Database Design

Entity-Relationship Model (ER model)

DR. MOHAMMED A. MOHAMMED

UNIVERSITY OF SULAIMANI | COLLEGE OF SCIENCE | COMPUTER DEPT.

2024 - 2025

Database design

- The process of designing a database can be divided into six steps:

1. Requirement Analysis
   - What data is to be stored?
   - What applications must be built.
   - What operations are the most frequent and subject to performance requirements.

2. Conceptual database design
   - High-level description of
     - ✓ Data to be stored
     - ✓ Constraints
   - Often carried out using the ER model

Database design (cont.)

3. Logical Database design
   - Choose a DBMS to implement our database design.
   - Convert the conceptual database design into a database schema in the data model of the chosen DBMS.
     - Convert ER schema into a relational database schema.

4. Schema Refinement
   - Analyze the collection of relations in our relational database.
   - Identify the potential problems.
     - Refine the schema

Database design (cont.)

5. Physical database design
   - ✓ Ensure that the design meets the performance requirement.
   - ✓ Build index, clustering some tables etc.
   - ✓ Redesign parts of the database schema.

6. Application and security design
   - Write application programs.
   - Identify data that can be accessible by certain types of users.
   - Take steps to ensure that access rules are enforced.

Steps of Database design

1. Requirement analysis
2. Conceptual database design - ER
3. Logical database design – relational
4. Schema refinement
5. Physical database design
6. Application and security design

Entity-Relationship Model

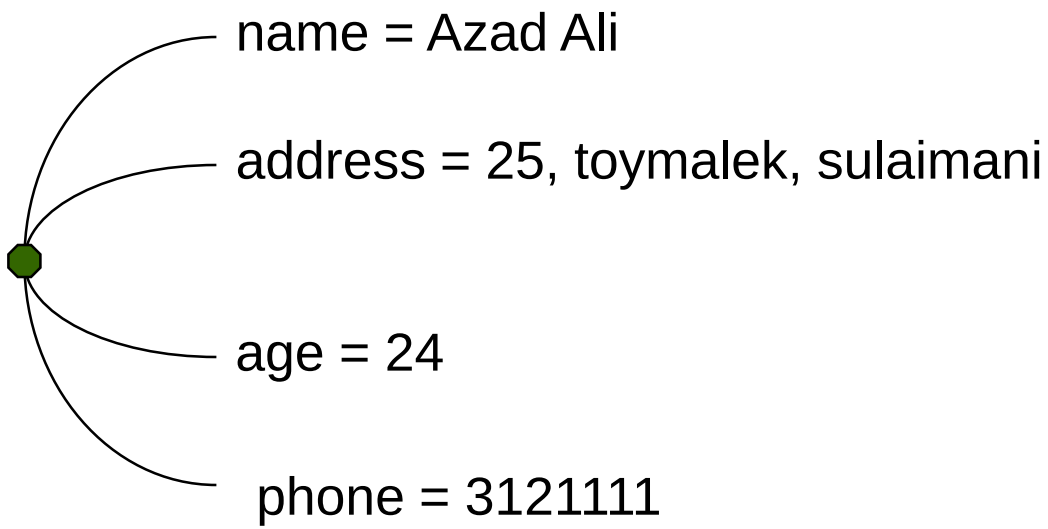
- **Entity-Relationship** (ER) model is a popular conceptual data model.
- This model is used in the **design** of database applications
- The model describes **data to be stored** and the **constraints** over the data.

Entities and attributes

- E-R model views the real world as a collection of **entities** and **relationships** among entities.
- An **entity** is an object in the real world that is distinguishable from other objects.
  - Examples
    - A classroom
    - A teacher
    - The address of the teacher

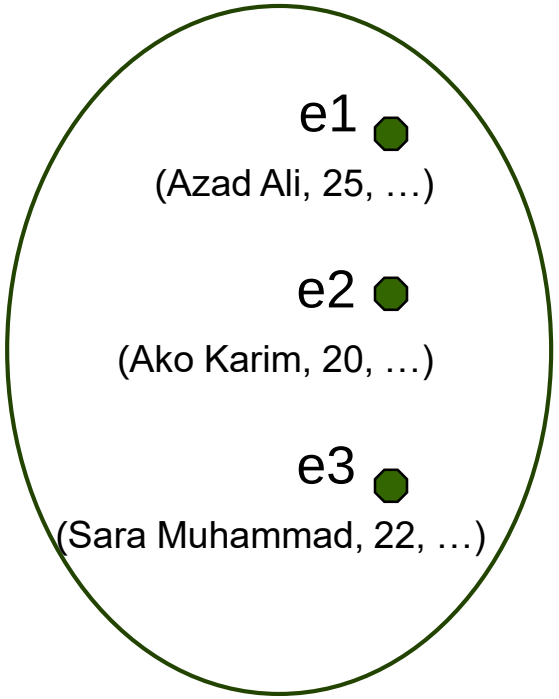
Entities and attributes

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



Entities and attributes

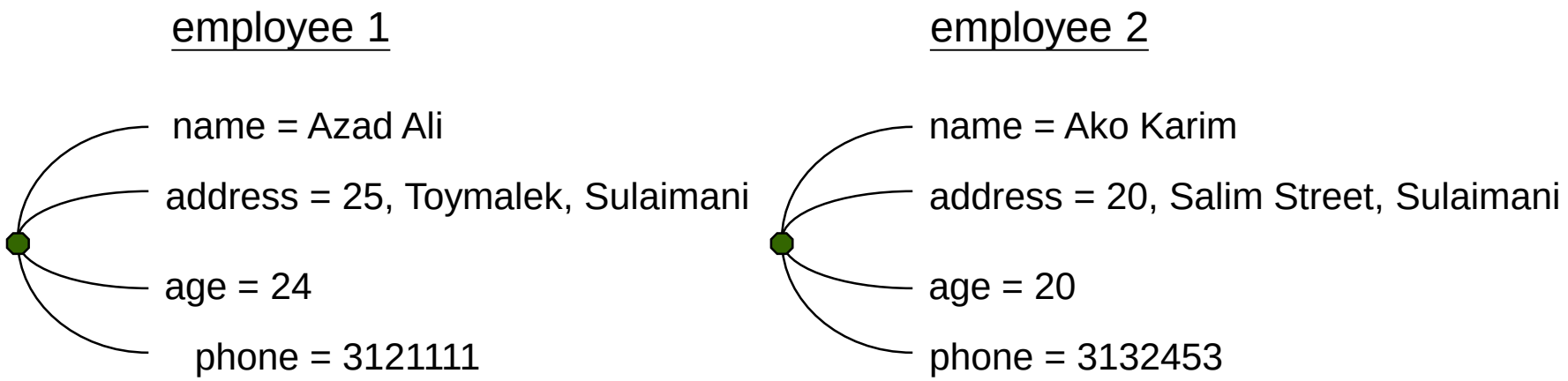
- An **entity set** is a collection of entities of the same type.



Entities and attributes

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

employee = (name, address, age, phone)



Entities and attributes

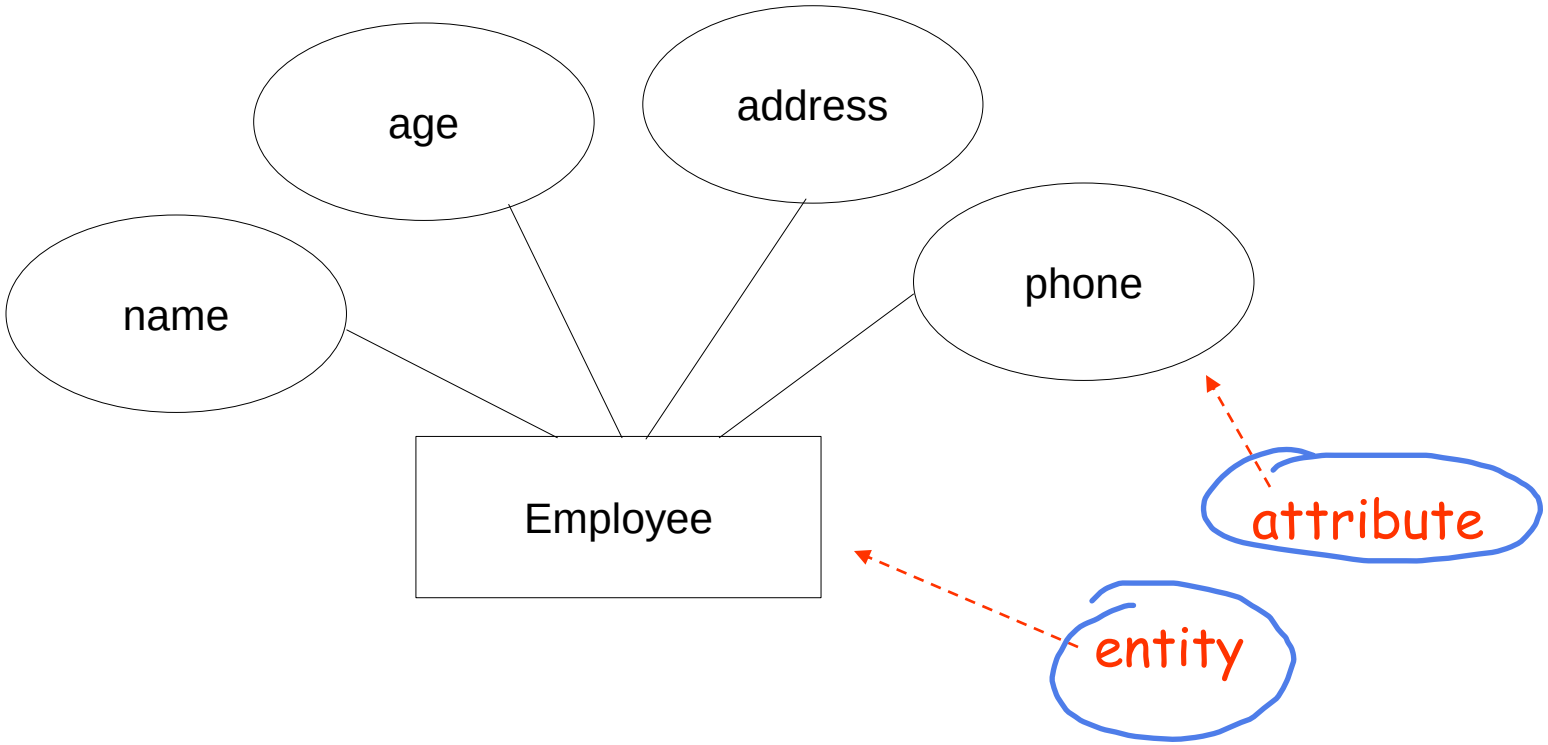
- For each attribute associated with an entity set, we identify a **domain** of possible values.

(data type)

  - **Example**
    - The domain associated with the attribute name might be the set of 20-character strings.
    - The domain associated with the attribute age might be an integer.

Entity-Relationship diagram (**E-R diagram**)

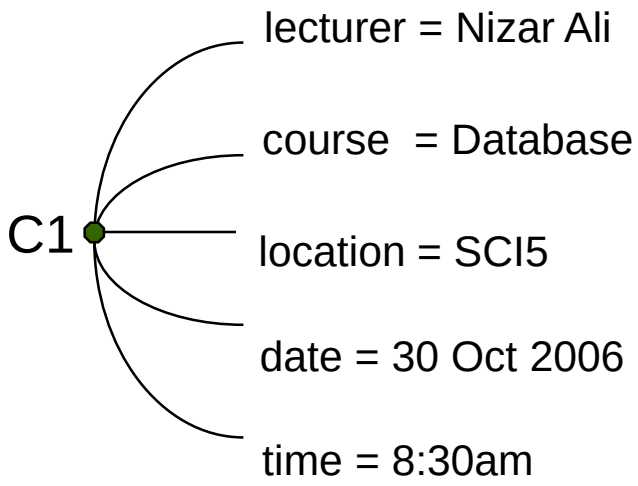
- The E-R model can be presented graphically by an E-R diagram.

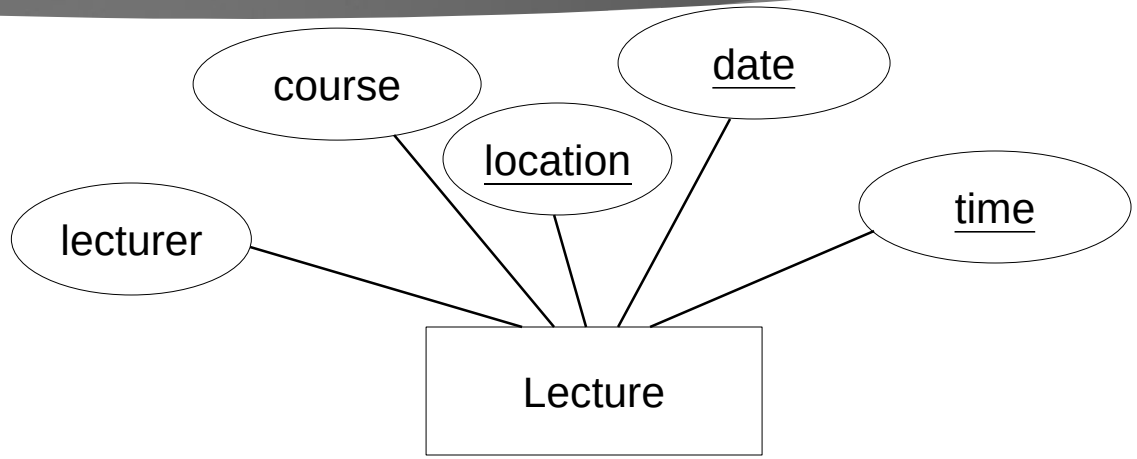


Key

- A key is a minimal set of attributes whose values uniquely identify an entity in the set.
  - E.g. Apartment (street, number, floor, flat, size, number_of_rooms, year)
- A key is also called a candidate key.
- There could be more than one candidate key.

Key





- **For example:**
- {location, date, time} is a key.
- {lecturer, date, time} is also a key.
- These are candidate keys, we can choose one to be the **primary key**.

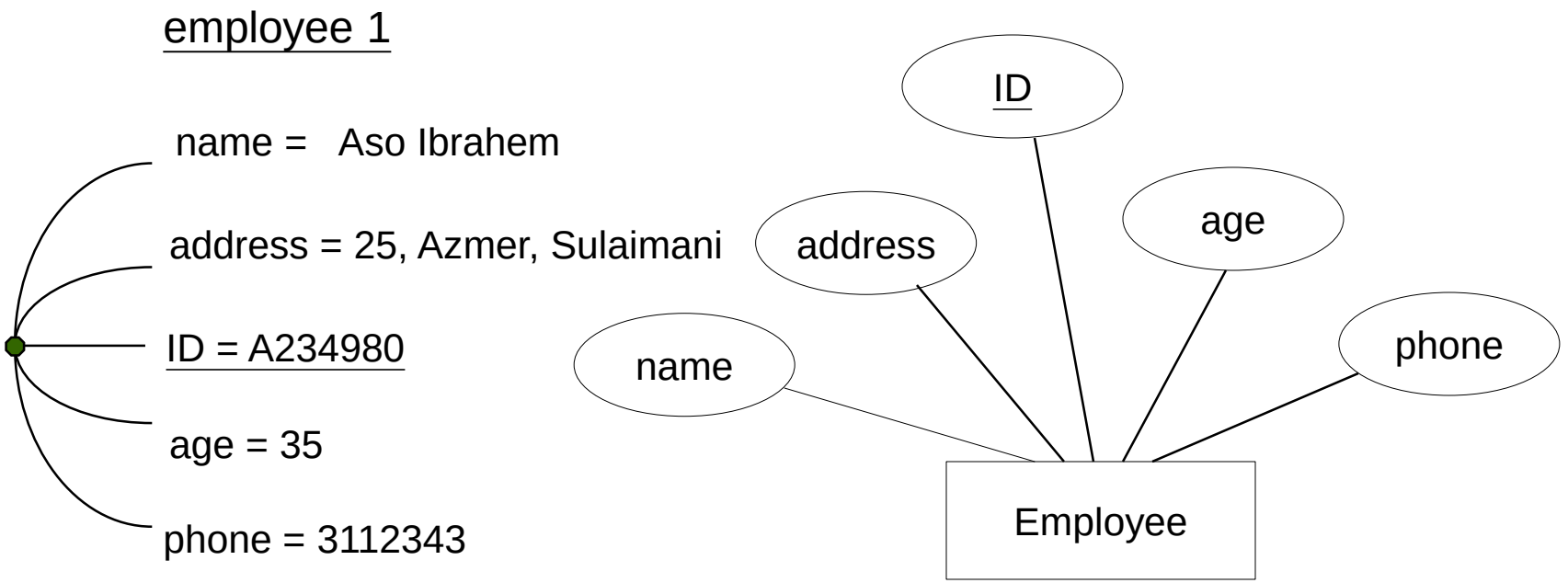
Key

- The key should depend on the real life possibility rather than on the current set of the data.

- **For example:**
- In a database which contains only two employees aged 30 and 40, the age may distinguish each employee.
- However, there can be in the future a new employee with the same age as an existing employee.

  Age is therefore **not** a key.

Key

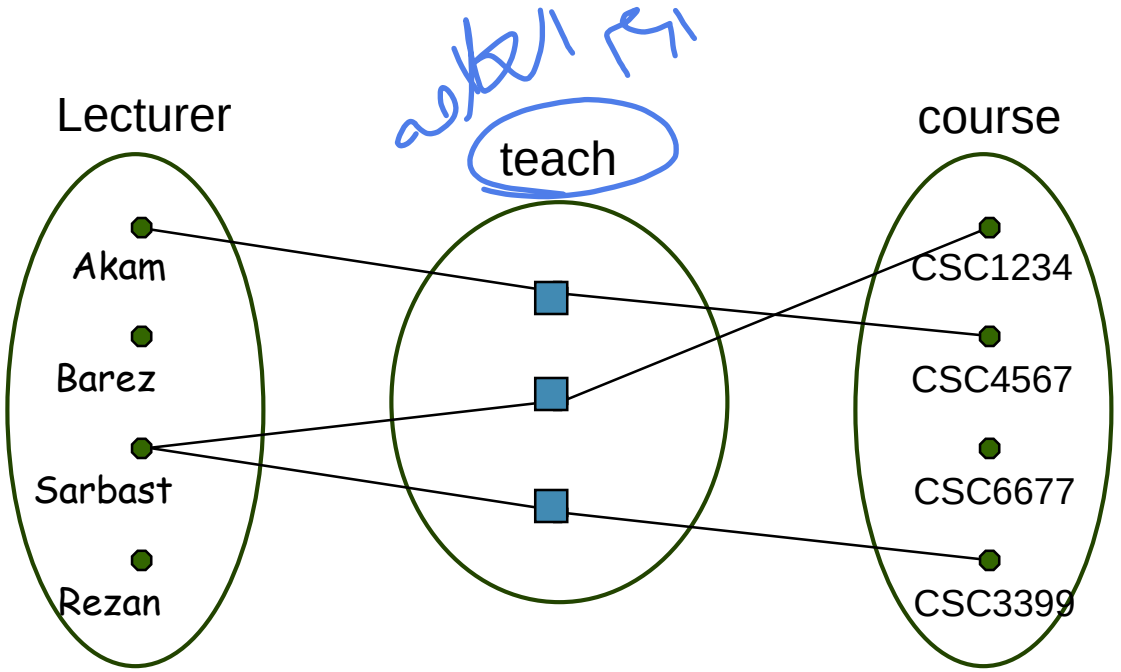


- For employee, if (name, address, age, phone) still cannot be a key, or we think it is too cumbersome, we can add an extra attribute as a key.

Relationships

- A relationship is an association among two or more entities.

Example: E1 = (Akam, Barez, Sarbast, Rezan) , *E2* = (CSC1234, CSC4567, CSC6677, CSC3399)



Relationships

- Relationship: teach

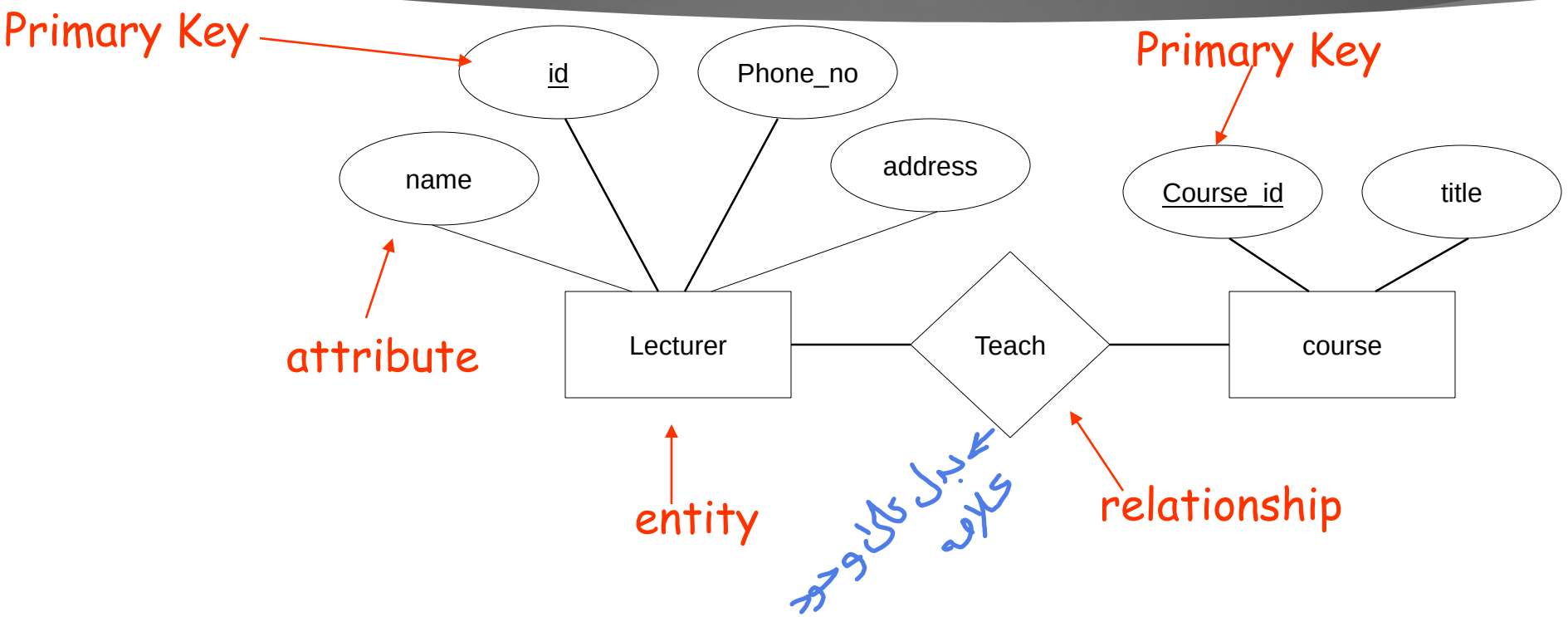
  (Akam,CSC4567), (Sarbast, CSC1234),(Sarbast,CSC3399)

$$\{(e_1,\cdots,e_n)\,|\,e_1 \in E_1,\cdots e_n \in E_n\}$$

- As with entities, we may wish to collect a set of similar relationships into a relationship set.
- A relationship set can be seen as a set of n-tuples:
- The teach relation can be represented by the set of ordered pairs:

  {(Akam,CSC4567), (Sarbast, CSC1234),(Sarbast,CSC3399)}

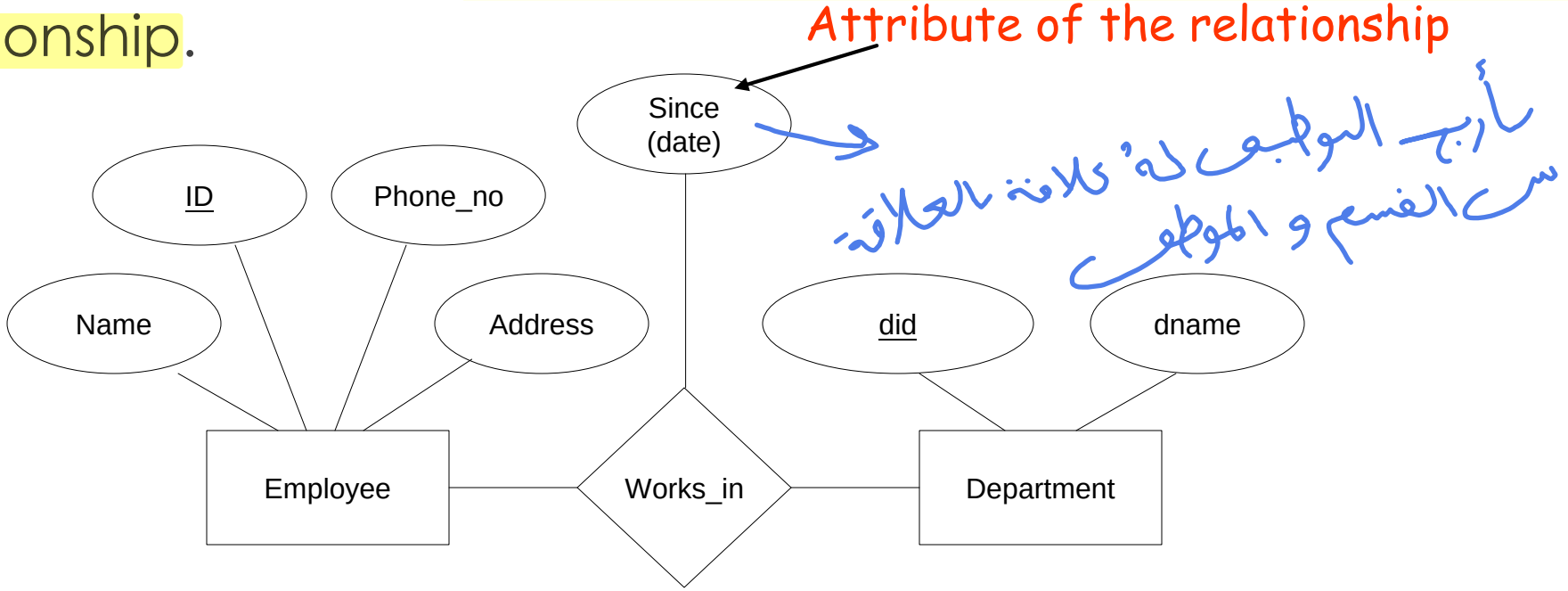
Relationships



- A relationship set can also be represented by an E-R diagram.

Relationships – Descriptive attribute

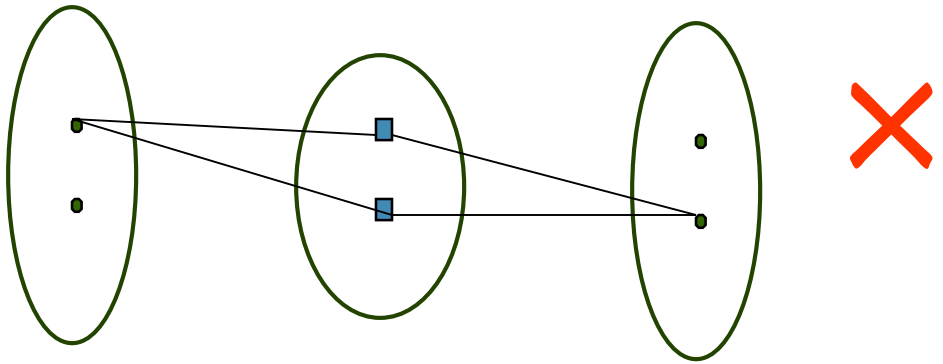
- A relationship can also have descriptive attributes.
- Descriptive attributes are used to record information about the relationship.



Relationships

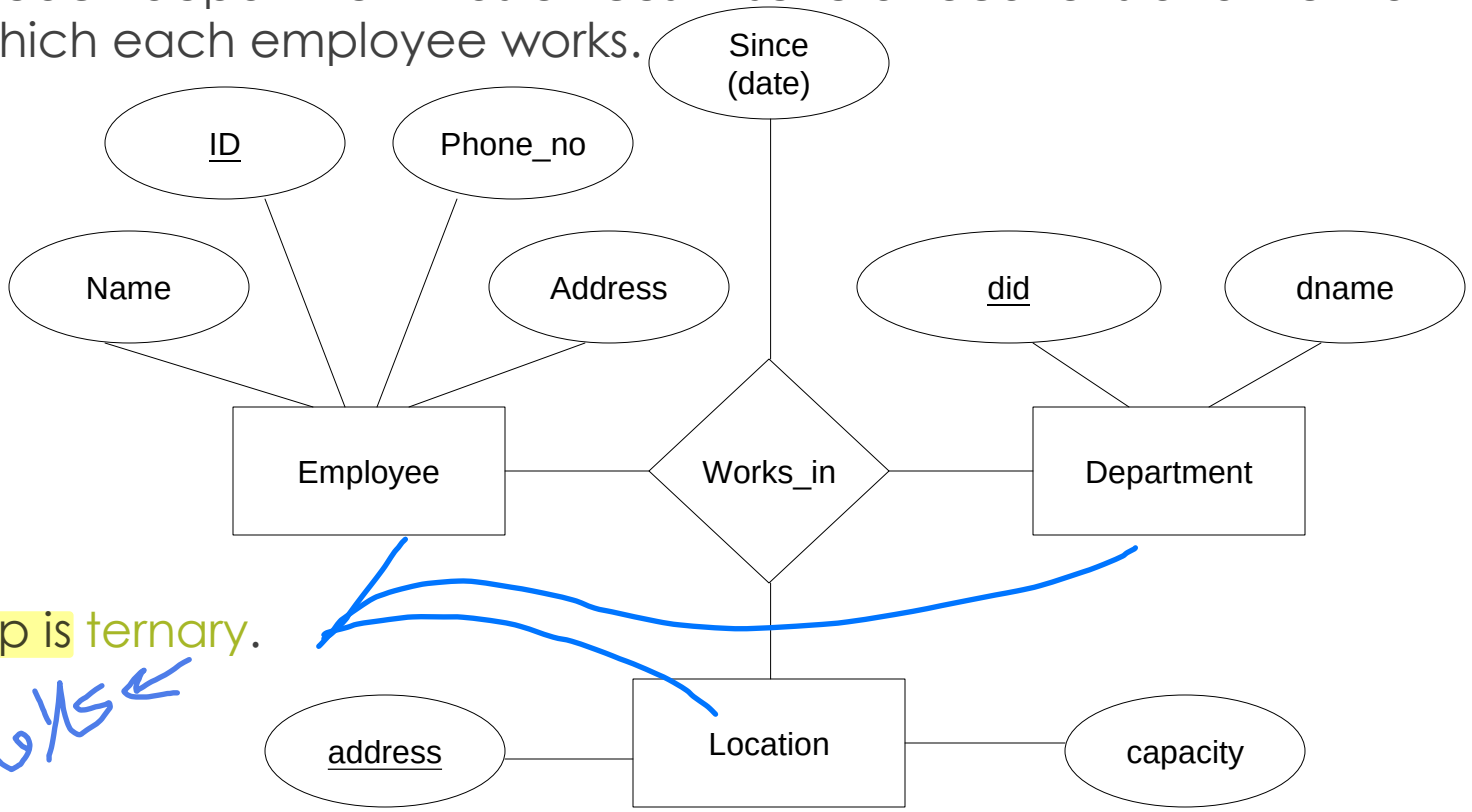
- A relationship must be uniquely identified by the participating entities, without reference to the descriptive attributes.
  - In the previous example, each relationship must be uniquely identified by the combination of the employee id and the department did.
- Thus, for each employee-department pair, we cannot have more than one associated "since" value.

(Azad-ID, Dept ABC, 1-2003) (Azad-ID, Dept ABC, 2-2004)



Ternary Relationships

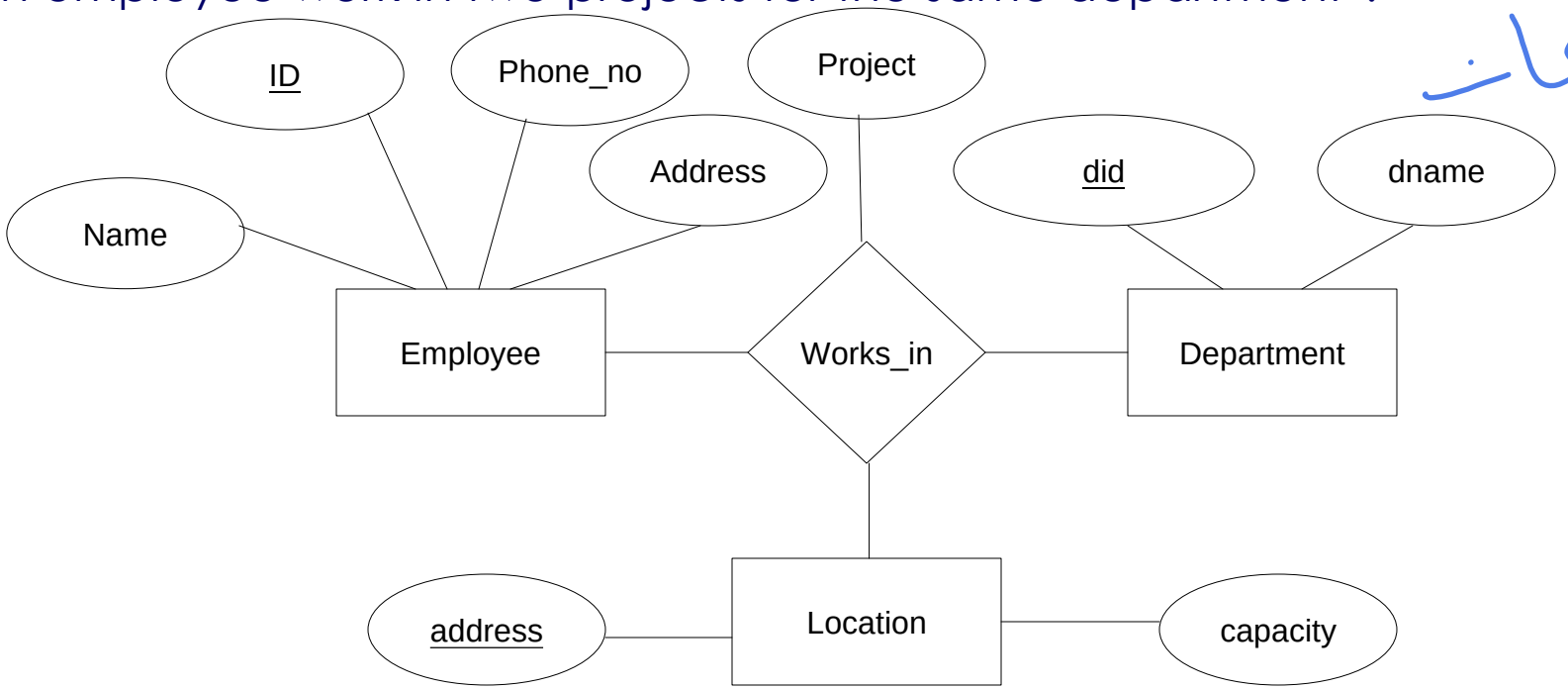
- Suppose now each department has offices in several locations and we want to record the locations at which each employee works.



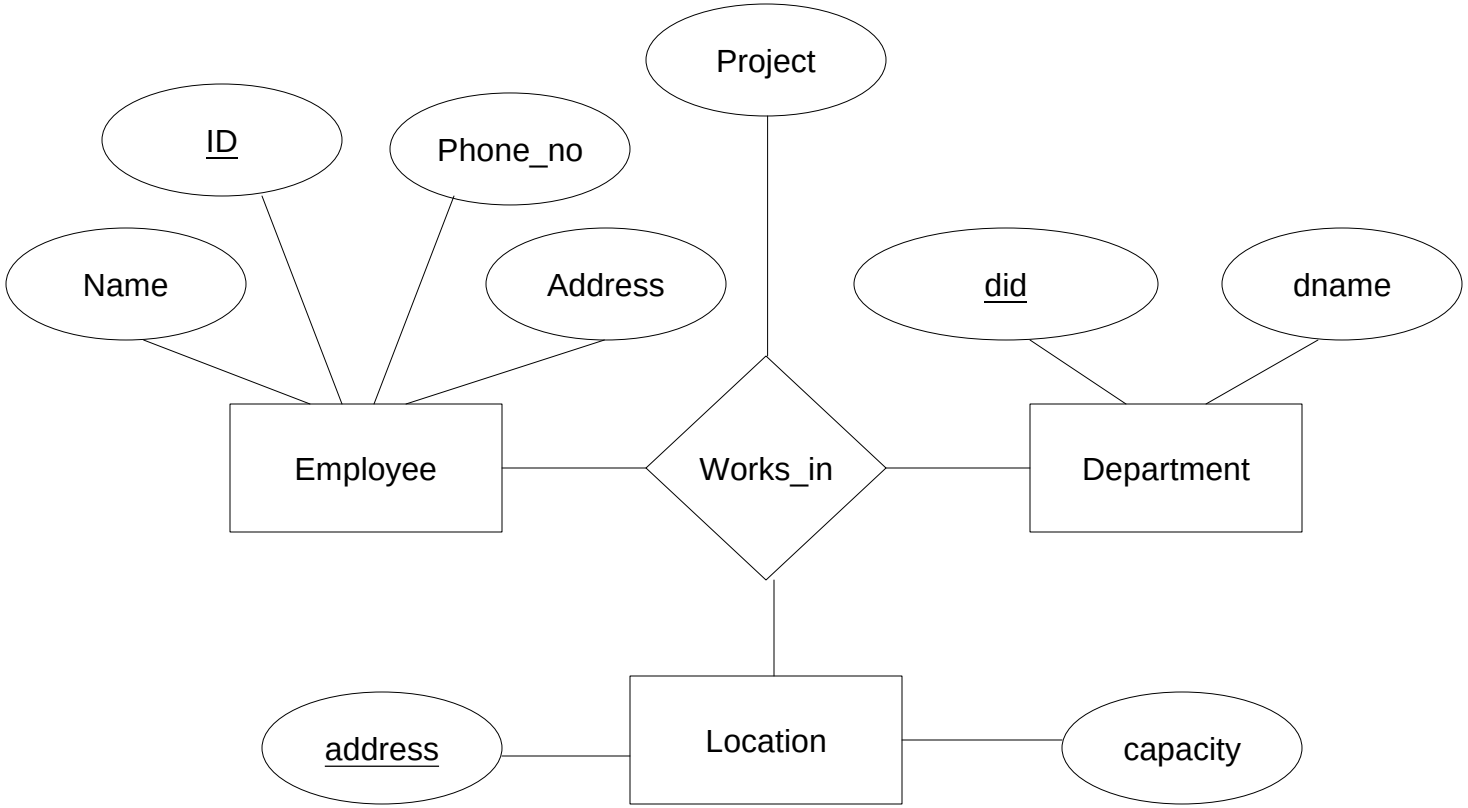
- This relationship is ternary.

Ternary Relationships

1. Can an employee work in two locations for the same department ?
2. Can an employee work in two projects for the same department ?



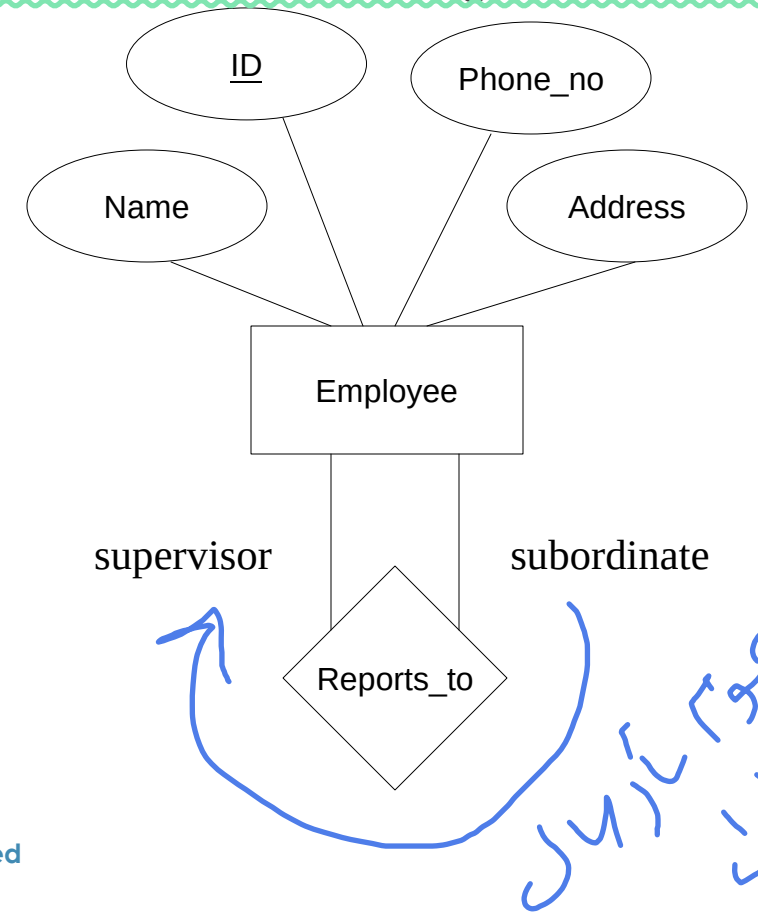
هذا يعني علاقة
من نوع العلاقات
many to many
⋮

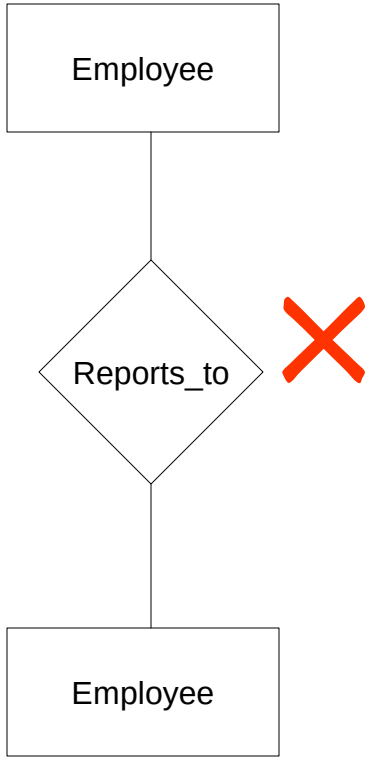


- (emp1, dept1, location1, proj1)
- (emp1, dept1, location1, proj2) ✗
- (emp1, dept1, location2, proj2)
- (emp1, dept2, location1, proj1)

Recursive Relationship

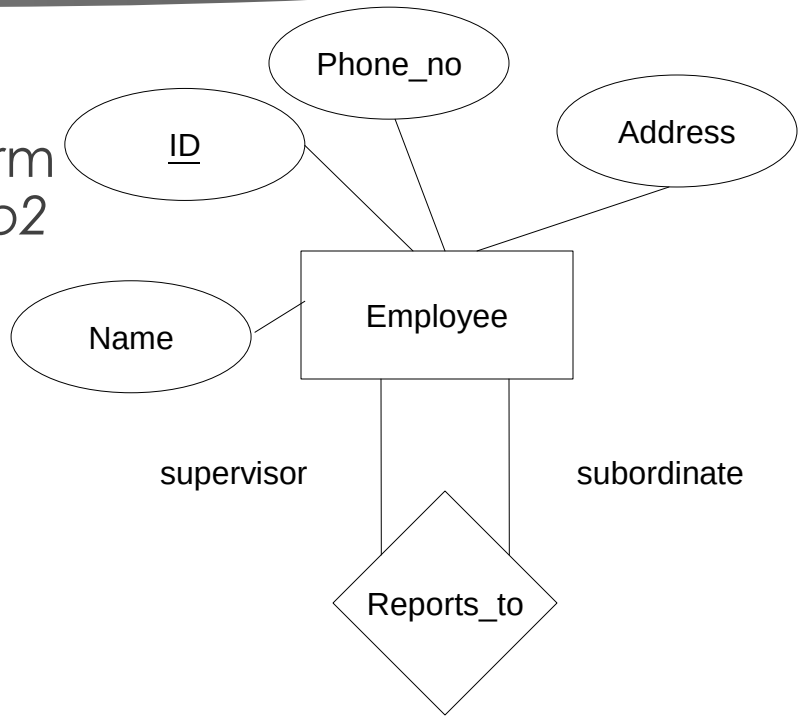
- Sometimes a relationship might involve two entities in the same entity set.





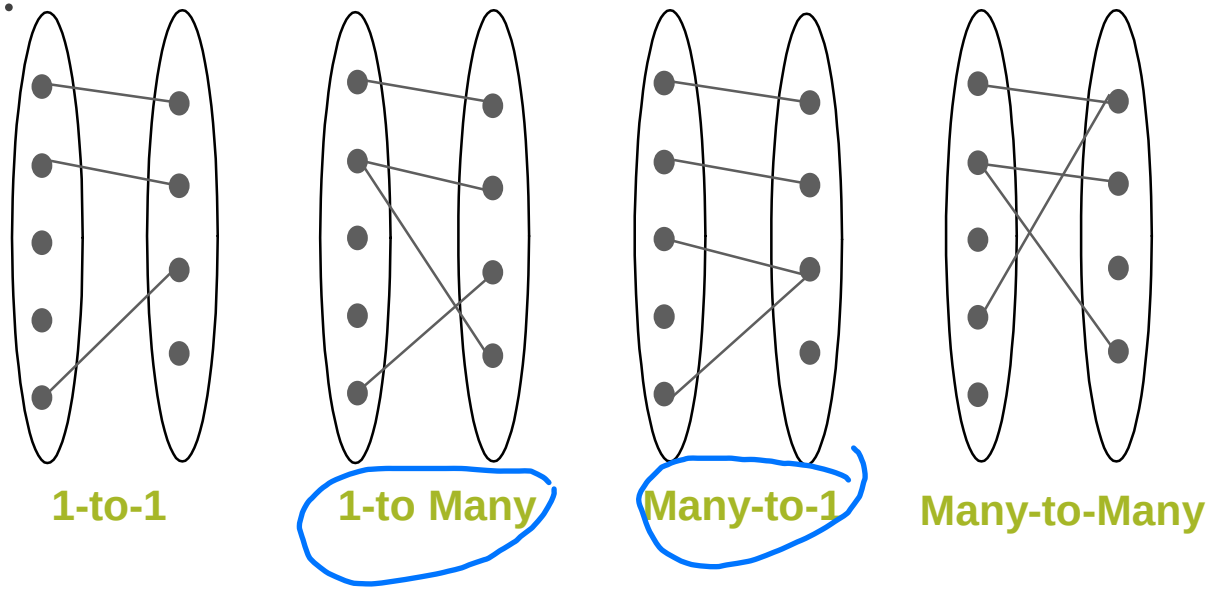
Recursive Relationship

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



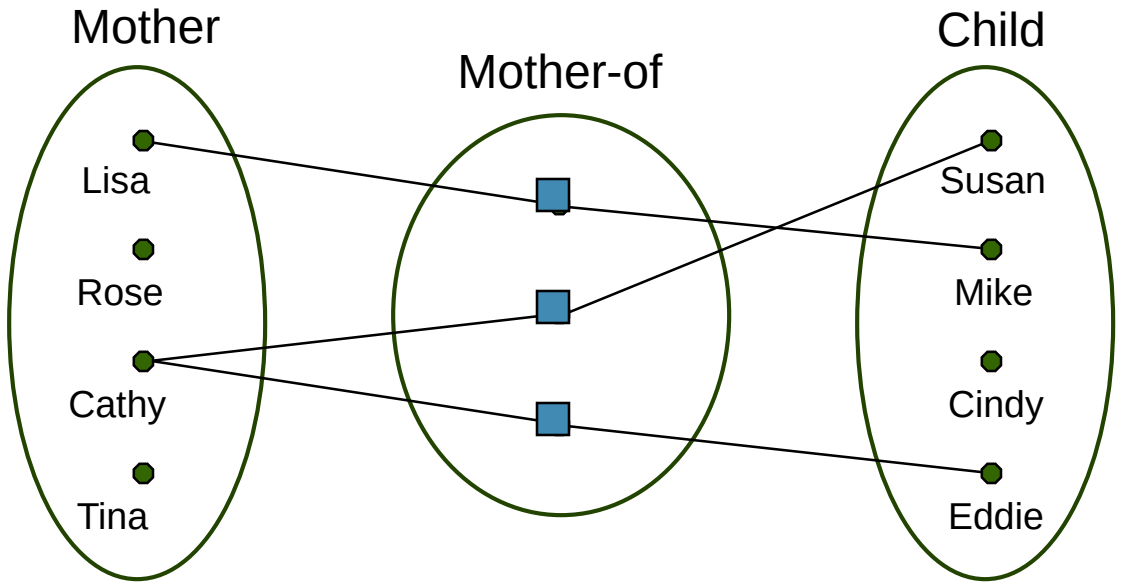
Key constraints (mapping constraints)

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



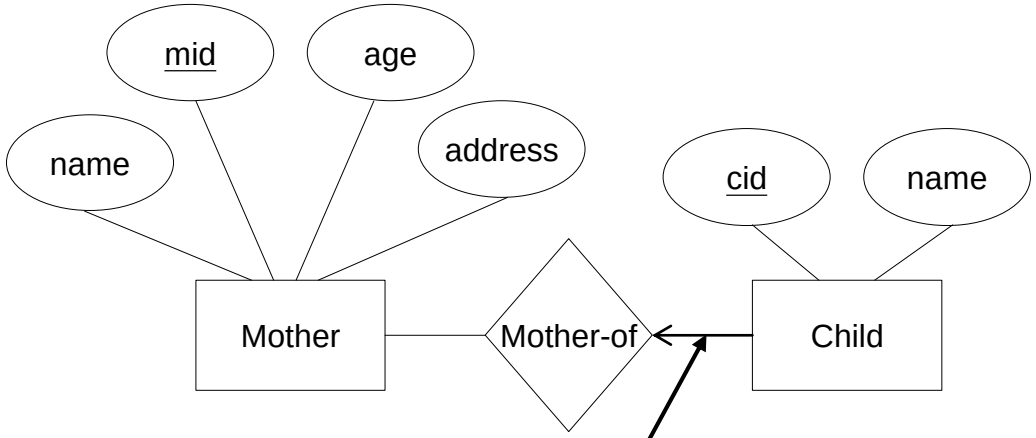
One-to-many

- One-to-many constraint from A to B:
- An entity in **B** can be associated with at most one entity in **A**.
- Each child can appear in at most one mother-child relationship.



One-to-many

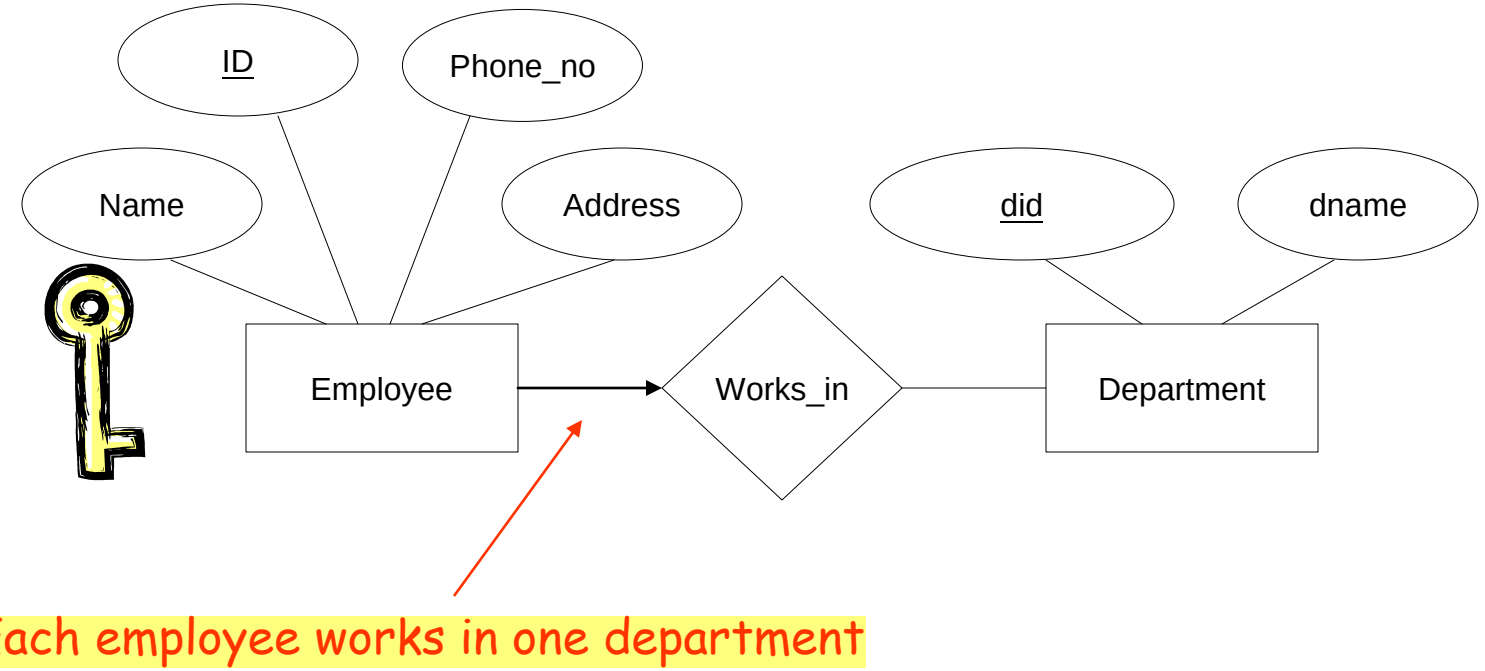
- **Child** has a key constraint in the mother-of relationship set.
- This restriction can be indicated by an **arrow** in the E-R diagram.



Intuitively, the arrow states that given a child entity, we can uniquely determine the mother-of relationship.

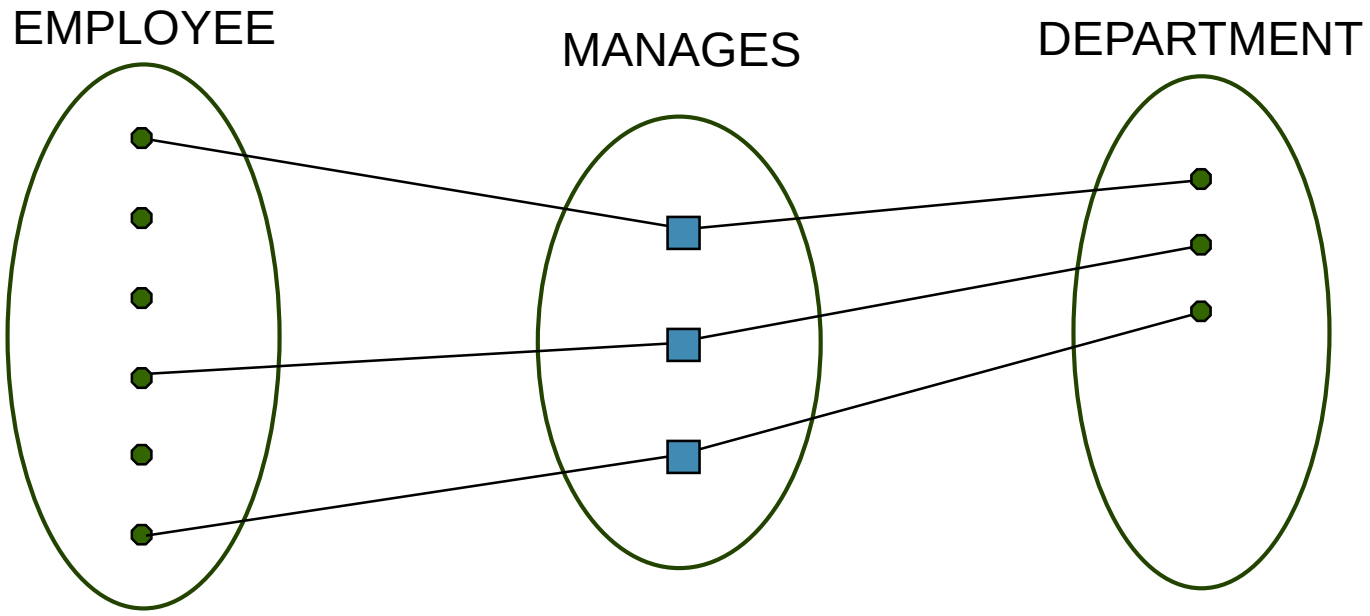
Many-to-one

- Similar to one-to-many

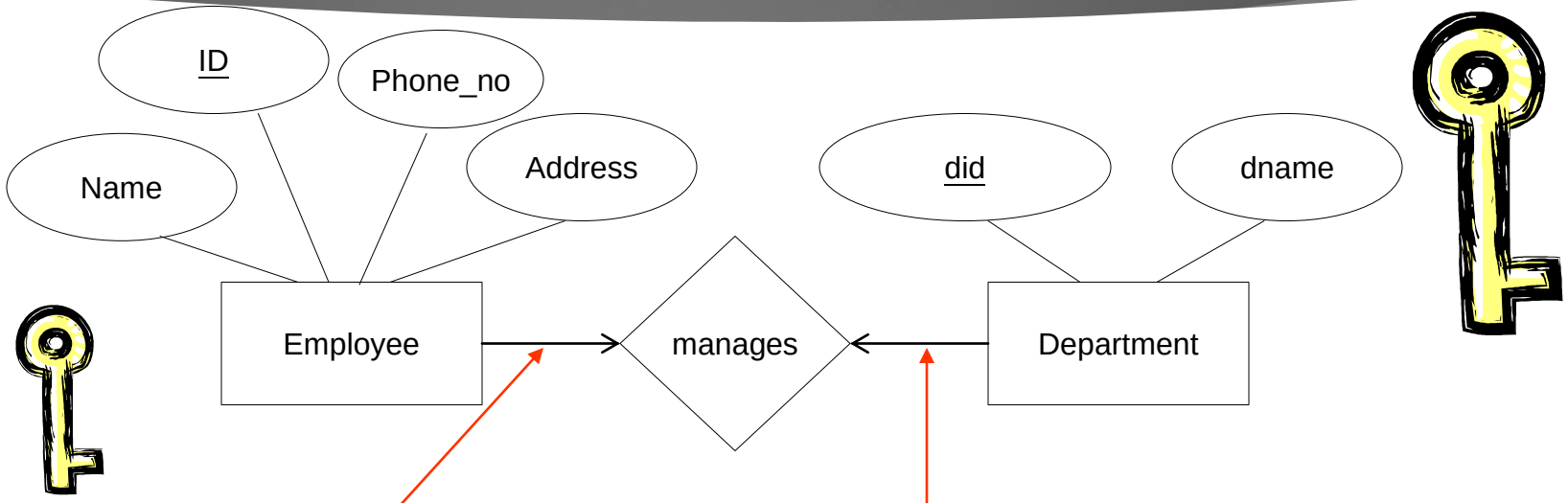


One-to-one

- If the relationship between **A** and **B** satisfies the **one-to-one** mapping constraint from **A** to **B**, then
- an entity in **A** is related to **at most one** entity in **B**, and
- an entity in **B** is related to **at most one** entity in **A**.



One-to-one

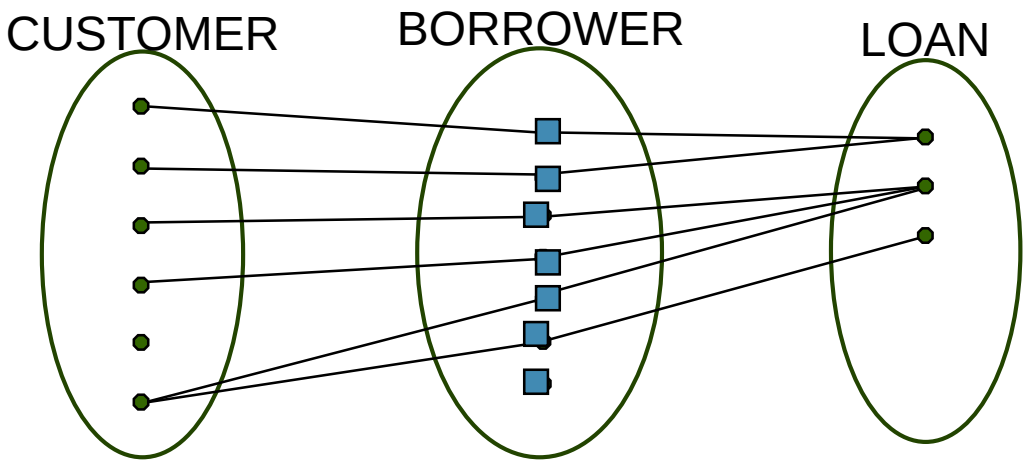


An employee can associate with at most one department via the Relationship "manages".

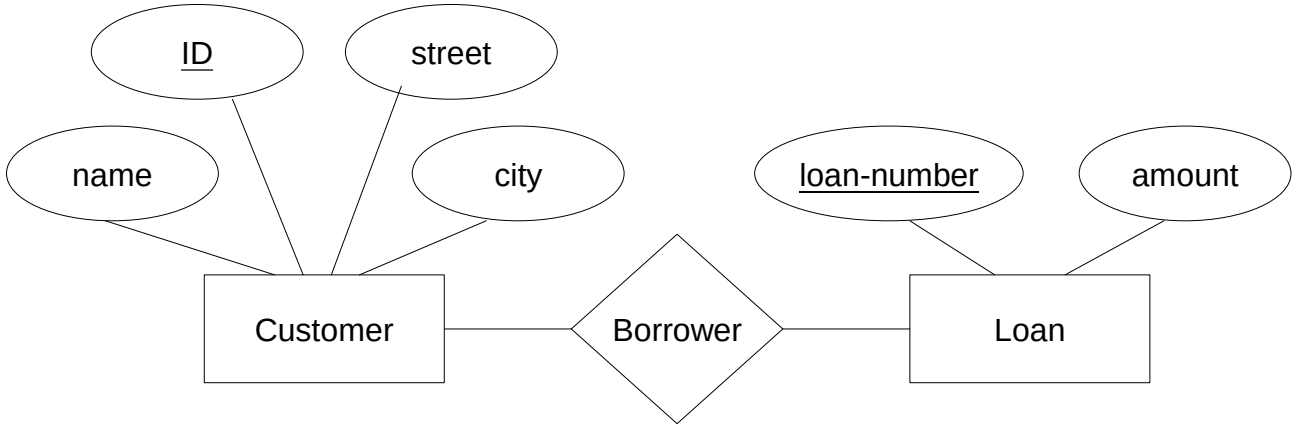
A department can associate with at most one employee via the relationship "manages"

Many-to-many

- An entity in **A** is associated with **any number** of entities in **B**, and an entity in **B** is associated with **any number** of entities in **A**.
- In fact, it means that there is no restriction in the mapping



Many-to-many

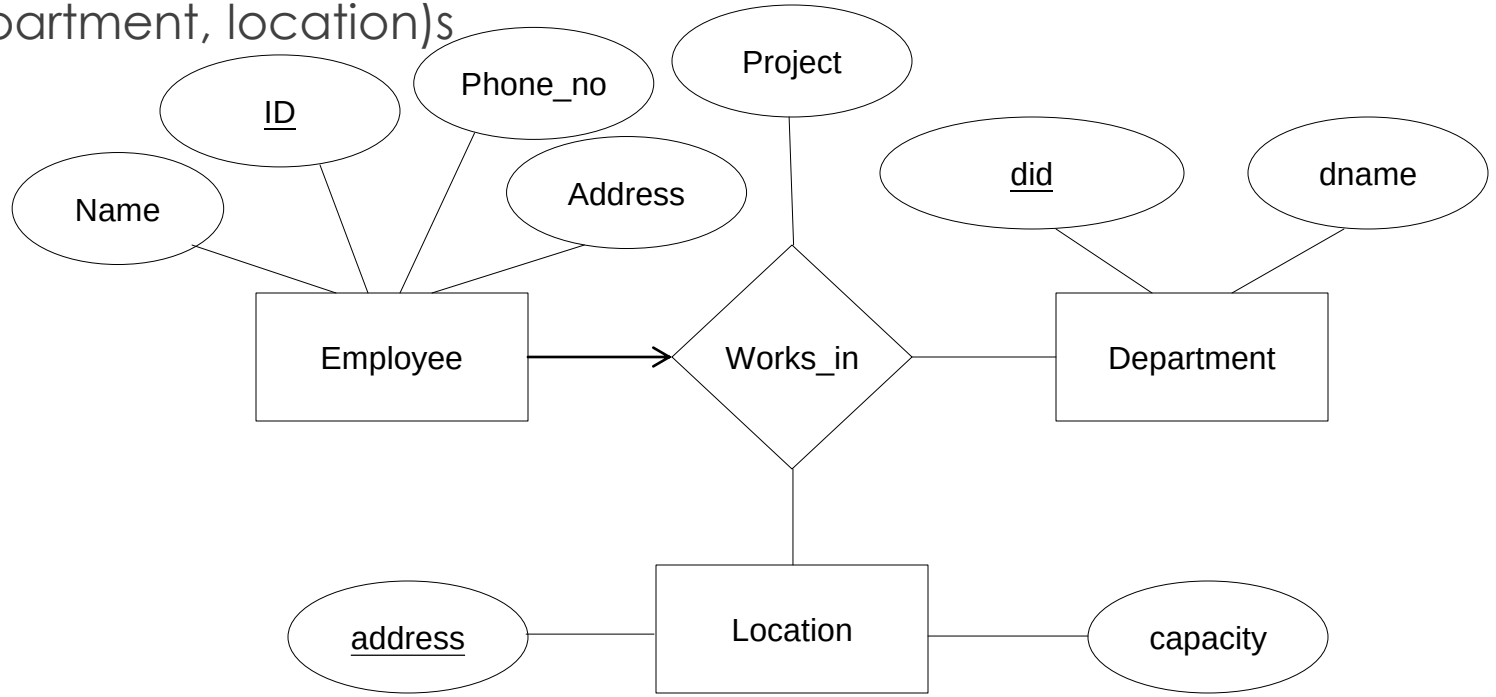


- A customer can associate with **several** loans (possibly 0) via Borrower
- A loan can associate with **several** customers (possibly 0) via Borrower

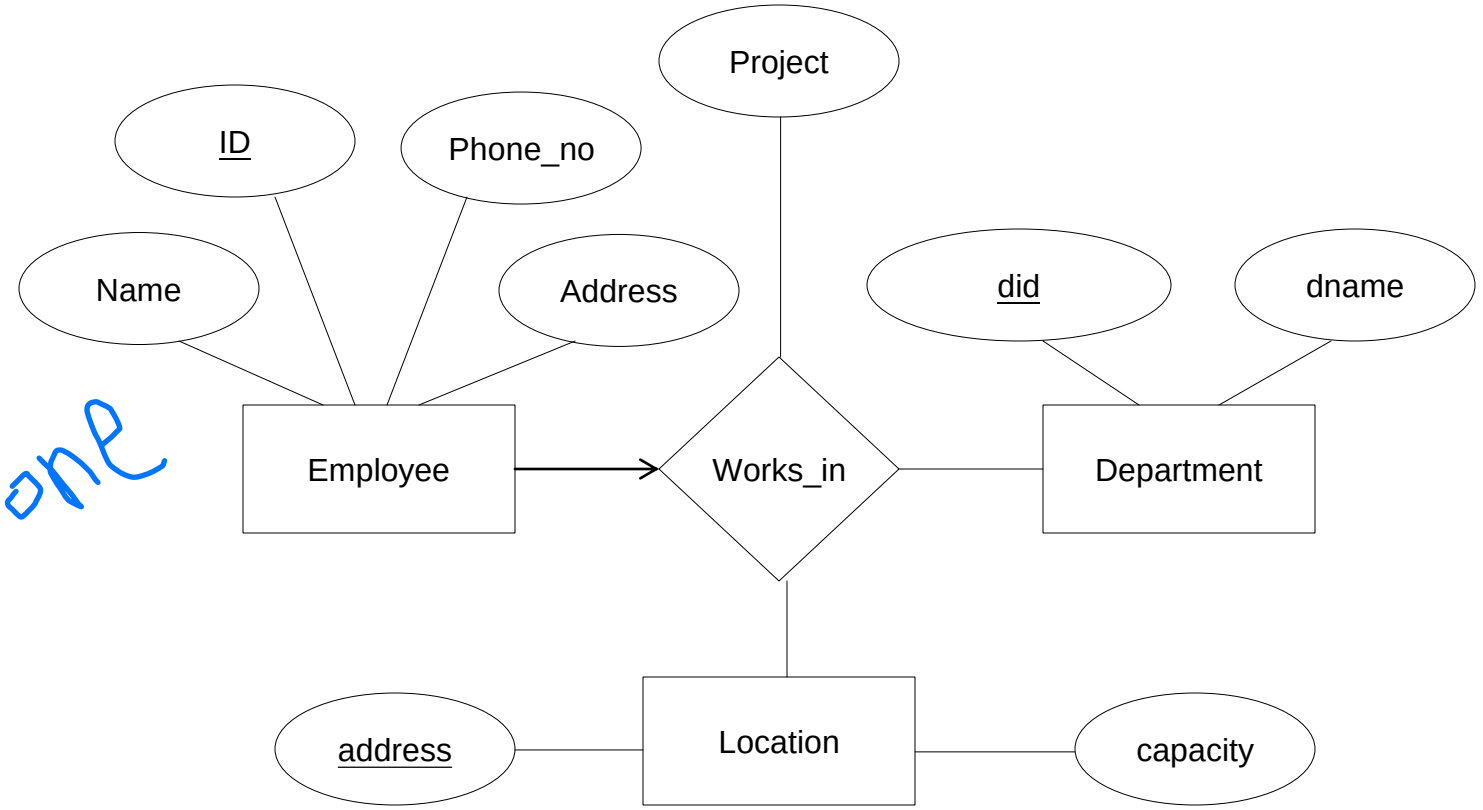
Ternary Relationship Constraint

The key constraint is not limited to binary relationships.

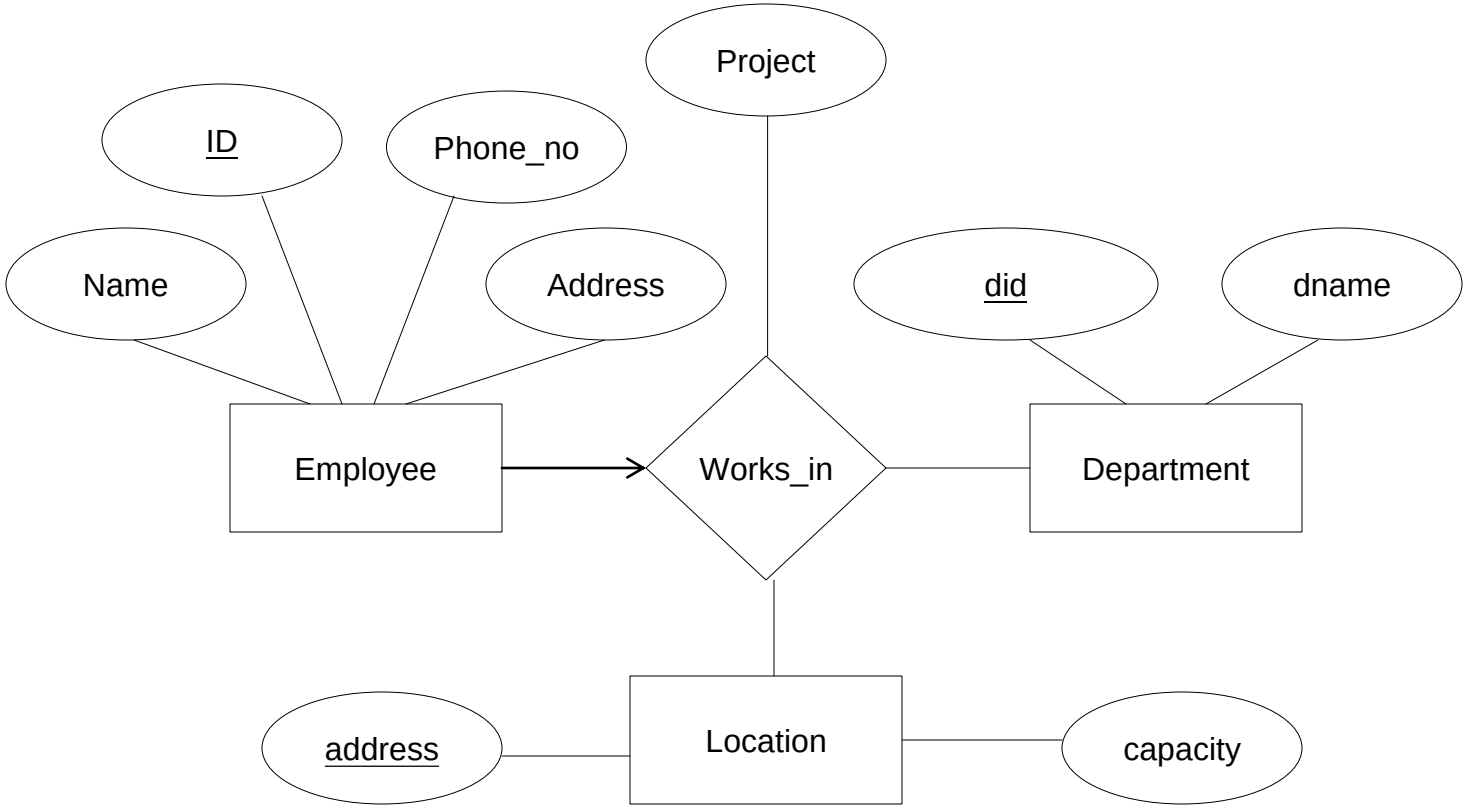
e.g. an employee works in at most one department at one location. **employee** is a key in (employee, department, location)s



- Can an employee work in two locations for the same department ? No
- Can she work in two locations? (No)
- Can an employee work in two projects for the same department ? (NO)
- Can he work in two projects?



many to one



- Given (emp1, dept1, location1, proj1)
- (emp1, dept1, location2, proj1) ×
- (emp1, dept2, location1, proj1) ×
- (emp1, dept1, location1, proj2) ×

Keys for a relationship set

The concept of keys is also used to identify a relationship, as in entities.

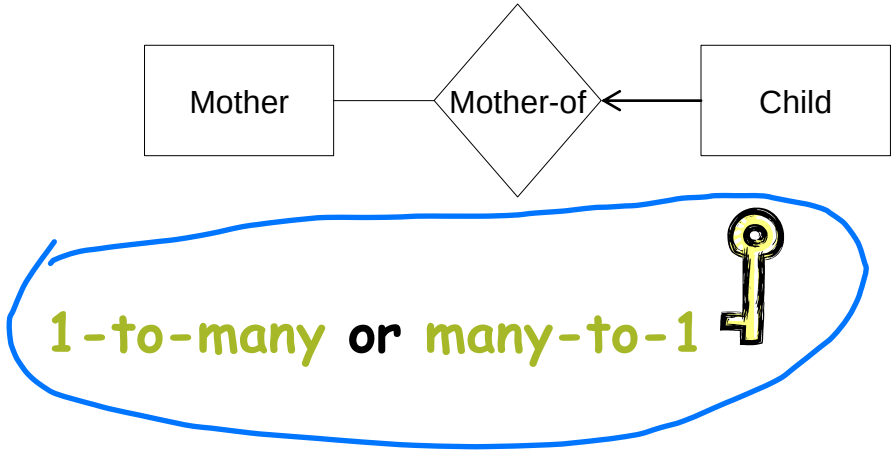
Key of relationship set **R** = a set of attributes that can uniquely identify a relationship in R.

Key constraints

- An entity set E has a key constraint in a relationship set R
- The key of E can be used as the key in R.

  e.g. child-mother: is a many-to-one relationship,

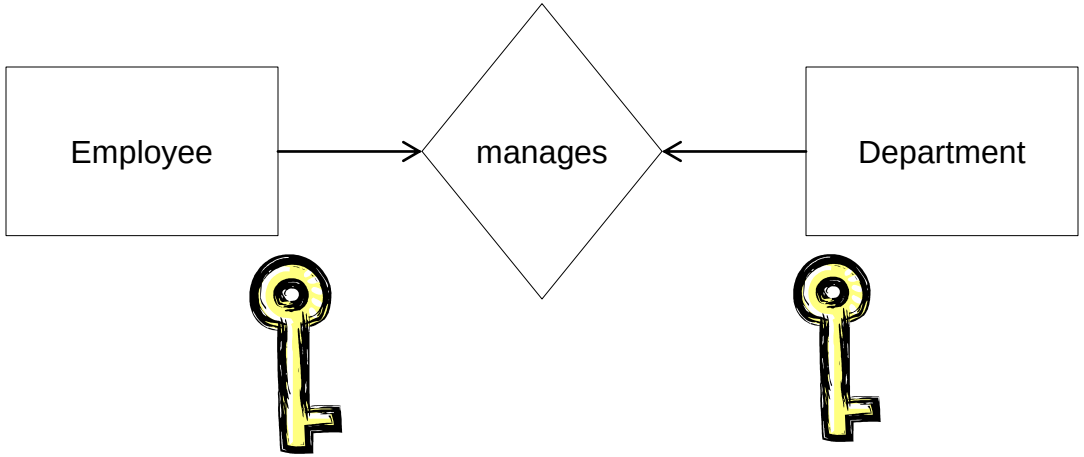
  entity Child has a key constraint.
  - Child can be the primary key in the relationship set.



One-to-one

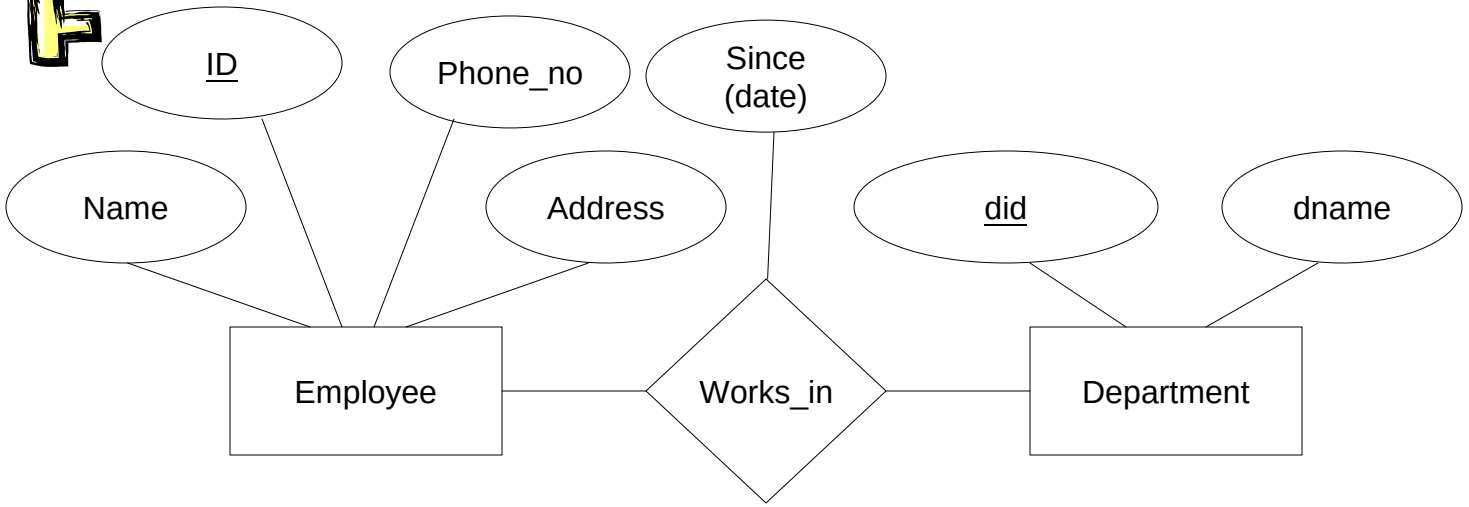
- For an one-to-one relationship between two entity sets E and F, key(E) and key(F) are both keys for the relationship set.

  E.g. "manages" relation



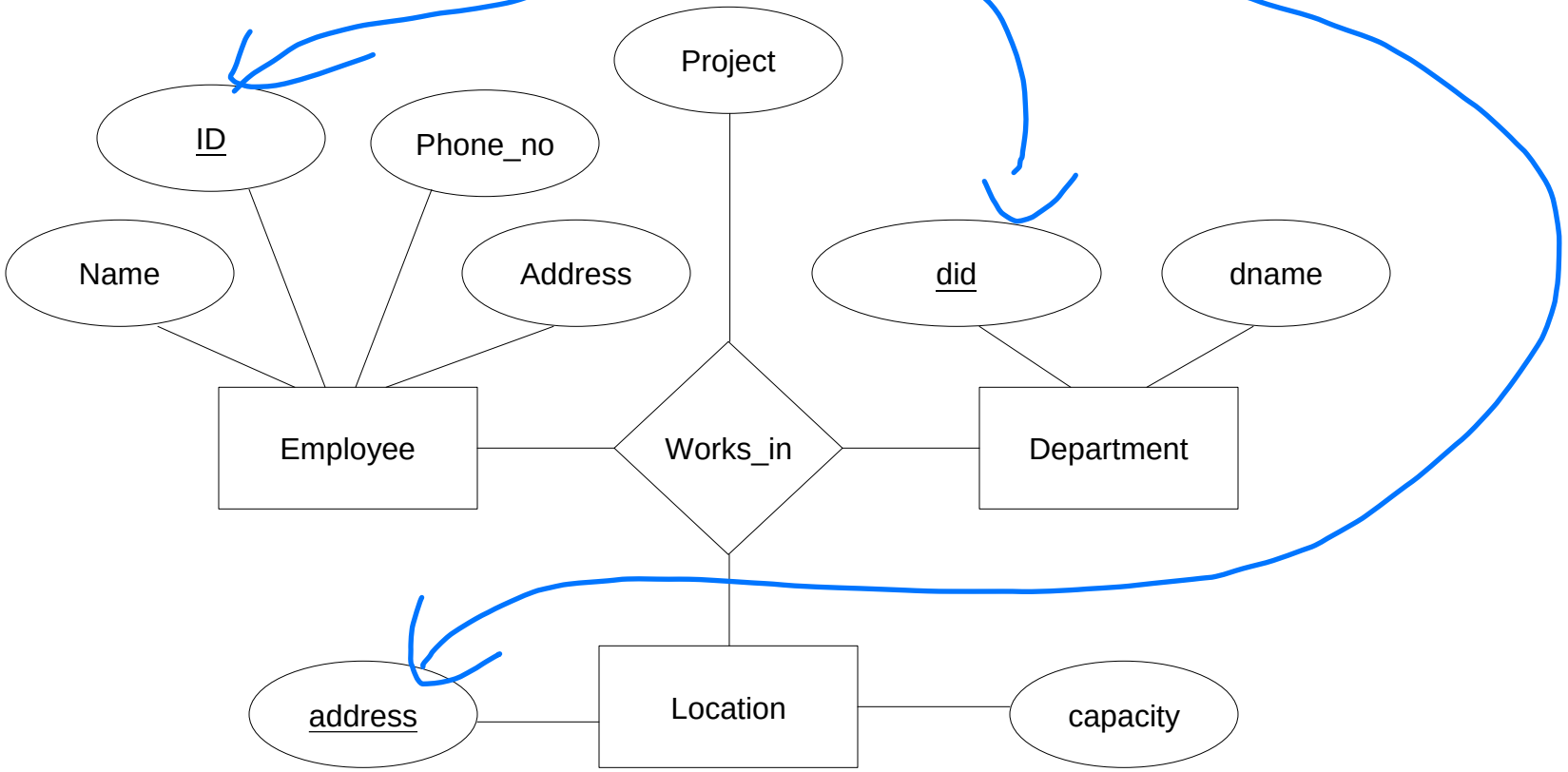
Many-to-many

Key for Works_in = (employee_id, department_id)

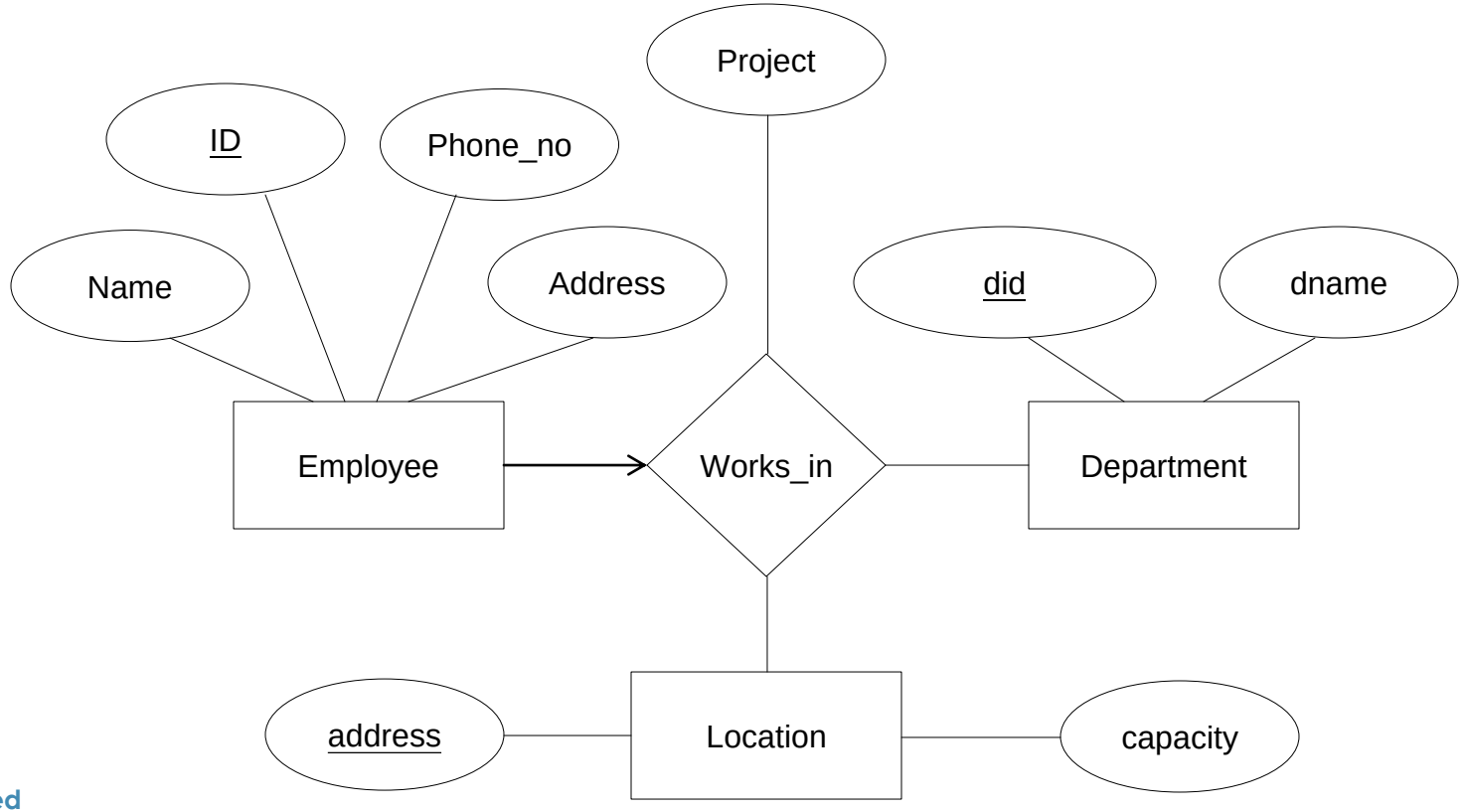


For a relationship among **E1**, …, **Ek**, with no mapping constraint, the primary key is normally the union of the primary keys for E1, …, Ek.

What is the key for this relationship ?



What is the key for this relationship ?



Participation constraints

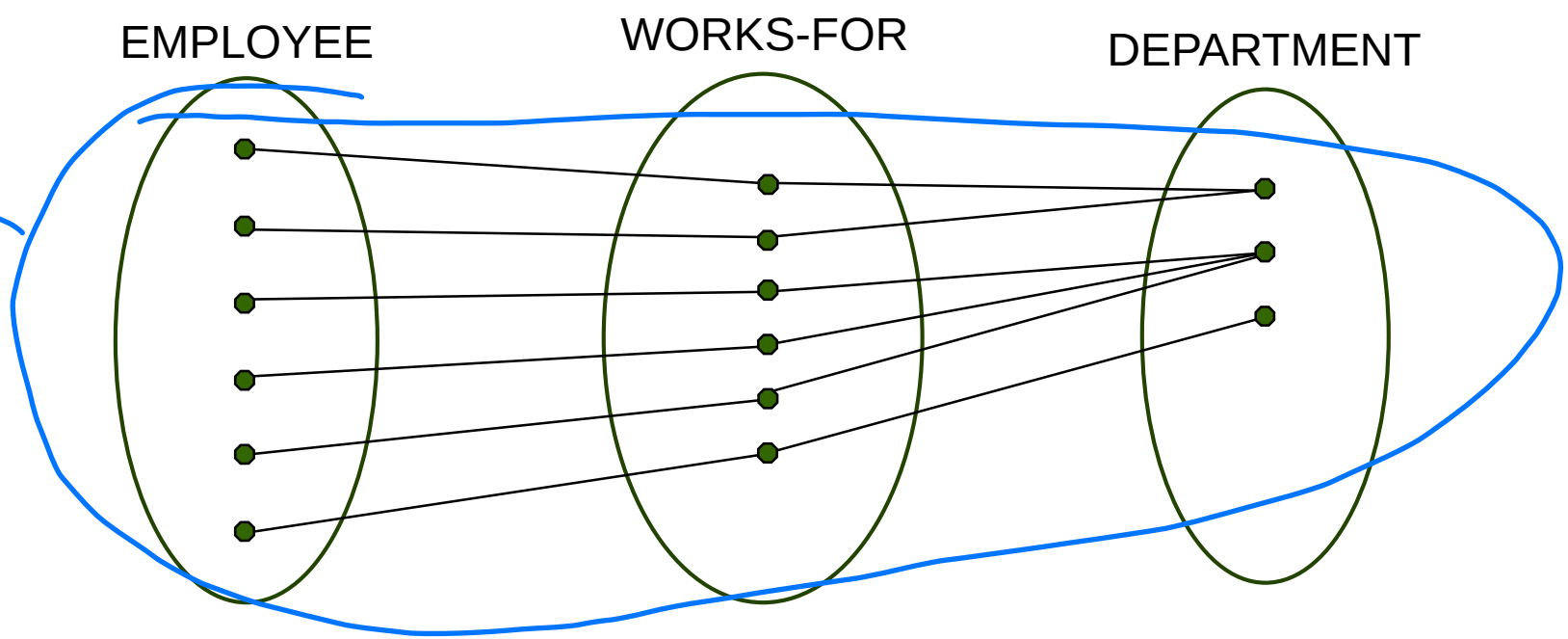
- Suppose every department is required to have a manager.
- Such a constraint is an example of participation constraint.

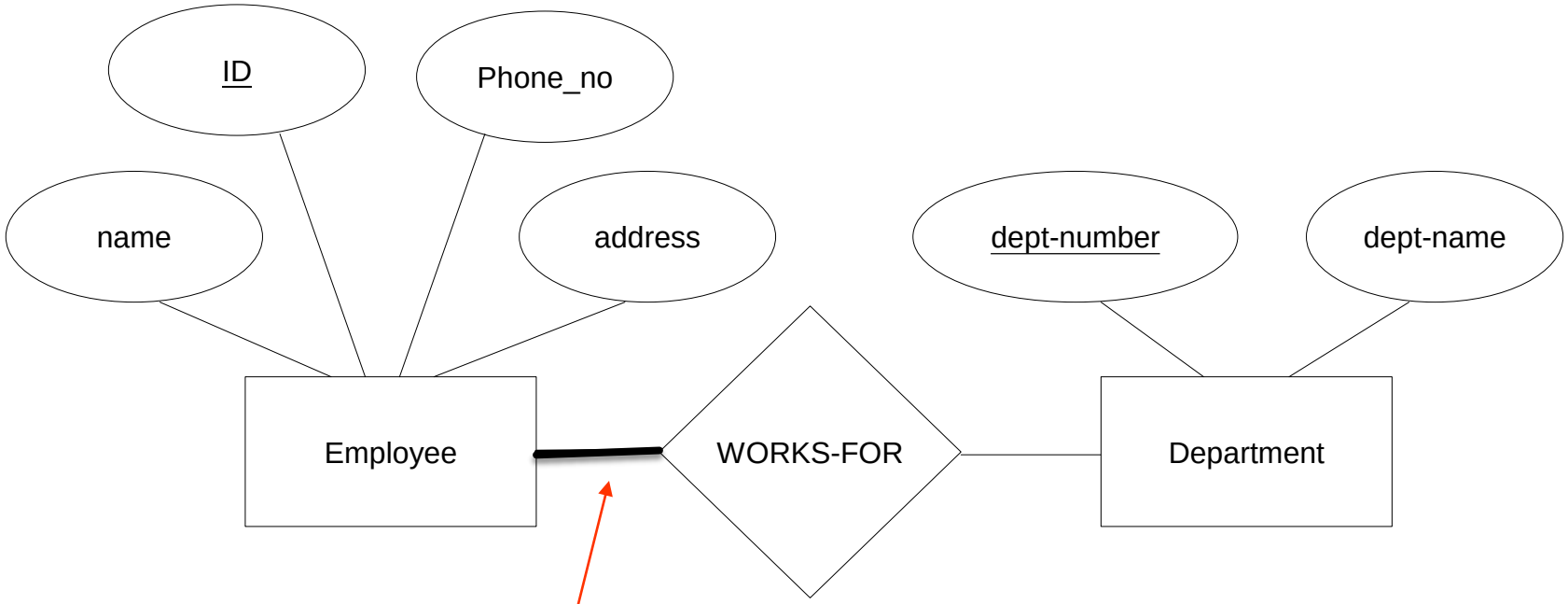
Participation constraints

- A participation constraint imposes some requirements on whether an entity participates in a relationship.
  - **Total participation** - each entity in the entity set must participate in at least one relationship.
  - **Partial participation** - an entity in the entity set may not participate in a relationship.

Participation constraints

- **Every** employee must work for some department.
- The participation of EMPLOYEE in WORKS-FOR is **total** participation.

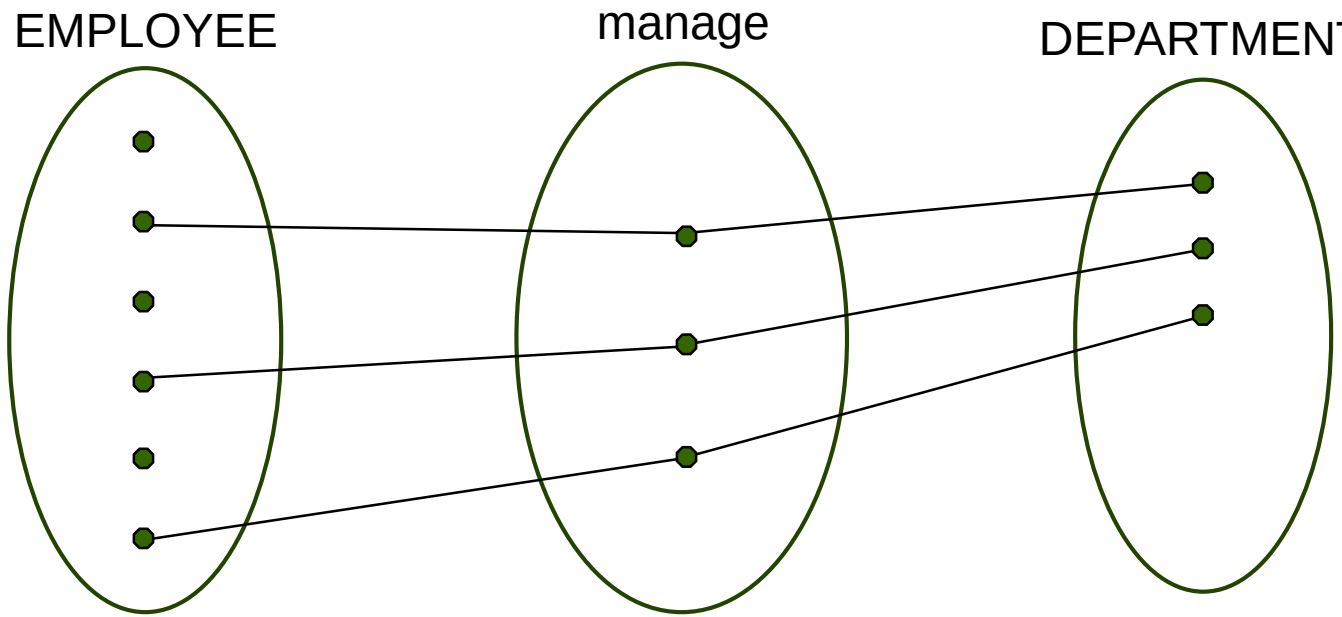


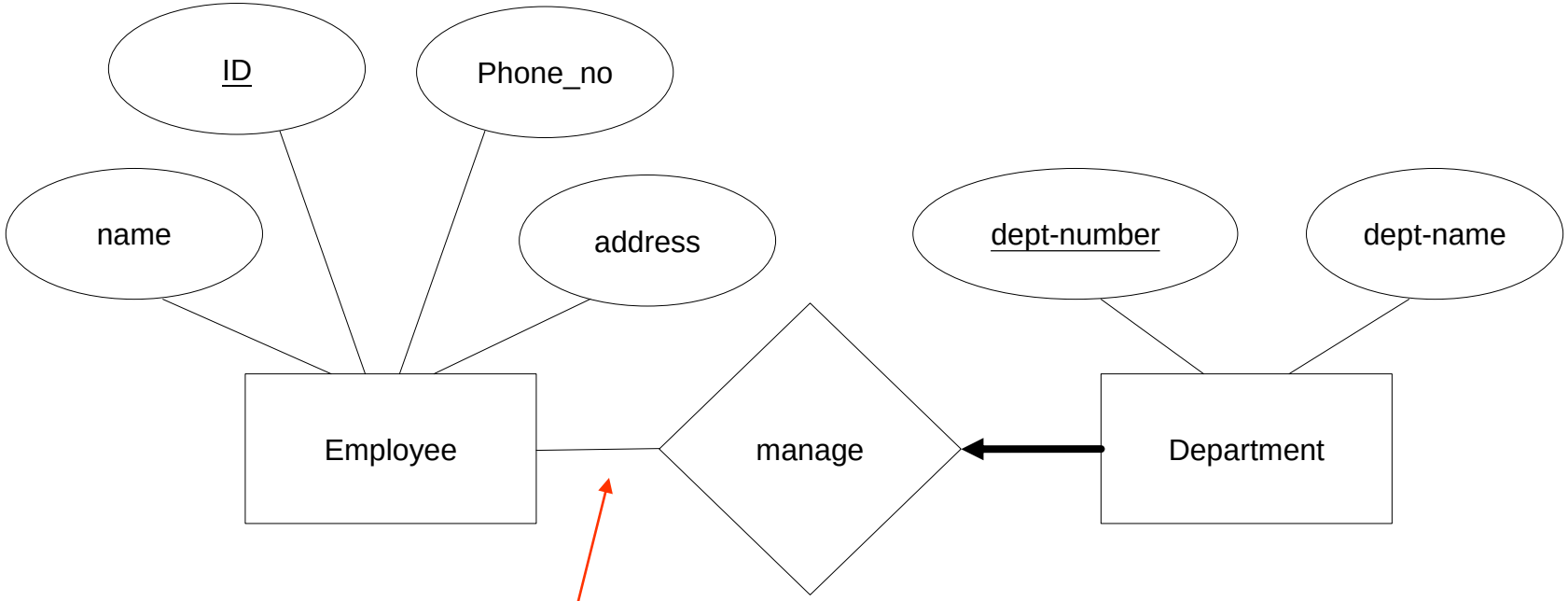


If the participation of an entity set in a relationship set is **total**, the two are connected by a **thick** link; independently, the presence of an arrow indicates a key constraint.

Participation constraints

- **Some** employee must manage departments.
- **Each** department must be managed by an employee.
- The participation of EMPLOYEE in manage is **partial** participation.





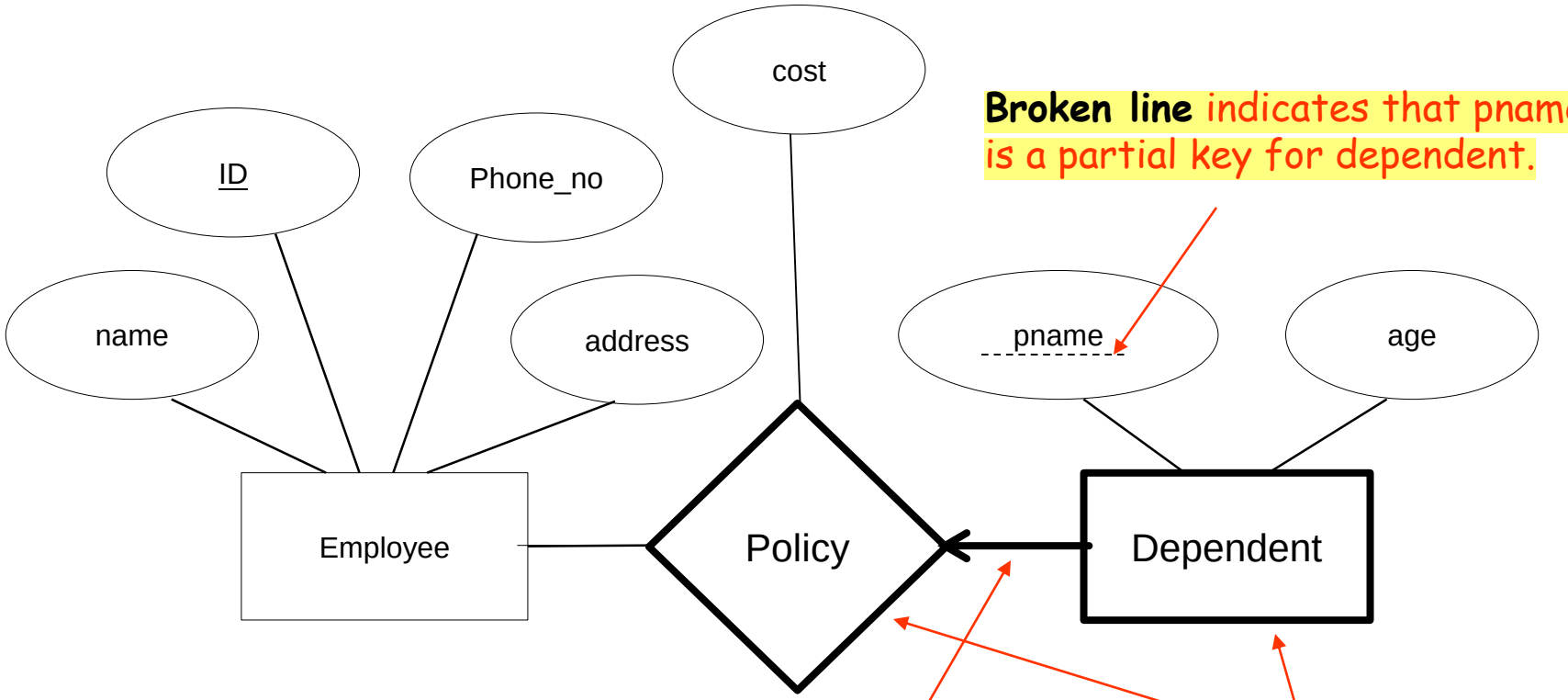
If the participation of an entity set in a relationship set is **partial**, the two are connected by a **thin** link; independently, the presence of an arrow indicates a key constraint.

Entities

- **There are two kinds of entities:**
- **Strong entity:**
  - An entity which has a super key.
  - Each entity can be distinguished from other entities in the same set.
- **Weak entity:**
  - Is an entity without super key.
  - May not be able to distinguish themselves from others without associations with entities in other entity sets.

Example:

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



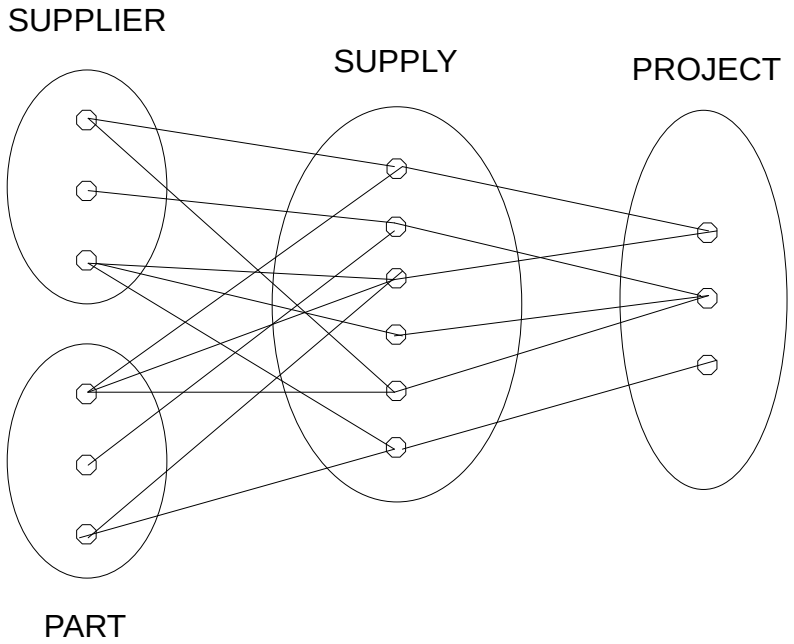
Broken line indicates that pname is a partial key for dependent.

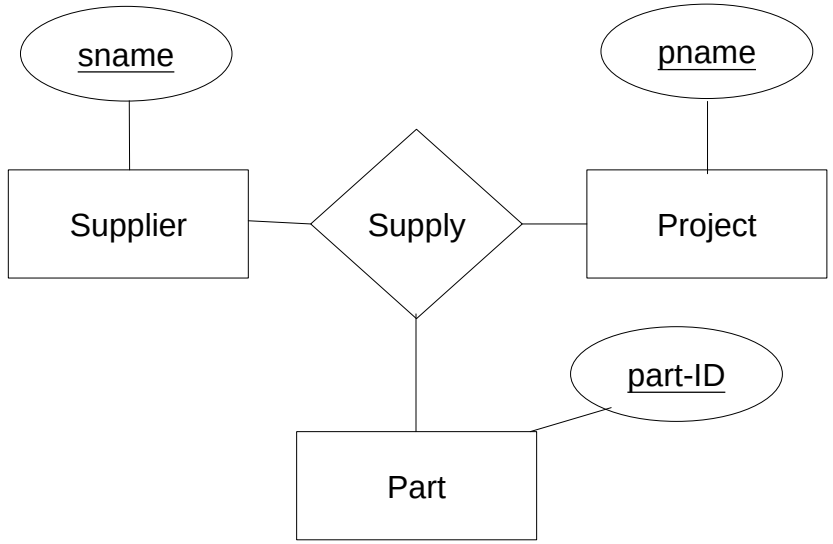
The arrow from Dependent to Policy indicates that each Dependent entity appears in at most one Policy relationship. The arrow is **thick** because of the total participation constraint.

To underscore the fact that Dependent is a weak entity and Policy is its identifying relationship, we draw both with **dark lines**.

Non-binary relationship set

- Suppose n ≥ 2, for E1, E2, ..., En,





Non-binary relationship set

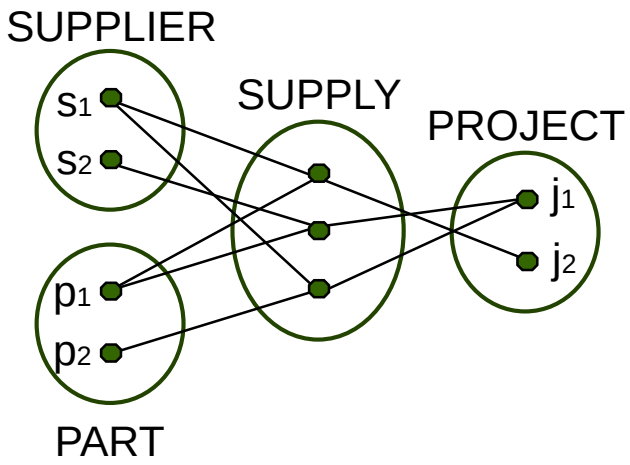
- A relationship set is a subset of the Cartesian product

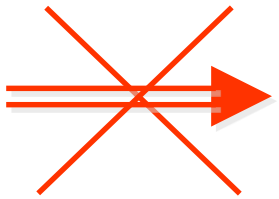
  E1 x E2 x ... x En

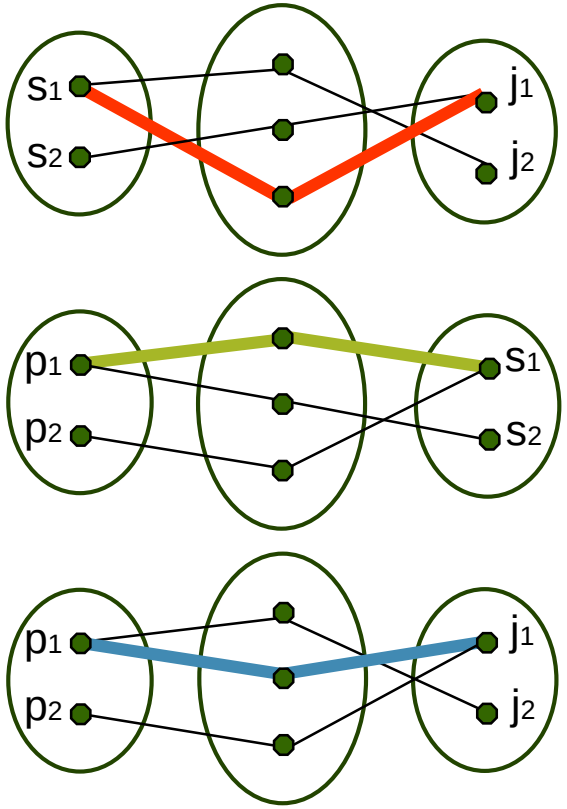
  - n = the degree of the relationship set

- In general, a non-binary relationship set **cannot be** replaced by a number of binary relationship sets.

Consider supplier s1, project j1, and part p1



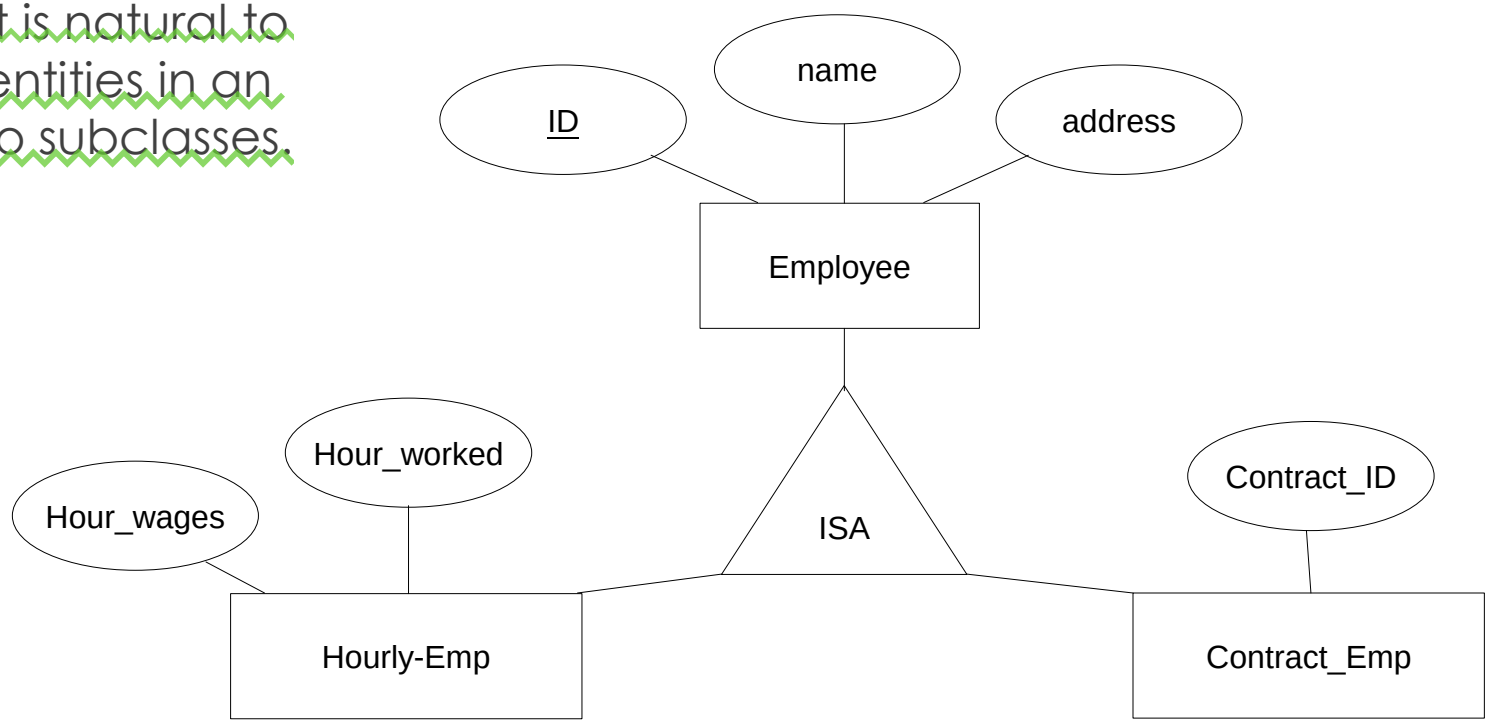




Existence of (s1,p1), (j1,p1) and (s1,j1),
does not imply existence of (s1,j1,p1) necessarily
- This is known as the **connection trap**

Class Hierarchies

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



Class Hierarchies

- Attributes are inherited by the entity set in the subclass.
  - E.g. the attributes defined for an Hourly_Emp entity are the attributes for Employees plus that of Houly_Emp.
- A class hierarchy can be viewed in one of the two ways.

1. A class is specialized into subclasses.
2. The subclasses are generalized by a superclass.

ISA Hierarchies

- We can specify two kinds of constraints with respect to ISA hierarchies,
  - Overlap constraints
    - Determine whether two subclasses are allowed to contain the same entity.

      E.g. an employee can be an **Hourly_Emp** as well as a **Contract_emp** entity
  - Covering constraints
    - Determine whether the entities in the subclasses collectively include all entities in the superclass.

      e.g. every Employees entity also have to be an **Hourly_Emps** or a **Contract_emps**.

Reasons for using hierarchy

- Add descriptive attributes that make sense only for the entities in a subclass.
  - E.g. Hourly_wages does not make sense for a Contract_Emp entity.

- Identify the set of entities that participate in some relationship.
  - E.g. manages relationship : the participating entity sets are Senior_Emp and Department to ensure that only senior employees can be managers.

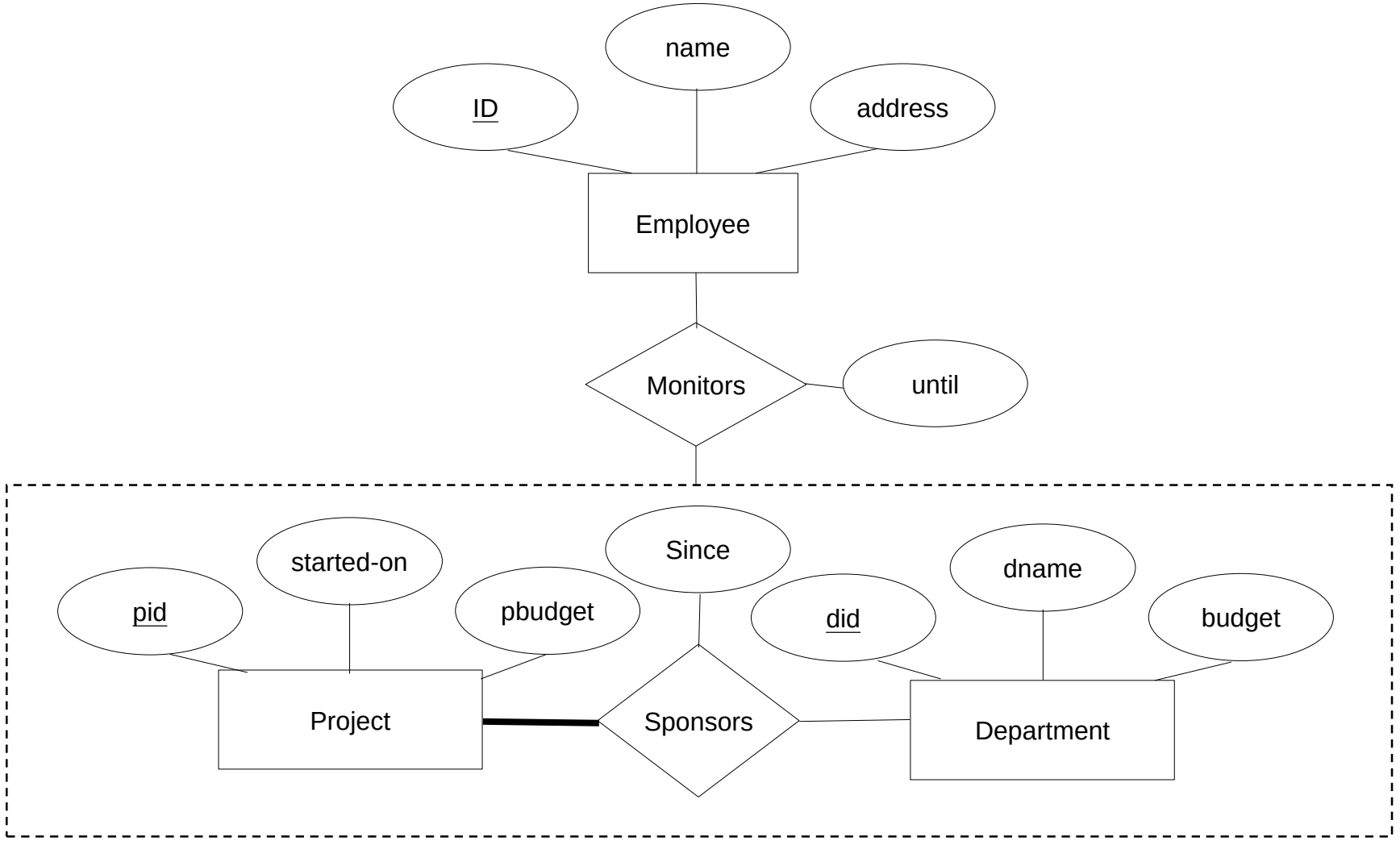
Aggregation

- Sometimes, we have to model a relationship between a collection of entities and *relationships*.

- **Aggregation** allows us to indicate that a relationship set (identified through a dashed box) participates in another relationship set.

Aggregation

- Example
  - Consider an entity set called project.
  - Each project entity is sponsored by one or more departments.
  - A department that sponsors a project might assign employees to monitor the sponsorship.
  - Intuitively, monitors should be a relationship set that associates a Sponsors relationship (rather than a project or department entity) with an Employee entity.


name
ID
address
Employee
Monitors
until
Since
started-on
pid
pbudget
did
dname
budget
Project
Sponsors
Department

many-to-many

Conceptual Design with the E-R model

- Developing an ER diagram presents several choices, including the following:
  - Should a concept be modeled as an entity or an attribute?
  - Concept modeled as an entity or a relationship?
  - What are the relationship sets and their participating entity sets?
  - Should we use binary or ternary relationships?
  - Should we use aggregation?

Entity versus Attribute

- Should **address** be an attribute of Employees or an entity (connected to Employees by a relationship)?
  - If we may have several addresses per employee, *address* must be an entity

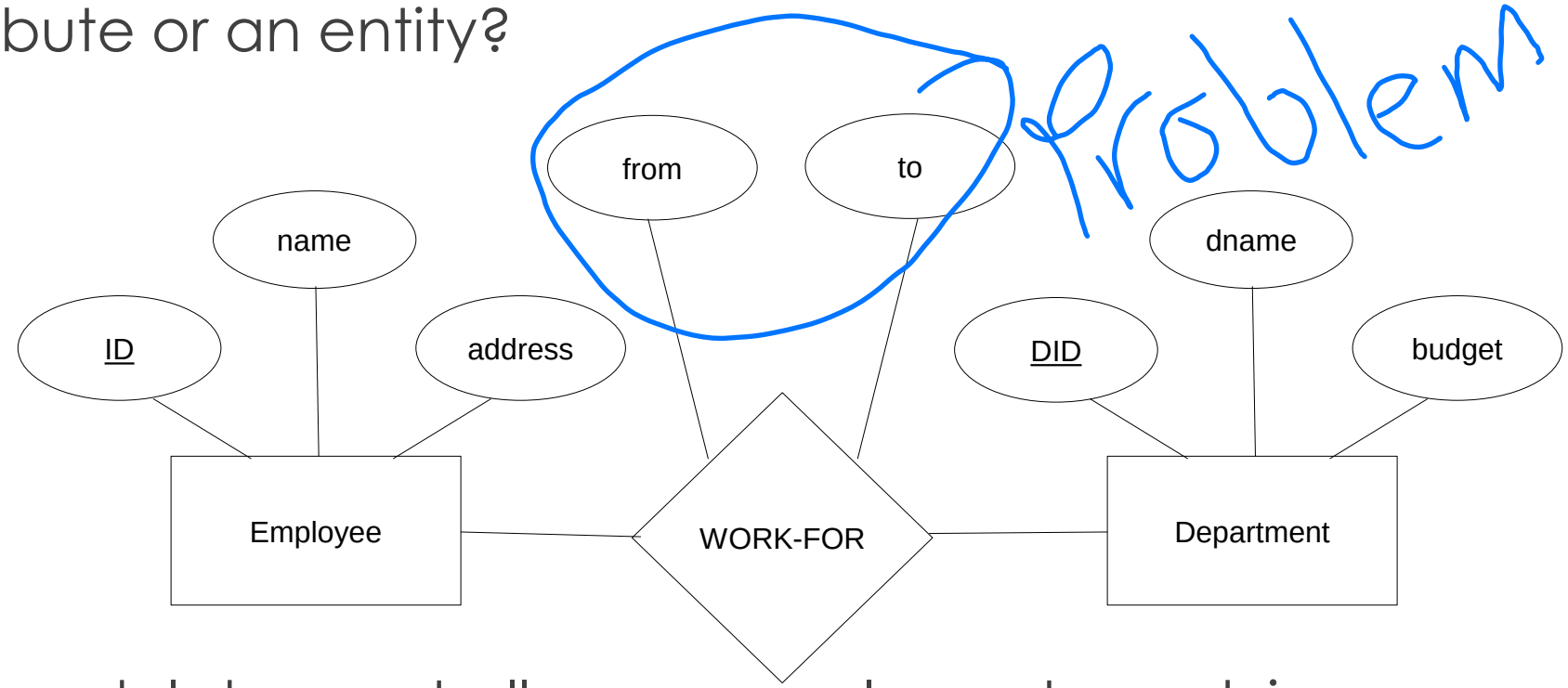
    (attributes cannot be set-valued)
  - If the structure (city, street, etc) is important, e.g. we want to retrieve employees in a given city, address must be modeled as an entity *(attribute values are atomic)*
  - City, street, etc can be modeled as attributes in case of one address for each employee.

Entity versus Attribute

ID	Name	Age	Phone
101	Azad	25	04770

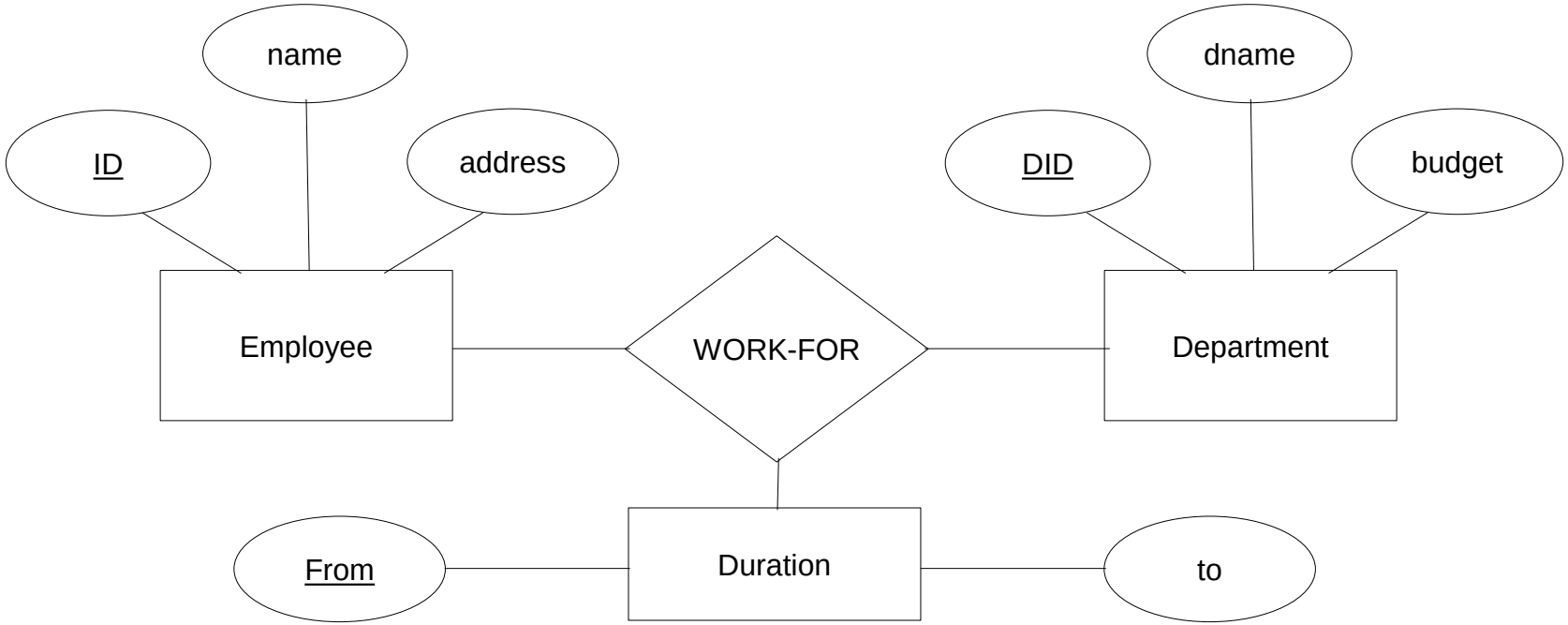
ID	City	Street	House
101	Sulaimani	Slaim	55
101	Sulaimani	Baxtiary	16

- Should **duration** of an employee working in a department be an attribute or an entity?



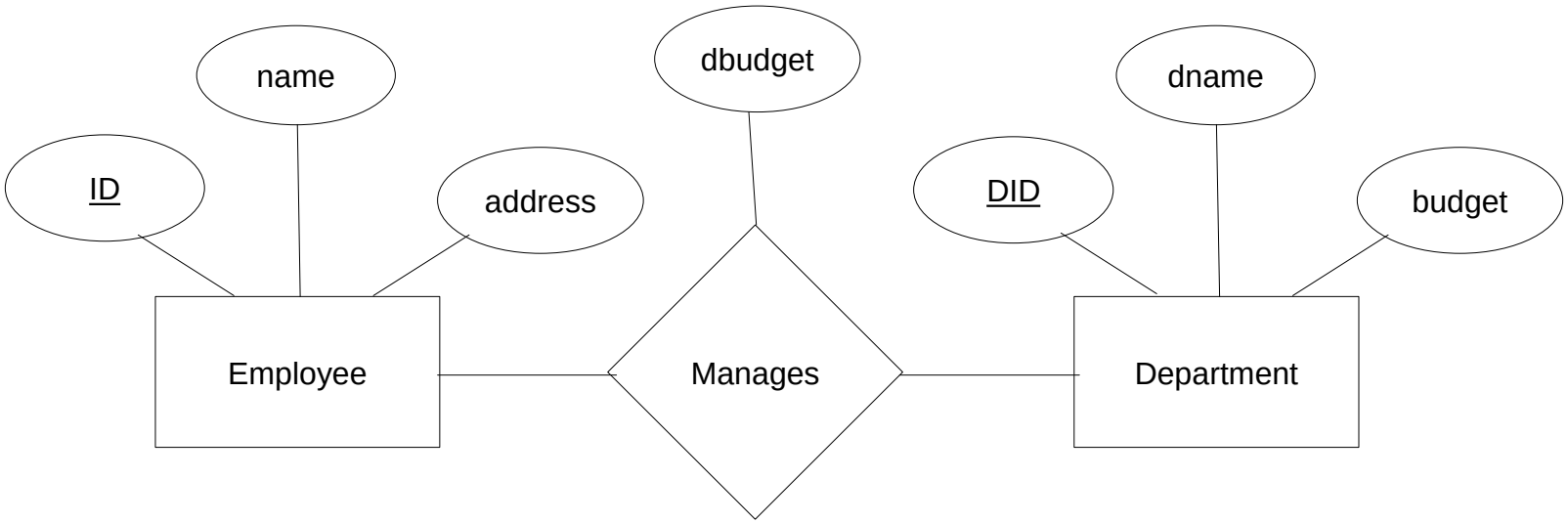
- This ER model does not allow an employee to work in a department for two or more periods.
  - Because a relationship is uniquely identified by the participating entities.

- The problem can be addressed by introducing an entity set called Duration.

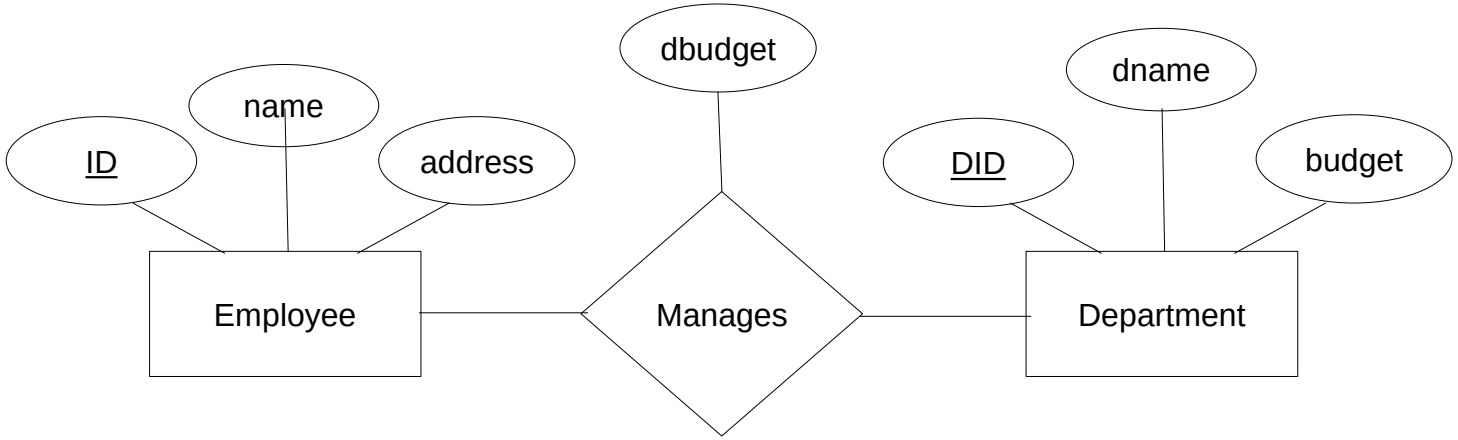


Entity versus Relationship

- Suppose each department manger is given a budget (**dbudget**).



- This approach is natural if we assume that a manager receives a separate budget for each department.



- What if the budget is a sum that covers all departments managed by that employee?
  - In this case, each manages relationship that involves a given employee will have the same value in the dbudget field, leading to redundant storage of the same information.

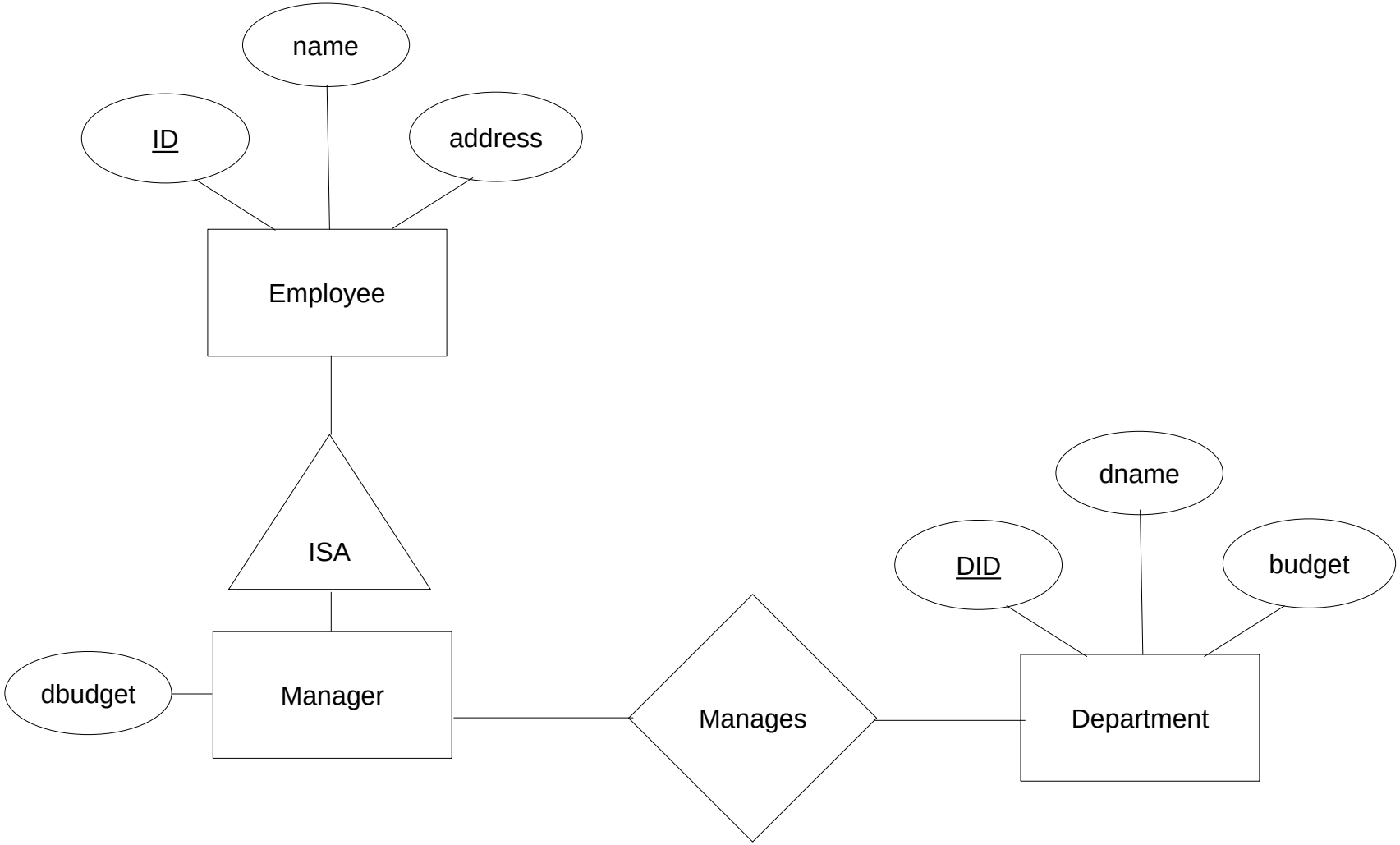
    (Tim, dept A, $200000) (Tim, dept B, $200000)

  - It is also misleading; it suggests that the budget is associated with the relationship, when it is actually associated with the manager.

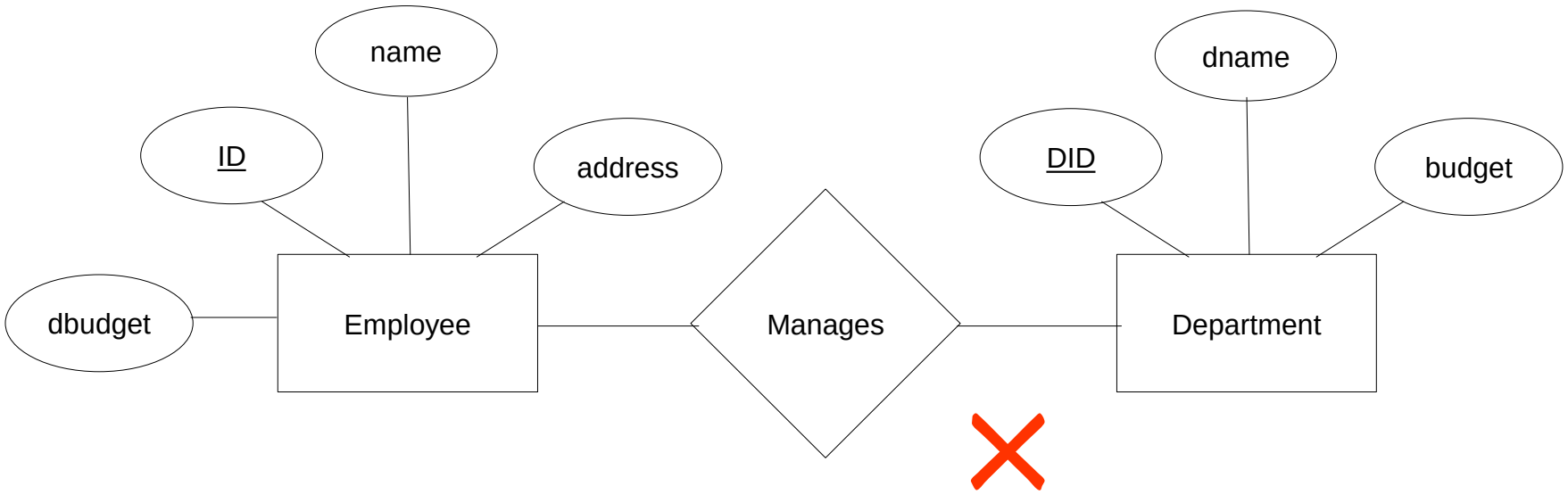
EID	DID	Dbu
101	CS	500
101	Math	400

EID	DID	Dbu
101	CS	900
101	Math	900

- We can address the problem by introducing a new entity set called manager with the **ISA** hierarchy.

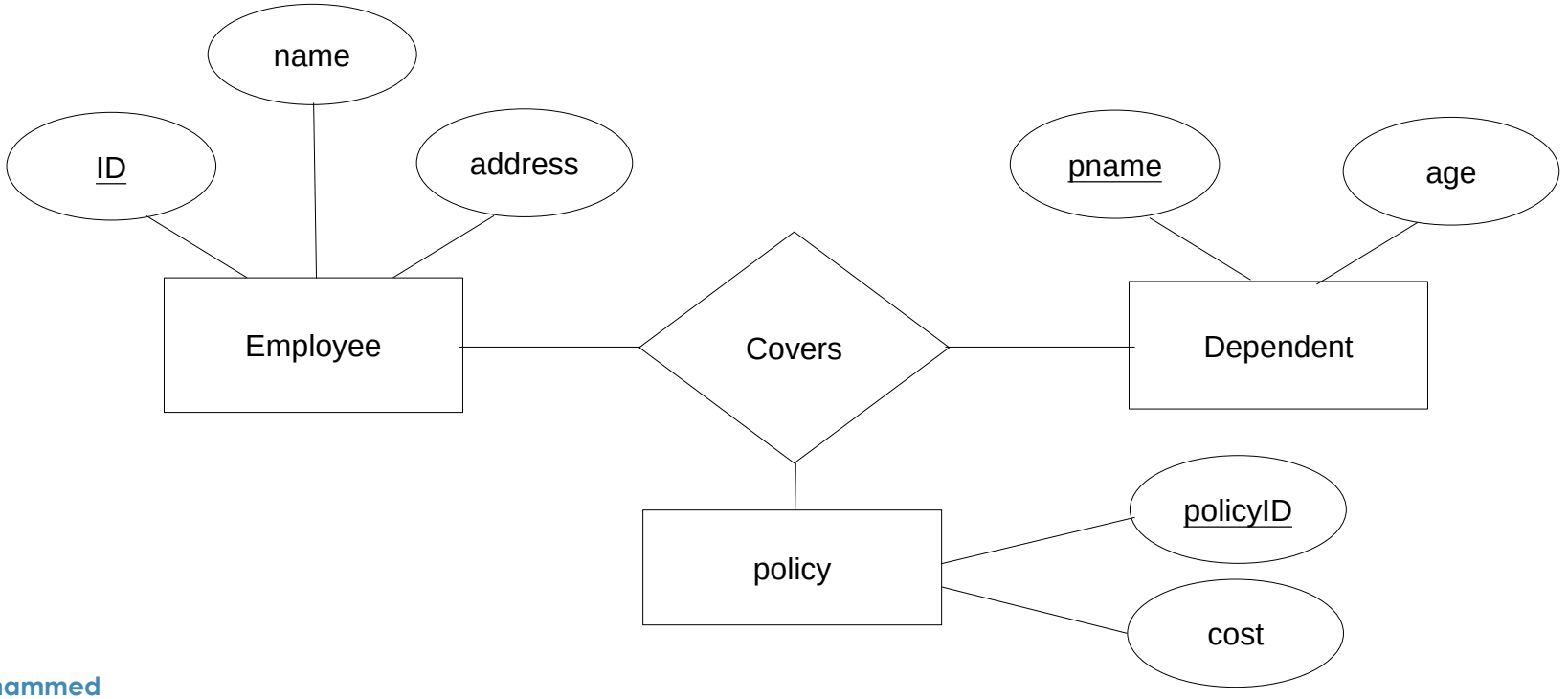


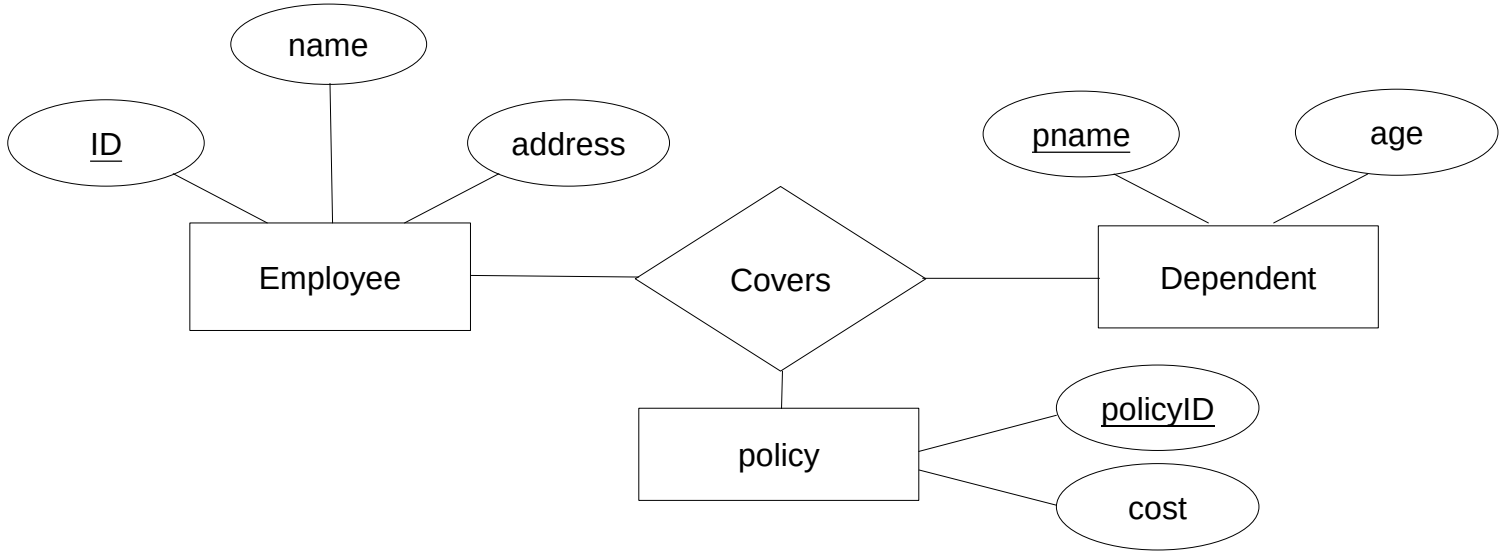
- Why not simply make the budget an attribute of Employee ?



Binary versus Ternary Relationships

- The following ER diagram models the situation that an employee can own several policies, each policy can be owned by several employees, and each dependent can be covered by several policies.

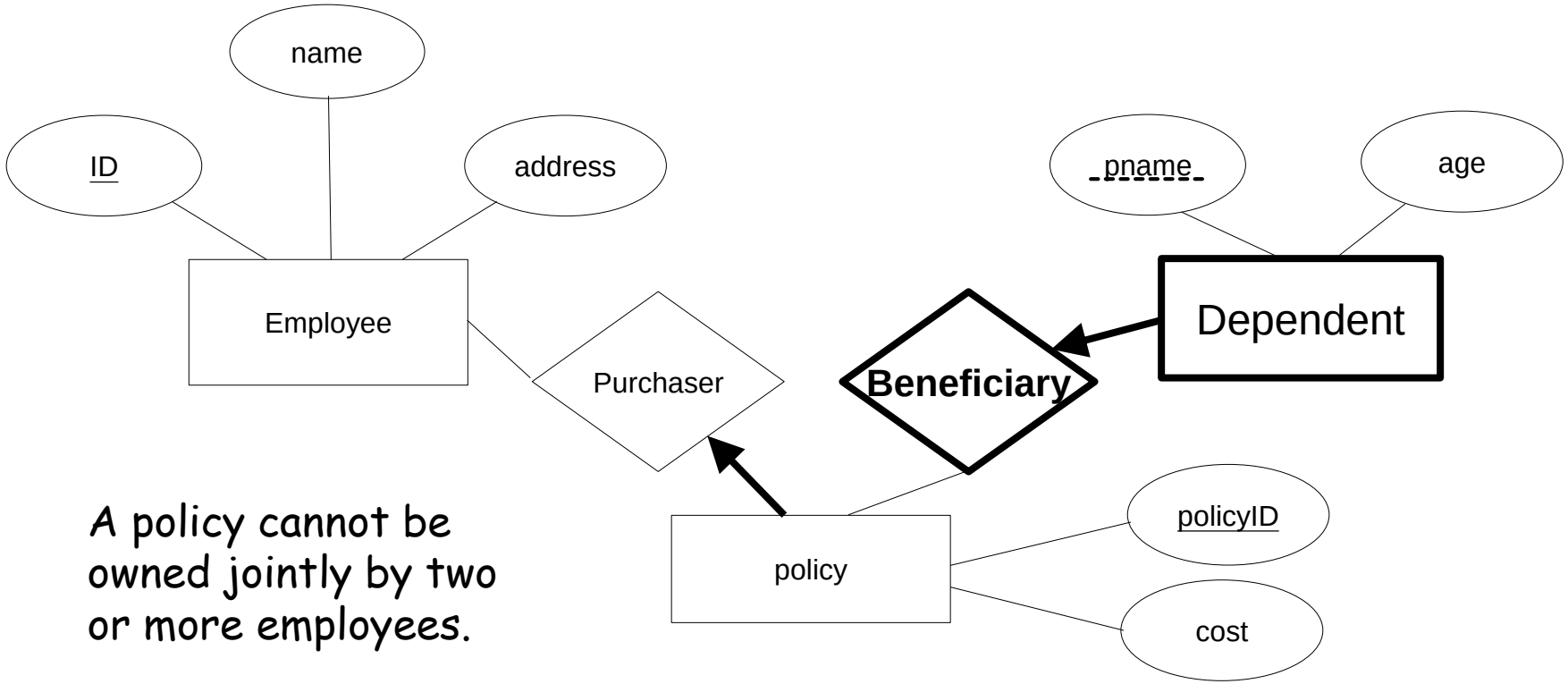




- Suppose we have the following additional requirements:
  - A policy cannot be owned jointly by two or more employees. (Impose a key constraint on Policy ?).
  - Every policy must be owned by some employee.

    (Impose a total participation constraint on Policy ?).
  - Dependents is a weak entity set, and each dependent entity is uniquely identified by (pname, policyID).

- Here is a solution

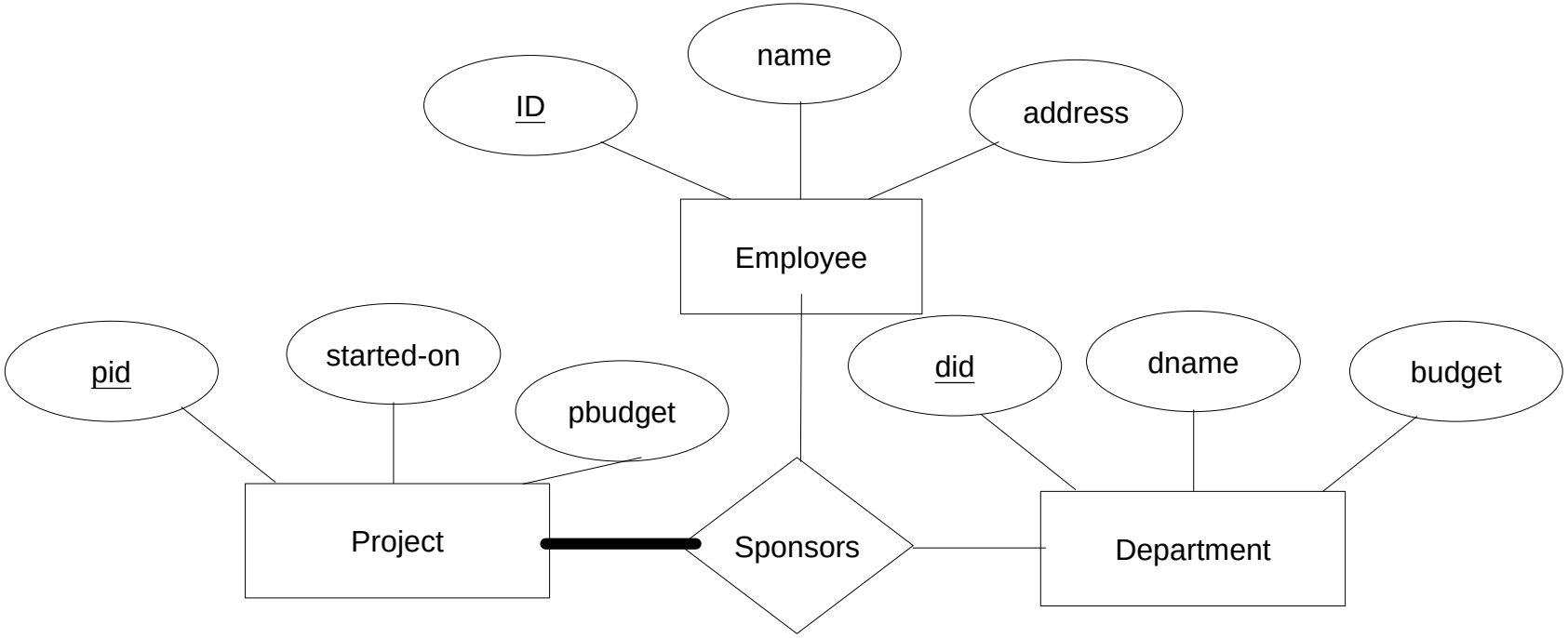


A policy cannot be owned jointly by two or more employees.

Every policy must be owned by some employee.

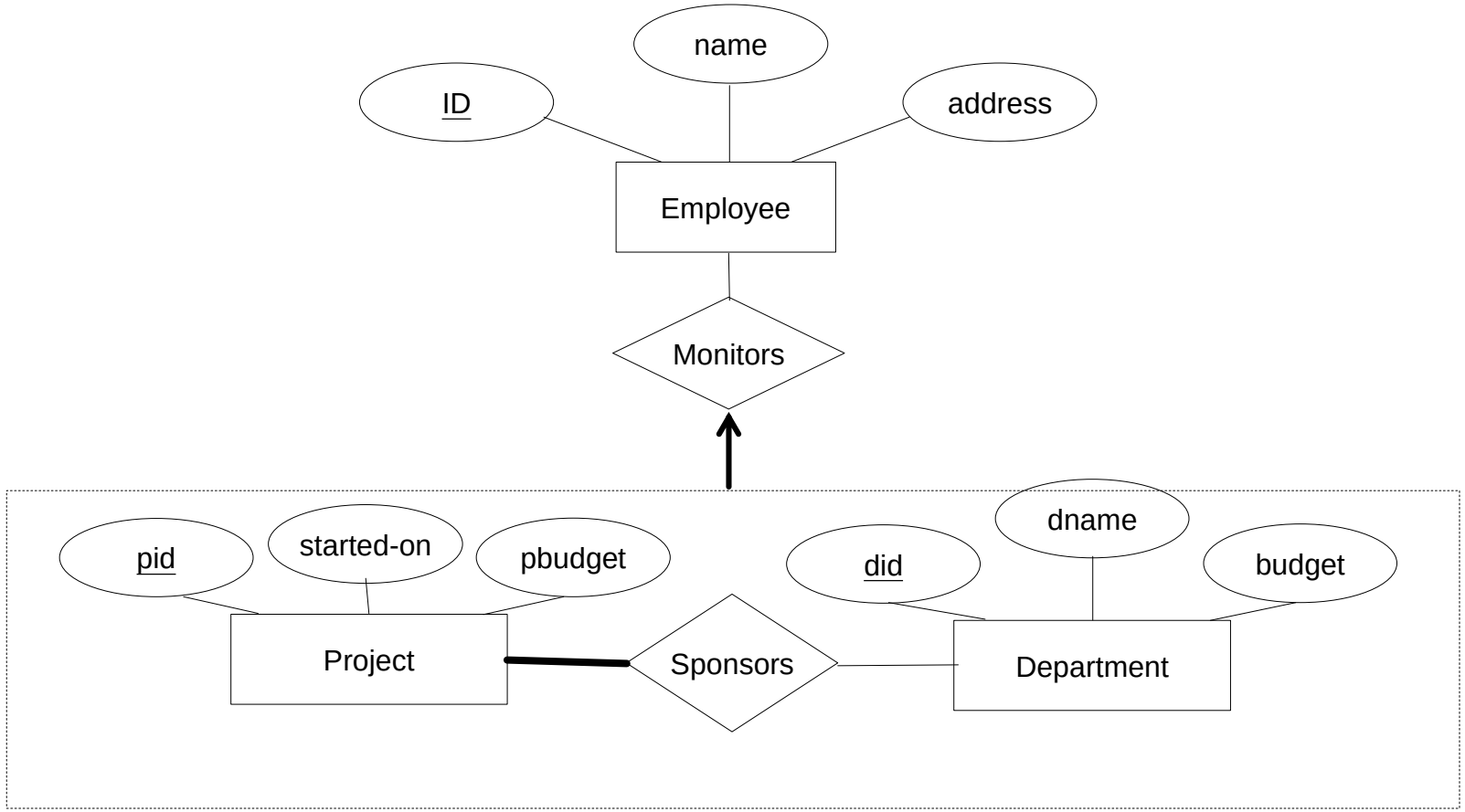
Dependent is a weak entity set, each dependent entity is uniquely identified by (pname, policyID).

Aggregation versus Ternary Relationships



- However, suppose we have a constraint that each sponsorship can be monitored by at most one employee.

- If aggregation is used, we can draw an arrow from the aggregated relationship sponsors to the relationship monitors.



Preferred resources

1) ***Fundamentals of Relational Database Management Systems*** by S. Sumathi
2) ***Database Management System*** by Raghu Ramakrishnan & Johannes Gehrke
3) ***Fundamentals of Database Systems*** by Ramez Elmasri & Shamkant B. Navathe
4) Solution Manuals for ***Database Management System*** by Raghu Ramakrishnan & Johannes Gehrke

Any Question ?