

Data Modeling Using the Entity-Relationship (ER) Model

Part 3

Weak Entity Types

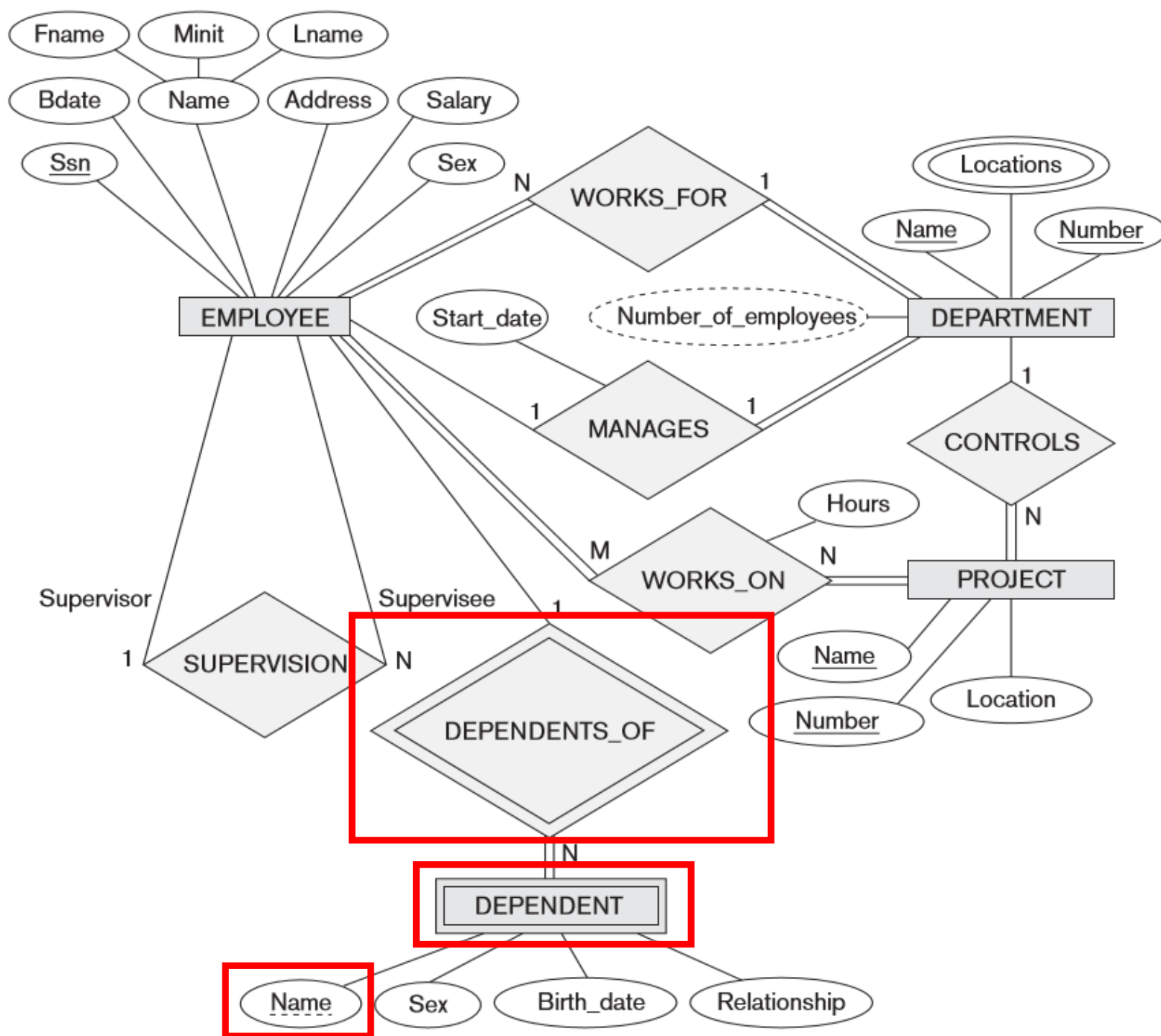
- Entity types that do not have key attributes of their own are called **weak entity types**.
- In contrast, **regular entity types** that do have a key attribute are called **strong entity types**.

- Entities belonging to a weak entity type are identified by being related to specific entities from another entity type in combination with one of their attribute values.
- We call this other entity type the **identifying** or **owner entity type**, and we call the relationship type that relates a weak entity type to its owner the **identifying relationship** of the weak entity type.
- A weak entity type always has a *total participation constraint* (existence dependency) with respect to its identifying relationship because a weak entity cannot be identified without an owner entity.

- Consider the entity type `DEPENDENT`, related to `EMPLOYEE`, which is used to keep track of the dependents of each employee via a 1 : N relationship.
- In our example, the attributes of `DEPENDENT` are `Name` (the first name of the dependent), `Birth_date`, `Sex`, and `Relationship` (to the employee).
- Two dependents of *two distinct employees* may, by chance, have the same values for `Name`, `Birth_date`, `Sex`, and `Relationship`, but they are still distinct entities.
- They are identified as distinct entities only after determining the *particular employee entity* to which each dependent is related.
- Each employee entity is said to *own* the dependent entities that are related to it.

- A weak entity type normally has a **partial key**, which is the attribute that can uniquely identify weak entities that are *related to the same owner entity*.
- In our example, if we assume that no two dependents of the same employee ever have the same first name, the attribute Name of DEPENDENT is the partial key.
- In the worst case, a composite attribute of *all the weak entity's attributes* will be the partial key.

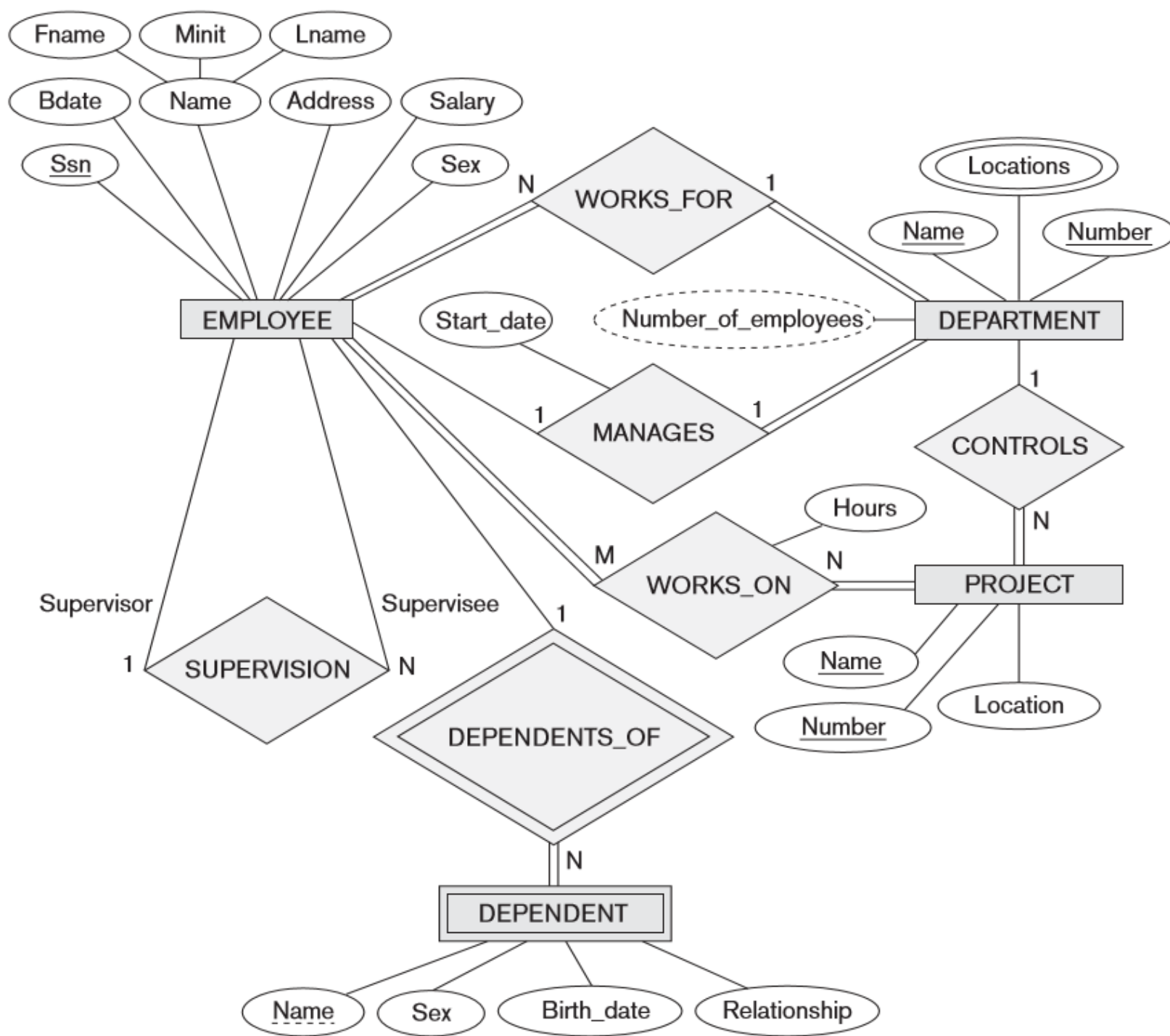
- In ER diagrams, both a weak entity type and its identifying relationship are distinguished by surrounding their boxes and diamonds with double lines.
- The partial key attribute is underlined with a dashed or dotted line.



- Weak entity types can sometimes be represented as complex (composite, multivalued) attributes.
- In the preceding example, we could specify a multivalued attribute Dependents for EMPLOYEE, which is a composite attribute with component attributes Name, Birth_date, Sex, and Relationship.
- The choice of which representation to use is made by the database designer.
- One criterion that may be used is to choose the weak entity type representation if there are many attributes.
- If the weak entity participates independently in relationship types other than its identifying relationship type, then it should *not* be modeled as a complex attribute.


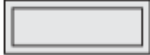









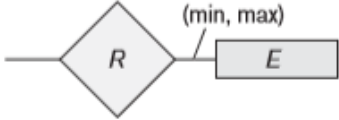
- In general, any number of levels of weak entity types can be defined; an owner entity type may itself be a weak entity type.
- In addition, a weak entity type may have more than one identifying entity type and an identifying relationship type of degree higher than two.

Refining the ER Design for the COMPANY Database



ER Diagrams, Naming Conventions, and Design Issues

Summary of Notation for ER Diagrams

Symbol	Meaning
	Entity
	Weak Entity
	Relationship
	Identifying Relationship
	Attribute
	Key Attribute
	Multivalued Attribute
	Composite Attribute
	Derived Attribute
	Total Participation of E_2 in R
	Cardinality Ratio 1: N for $E_1:E_2$ in R
	Structural Constraint (min, max) on Participation of E in R

Design Choices for ER Conceptual Design

- In general, the schema design process should be considered an iterative refinement process, where an initial design is created and then iteratively refined until the most suitable design is reached.

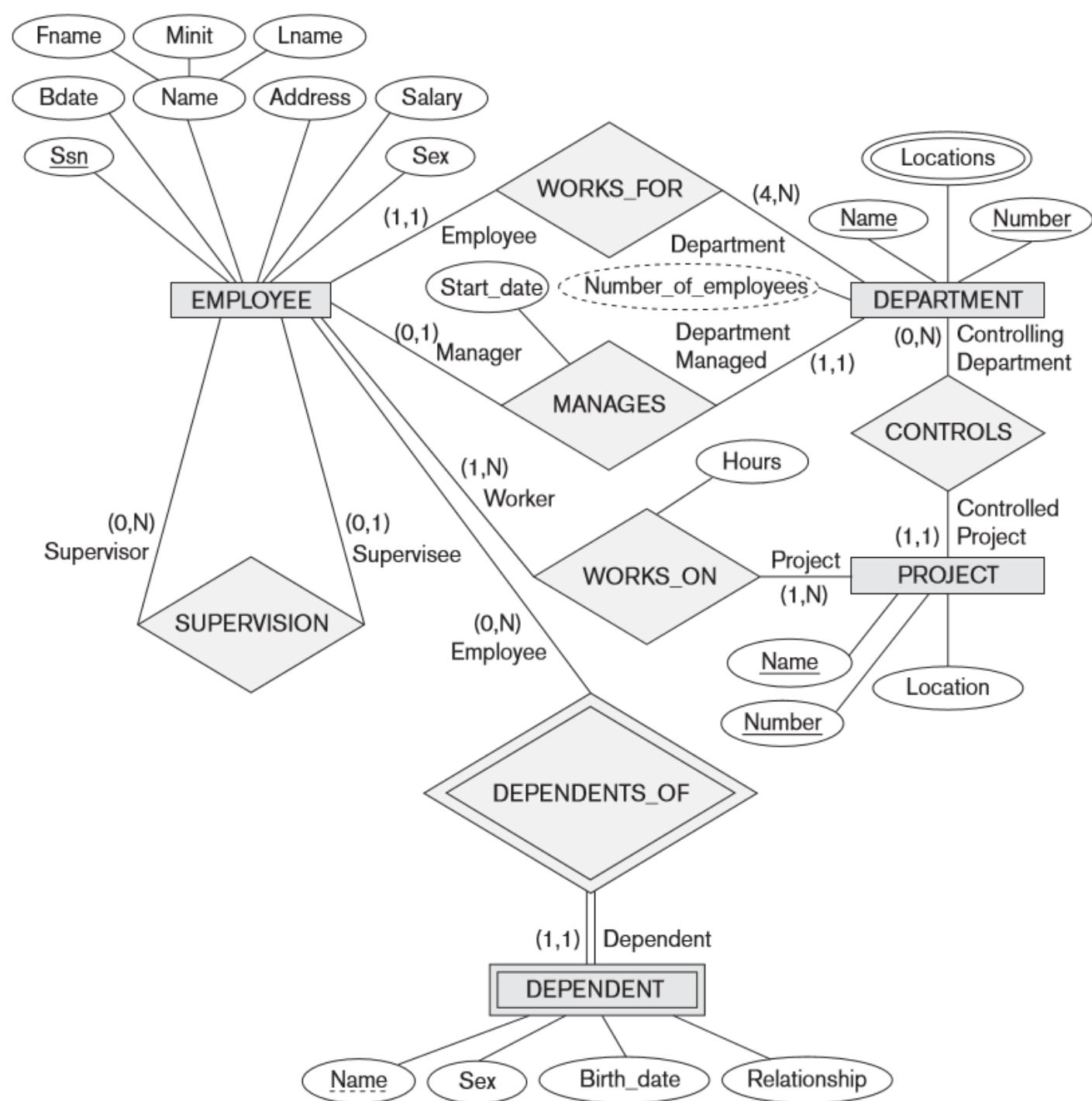
- A concept may be first modeled as an attribute and then refined into a relationship because it is determined that the attribute is a reference to another entity type.
- It is important to note that in our notation, once an attribute is replaced by a relationship, the attribute itself should be removed from the entity type to avoid duplication and redundancy.

- Similarly, an attribute that exists in several entity types may be elevated or promoted to an independent entity type.

- An inverse refinement to the previous case may be applied—for example, if an entity type `DEPARTMENT` exists in the initial design with a single attribute `Dept_name` and is related to only one other entity type, `STUDENT`.
- In this case, `DEPARTMENT` may be reduced or demoted to an attribute of `STUDENT`.

Alternative Notations for ER Diagrams

- We can have one alternative ER notation for specifying structural constraints on relationships.
- This notation involves associating a pair of integer numbers (\min , \max) with each *participation* of an entity type E in a relationship type R , where $0 \leq \min \leq \max$ and $\max \geq 1$.
- The numbers mean that for each entity e in E , e must participate in at least \min and at most \max relationship instances in R *at any point in time*.
- In this method, $\min = 0$ implies partial participation, whereas $\min > 0$ implies total participation.

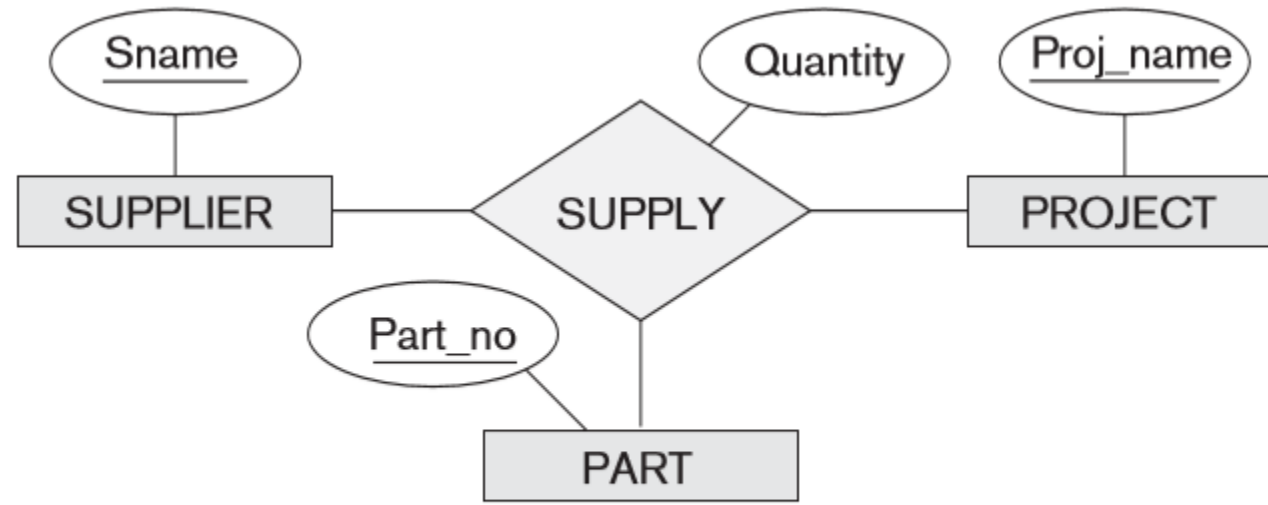


Relationship Types of Degree
Higher than Two

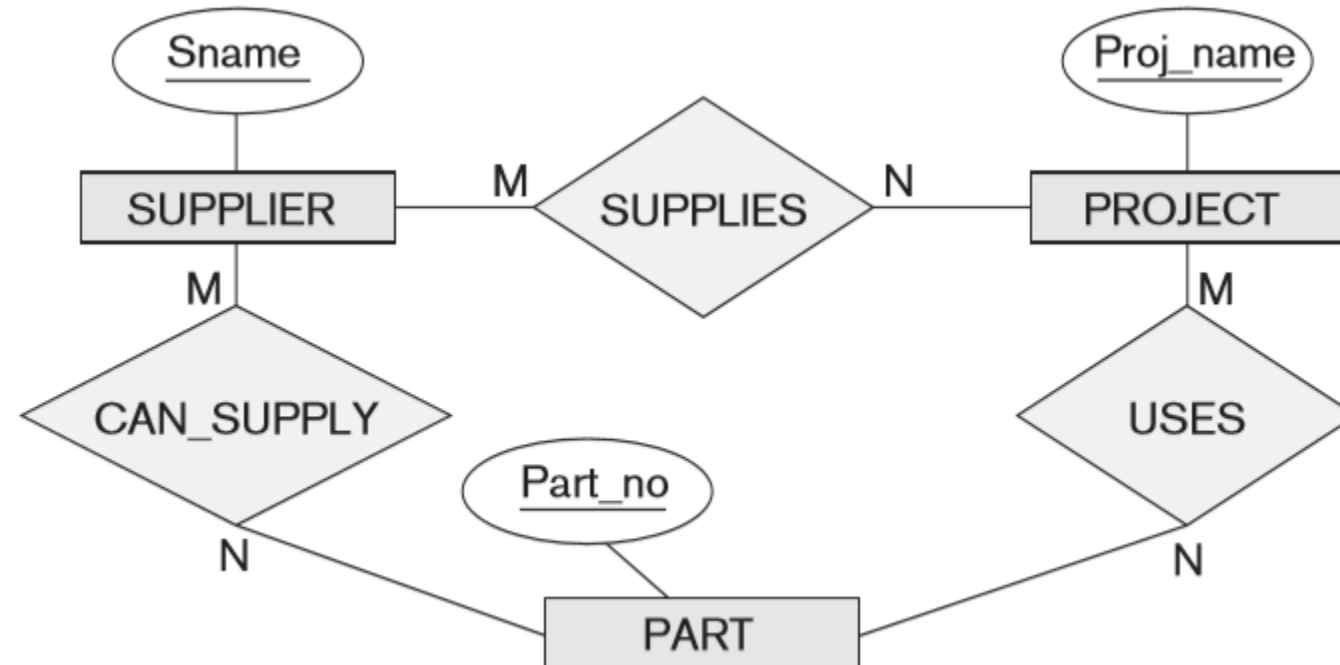
Choosing between Binary and Ternary (or Higher-Degree) Relationships

- In general, a relationship type R of degree n will have n edges in an ER diagram, one connecting R to each participating entity type.

(a)

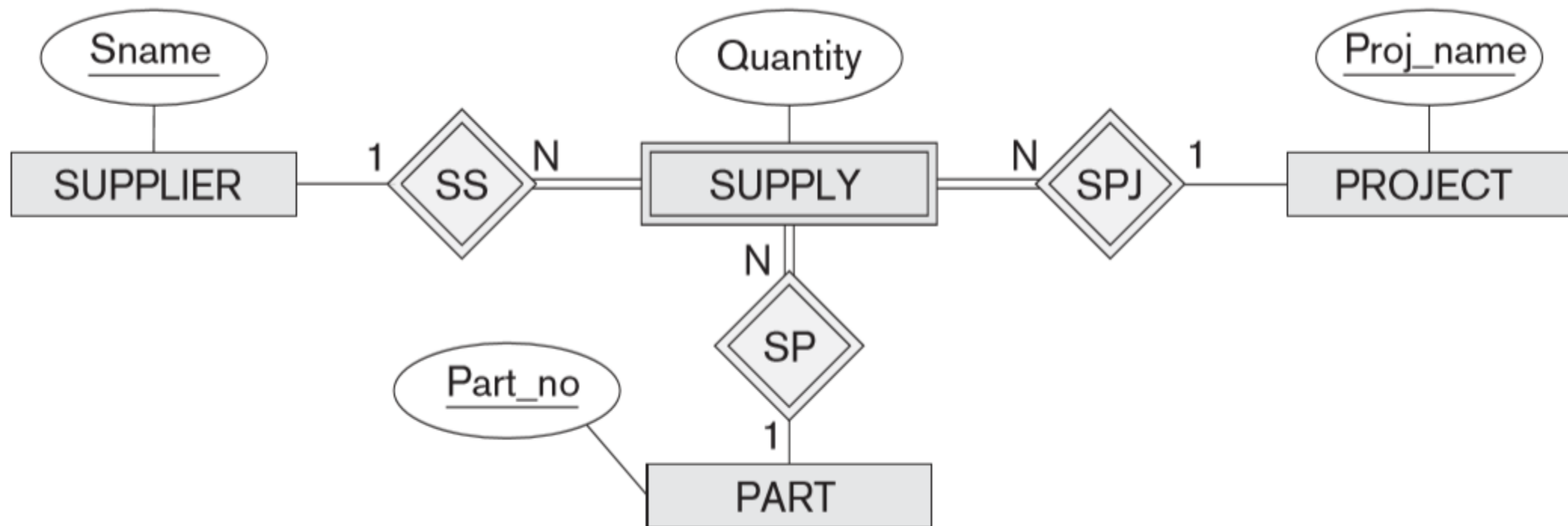


(b)

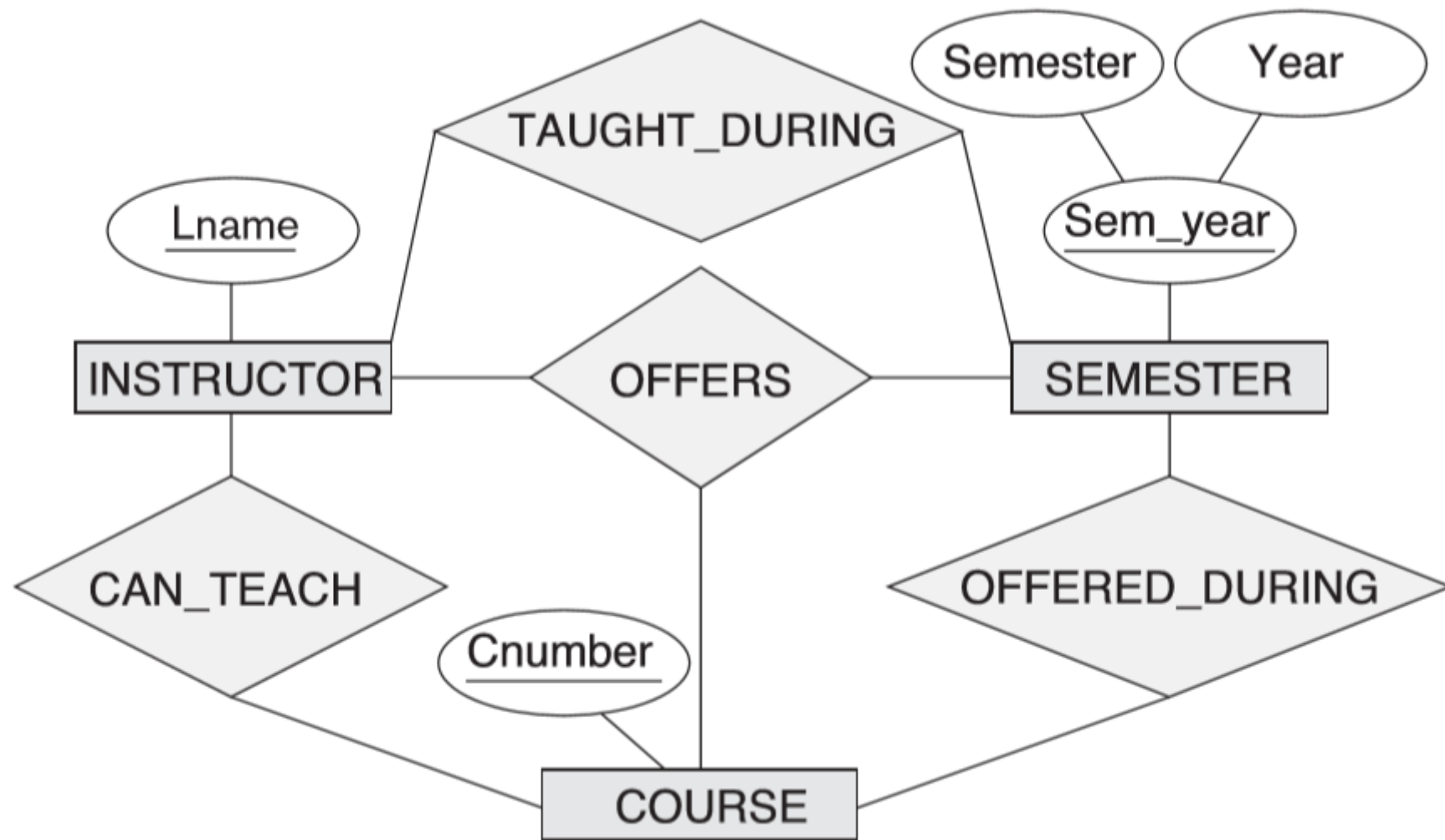


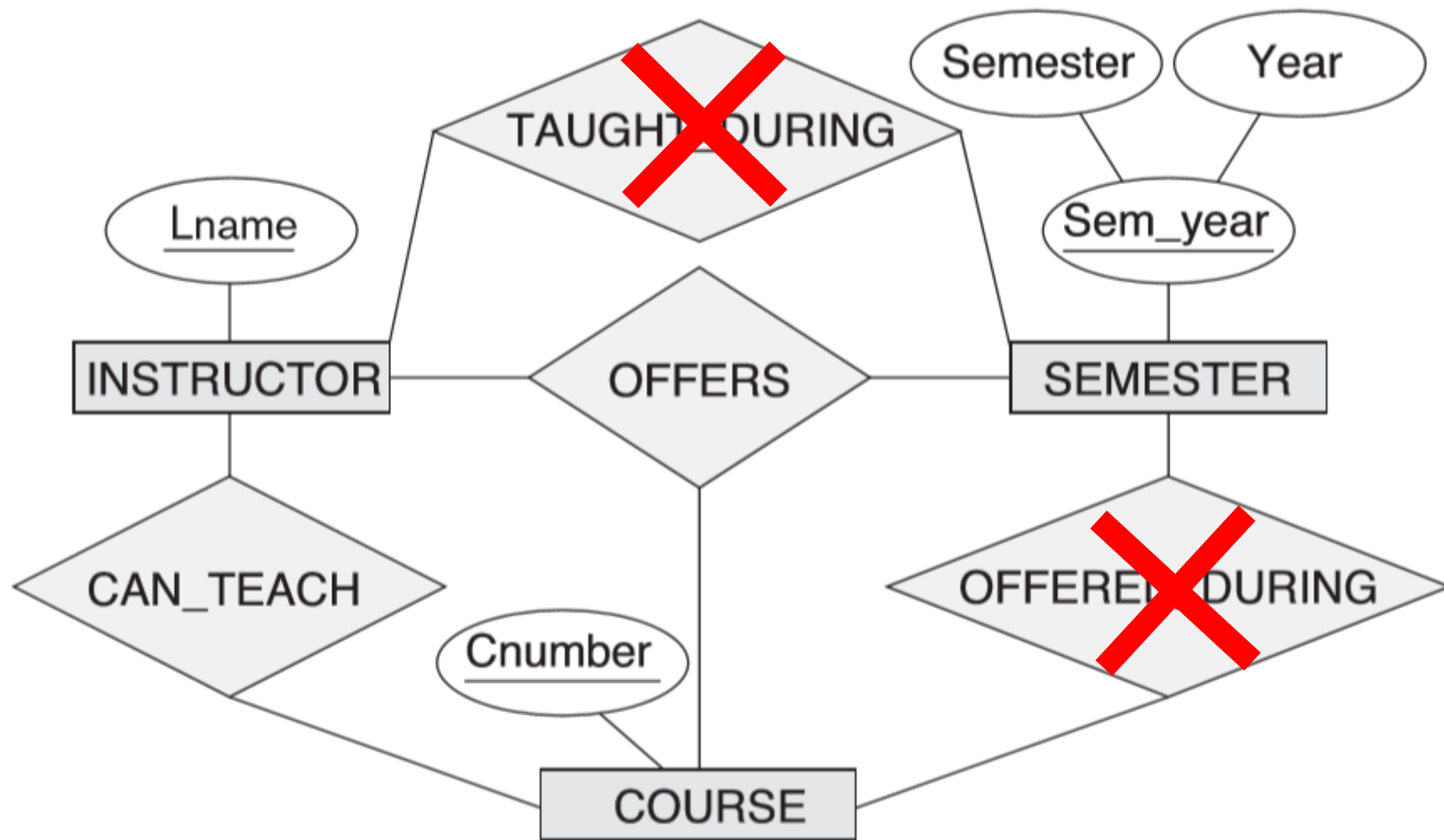
- It is often tricky to decide whether a particular relationship should be represented as a relationship type of degree n or should be broken down into several relationship types of smaller degrees.
- The designer must base this decision on the semantics or meaning of the particular situation being represented.
- The typical solution is to include the ternary relationship *plus* one or more of the binary relationships, if they represent different meanings and if all are needed by the application.

- Some database design tools are based on variations of the ER model that permit only binary relationships.
- In this case, a ternary relationship such as `SUPPLY` must be represented as a weak entity type, with no partial key and with three identifying relationships.

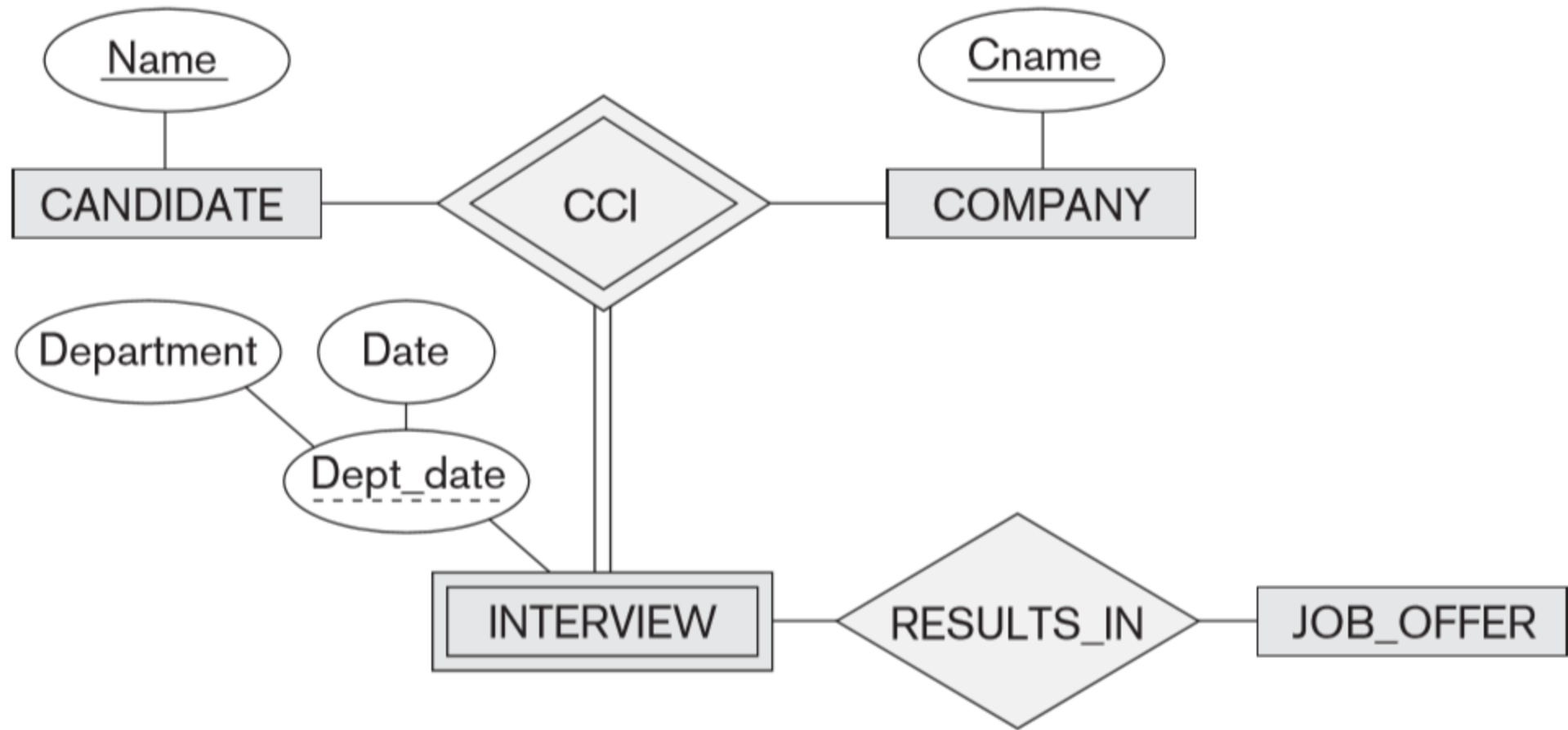


- Another ternary relationship type OFFERS represents information on instructors offering courses during particular semesters





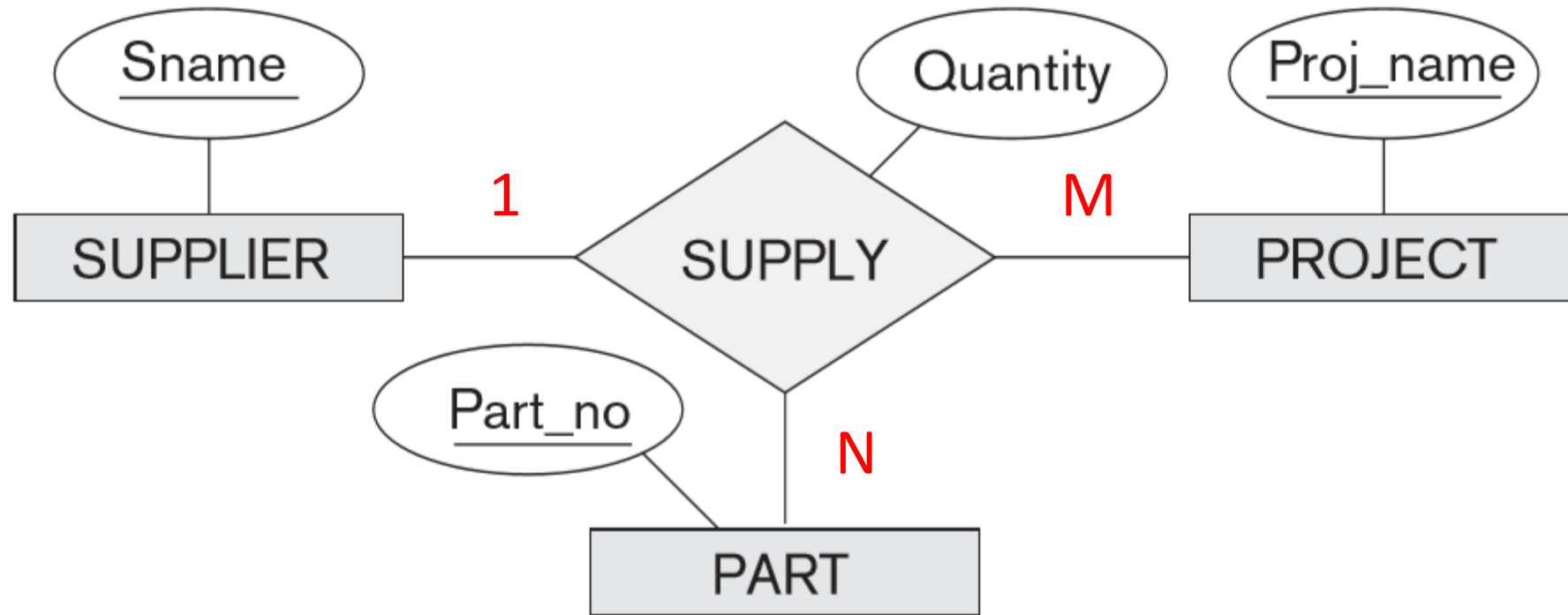
- Notice that it is possible to have a weak entity type with a ternary (or n -ary) identifying relationship type.
- In this case, the weak entity type can have *several* owner entity types.

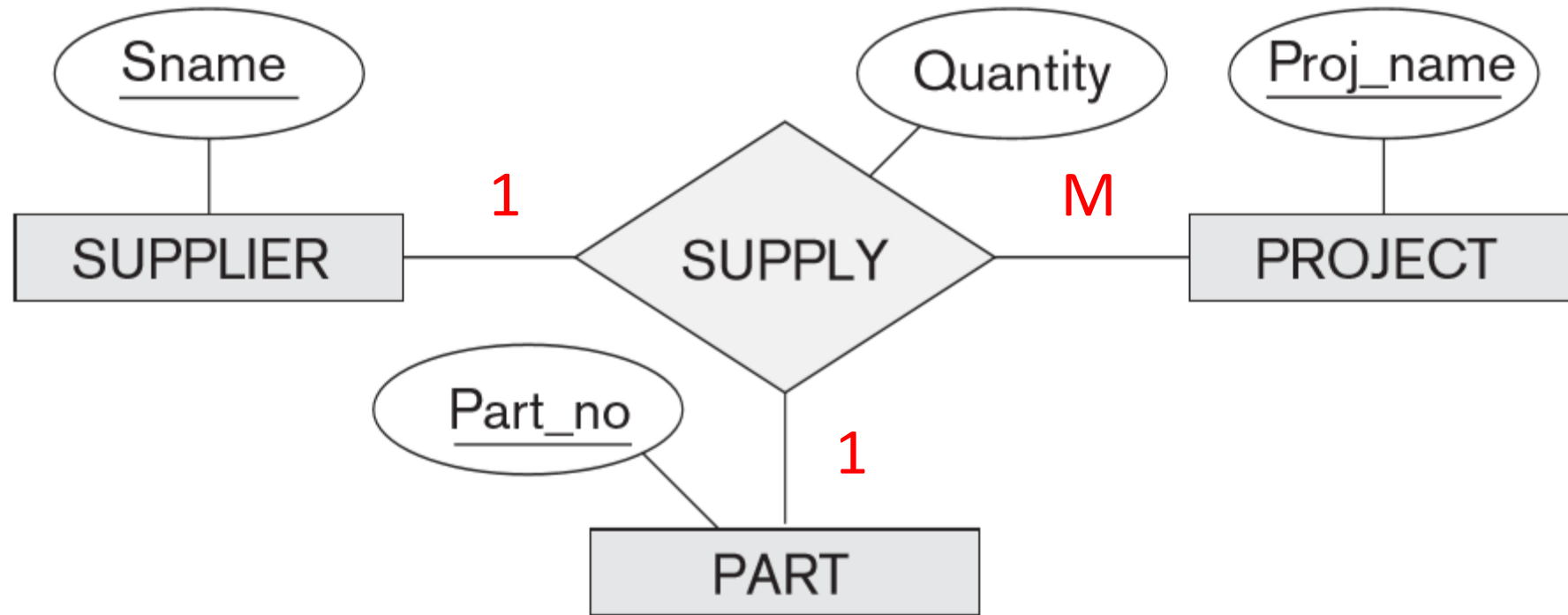


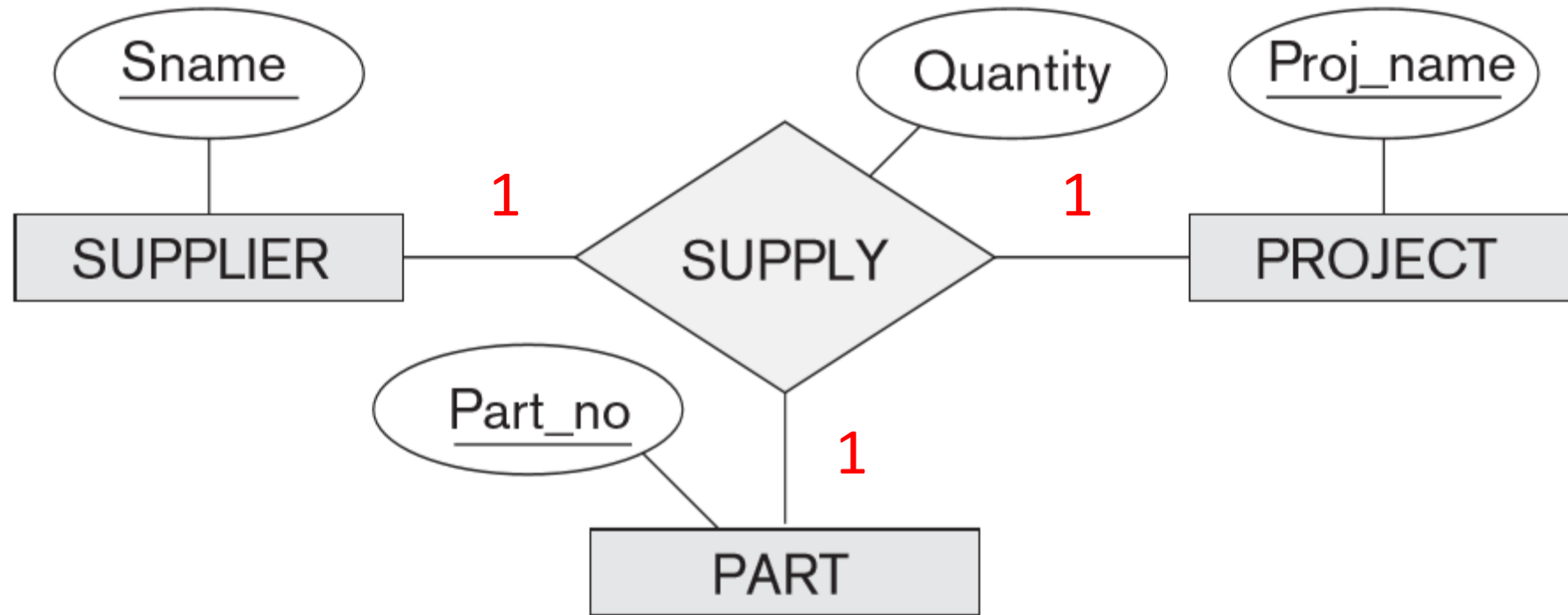
Constraints on Ternary (or Higher-Degree) Relationships

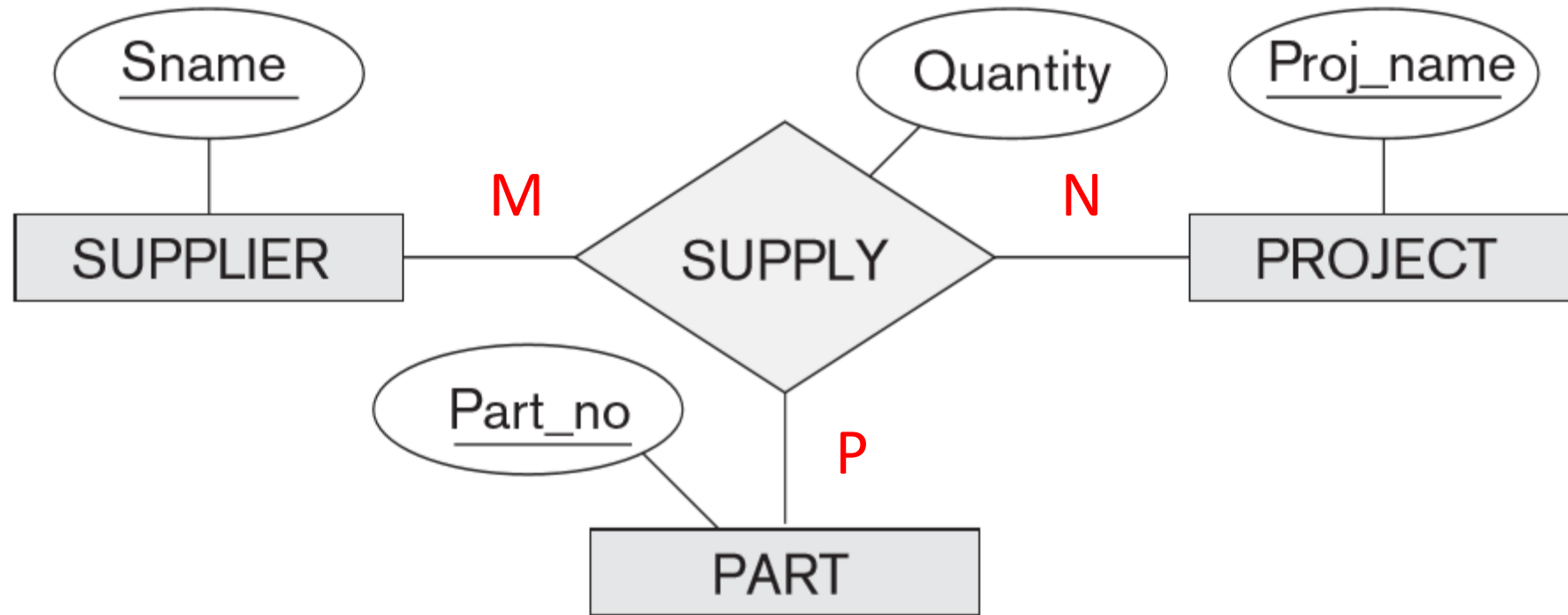
- There are two notations for specifying structural constraints on n -ary relationships, and they specify different constraints.
- They should thus *both be used* if it is important to fully specify the structural constraints on a ternary or higher-degree relationship.

- The first notation is based on the cardinality ratio notation.









- The second notation is based on the (\min, \max) notation.
- A (\min, \max) on a participation here specifies that each entity is related to at least *min* and at most *max relationship instances* in the relationship set.