IN 1400 - Fundamentals of Databases and Database Design

DATABASE ANALYSIS AND DESIGN 2

Week 3

Outline

- Naming Relationships
- Entity Set and Value Set
- Relationship Sets and Instances
- Relationship Types
- Examples

Exercise

You need to filter out unwanted information

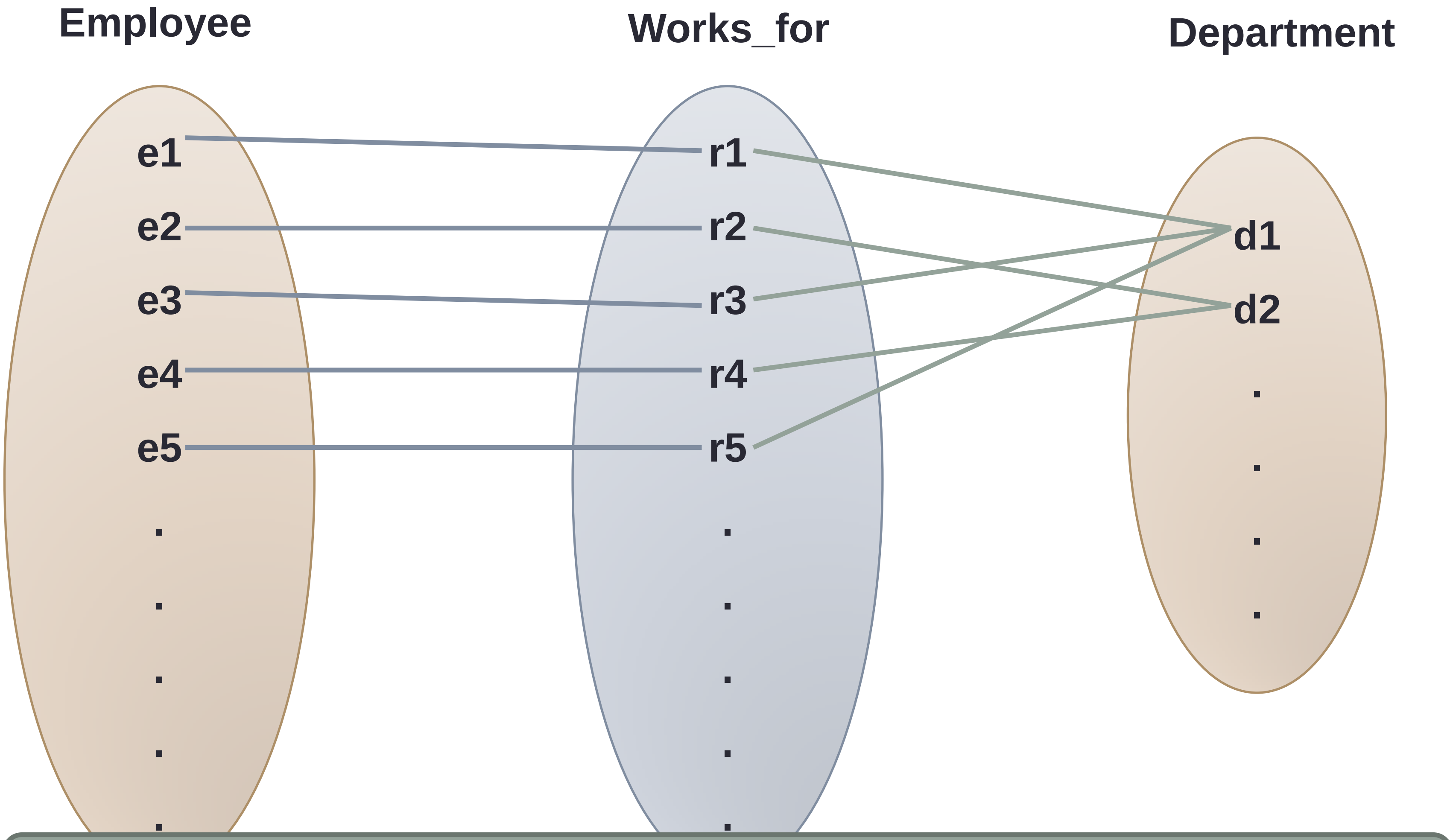
Scenario	Entities	Relationship
An employee works in a department		
A student studies DBMS, OOP, web development		
An employee has three children		
Some lecturers are assigned to exam invigilation		
A module will be cancelled if there are no registrants		
A company sells 10 products. Customers can order products by phone.		
A teacher calculates the average mark of each student. The marks are recorded in report cards which are sent to parents.		

Entity Set and Relationship Set

- Entity Set - A collection of similar entities
  - E.g. All employees
  - All entities in an entity set have the same set of attributes
  - Each entity set has a key

- Relationship Set – A collection of similar relationships
  - A set of relationship instances r_i , where each r_i associate n individual entities (e_1, e_2….. e_n)
  - Each relationship instance r_i in a relationship type, include exactly one entity from each participating entity type.

Some instances of Works_for relationship set



Each relationship instance in the relationship set Works_for associates one employee instance and one department instance.

Cardinality Constraints

- The number of instances of one entity that can or must be associated with each instance of another entity.

- Maximum cardinality (type of relationship)
  - Describes the maximum number of instances in which an entity participates in a relationship.
    - One to one
    - One to many
    - Many to many

- Minimum cardinality
  - Describes the minimum number of entity instances that must participate in a relationship
    - If zero, then optional (in partial participation)
    - If one or more, then mandatory (in total participation)

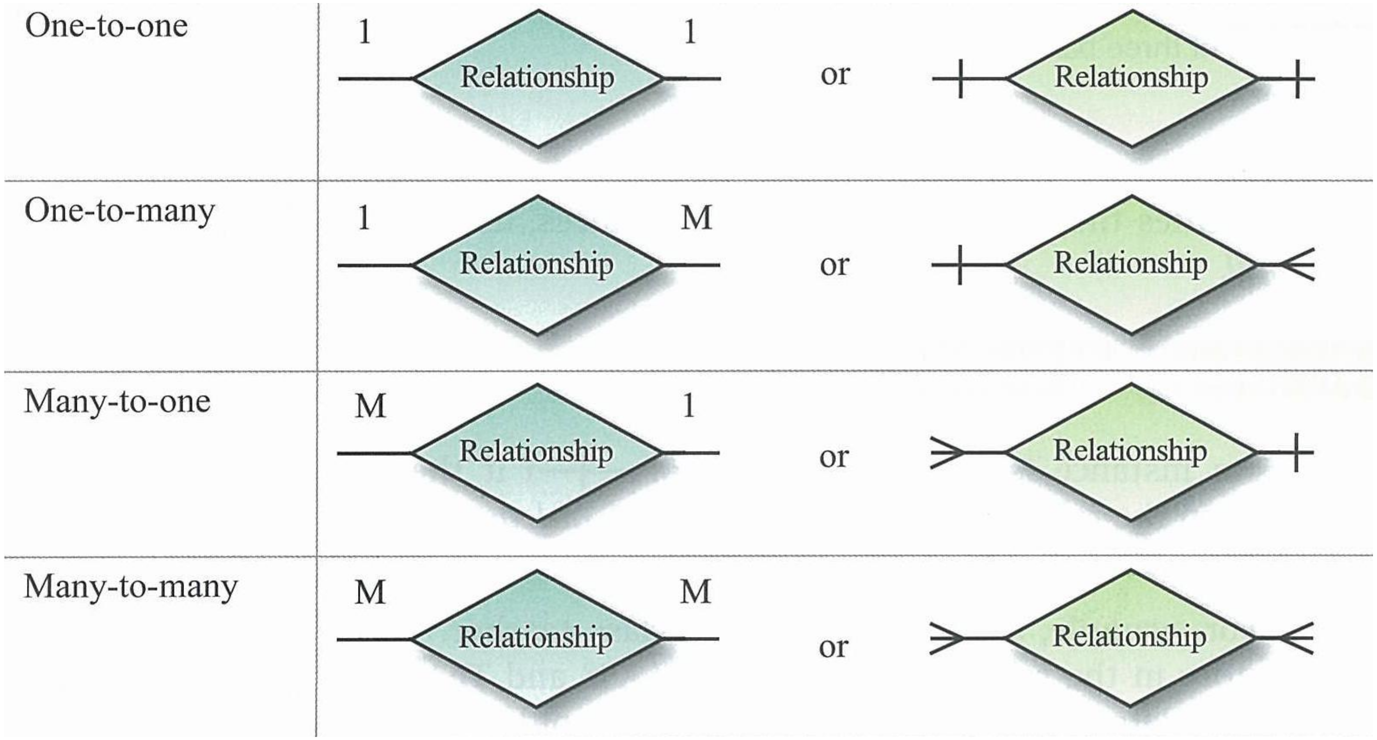
Cardinality Notations

- The crow's foot notation is widely accepted as the most intuitive style
- In addition, OMT, IDEF, Bachman, or UML notation are used to indicate cardinality.

Chen notation

Crow's foot notation

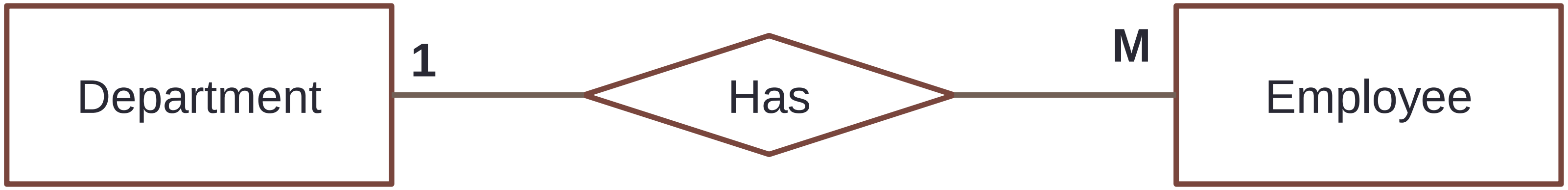
Maximum cardinality



Cardinality in Chen Notation

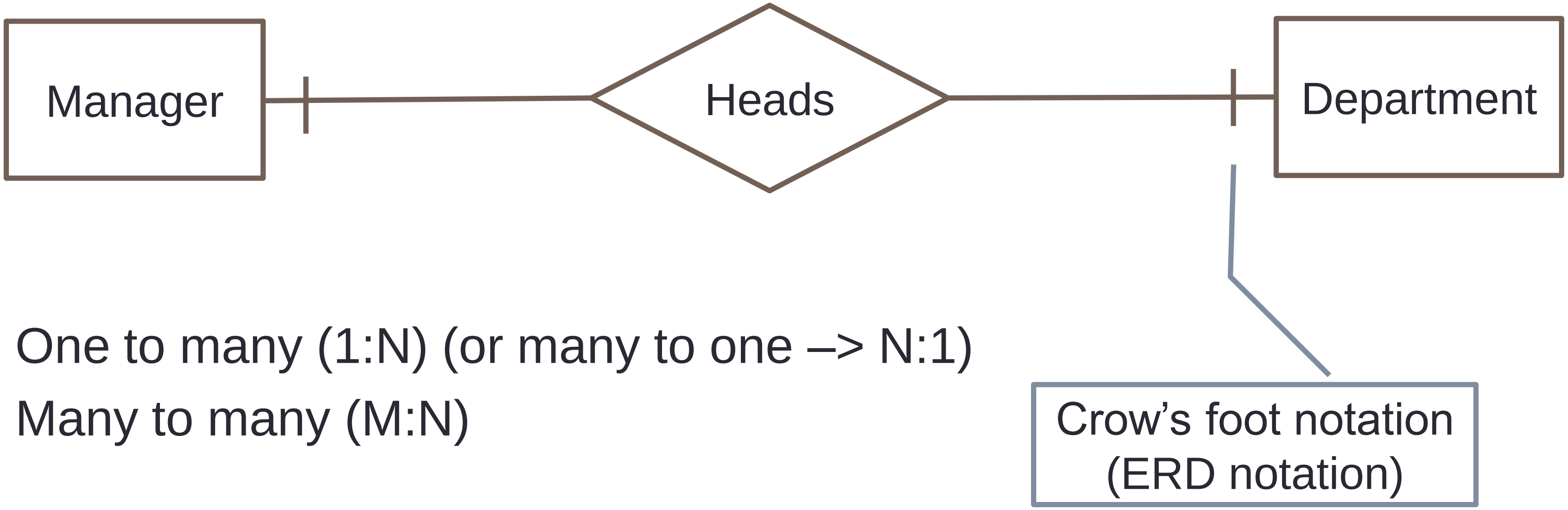
- Cardinality in ER diagrams using Chen notation

- E.g. In a department, can have multiple Employees.
- The relationship in this case follows a “one to many” model.



Cardinality Ratios

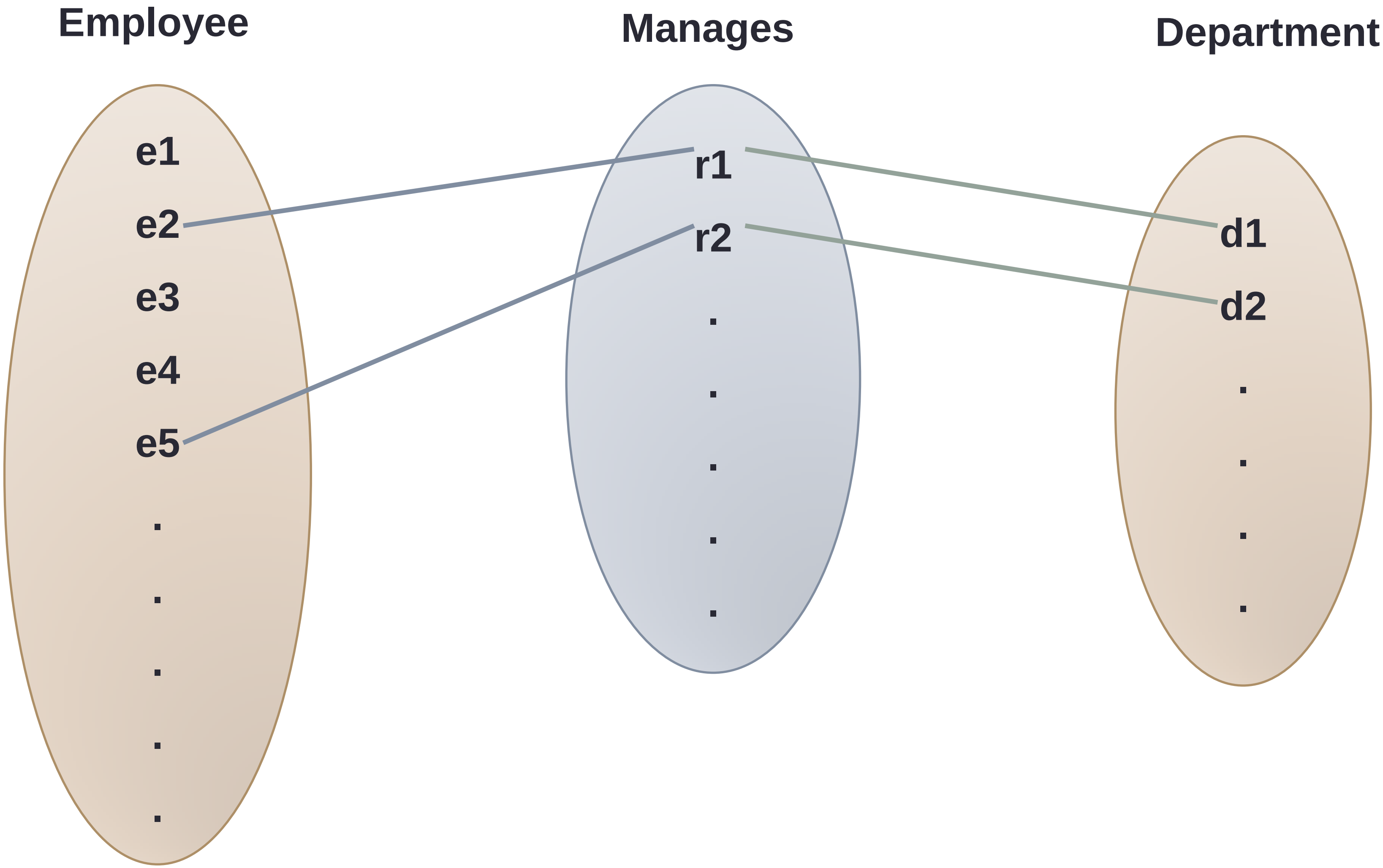
- This leads to 3 types of relationship
  - One to one (1:1)
    - One instance of the first entity can correspond to only one instance of the second entity
    - E.g. A manager head one department and vice versa



Crow's foot notation (ERD notation)

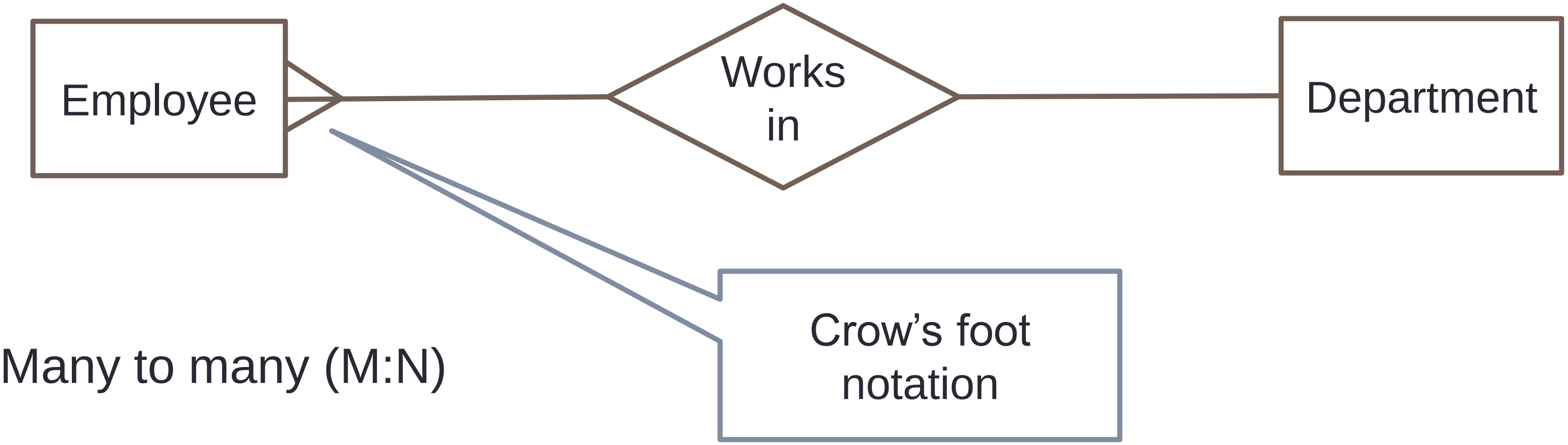
  - One to many (1:N) (or many to one –> N:1)
  - Many to many (M:N)

1:1 Relationship: Manages



Cardinality Ratios

- This leads to 3 types of relationship
  - One to one (1:1)
  - One to many (1:N) (or many to one –> N:1)
    - One instance of the first entity can correspond to more than one instance of the second entity
    - E.g. An employee works in one department or one department has many employees

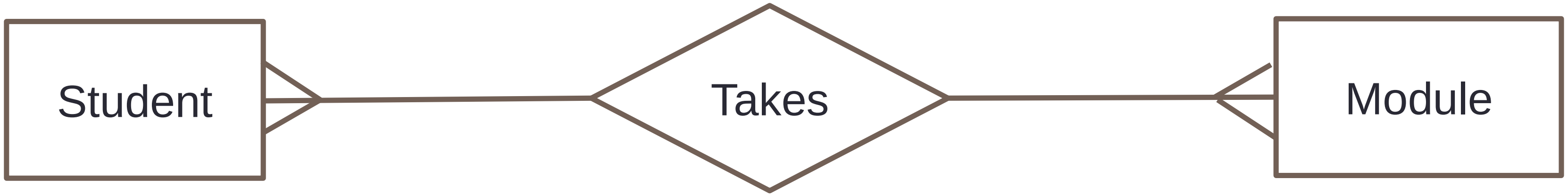


  - Many to many (M:N)

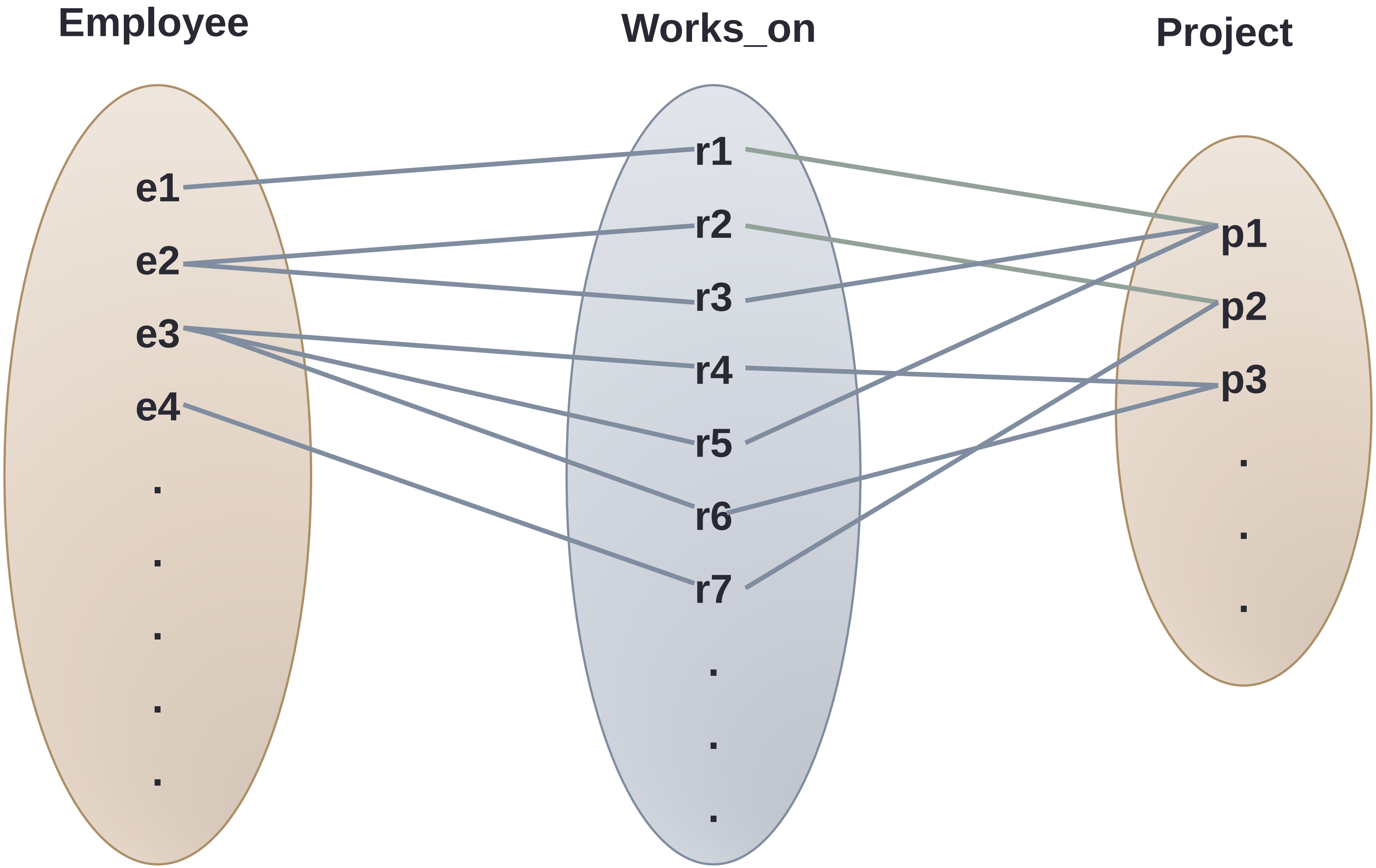
Cardinality Ratios

- This leads to 3 types of relationship
  - One to one (1:1)
  - One to many (1:N) (or many to one –> N:1)

  - Many to many (M:N)
    - More than one instance of the first entity can correspond to more than one instance of the second entity
    - E.g. Each student takes several modules, and each module is taken by several students



M:N Relationship: Works_on



Examples

Example	Relationship
A student has one and only one student card. Each student card is owned by one and one only student.	
A student may borrow some books from the library. A book in the library may be borrowed by a student.	
A student may apply for some scholarships. Each scholarship may be applied for by some students.	
Student takes at least one course. A course is taken by at least one student.	
A teacher may be in charge of a class. Each class must be in charge of by one teacher.	

Participation Constraint

- Specifies whether the existence of an entity depends on its being related to another entity via the relationship type.

- Specifies the minimum number of relationship instances that each entity can participate in, and is sometimes called the **minimum cardinality** constraint.

- Cardinality ratio and participation constraint together are referred as **structural constraints** of a relationship type.

- Two types of participation constraints
  1. Total participation (also known as existence dependency)
  2. Partial participation

Total participation

- An instance of one entity **cannot exist without** the existence of some **other related entity**.

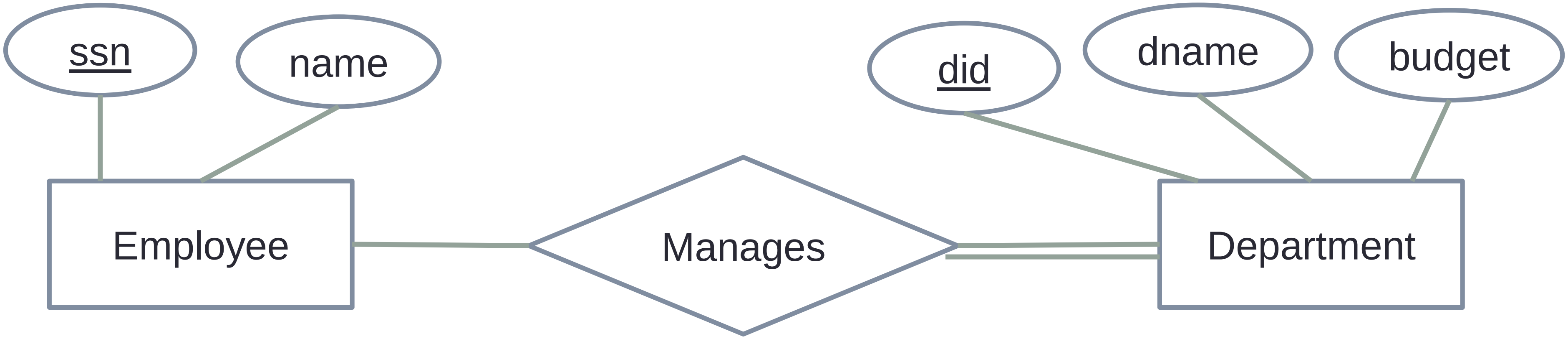
- E.g.
- If a company policy states that *every employee must work for a department*, then an employee entity can exist only if it participates in at least one Works_for relationship instance.

- Participation of employee in Works_for relationship is called Total Participation.

- Every entity in “the total set” of employee entities must be related to a department entity via Works_for

Total participation

- Does every department have a manager?
  - If so, this is a *total participation constraint*: the participation of Departments in Manages is said to be total.



Partial Participation

- E.g.
- Every employee does not manage a department.
- Therefore, the participation of Employee in the Manages relationship type is partial.

- Some or “part of the set of” employee entities are related to some department via Manages, but not necessarily all.

Structural Constraints on Relationships using the (Min, Max) Notation

- Associate (min, max) with each participation of an entity type E in a relationship type R.

- $0 \le min \le max$ and $max \ge 1$

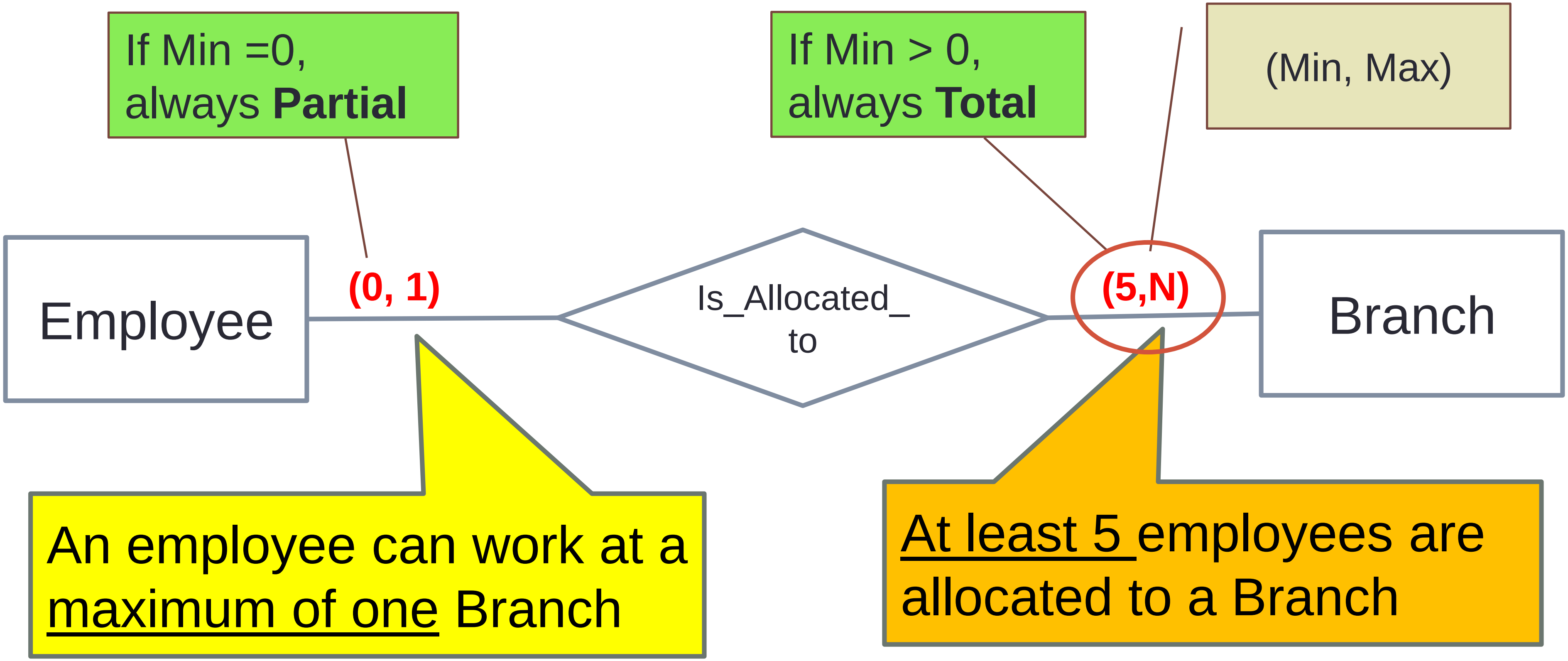
- For each entity e in E, e must participate in at least min and at most max relationship instances in R, at any point in time.

- min = 0 means partial participation, min > 0 means total participation

- One can either use (min, max) notation instead of cardinality ratio/single-line/double-line.

Displaying Participation Constraints using the (Min, Max) Notation

- At least 5 staff members are allocated to a Branch. A member of Staff need not work at a Branch office.



An employee can work at a <u>maximum of one</u> Branch

<u>At least 5 </u>employees are allocated to a Branch

Example

- A university consists of a number of departments. Each department offers several courses. A number of modules make up each course. Students enrol in a particular course and take modules towards the completion of that course. Each module is taught by a lecturer from the appropriate department, and each lecturer tutors a group of students

- Identify Entities
- Identify Relationships

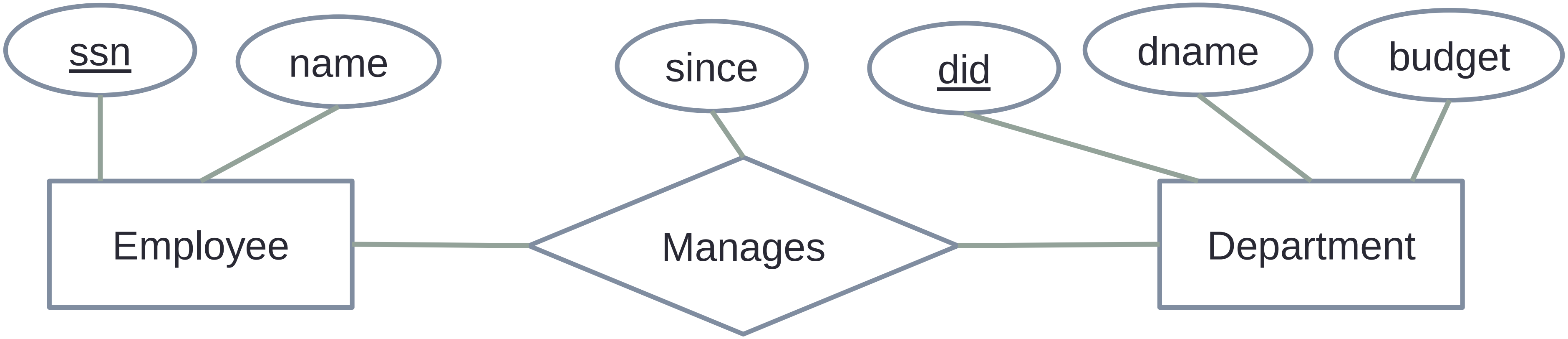
Attributes of a Relationship Type

- Relationship types also can have attributes

- E.g. to record number of hours per week that an employee works on a particular project. An attribute *Hours* could be included for the *Works_on* relationship type.

- E.g. to include the data which manager started to mange the department, an attribute *StartDate* for *Manages* relationship type

Relationship Attributes



Attributes of a Relationship Type

- Attributes for 1:1 relationship type can be migrated to one of the participating entity types
  - *E.g. StartDate* attribute of *Manages* relationship can be an attribute of either *Employee* or *Department*.
  - Since it is 1:1 relationship, each employee and each department participates in at most one relationship instance.
  - Conceptually it belongs to manages

- Attributes for 1:N relationship type can only be migrated to the entity type on the N-side of the relationship
  - E.g. each employee works for only one department.
  - StartDate attribute (indicating employee work start date) can be migrated to Employee entity type

- The decision where to place the relationship attribute (as entity type attribute or relationship type attribute) can be decided by the schema designer.

Attributes of a Relationship Type

- Attributes of M:N relationship type, cannot be migrated to participating entity types.
  - Some attributes are determined by combination of participating entities in a relationship instance.
  - E.g. Consider each employee working on multiple projects.
  - *Hours* attribute (indicating the number of hours an employee works on each project) of the *Works_on* relationship is determined by the combination of *Employee* and *Project*.
  - Such attributes must specified under relationship type.

Degree of Relationship

- It describes the number of entities involved in a relationship.
  - Unary (one entity)
  - Binary (two entities)
  - Ternary (three entities)
  - N’ary (more than 3 entities)

- Binary relationship is the most common relationship in ERD

Unary Relationship

- Unary (recursive): only 1 entity

- Exists when an association is maintained within a single entity.

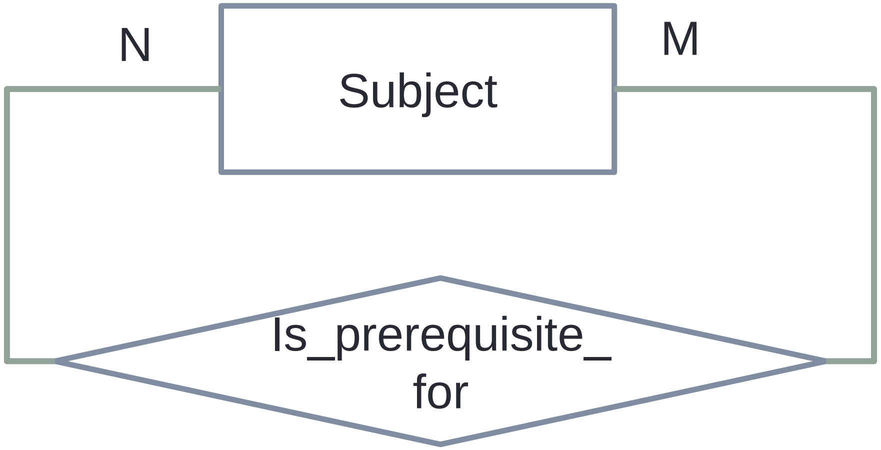
- If the same entity participates more than once in a relationship it is known as a recursive relationship.
  - Recursive relationships occur within unary relationships.

- The relationship may be one to one, one to many or many to many.

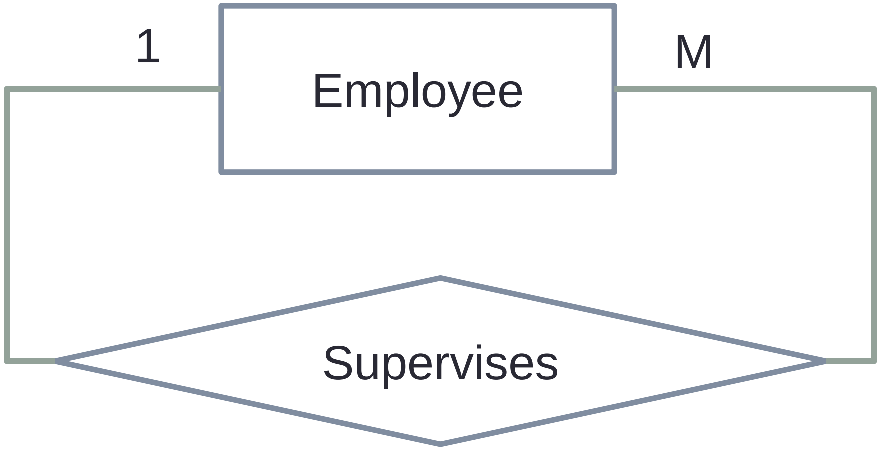
Recursive Relationship

- Example
  - Subjects may be prerequisites for other subjects.
  - A customer can refer multiple other customers
  - An employee can be a supervisor and be supervised

M:N unary relationship:
A Subject may have many other Subjects as prerequisites and each Subject may be a prerequisite to many other Subjects

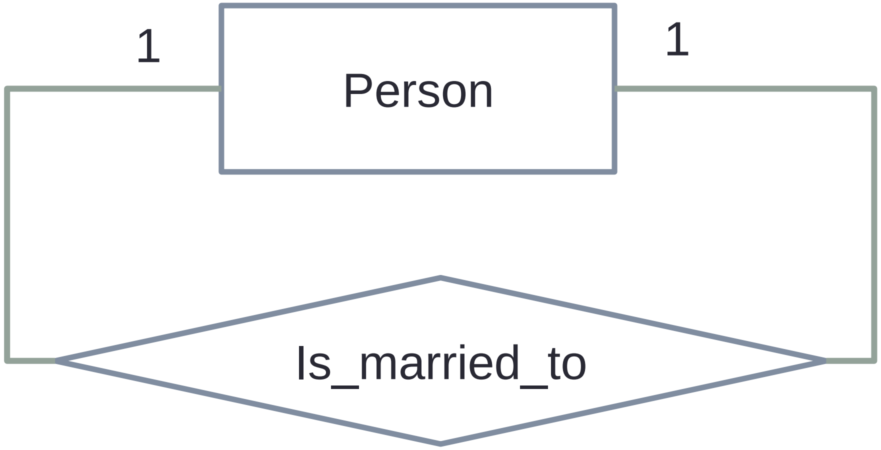


Recursive Relationship



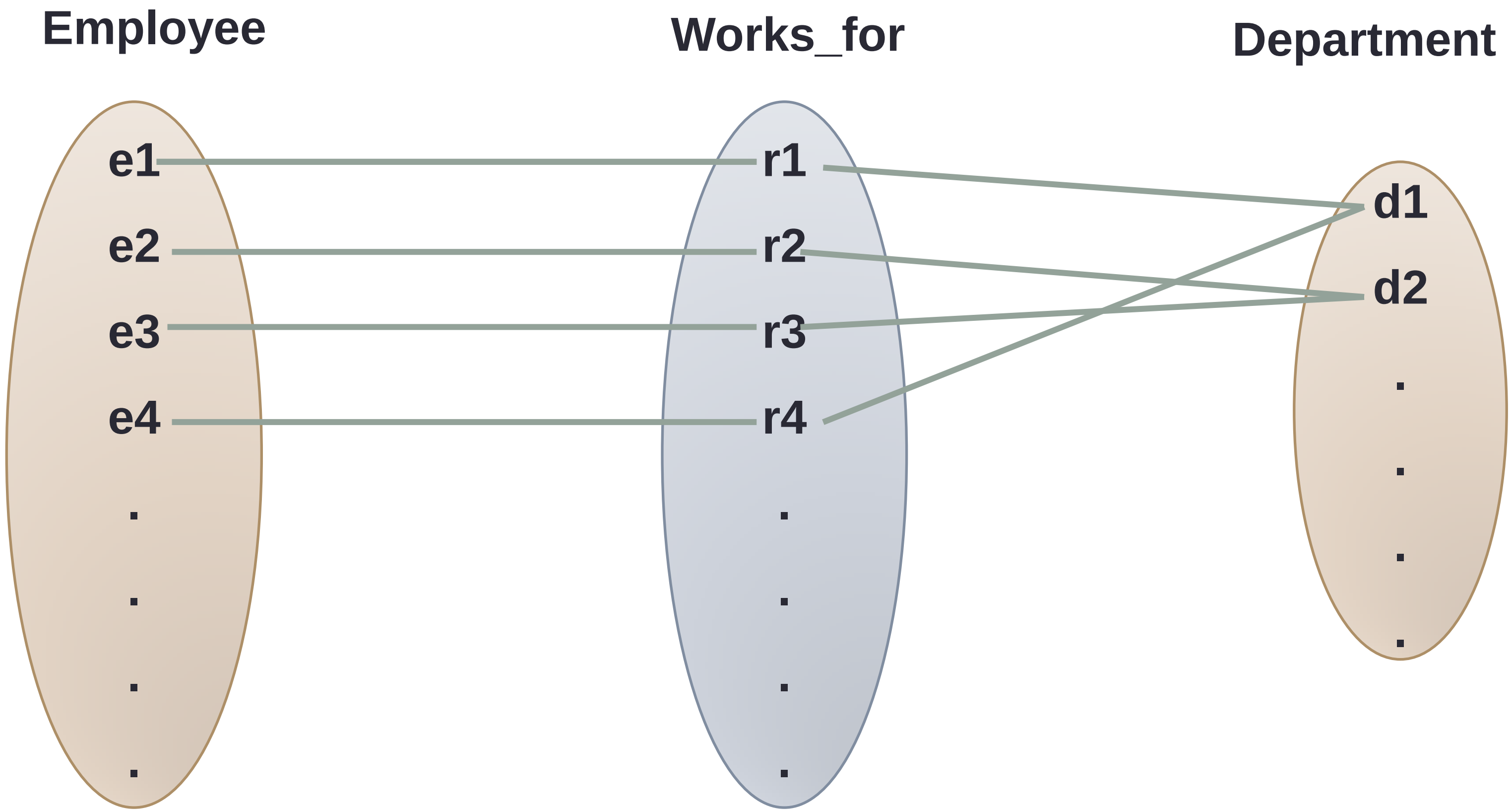
1:M unary Relationship
Employee may supervise many Employees, but an Employee is supervised by one Employee.

1:1 unary Relationship
A Person may be married to only one Person.



Binary Relationship

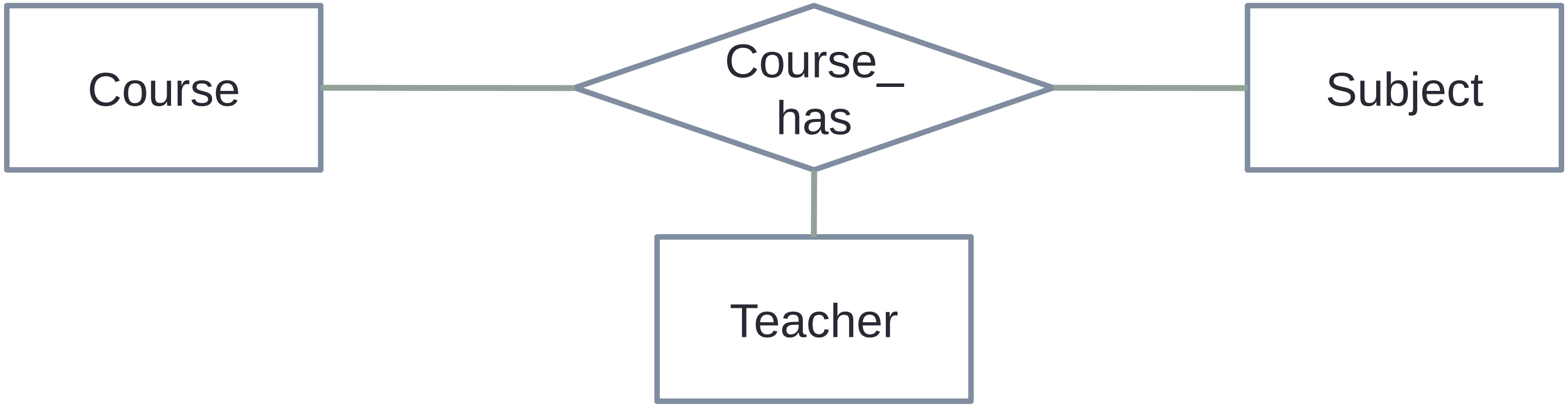
- A relationship type of degree two.

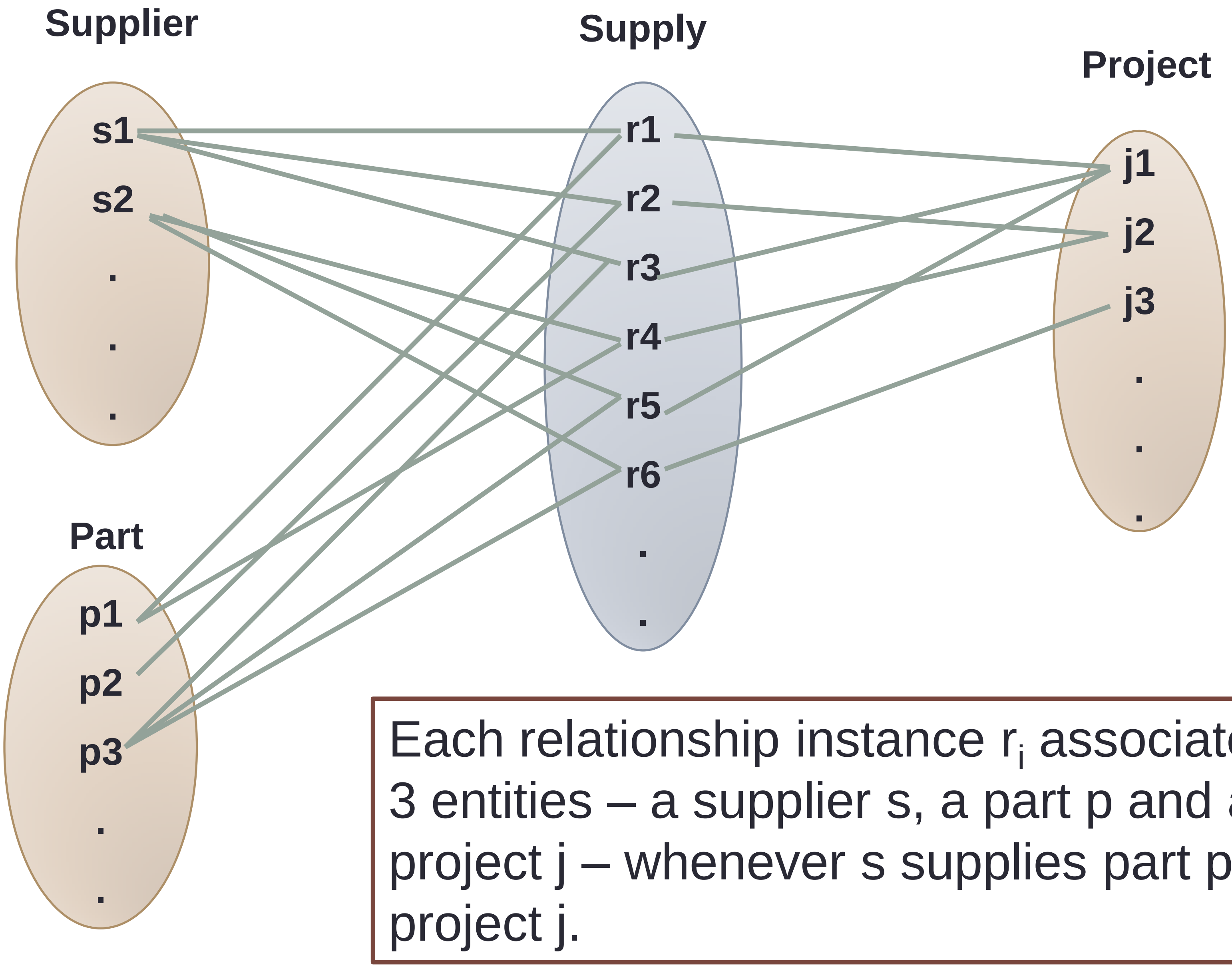


Ternary Relationship

- Ternary: 3 entities are required in this relationship

- E.g.
  - A doctor prescribes a drug to a patient
  - A supplier supplies a part to a project.
  - The University might need to record which teachers taught which subjects in which courses.



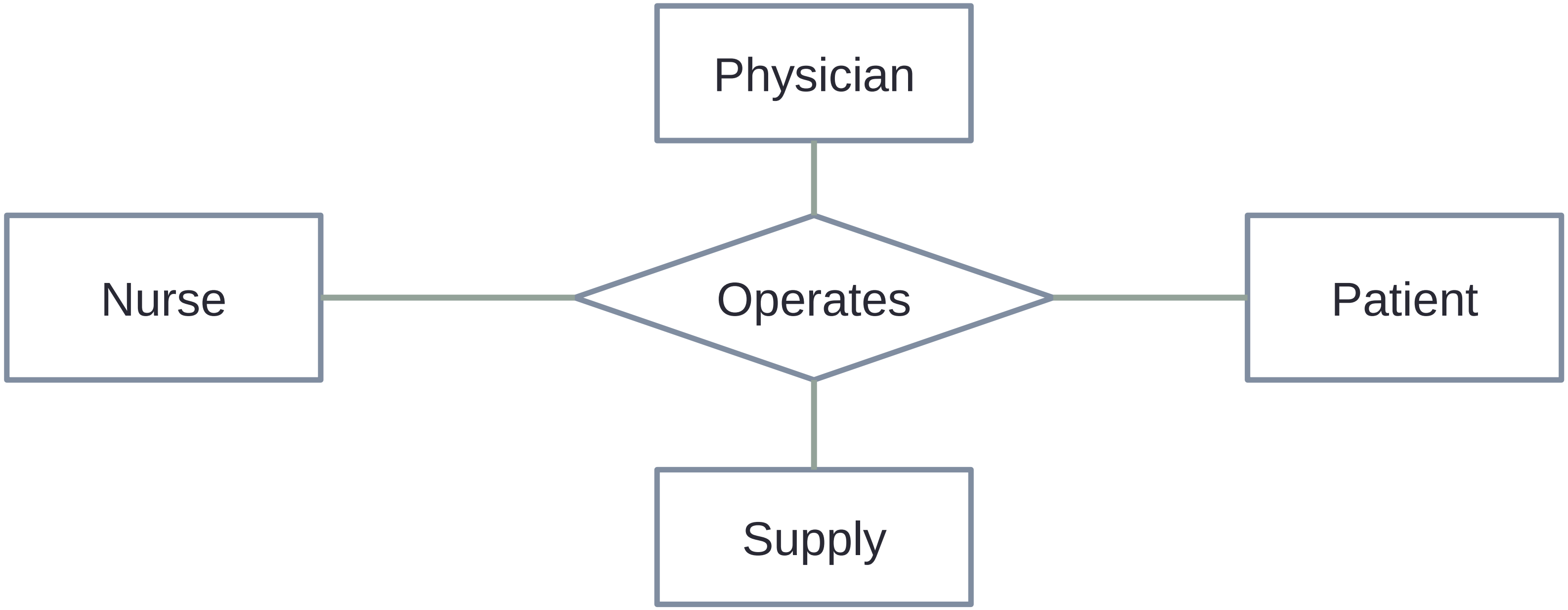


Each relationship instance r_i associates 3 entities – a supplier s, a part p and a project j – whenever s supplies part p to project j.

N’ary Relationship

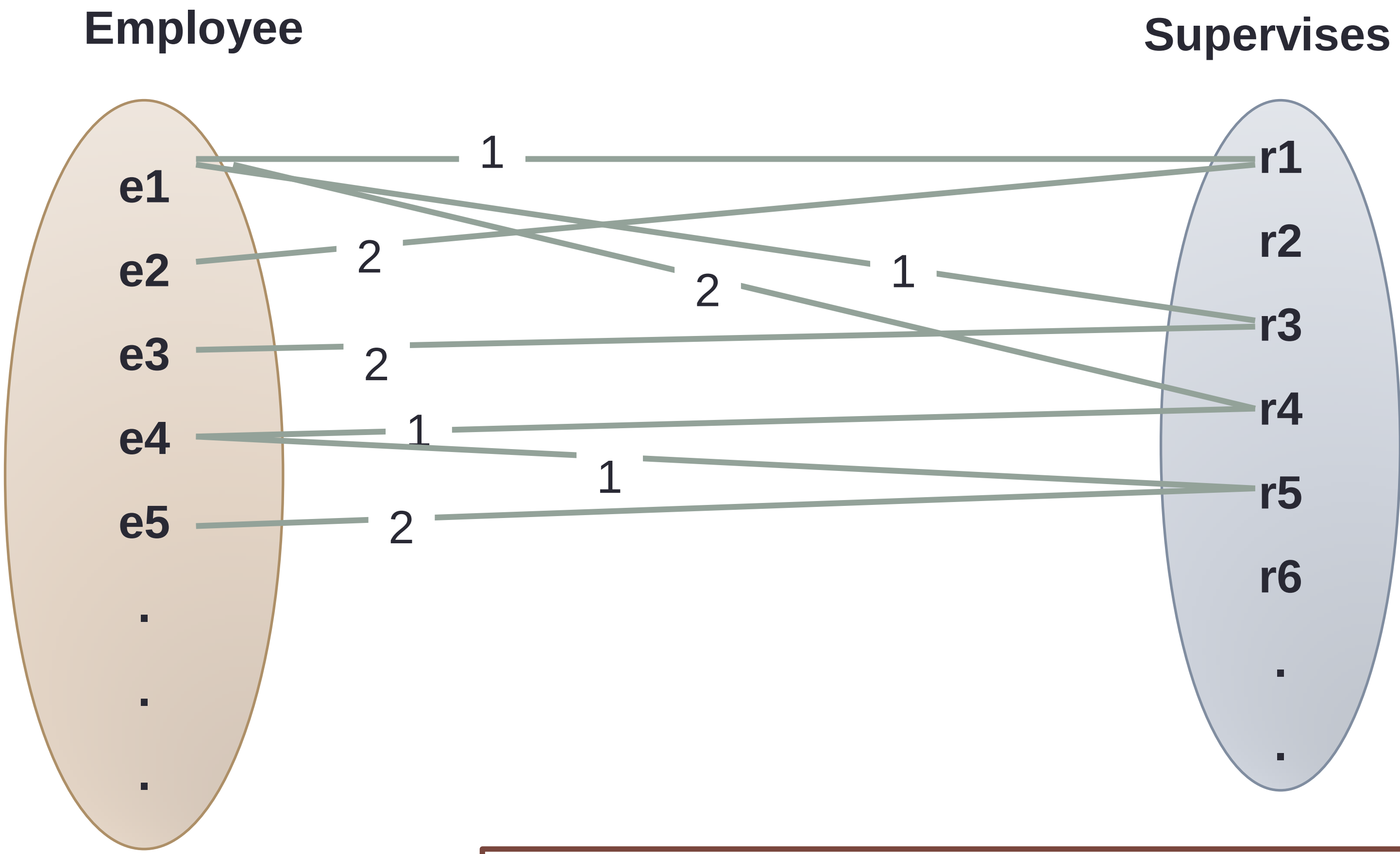
- More than 3 entities participate in a relationship

- E.g. With 4 entities
  - A physician operates on a patient, with certain nurses and supplies participating in this operation at the same time



Role Names

- Each entity type that participates in a relationship type plays a particular role in the relationship.
- The role name signifies the role that a particular entity from the entity type plays in each relationship instance.
- E.g. In the *Works_for* relationship type, Employee plays the role of employee or worker and Department plays the role of *department* or *employer*.
- Role names are not technically necessary in relationship types where all the participating entity types are distinct.
  - As each participating entity can be used as the role name
- Role names are important if the <u>entity type participates more than once in a relationship type in different roles</u>
  - Recursive relationship



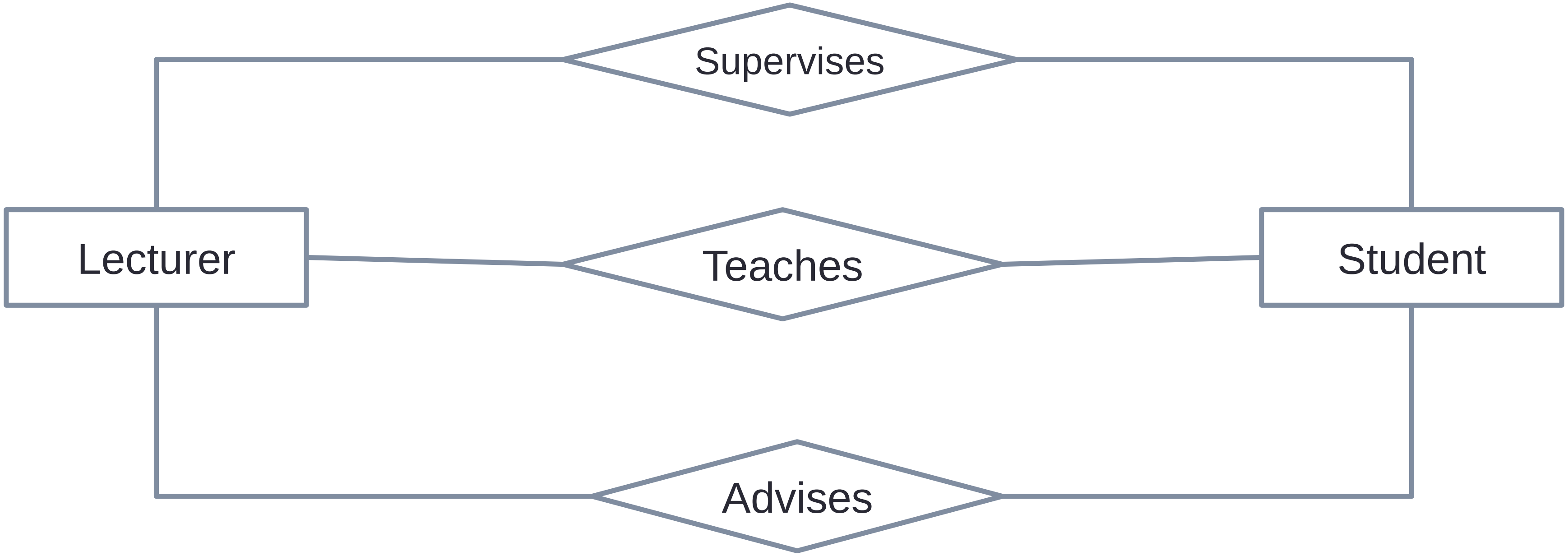
Employee in the supervisor role is 1 and Employee in subordinate role is 2

Relationship Modelling Considerations

- Multiple relationships
- Transitive relationship
- Attributes of relationships
- Promoting relationship to entity

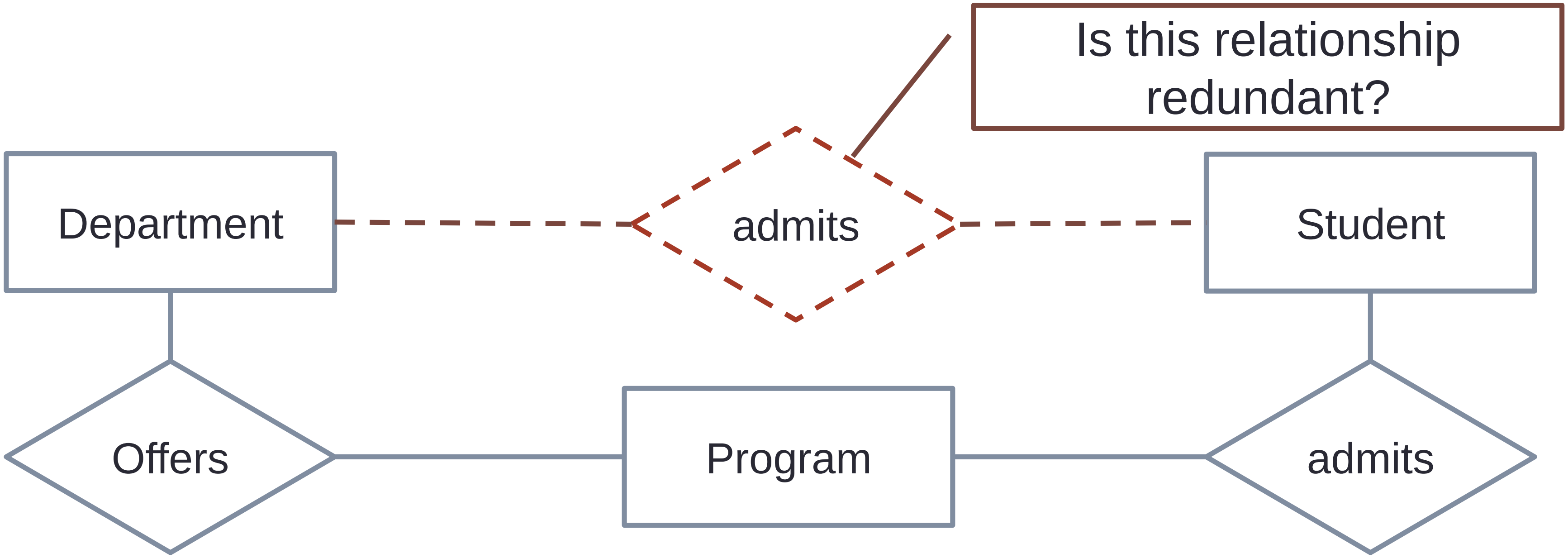
Multiple Relationship

- Multiple relationships can exists between entities, as long as they are independent or different.



Transitive Relationship

- Entities can be related indirectly by two relationships.
- A relationship is redundant if it can be completely represented by alternate transitive relationships



Is this relationship redundant?

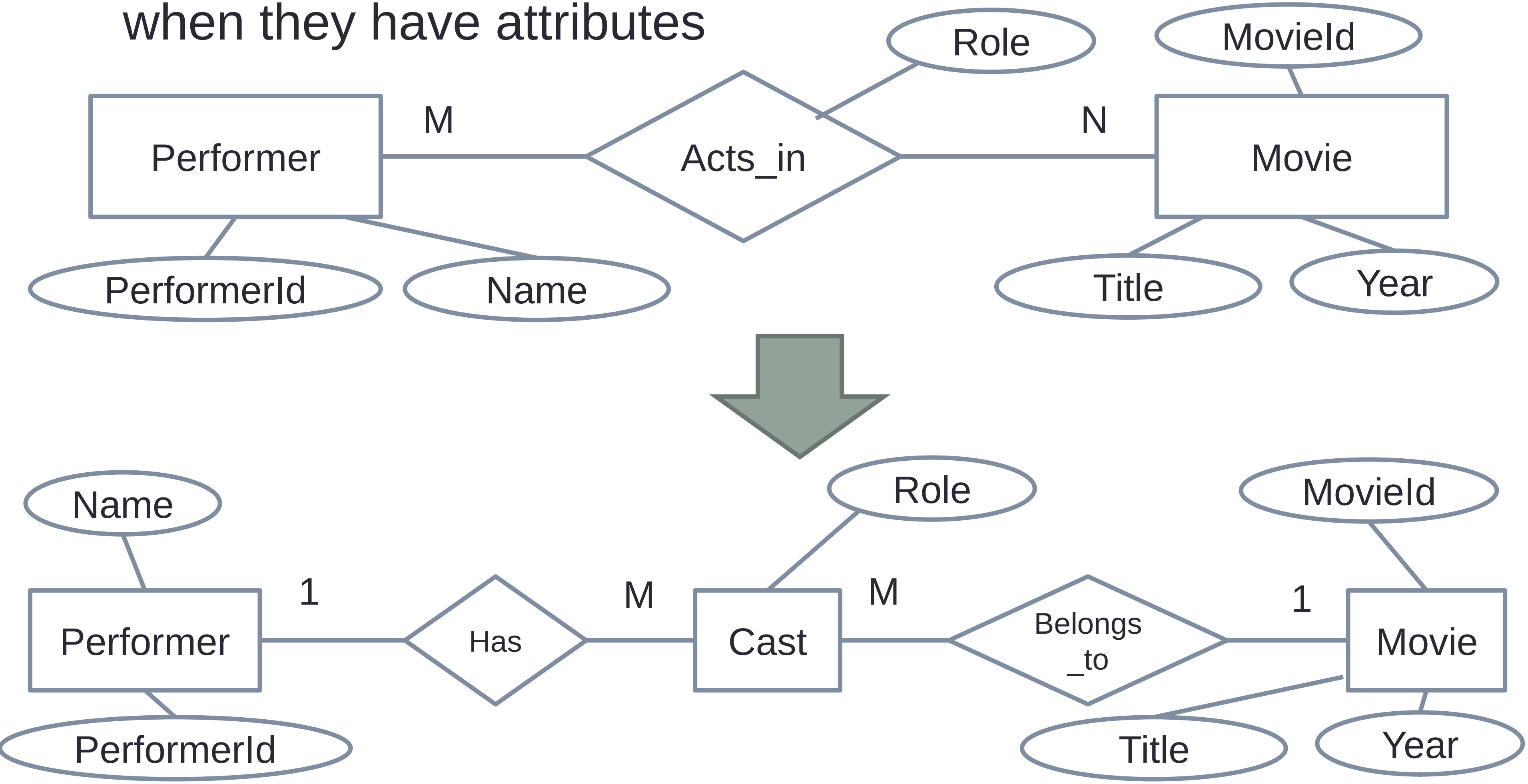
Can Department and Student be related indirectly through these two relationships?

One to one Relationship

- **Some** relationships between entities (e.g. A and B), **might** be redundant if
  - It is a 1:1 relationship between A and B
  - Every A is related to a B and every B is related to an A
- Example?

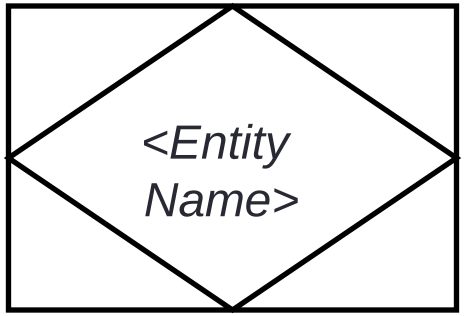
Relationship as an Entity

- Relationships can be modelled as entities, particularly when they have attributes



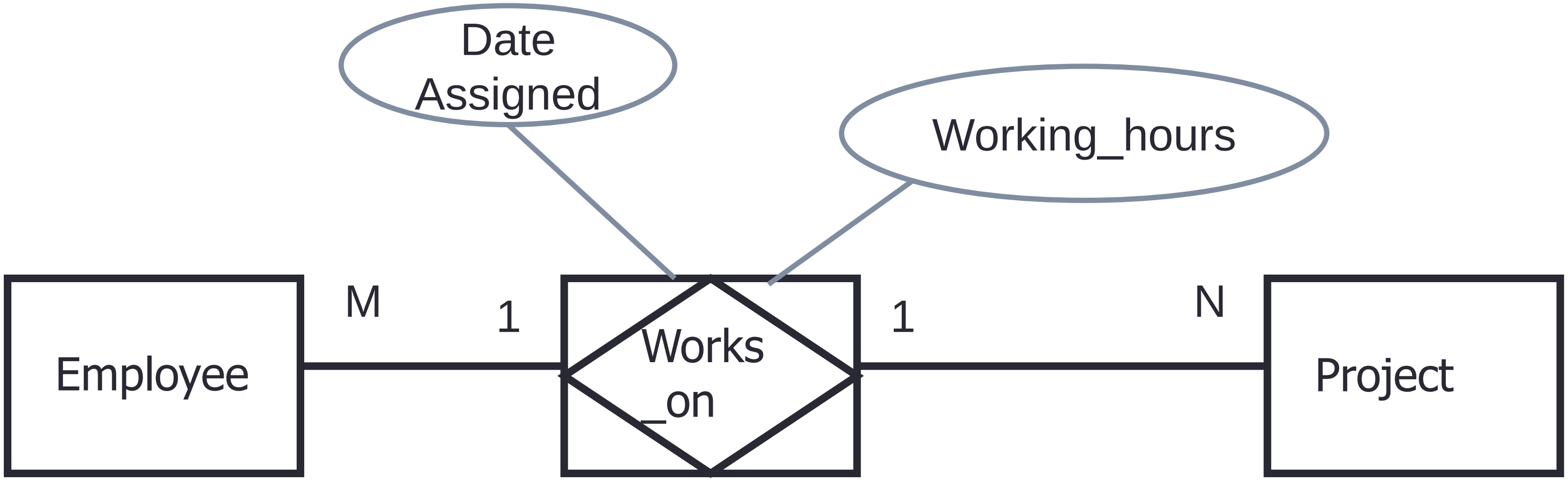
Associative Entities

- What happens when one or more attributes exist for a relationship?
  - Acts in relationship between Performer and Movie has Role attribute
- An associative entity is an entity type that associates the instances of more or more entity types and contains attributes that are specific to the relationship between those entity instances.



Converting a relationship to an associative entity

- Example
  - Employee’s working hours can vary by project (employee to project, many-to-many)



Weak Entity

- A weak entity depends on the existence of another entity.

- Weak entity is an entity type that, in addition to being <u>**existence dependent**</u>, has a primary key that has been totally or partially constructed from the entity it depends on.
  - A weak entity <u>cannot be identified by its own attributes</u>.

- A weak entity can be identified uniquely **only** by considering the primary key of another (*owner*) entity.
  - Owner entity type also known as Identifying entity type, dominant entity type, parent entity type

- If any entity has a key attribute, they are called as **strong entity types**

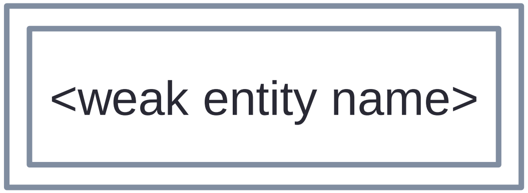
Weak Entity

- Usually a weak entity type has a **partial key**.
  - Partial key is set of attributes that can uniquely identify weak entities that are related to the same owner entity.

- It uses a Owner’s key (known as foreign key) combined with its attributed to form the primary key.

- E.g. The order item will be meaningless without an order so it depends on the existence of order.

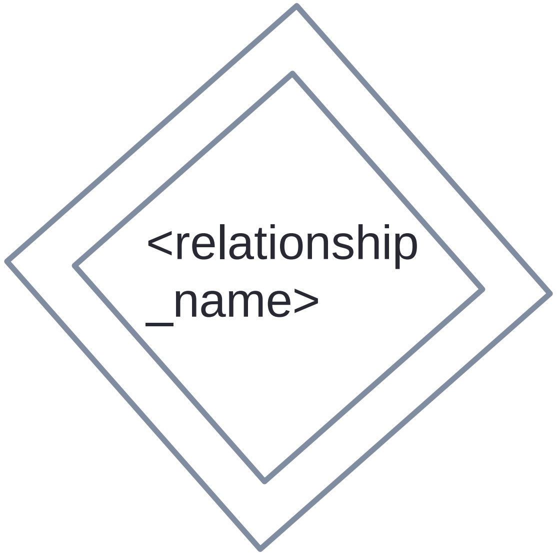
Notation



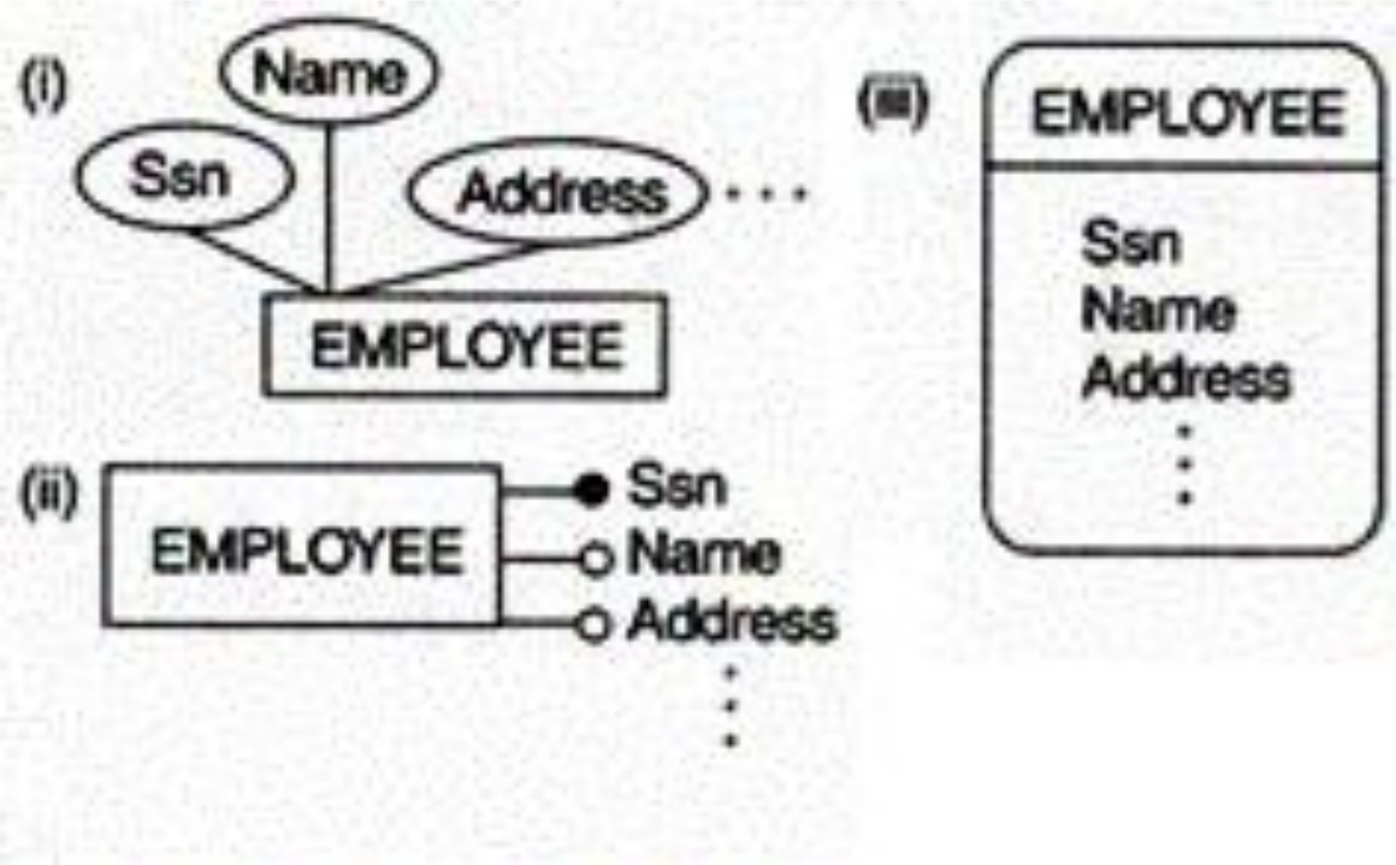
Identifying Relationship

- Relationship type that relates a weak entity type to its owner.
- A weak entity has total participation with respect to its identifying relationship.
  - As weak entity cannot be identified without an owner entity

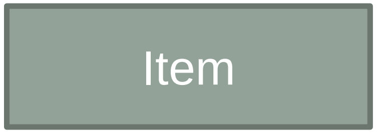
Notation



Alternative Notation to Display Attributes

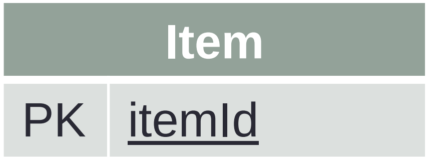


Level of Entity Attribute Display



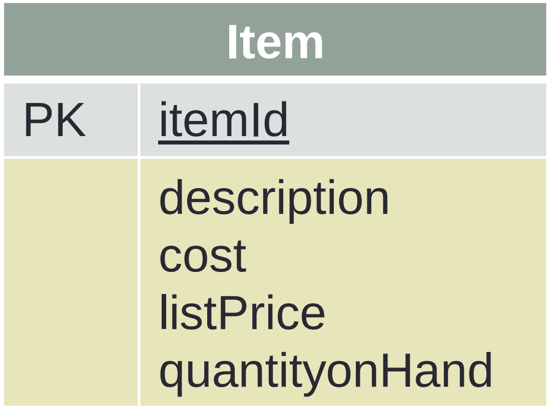
Entity with no attributes

In conceptual design, just to show their relationships



Entity showing only key attributes

In conceptual design, We can see what attributes involved in the relationship



Entity showing all attributes

How to draw ER diagrams?

- Identify all the relevant entities in a given system and determine the relationships among these entities.
- An entity should appear only once in a particular diagram.
- Provide a precise and appropriate name for each entity, attribute, and relationship in the diagram.
- Remove vague, redundant or unnecessary relationships between entities.
- Never connect a relationship to another relationship.

Benefits of ER Diagrams

- ER diagrams are easy to understand and do not require a person to undergo extensive training to be able to work with it efficiently and accurately.
  - This means that designers can use ER diagrams to easily communicate with developers, customers, and end users, regardless of their IT proficiency.
- ER diagrams are readily translatable into relational tables which can be used to quickly build databases.
- ER diagrams can directly be used by database developers as the blueprint for implementing databases in specific software applications.
- ER diagrams may be applied in other contexts such as describing the different relationships and operations within an organization.