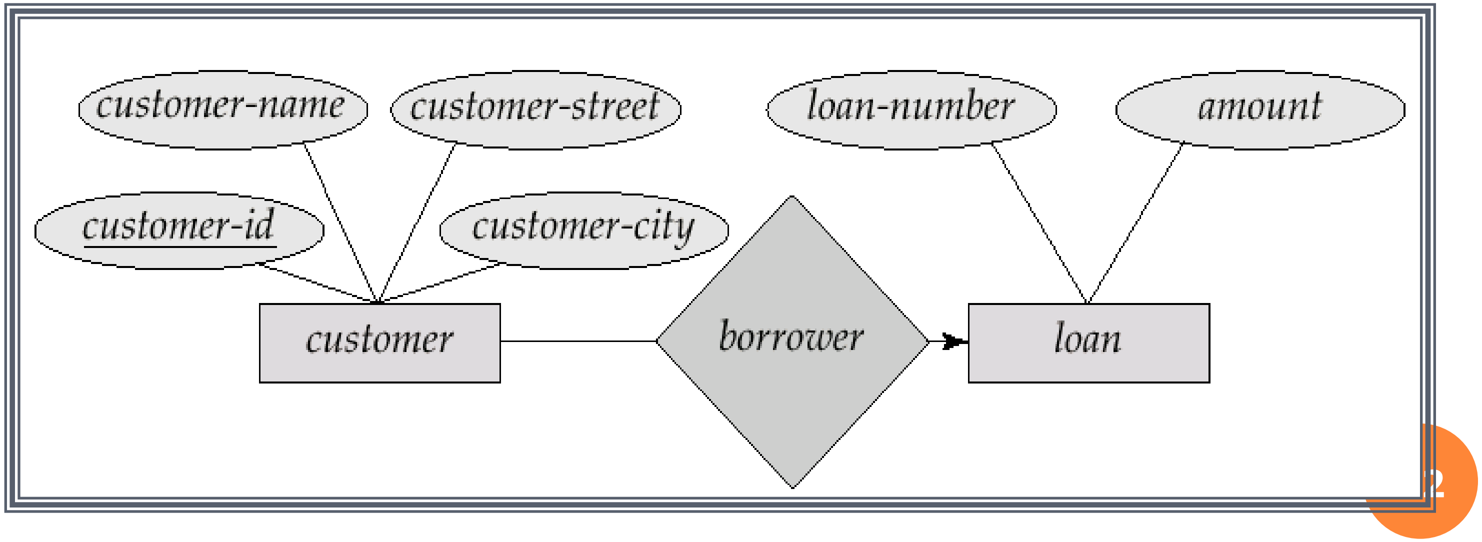
Many-to-1

# Many-to-one relationship

- A loan is associated with several (including 0) customers via borrower,
- A customer is associated with at most one loan via borrower

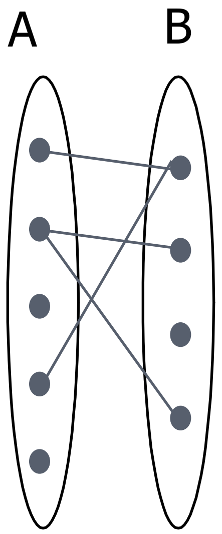


Many-to-1



# Many-to-many Relationship

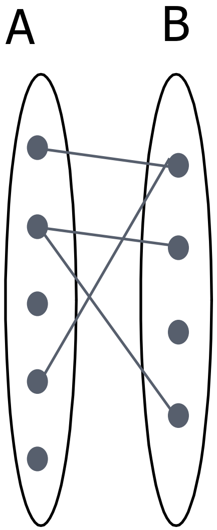
- Many-to-many Relationship
  - An entity in A is associated with any number of entities in B
  - An entity in B is associated with any number of entities in A
  - That is, there is no restriction in the mapping.



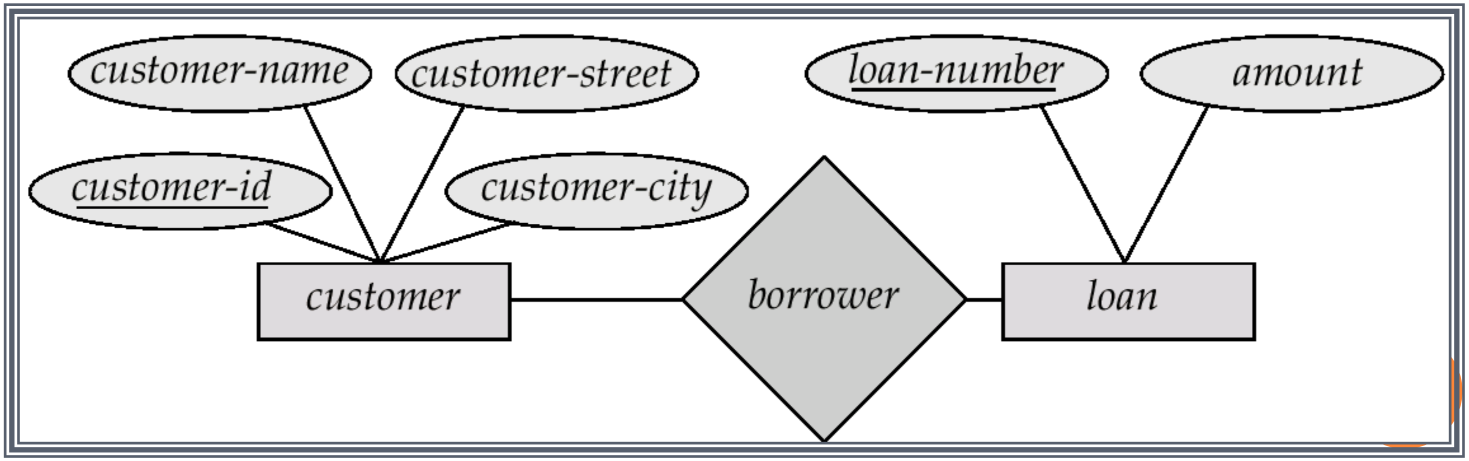
Many-to-Many

# Many-to-many Relationship

- A customer is associated with several (possibly 0) loans via borrower
- A loan is associated with several (possibly 0) customers via borrower

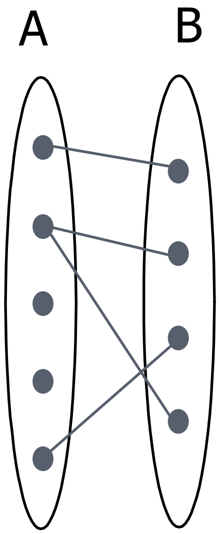


Many-to-Many



# Participation Constraint

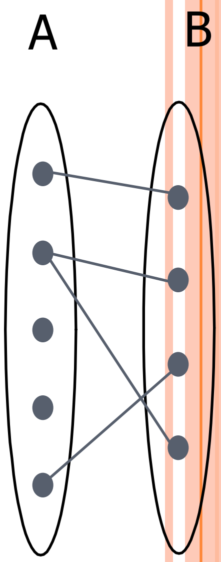
- The above constraints (e.g., 1-to-many) tells us that a customer borrows some loans.
- A natural question to ask is to whether every loan is borrowed by at least one customer.
- Suppose that each loan is borrowed by at least one customer. Such a constraint is called a **participation constraint**.



1-to Many

# Participation Constraint

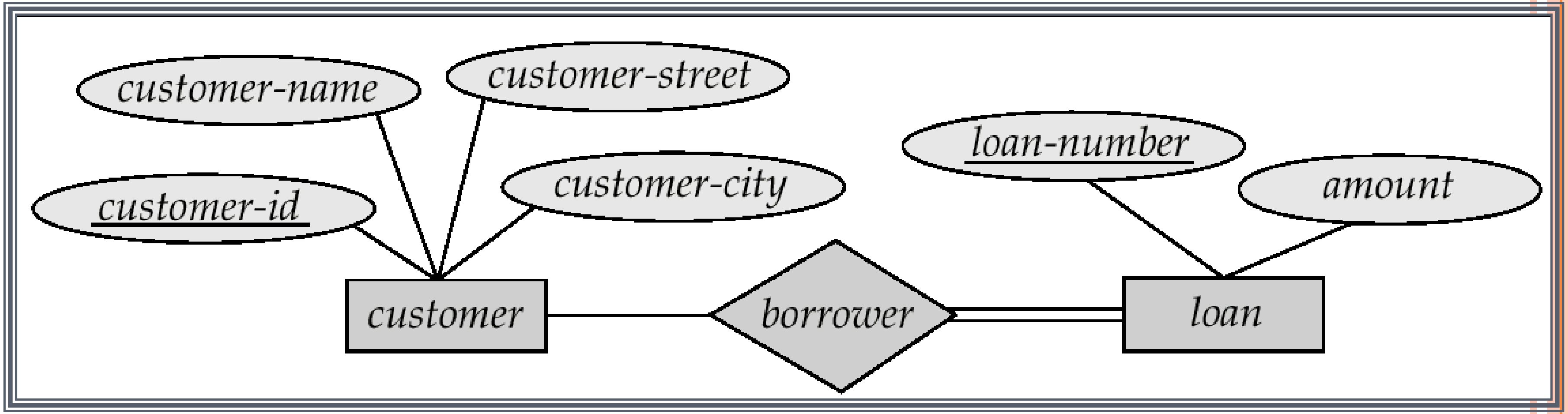
- We can classify participation in relationships as follows.
  - **Total**
    - Each entity in the entity set must be associated in at least one relationship
  - **Partial**
    - Each entity in the entity set may (or may not) be associated in a relationship



**1-to Many**

# Participation Constraint

- The participation of loan in borrower is **total**
  - Every loan must be borrowed by a customer
- The participation of customer in borrow is **partial**
  - Some customers may (or may not) borrow loans



# OUTLINE

- Entity
- Relationship
  - Binary relationship
- Weak Entity/Strong Entity
- Class Hierarchy
- Relationship
  - Non-Binary relationship

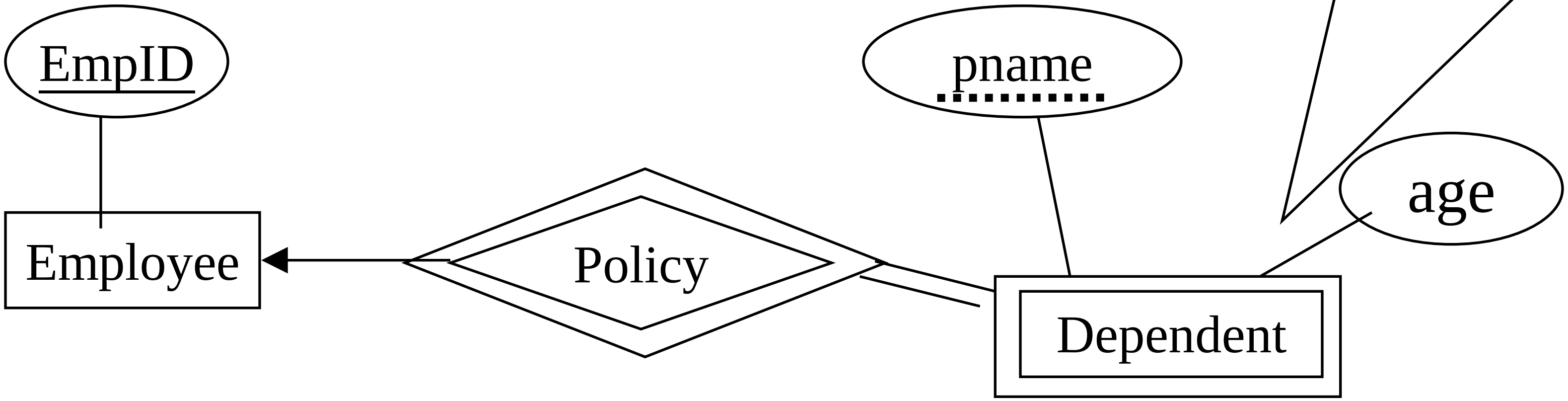
# WEAK ENTITIES

- **Strong Entity**
  - An entity can be uniquely identified by some attributes related to this entity
  - E.g., Employee has an attribute EmpID (which can be used to uniquely identify each employee)
- **Weak Entity**
  - An entity cannot be uniquely identified by all attributes related to this entity

# Weak Entities

- Example
  - Suppose employees can purchase insurance policies to cover their dependants.
  - The attribute of the dependents entity set are pname and age
  - The attribute pname cannot uniquely identify a dependent
  - Dependent is a weak entity set.
  - A dependent can only be identified by considering some of its attributes in conjunction with the primary key of employee (**identifying entity set**).
  - The set of attributes that uniquely identify a weak entity for a given owner entity is called a **discriminator** or **partial key**.

# WEAK ENTITIES



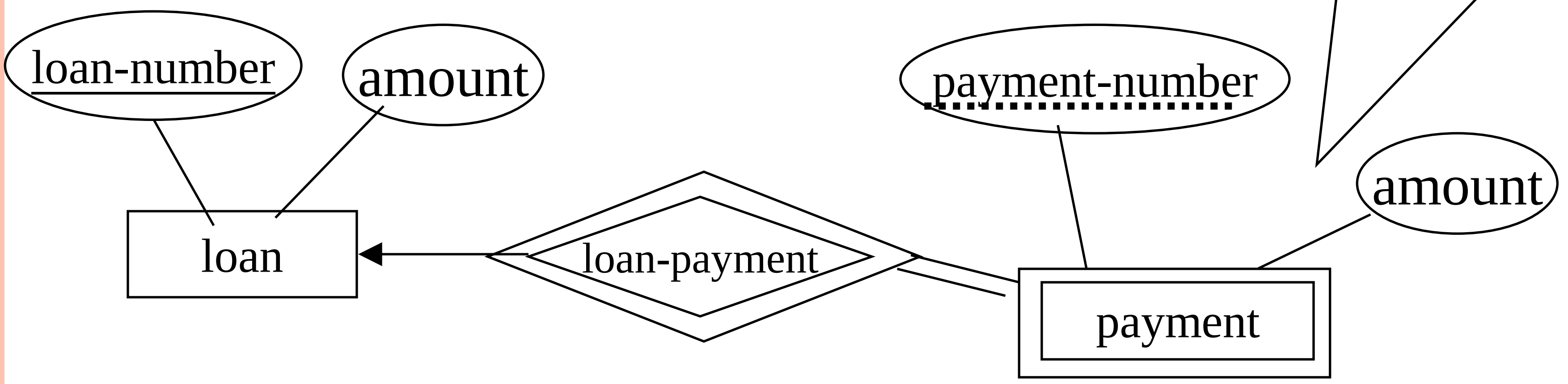
Identifying entity set = Employee
Discriminator = {pname}
Key = {EmpID, pname}

- A dependent cannot be uniquely identified by “pname”.
- Note:
  - A child may not be old enough to have a CNIC number
  - Even if he/she has a CNIC number, the company may not be interested in keeping it in the database.

- **Definition:** If a weak entity set W is dependent on a strong entity set E, we say that E **owns** W.
  - E.g., Employee owns Dependent

# Weak Entities

Identifying entity set = loan
Discriminator = {payment-number}
Key = {loan-number, payment-number}



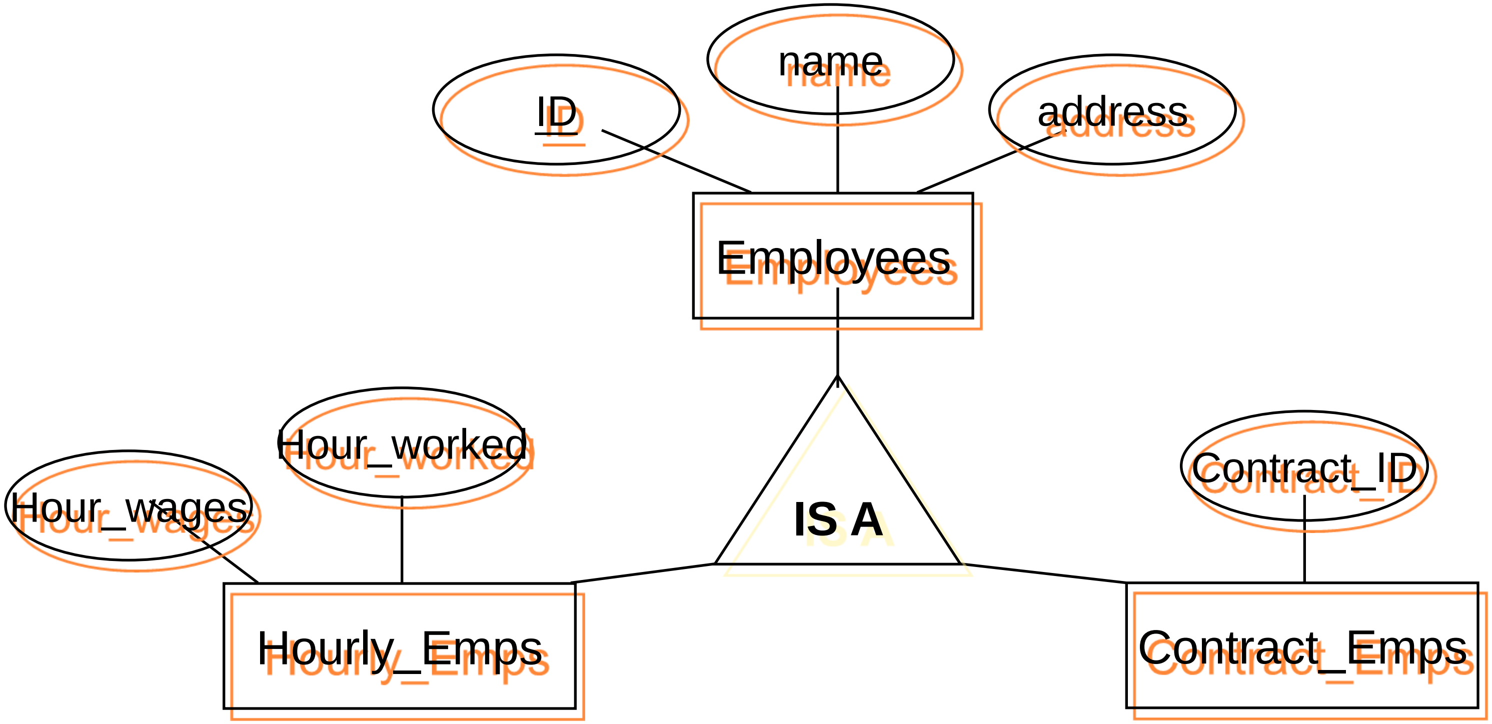
- A payment itself cannot be identified by “payment-number”
- loan owns payment

# OUTLINE

- Entity
- Relationship
  - Binary relationship
- Weak Entity/Strong Entity
- Class Hierarchy
- Relationship
  - Non-Binary relationship

# Class Hierarchy

- Sometimes, it is natural to classify the entities in an entity set into subclasses



# Class Hierarchy

- Attributes are **inherited** by the entity set in the subclass.
  - E.g., the attributes defined for an Hourly_Emps entity are the attributes for Employees plus that of Hourly_Emps
- A class hierarchy can be viewed in one of the two ways.
  - A class is specialized into subclasses.
  - The subclasses are generalized by a superclass.

# Class Hierarchy

- We can specify two kinds of constraints with respect to IS A hierarchies
  - **Overlap constraints**
    - Determine whether two subclasses are allowed to contain the same entity
      e.g., Can an employee be an Hourly_Emps as well as a Contract_emps entity?
  - **Covering constraints**
    - Determine whether the entities in the subclasses collectively include all entities in the superclass.
      E.g., Does every Employees entity also have to be an Hourly_Emps or a Contract_emps

# Class Hierarchy

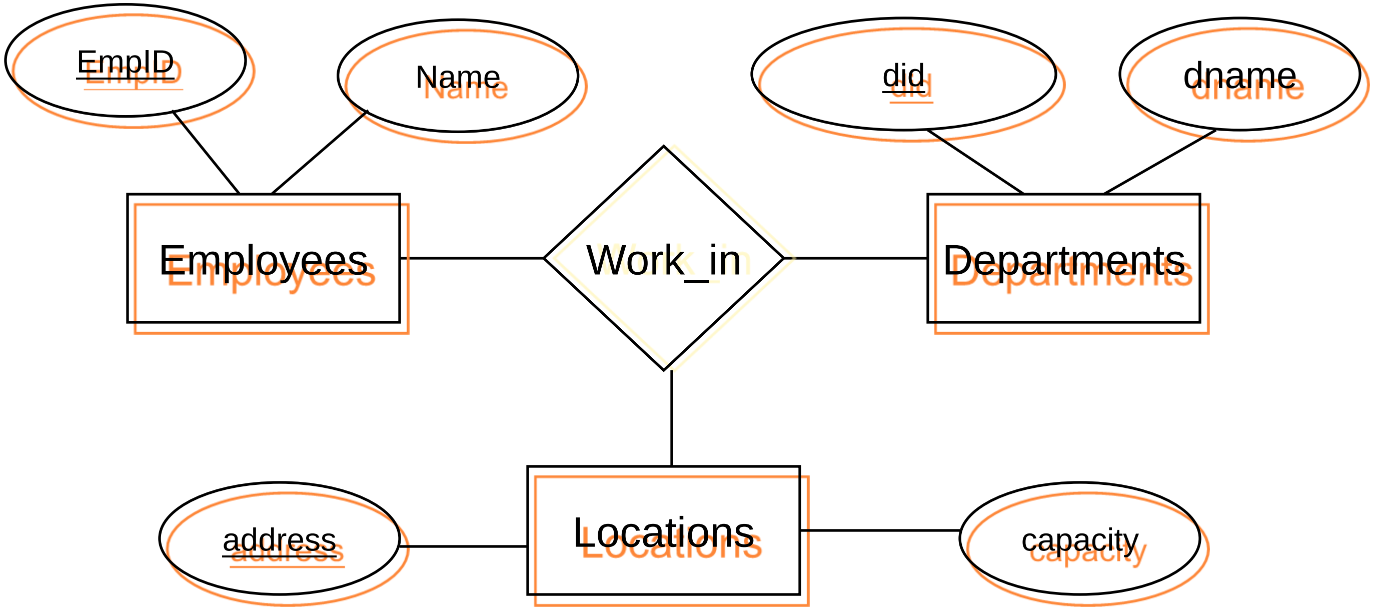
- Reason why we use class hierarchy
  - Add **descriptive attribute** that make sense only for the entities in a subclass
    - E.g., Hourly_wages does not make sense for a Contract_Emps entity.
  - Identify the set of entities that participate in some relationships
    - E.g., We may want to have a relationship called “Bonus” with Contract_Emps (not Hourly_Emps)

# Outline

- Entity
- Relationship
  - Binary relationship
- Weak Entity/Strong Entity
- Class Hierarchy
- Relationship
  - Non-Binary relationship

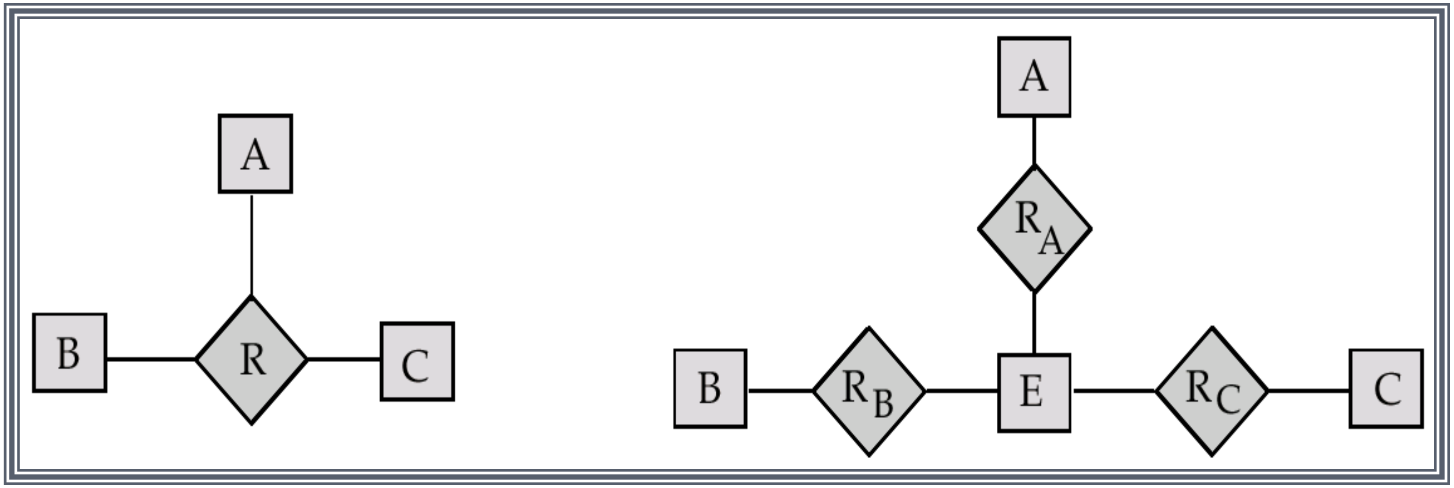
# NON-BINARY RELATIONSHIP

- **Ternary Relationship**
  (i.e., a relationship involving 3 entities)
- An employee works for the department in a location



# Non-binary Relationship

- In general, any non-binary relationship can be represented using binary relationships by creating an artificial entity set.



# Class Work

- Construct an ER diagram for a university department headed by a chair person. Department has a number of faculty members and students. Where each of student is enrolled in one/many classes/courses and each of faculty member teaches one/many courses.

# Home Work

- Construct an ER diagram for a car company that has a set of customers, each of whom owns one/many car. Each car has associate with it zero to any number of recorded accidents.