

# Unit 2: Entity-Relationship Model

Dr. Ahmed E. Khaled

Department of Computer Science



MPCS 53001- Databases

- What is an ER model?
- Entity vs Entity set
- The different types of attributes
- Primary Key vs Candidate Key
- Relationship vs Relationship set
- Relationship Degree
- Mapping Cardinalities
  - 1:1, 1:N, N:1, N:M
  - (Min : Max) Annotation
- Strong Entity set vs Weak Entity set
- Total vs Partial Participation

Step 1- *Requirements Analysis*

Step 2- *Conceptual Database Design* (The focus of this Unit)

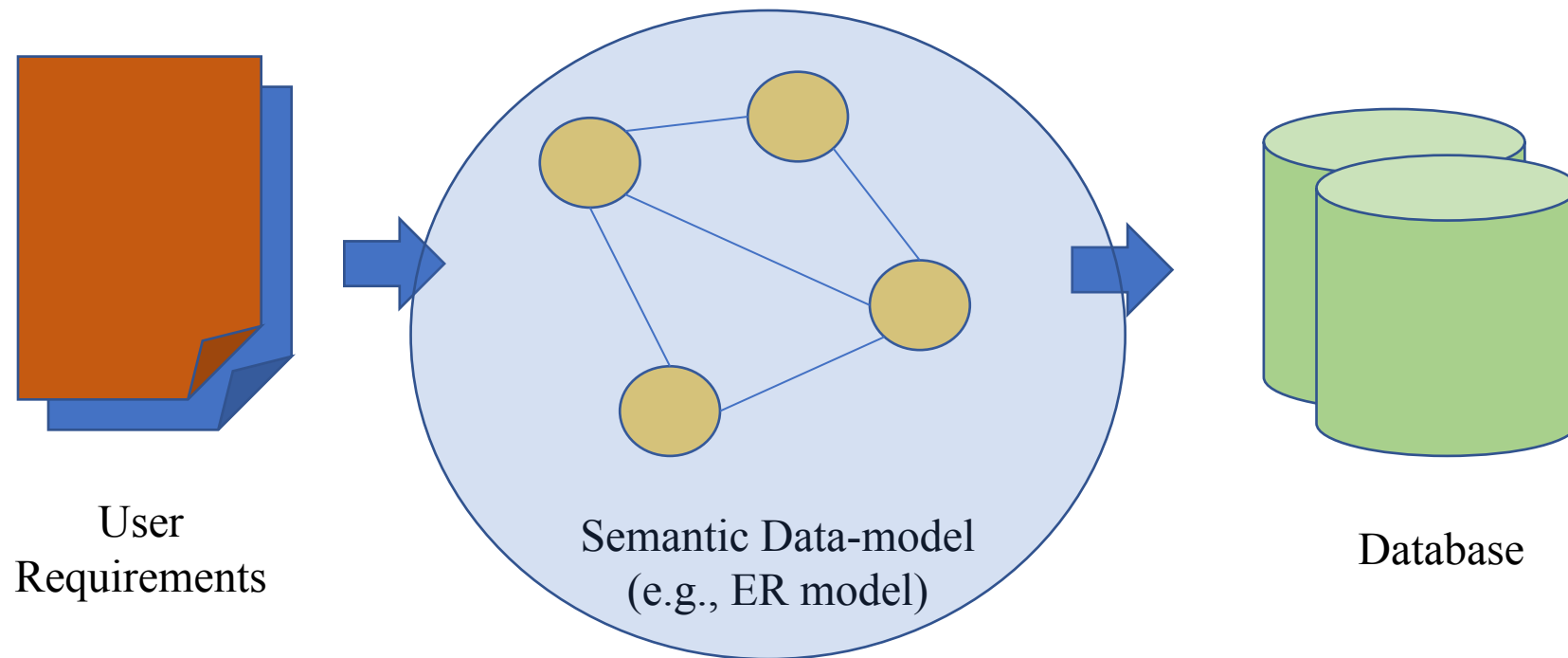
Step 3- *Logical Database Design*

Step 4- *Schema Refinement*

Step 5- *Physical Database Design*

Step 6- *Application Design*

- ER (Entity-relationship model/diagram) provides a way to move from an *informal description* of what users want (step 1 – requirement analysis) to a *more detailed description* that can be implemented in a DBMS (step 3 – logical design).
- ER is a semantic data-model used to develop an *initial design* for a database.



- The ER model allows designers to *describe* users' requirements and constraints using a set of high-level abstractions and structures.
- Such description of data is in terms of objects (referred to as *entities*), *attributes* to describe the entities, and links (referred to as *relationships*) between the different entities.

The **Entity-Relationship (ER) diagram** was invented by **Peter Chen** in **1976**, as a way to visually represent the structure of databases by focusing on entities (objects) and their relationships, which became a foundational tool in database design.

He introduced the concept in his seminal paper titled "*The Entity-Relationship Model—Toward a Unified View of Data.*"

# *Entity and Entity Set*

An entity is an *object in the real world*, either animate or inanimate, that can be identifiable and distinguishable from other objects.

- ✓ Taking a university database as an example, we have: Student, Faculty, Course, Exam, Classroom, ....
- ✓ Taking a company database as an example, we have: Employee, Departments, Offices, ....

However, since we would like to build a *high-level design*, it is often useful to identify a collection of *similar entities*, that is called an *entity set*.

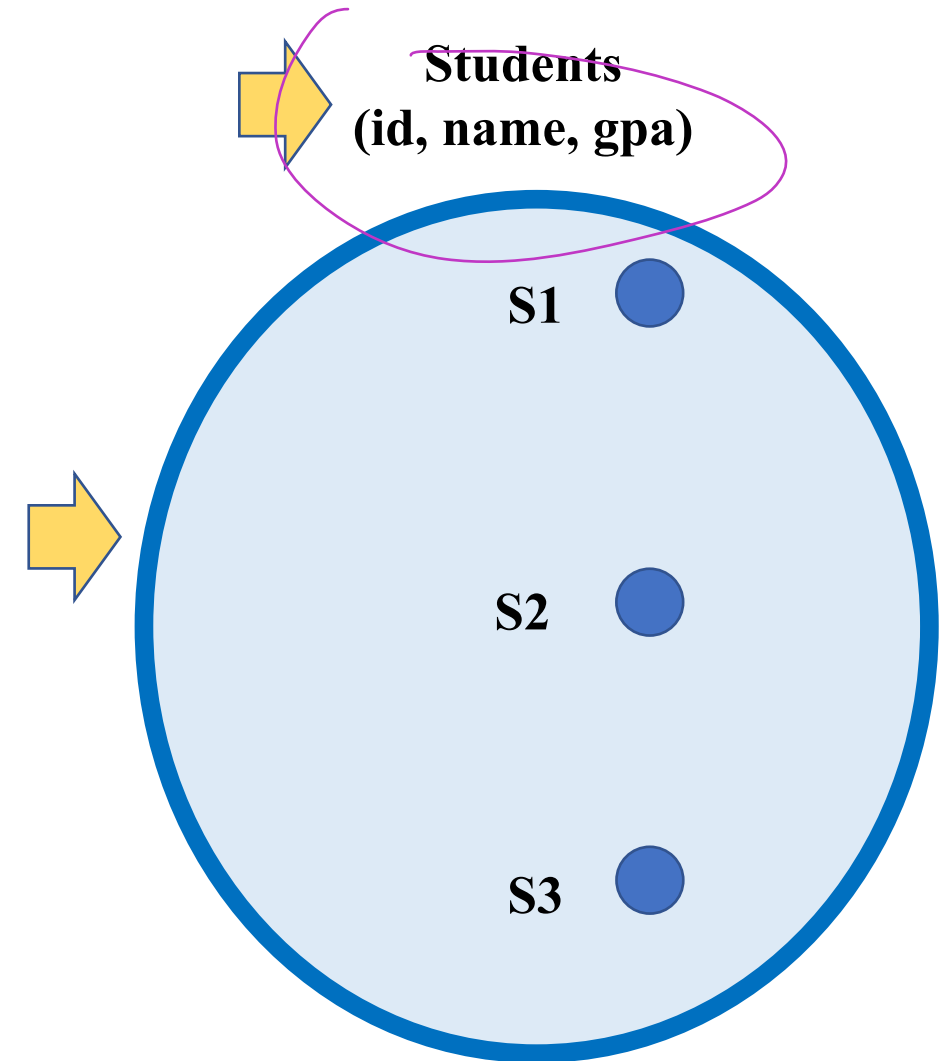
- Students entity set contains all the students of the school.
- Employees entity set contains all the employees of the company.

An ER model *defines and uses entity sets*, not individual entities  
(we would like to represent all students, not one specific student)

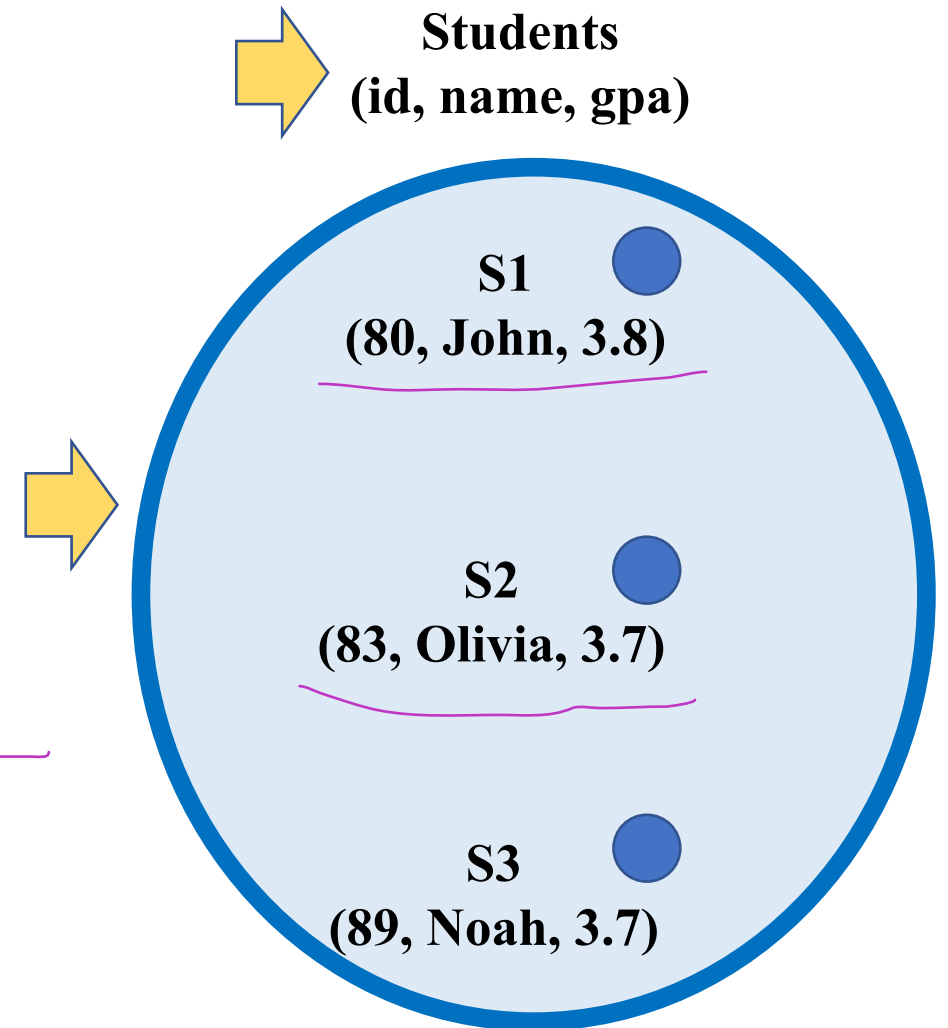


- An entity is described using a set of *attributes* and all entities in an entity set usually have the *same (common) attributes*.
- For example, each student (entity) in the set of students (entity set) can be represented through the following attributes:
  - ✓ Name
  - ✓ ID
  - ✓ GPA
  - ✓ Age
- The choice of attributes reflects the *level of details and information* we would like to capture and represent about entities.

- This Student entity set is represented with (Id, Name, GPA) as attributes, and we have 3 individual students (entities).



- This Student entity set is represented with (Id, Name, GPA) as attributes, and we have 3 individual students (entities).
- An *instance* of an entity set can be thought of as a snapshot of the set of entities at some instant in time.
- Each student (entity) has a value for each of these attributes (e.g., Student1 has id as 80, name as John, and 3.8 as GPA)



- For each attribute associated with an entity set, we should (when possible) identify a *domain of possible values*.
- For example:
  - ✓ The domain of attribute *Name* might be a set of *30-character string*.
  - ✓ The domain of attribute *Id* might be an *integer value* with a value between 1000 to 5999.
  - ✓ The domain of attribute *GPA* might be a *real value* with a value between 0.0 to 4.0.

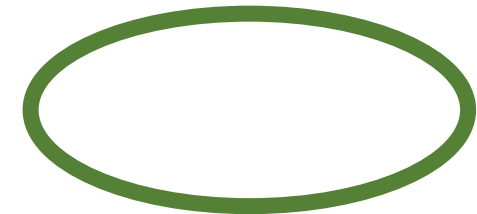
- The common abstract data types for attributes in an ER are:
  - ***Integer***: for attributes that store whole numbers (e.g., age, quantity, ID).
  - ***Decimal***: for attributes that store decimal numbers (e.g., salary, price, weight, balance)
  - ***String***: for attributes that stores one or an array of characters (e.g., name, address).
  - ***Text***: for attributes that store long-form text (e.g., description, comments).
  - ***Date***: for attributes that store data in the format YYYY-MM-DD (e.g., birth\_date, order\_date).
  - ***Time***: for attributes that store time values (e.g., start\_time, end\_time).
  - ***Boolean***: for attributes that store binary like true/false or yes/no values (e.g., is\_active, is\_verified).

## *Designing an ER Diagram – Step 1 (Entity Sets and Attributes)*

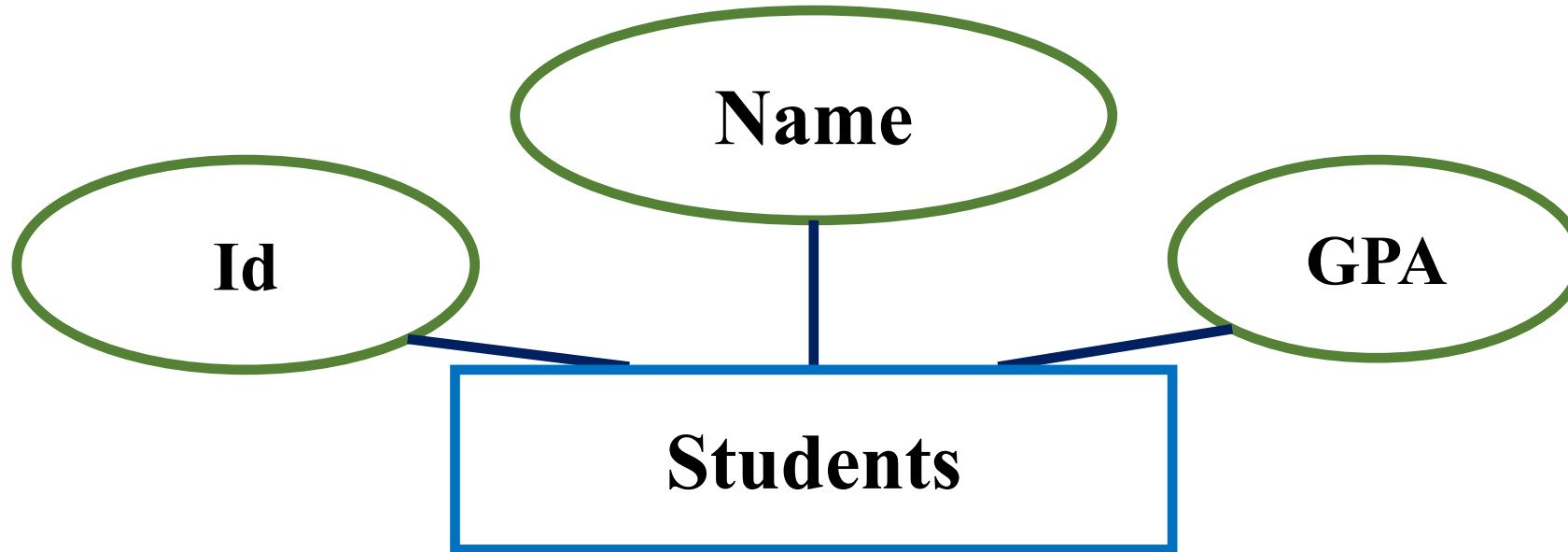
- An *entity set* is represented by a rectangle:



- An *attribute* is represented by an oval:

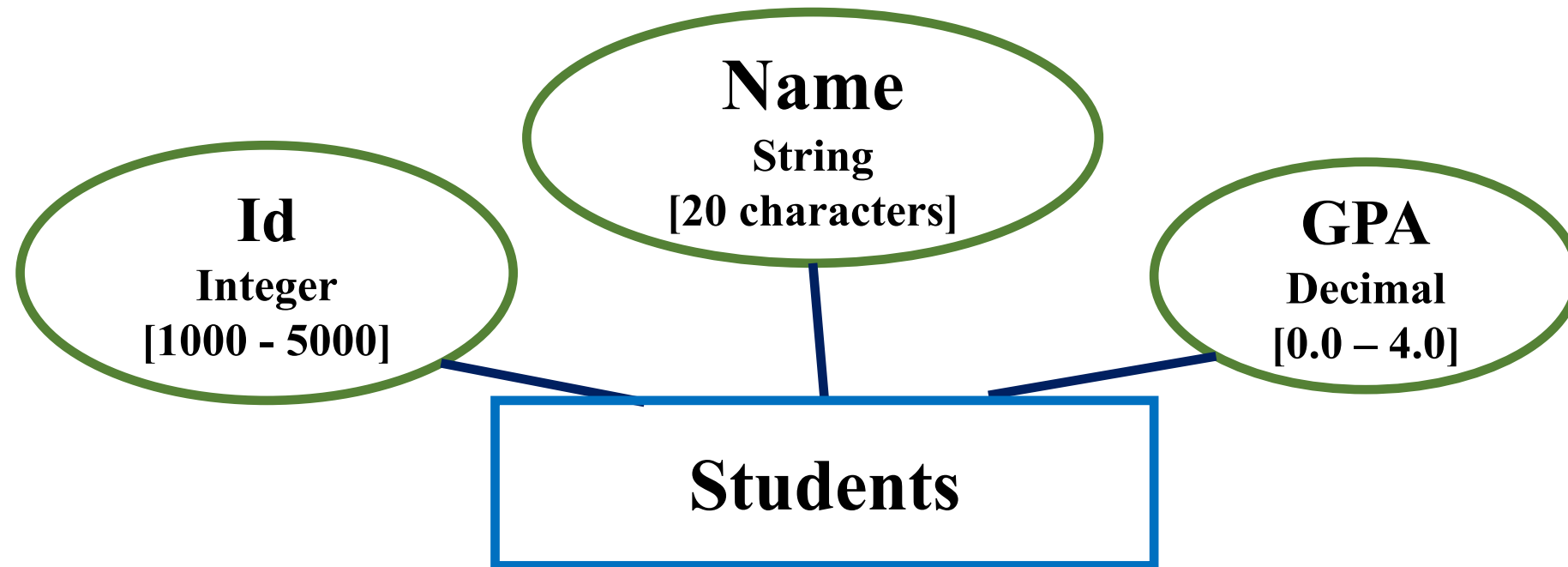


Take Students' *entity set* with *attributes* Id, Name, and GPA as an example, we can build the following ER that conceptually represents such entity set:





Take Students' *entity set* with *attributes* Id, Name, and GPA as an example, we can build the following ER that conceptually represents such entity set:

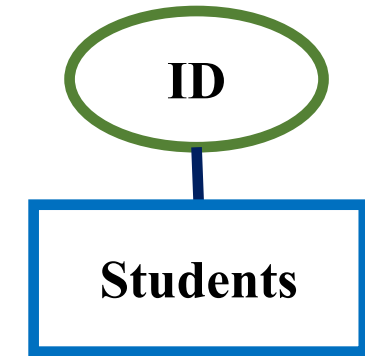


- The *domain information* is listed along with the attribute name.
- We will *omit* the domain info from the rest of the slides to keep the figures compact and readable.

# *The different types/categories of Attributes*

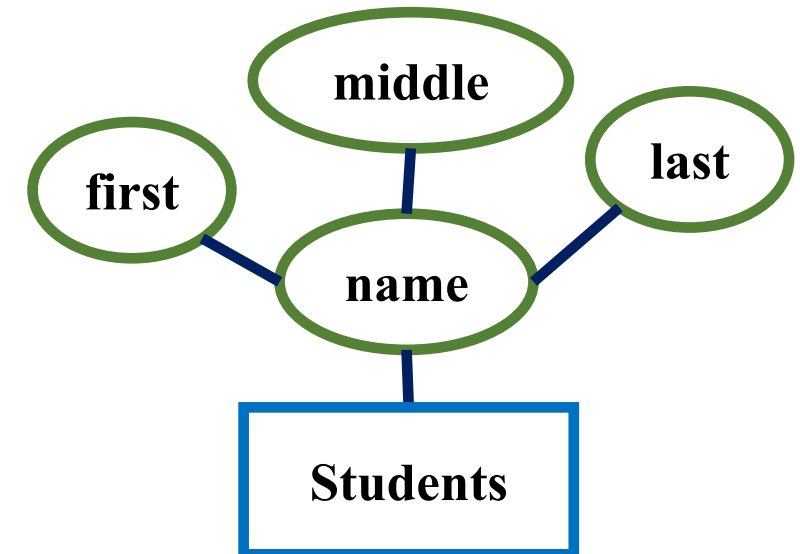
Type 1: ***Simple attribute (or atomic attribute):***

- ✓ An attribute of a ***single component*** with an independent existence.
- ✓ An attribute ***cannot be further subdivided*** into smaller components.
- ✓ For example, ***a student's id is an atomic value of 8 digits.***



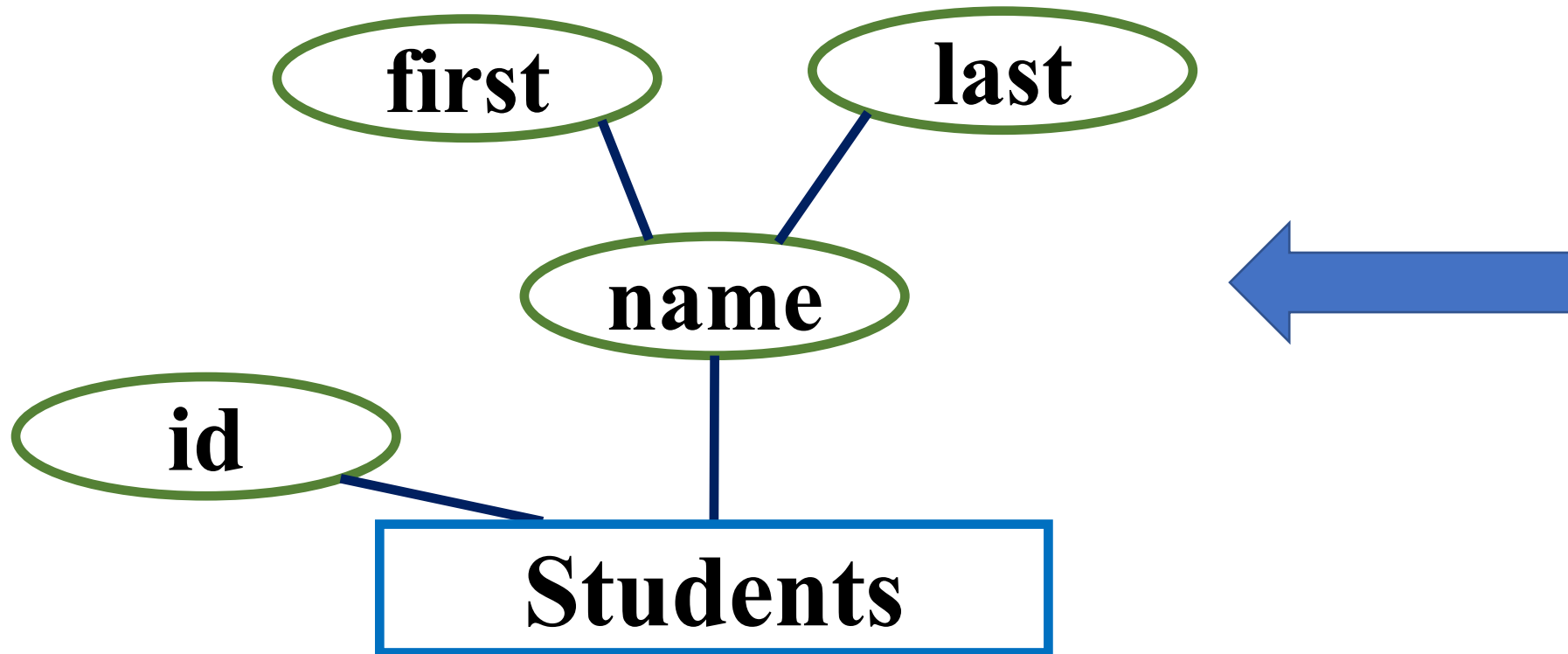
Type 2: ***Composite attribute***

- ✓ An attribute made of more than ***one simple attributes***, divided in a tree like structure.



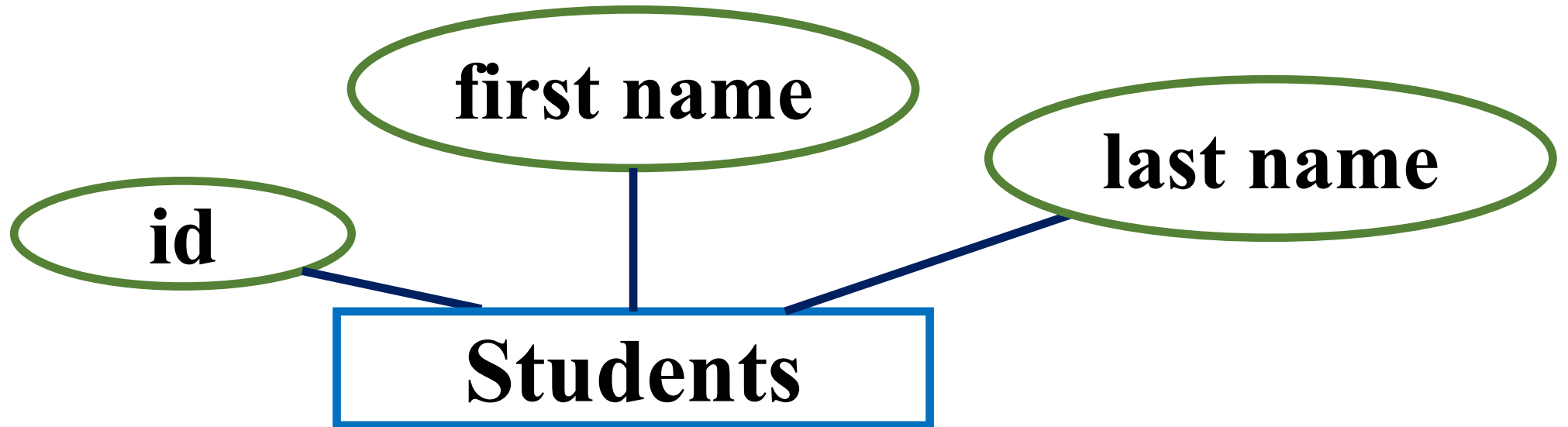
### *Composite attribute*

- ✓ Should student's name be a composite attribute? – Yes, if you would like to represent the first and last names *under the attribute name*



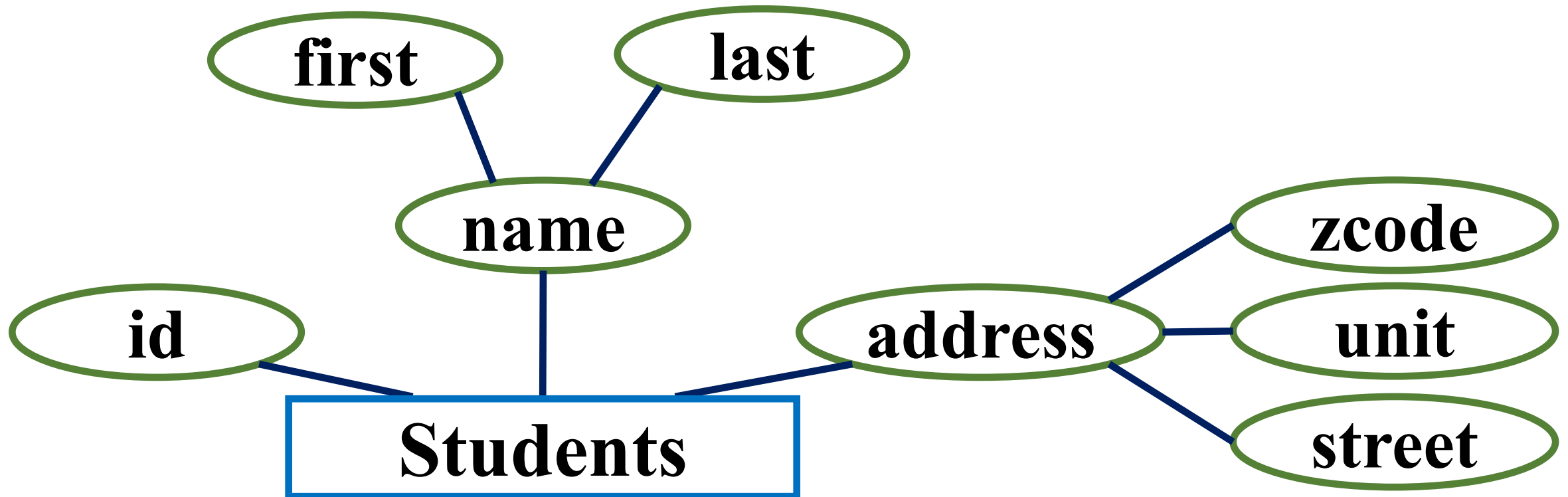
### *Composite attribute*

- ✓ Should student's name be a composite attribute? – No, if you would like to represent the first and last names *as simple attributes*



## *Composite attribute*

- ✓ Can student's address be a composite attribute? – Yes, if you would like to represent zip code, unit and street *under the address attribute*



### Type3: *Single-Value attribute*

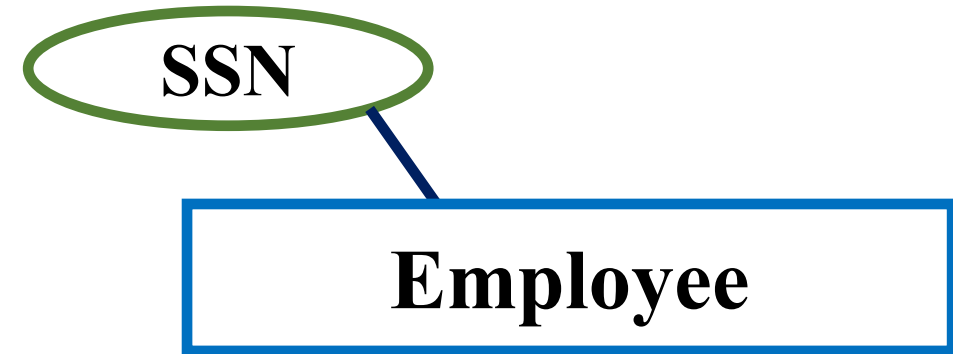
✓ An attribute that holds a maximum of one/single value.

✓ For example:

☐ Employee's SSN

☐ Student's ID

☐ Student's GPA

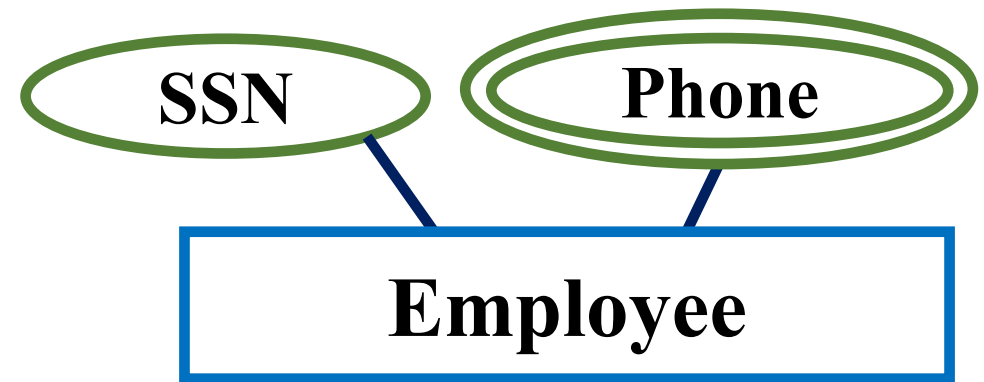


### Type 4: *Multi-Value attribute*

- ✓ An attribute that may carry *more than one value*, in this case the attribute is represented by a *double oval*.



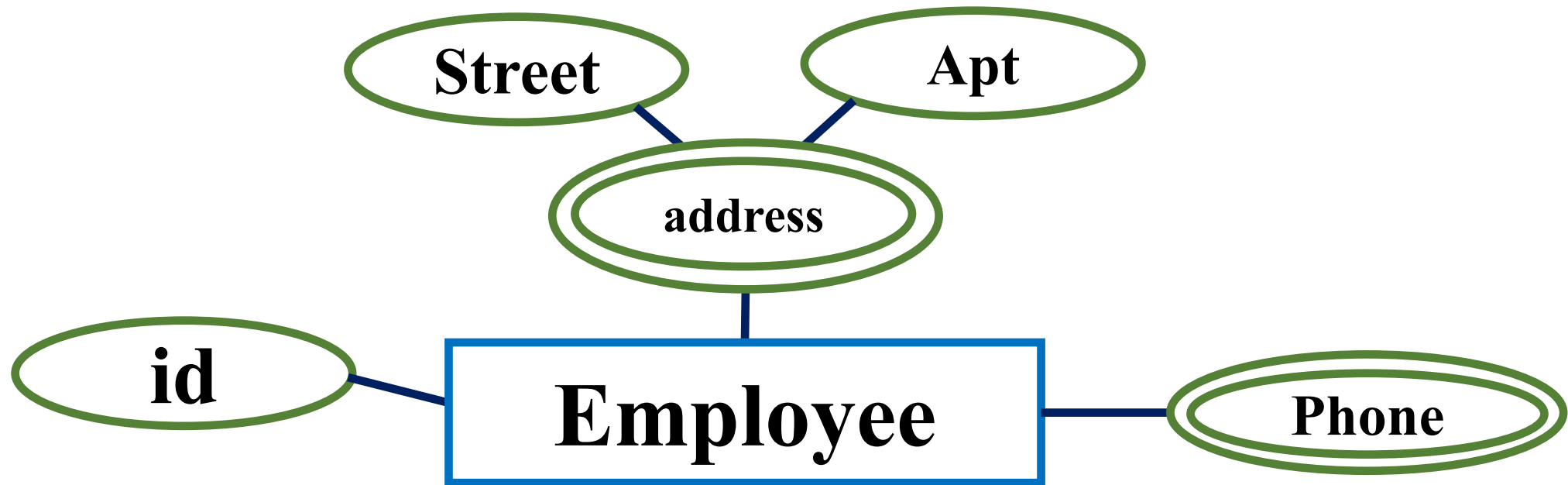
- ✓ For example: an employee can have *more than one phone number or email address*.





### *Multi-Value attribute*

- ✓ We can have an atomic or composite attribute, that is multi-valued as well – yes!

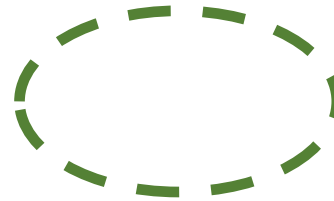


### Type 5: *Derived attribute*

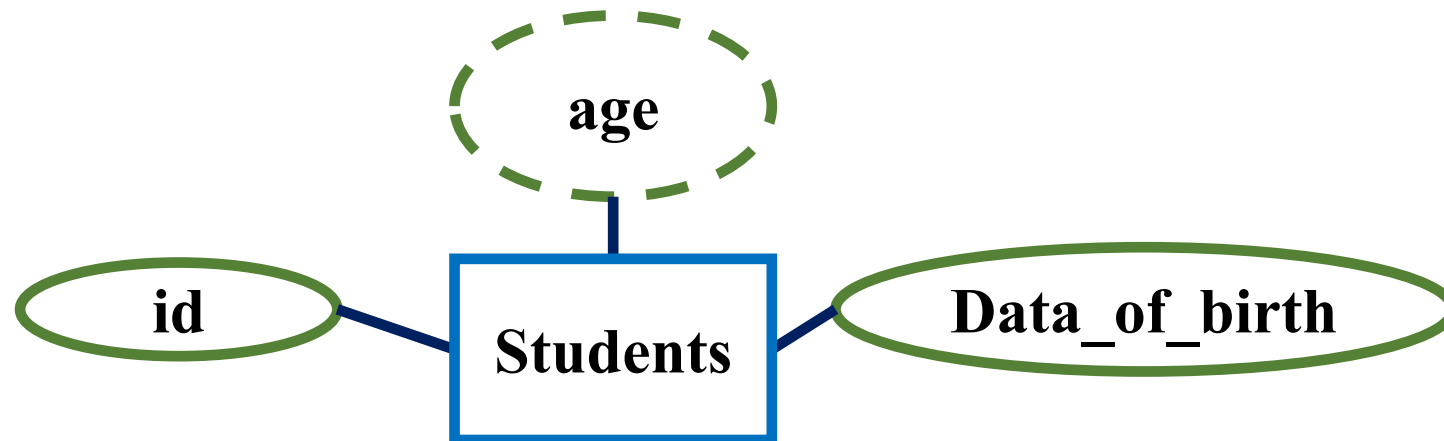
- ✓ Attributes that do not exist in the physical data set, but their values are *derived from other attributes* present in the data set.
- ✓ For example:
  - *average\_salary* can be derived from salary and number of employees.
  - *age* can be derived from data of birth.

### Type 5: *Derived attribute*

✓ This attribute is represented by a dotted oval.

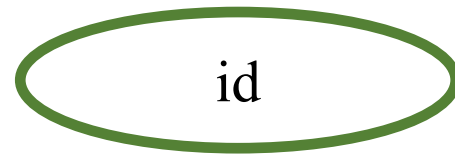


✓ The age of the student can be derived (computed) from his data\_of\_birth (another attribute).



As a summary - any attribute can be:

- Atomic single valued



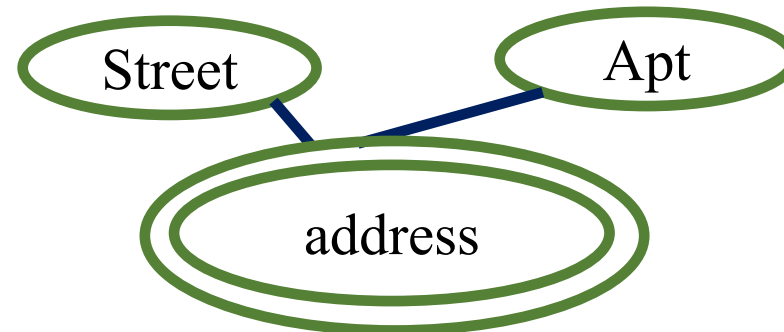
or Atomic Multi-valued



- Composite Single-valued



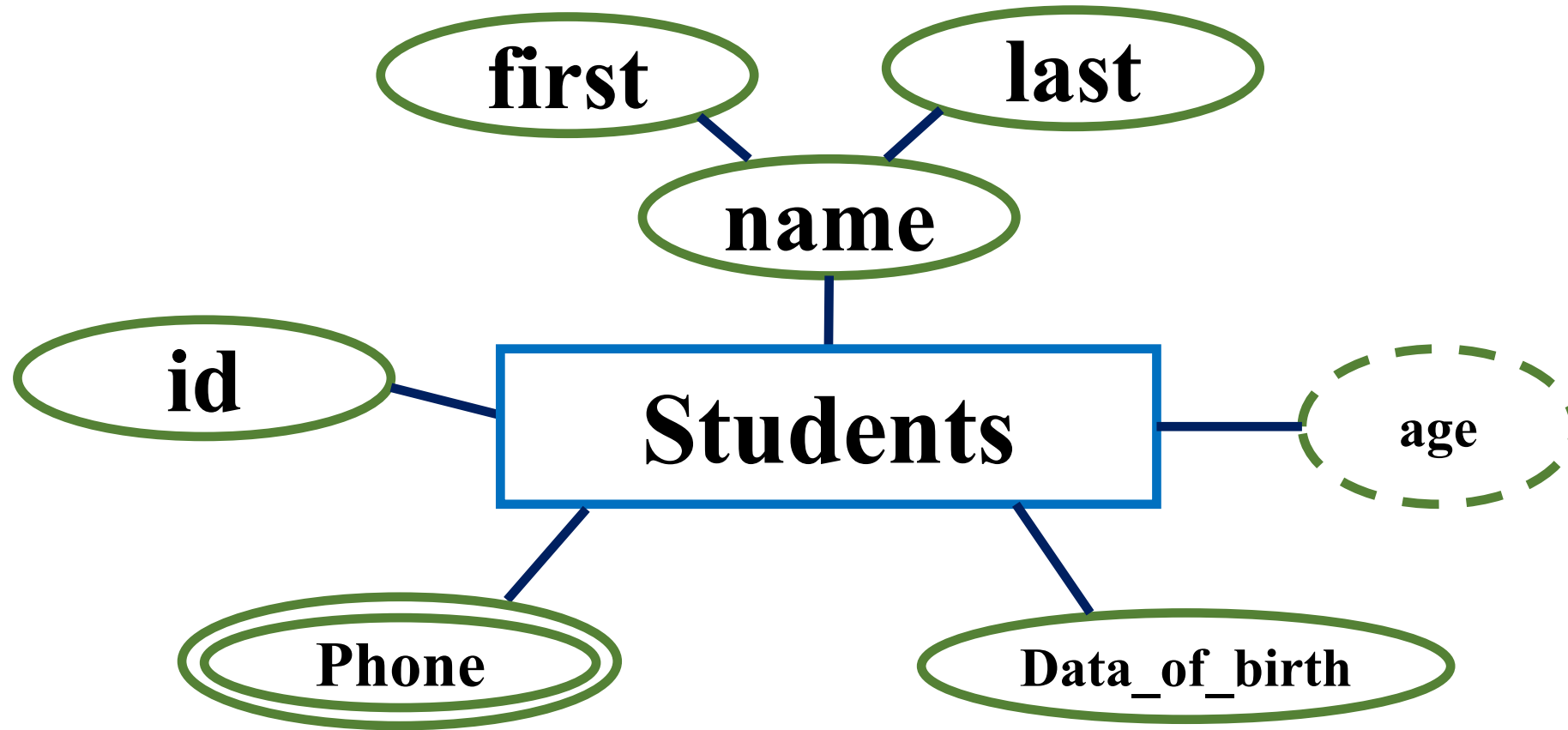
- Composite Multi-valued



- Derived



Put them all together in one diagram:



# *Primary key vs Candidate Key*

- As an entity set is composed of many entities, we need certain attribute or a set of attributes to *uniquely identify the individual entities within the entity set* - we call this the “*key*” of the entity set
- Certain worker within a set of workers can be uniquely identified given the worker’s SSN (no two workers have the same SSN)
- Certain worker within a set of workers cannot be uniquely identified given the worker’s first name (more than one worker can have the same first name)
- A key attribute cannot be *null* (empty or with no value).

### *Candidate Key:*

- The *minimal number of attributes* whose value(s) uniquely identify the individual entity.
- This implies that a candidate key cannot contain a *null* value.
- We can have *more than one candidate key* per entity set.



### *Candidate Key*

✓ From the following attributes, what are the possible candidate keys for an employees' entity set?

- ☐ Social Security Number (ssn)
- ☐ Id
- ☐ Name and Id
- ☐ Id and SSN
- ☐ Id and Department Id
- ☐ Office Id and Department Id

## *Candidate Key*

✓ From the following attributes, what are the possible candidate keys for an employees' entity set?

- ☐ Social Security Number (ssn) ←
- ☐ Id ←
- ☐ Name and Id
- ☐ Id and SSN
- ☐ Id and Department Id
- ☐ Office Id and Department Id ←

We can use:

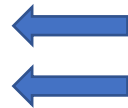
- The SSN as the key *or*
- The id as the key *or*
- Both office and department Ids as the key

These are 3 candidate keys for the employee entity set.

## *Candidate Key*

✓ From the following attributes, what are the possible candidate keys for an employees' entity set?

- ☐ Social Security Number (ssn)
- ☐ Id
- ☐ Name and Id
- ☐ Id and SSN
- ☐ Id and Department Id
- ☐ Office Id and Department Id



We can't use:

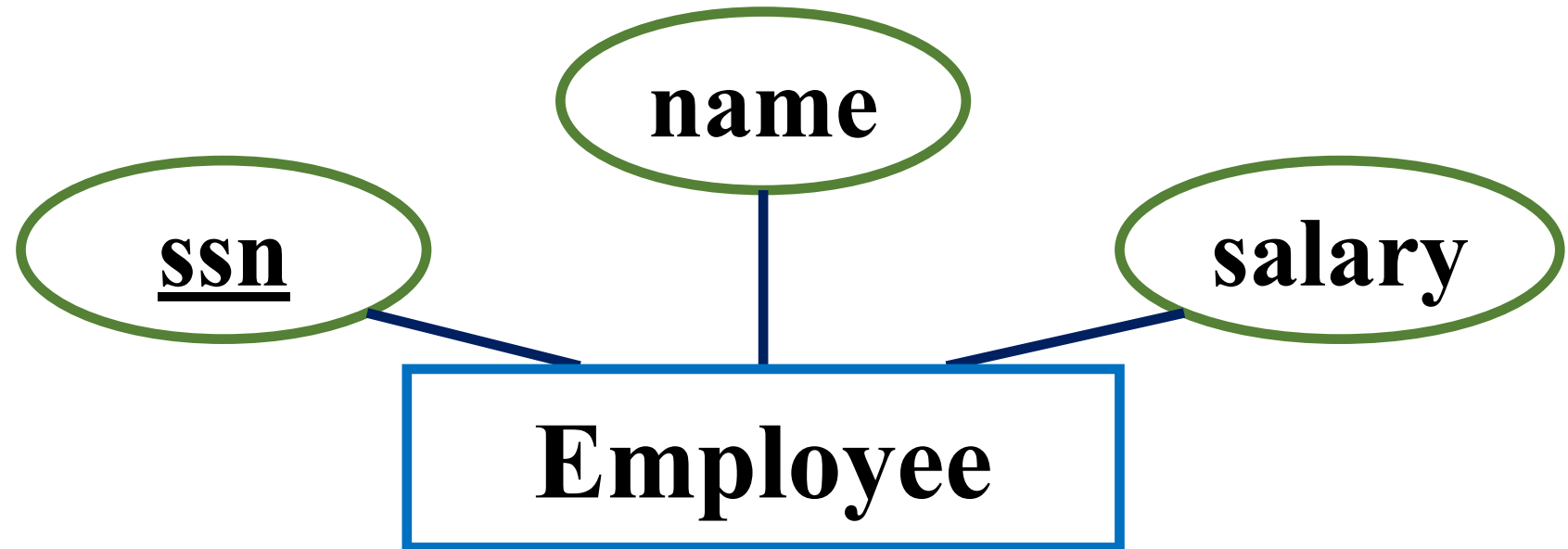
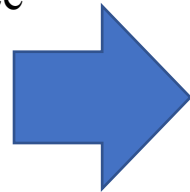
- Both name and id, as id only is enough
- Both id and SSN, as one of them is enough

- ✓ A primary key is *one of the candidate keys* chosen by the database designer to uniquely identify entities in an entity set.
- ✓ Each attribute in the primary key is underlined.

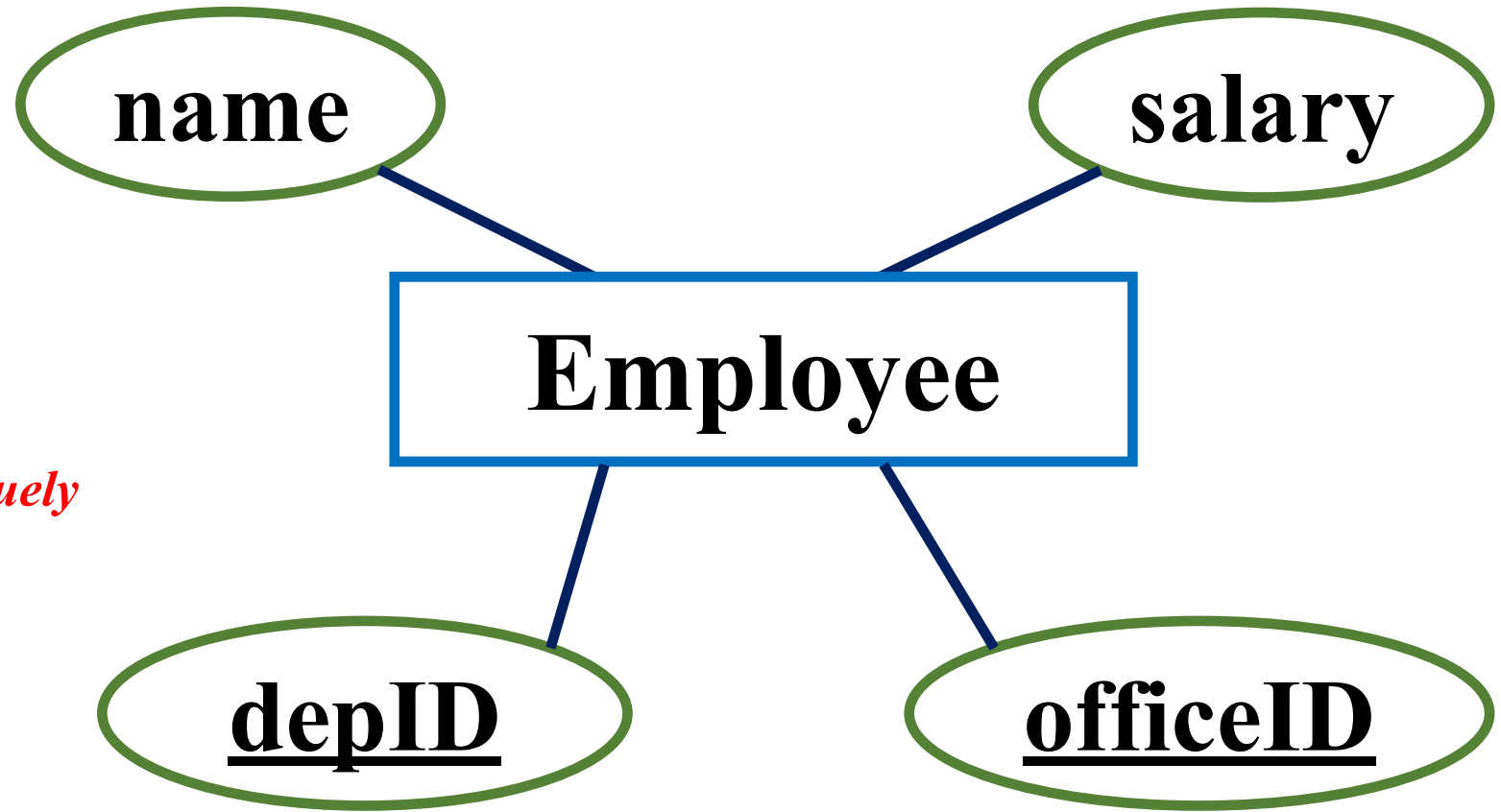


- Take Employees' *entity set* with *attributes* ssn (key), name, and salary as an example

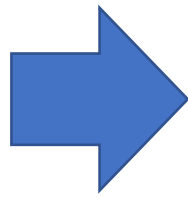
Attribute that *uniquely*  
identifies any employee



- Take Employee's *entity set* with *attributes* name, salary, office id (key) and department id (key) as an example



Both depID and officeID *uniquely* identifies any employee



Develop an ER diagram for Employees in certain company, in terms of:

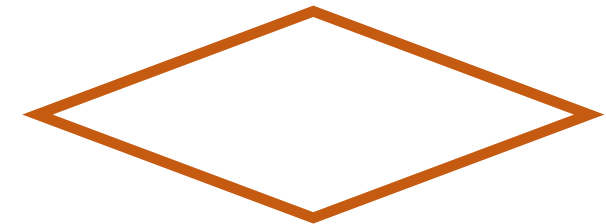
- SSN
- ID
- Name
- Address
- Department ID
- Salary
- Age
- Date of Birth
- Phone number

- Entity Set vs. Entity
- Single vs. Composite attribute
- Single vs Multi-valued attribute
- Derived attribute
- Candidate key
- Primary key

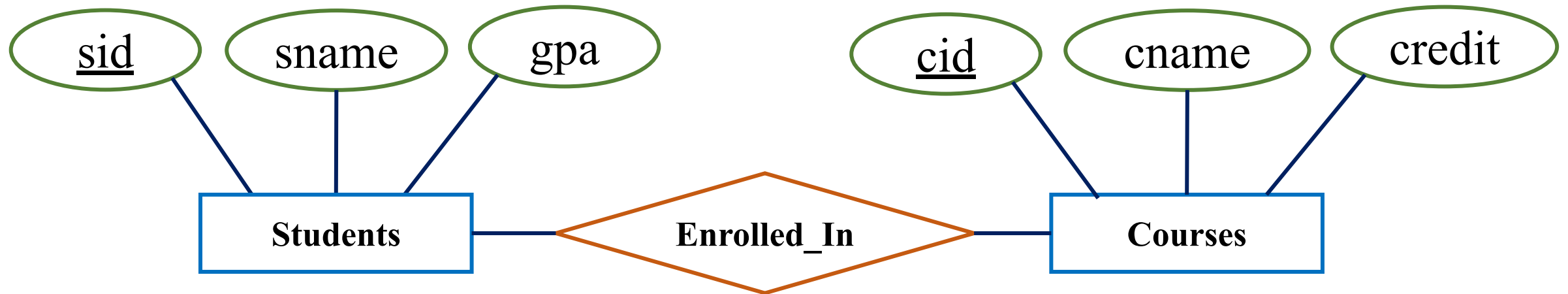


# *Relationship vs Relationship Set*

- A relationship is an association or a link between *two or more entities*.
  - John *works in* the computer department – is a relationship between two entities (one employee and one department).
  - Sophia *is enrolled in* the database course – is a relationship between two entities (one student and one course).
- As with entities, the ER diagram represents a set of similar relationships with a relationship set.
- A *relationship set* is a collection of similar relationships and is represented by a diamond-shaped box:

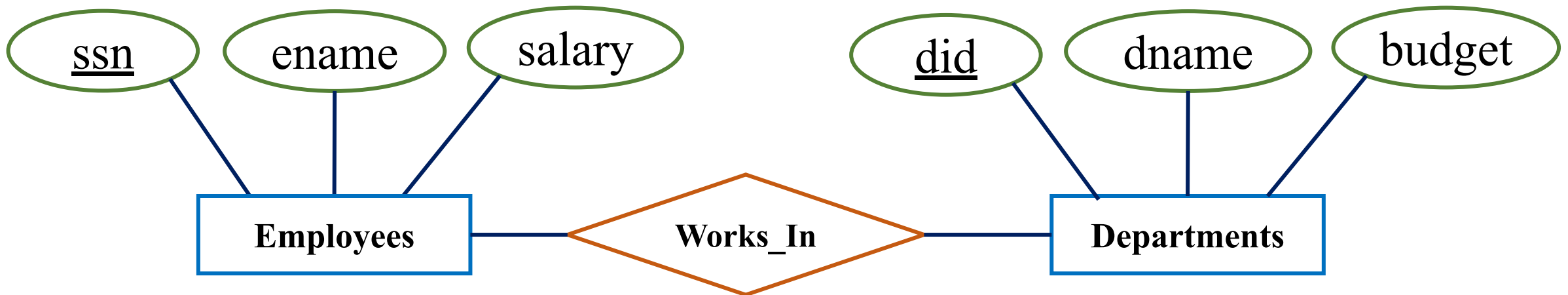


The *relationship set* “Enrolled\_In” represents a collection of similar relationships for students enrolled in courses (student 1 enrolled in course 3 and student 2 enrolled in course 5), can be represented as:



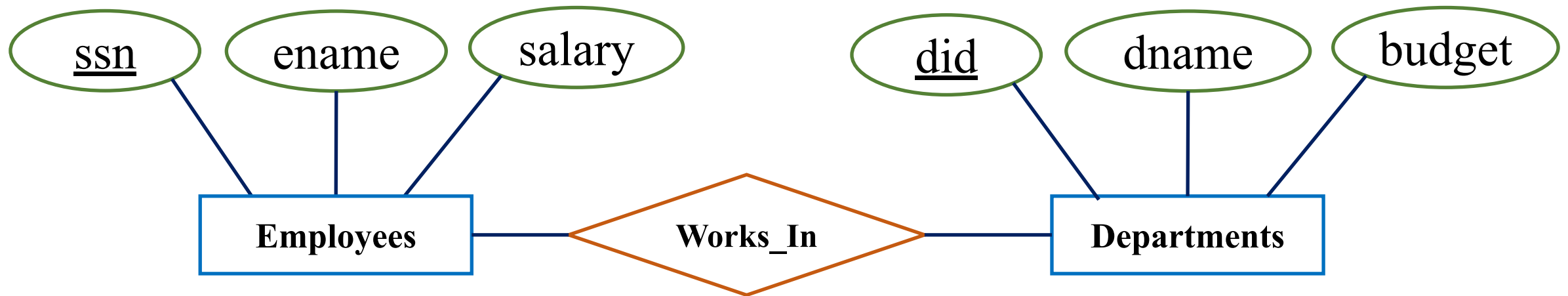
As mentioned before: In ER diagram, we represent entity sets (not entities) and relationship sets (not relationships).

The *relationship set* “Works\_In” where each relationship indicates a department in which an employee works in, can be represented as:

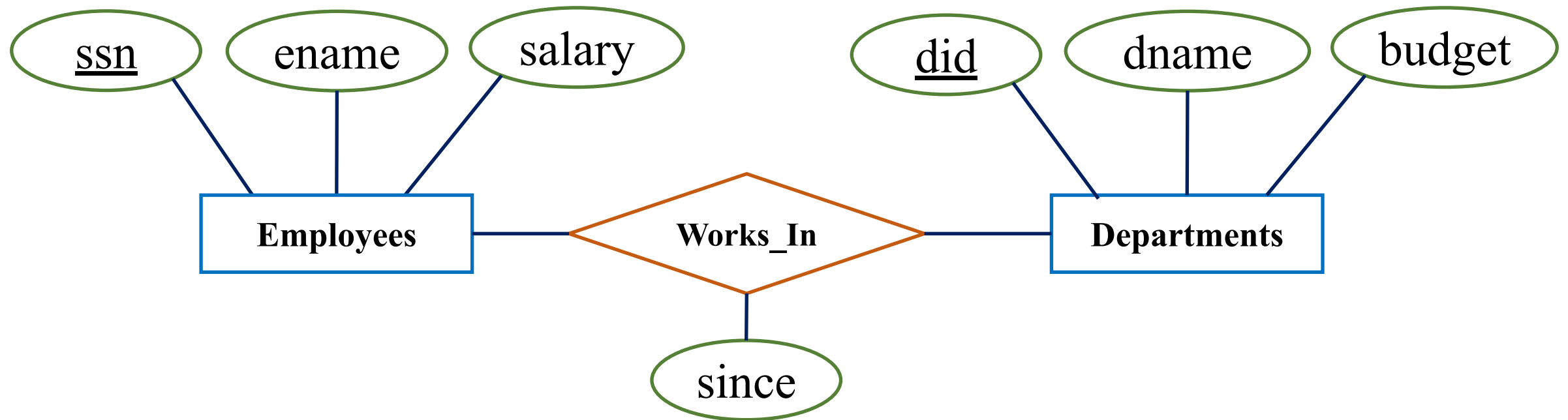


If we would like to capture information about *when* certain employee joined certain department - should we add such attribute to the employees set? should we add such attribute to the departments set?

- Yes we can! - however one employee may work in more than one department or one department may have many employees.
- This attribute is *logically related to the relationship* between the employee and the department

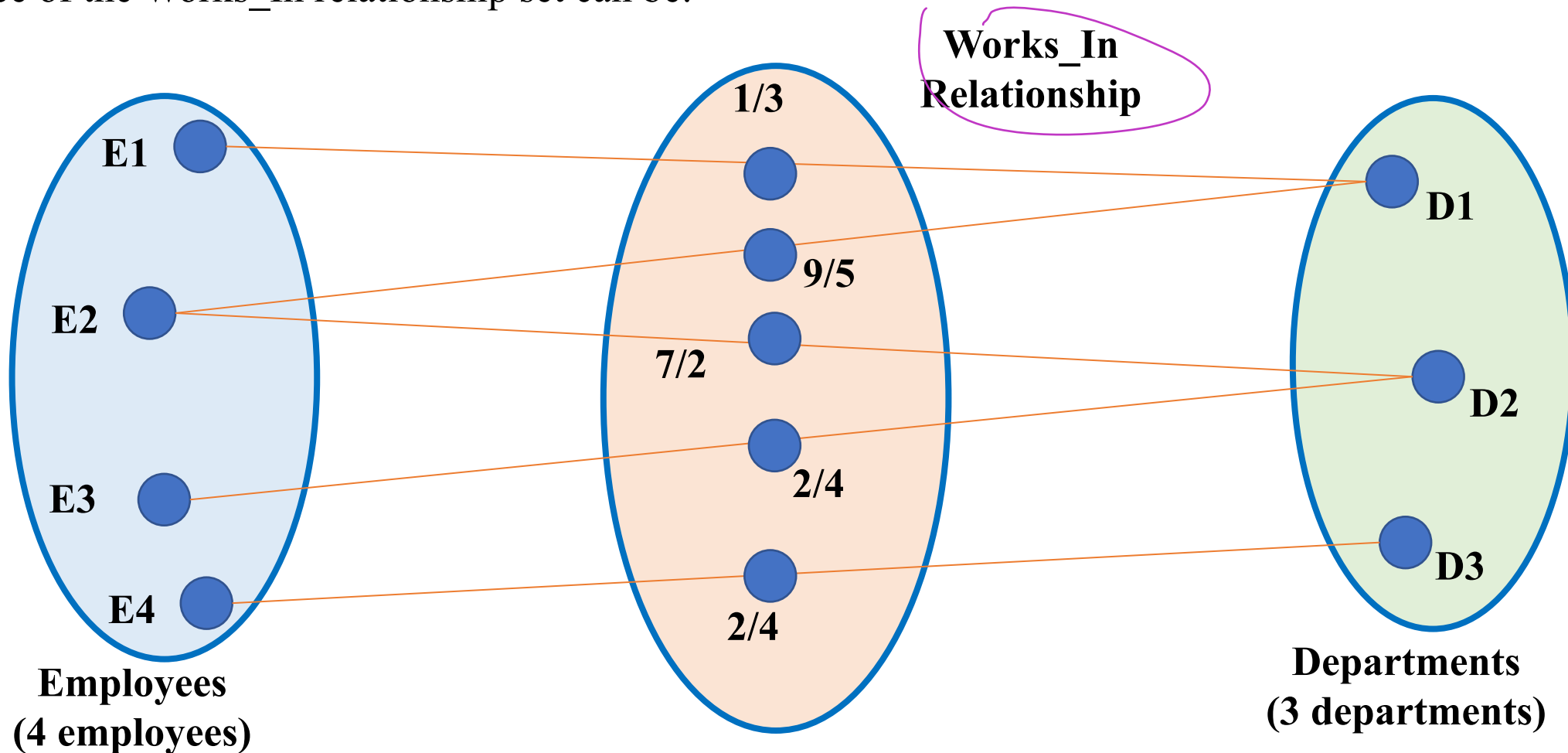


A relationship set -as an entity set- can *have attributes* that are used to record information about the relationship rather than about any of the participating entities.

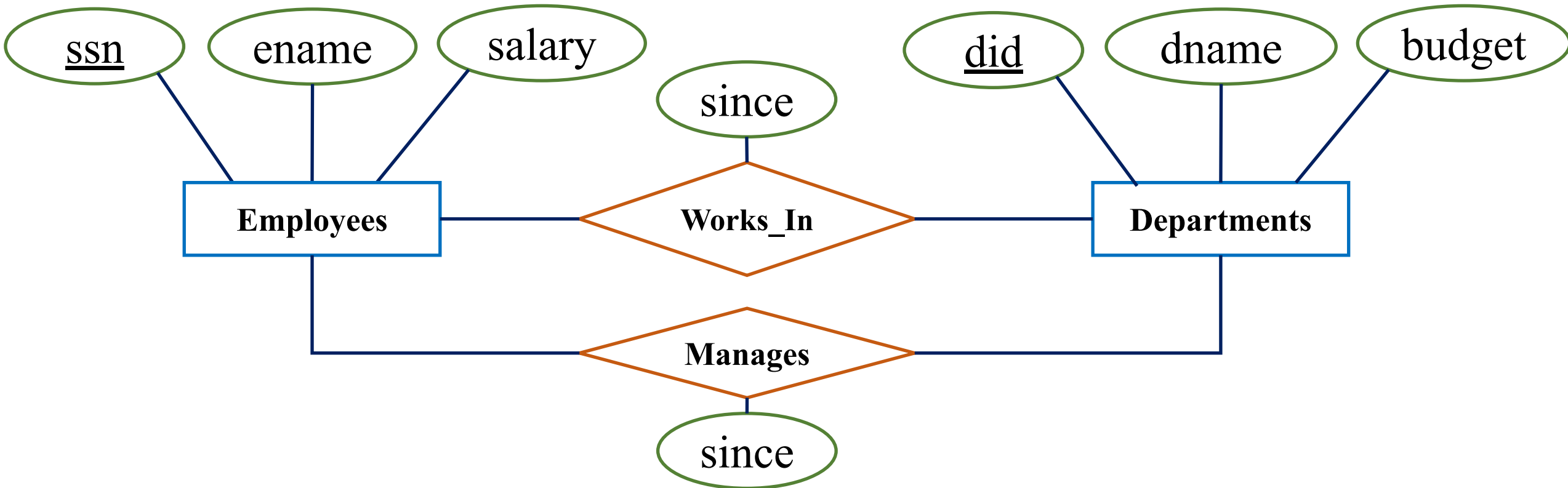


# Relationship vs Relationship Set

- An instance can be thought of as a *snapshot* of the relationship set *at a given time*.
- Instance of the Works\_In relationship set can be:



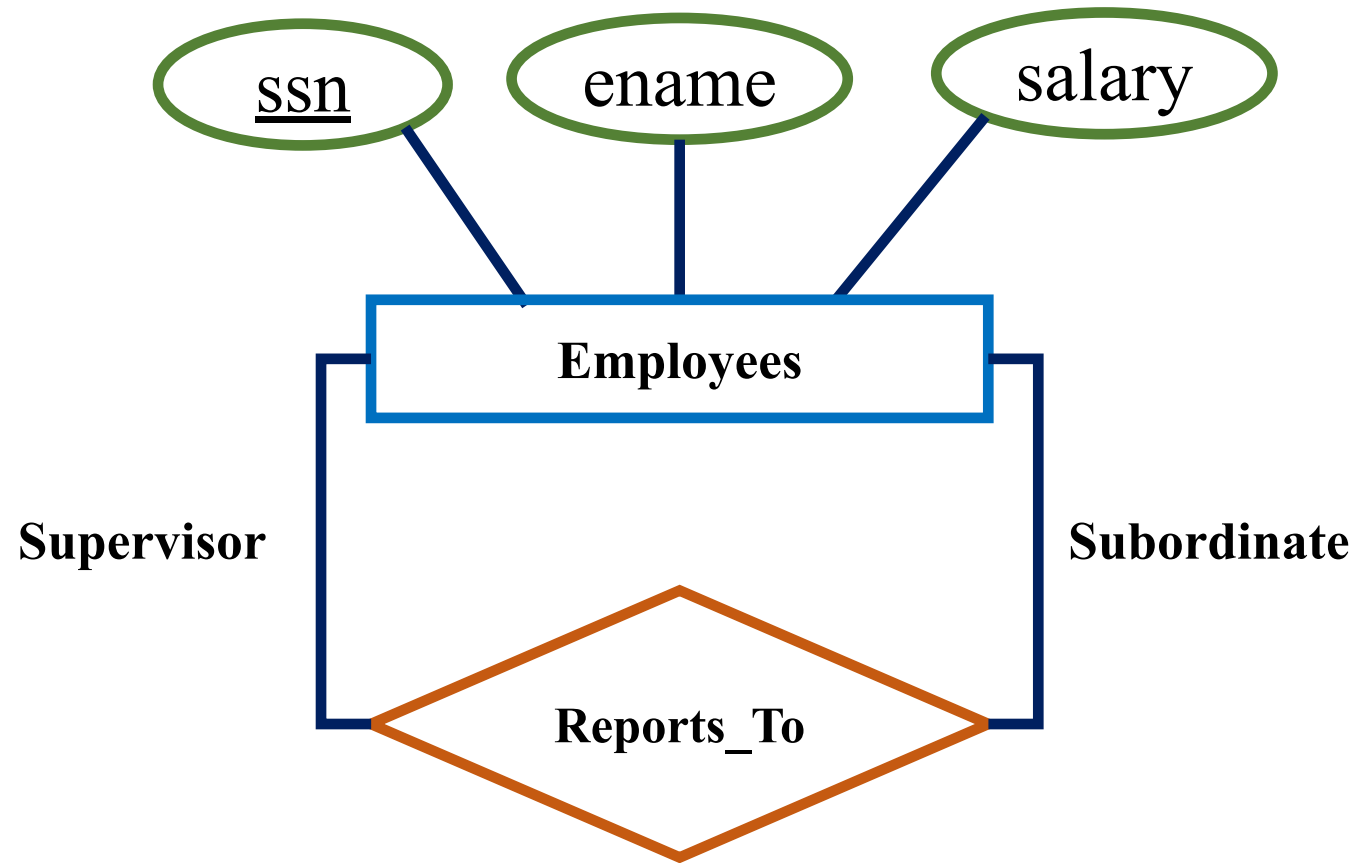
*Several* relationship sets may link the *same entity sets* - for example, we could also have a *Manages* and a *Works\_In* relationship sets involving Employees and Departments entity sets.





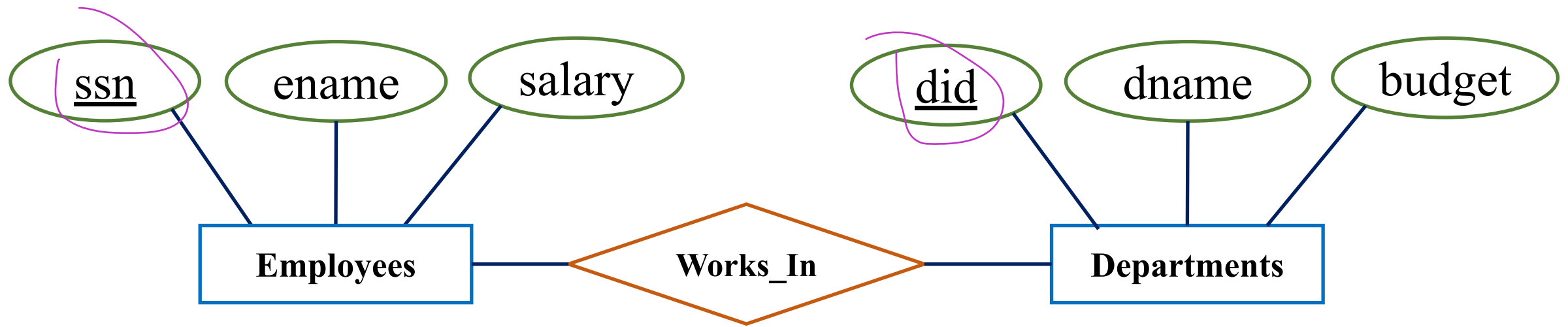
- Assume we have a company, where an employee can report to another employee – How to represent such relationship?
  - ***Solution 1:*** We can have two entity sets: one for employees and one for managers, and use normal relationships.
  - ***Solution 2:*** We can have one set for both employees and managers, but using self-referencing relationships.
- The entity sets that participate in a relationship set may not be distinct, sometimes a relationship might involve *two entities from the same entity set*.

- For example, since an employee *reports to* another employee, every relationship in Reports\_To is in the form of (emp1, emp2) where both emp1 and emp2 are entities in Employees.
- However they play different roles: emp1 reports to the managing employee emp2, which is reflected in the *role indicators* supervisor and subordinate.

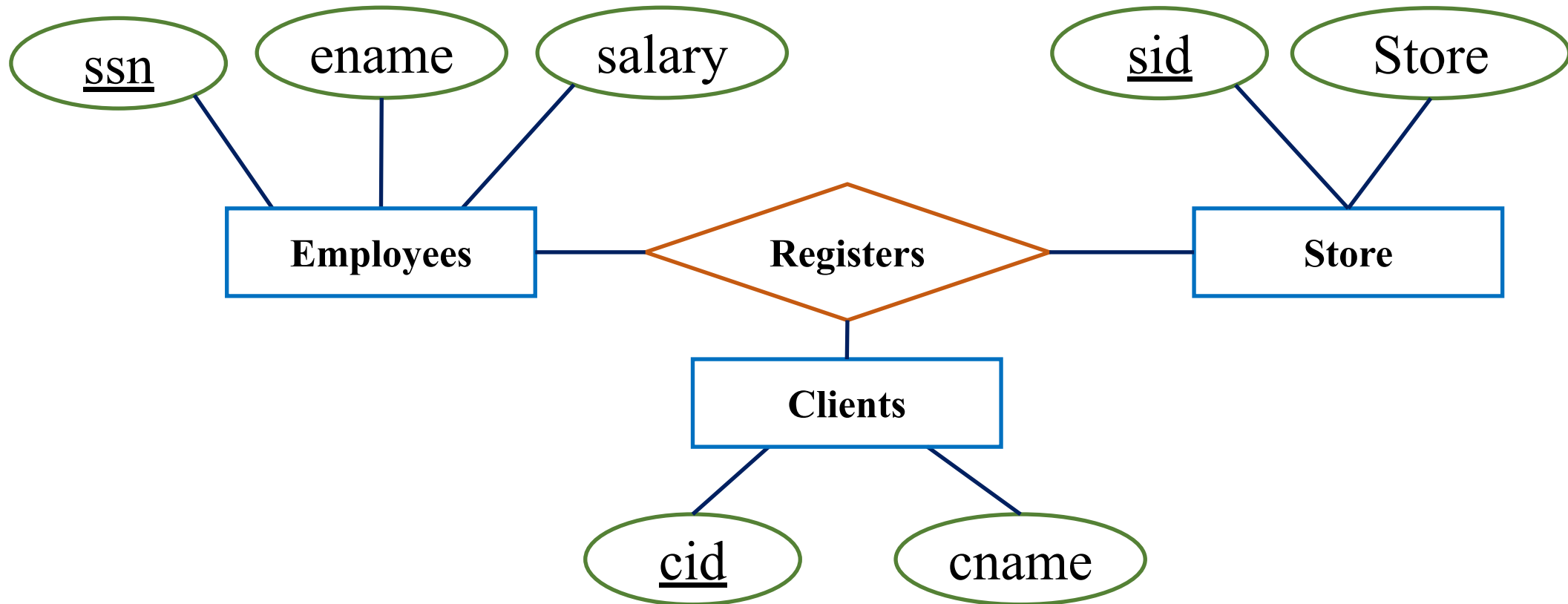


# *Degree of a Relationship*

- The degree of the relationship is the number of participating entities in a relationship.
- John *works in* the sales department, what is the degree of this relationship? – this is a *binary* relationship (Employee and Department).



John who *works in* the computer store *registers* Emma as a new client, what is the degree of this relationship? – this is a *ternary* relationship (Employee, Department, and Client).

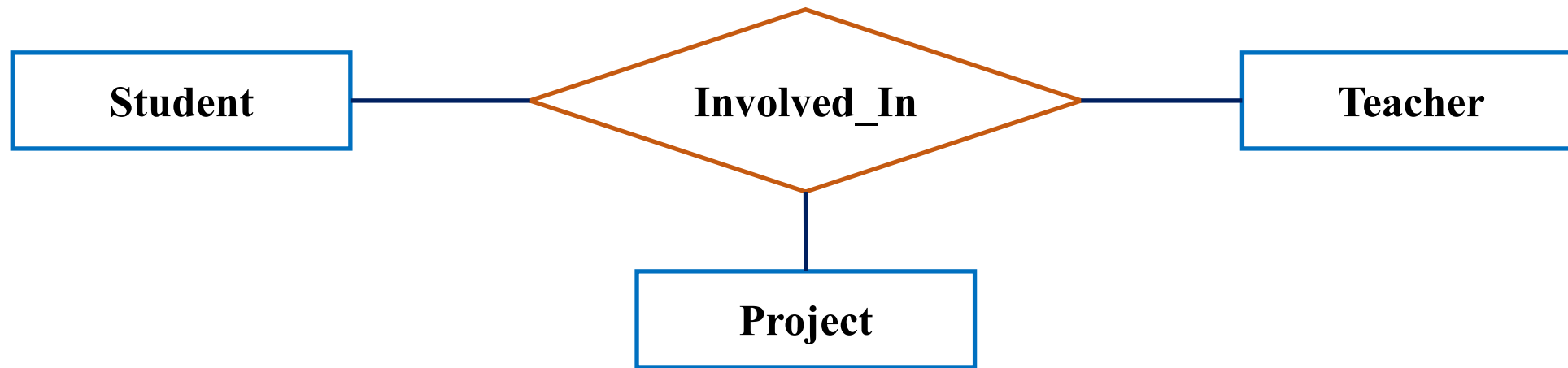


“Binary” is a relationship of degree 2

“Ternary” is a relationship of degree 3

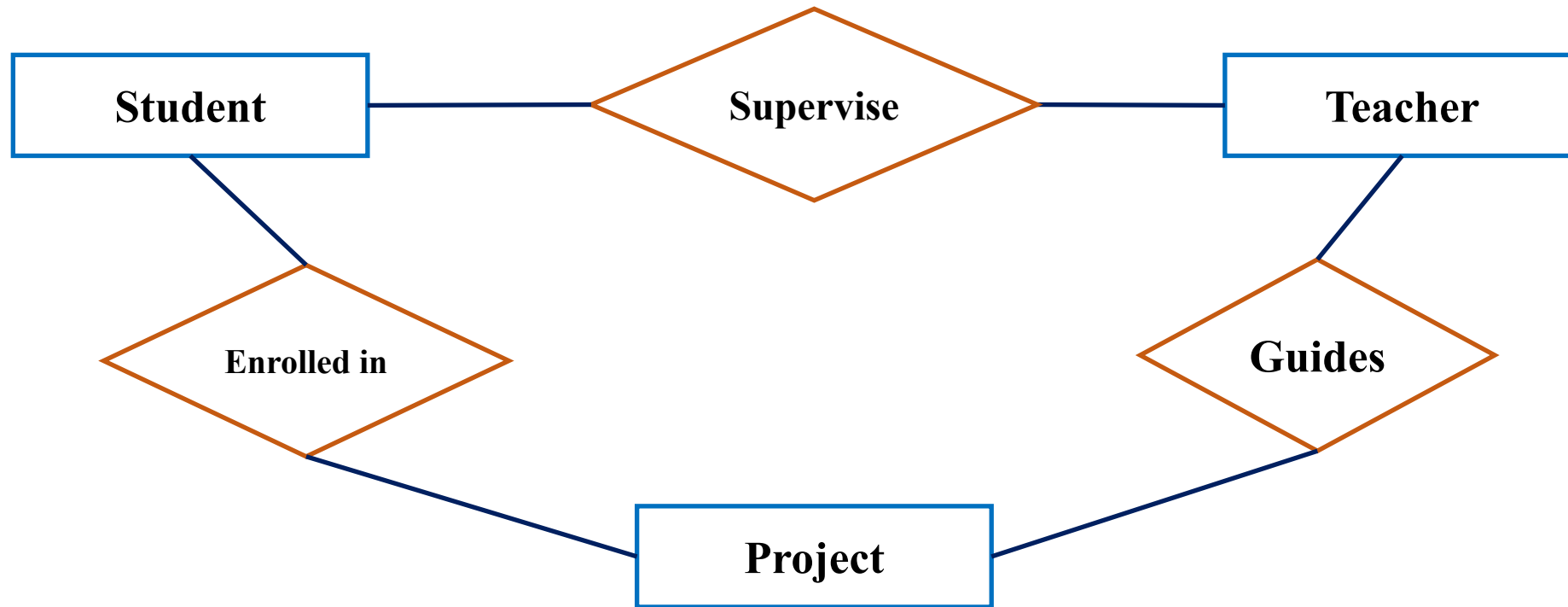
“N-ary” is a relationship of degree N

What is the relationship degree for each of the following?



Three entity sets all linked through one relationship set of degree 3

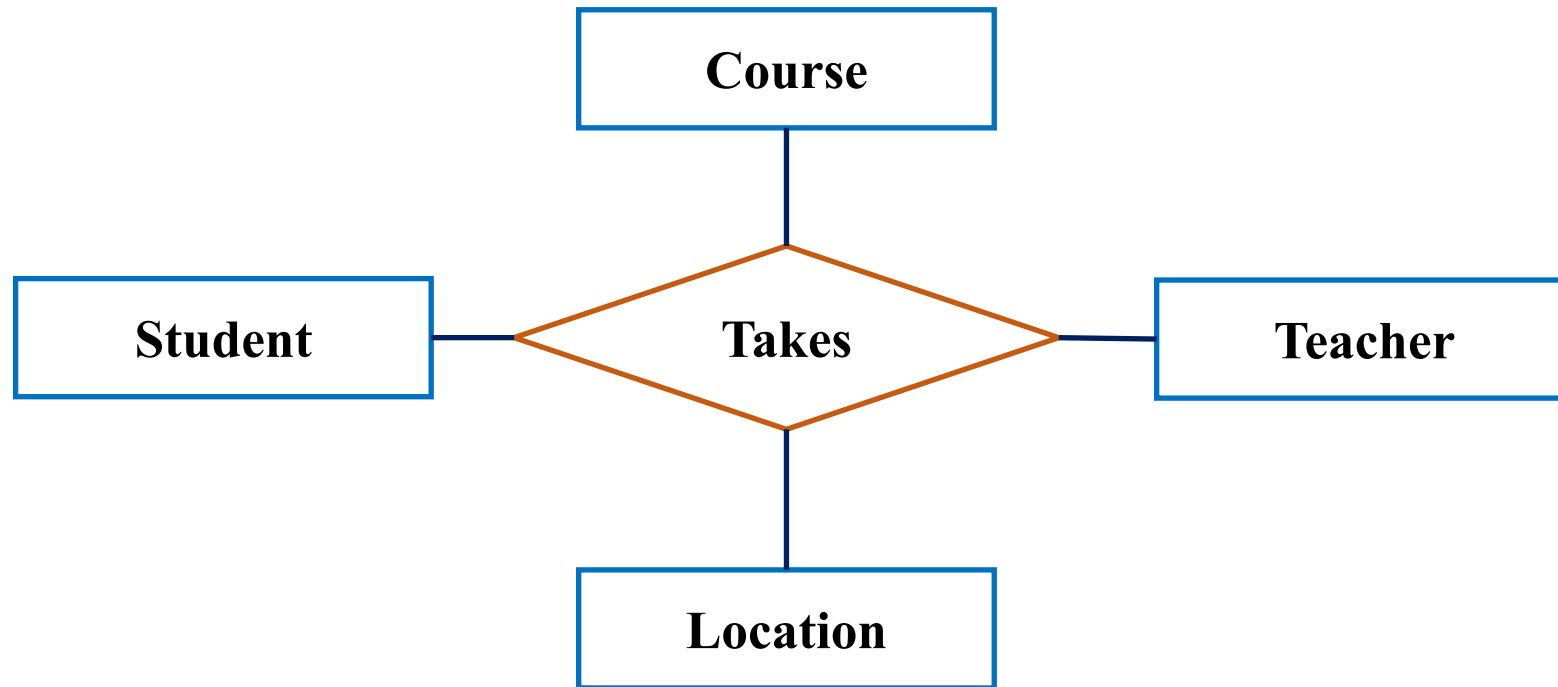
What is the relationship degree for each of the following?



Three entity sets linked through three relationship sets, each of degree 2



What is the relationship degree for each of the following?

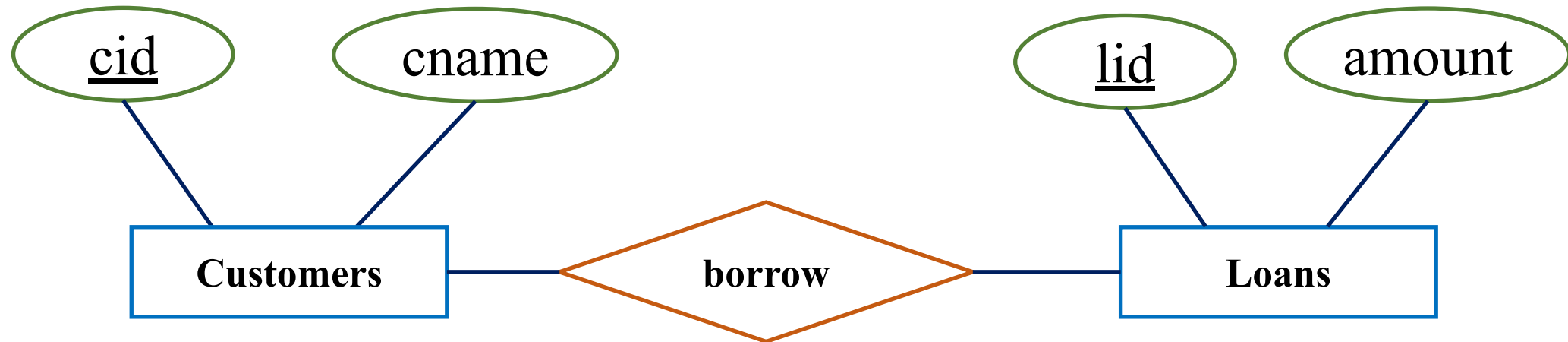


Four entity sets all linked through one relationship sets of degree 4

# *Mapping Cardinalities*

- Cardinality is defined as the number of entities in one entity set that *can be associated* with number of entities in other entity set via a relationship.
- For example:
  - ✓ **1** teacher (in teachers entity set) can teach *Many* course (in courses entity set).
  - ✓ **1** course (in courses entity set) can be taught by *only 1* teacher (in teachers entity set).
  - ✓ *Many* students (in students entity set) can enroll in *Many* courses (in courses entity set).

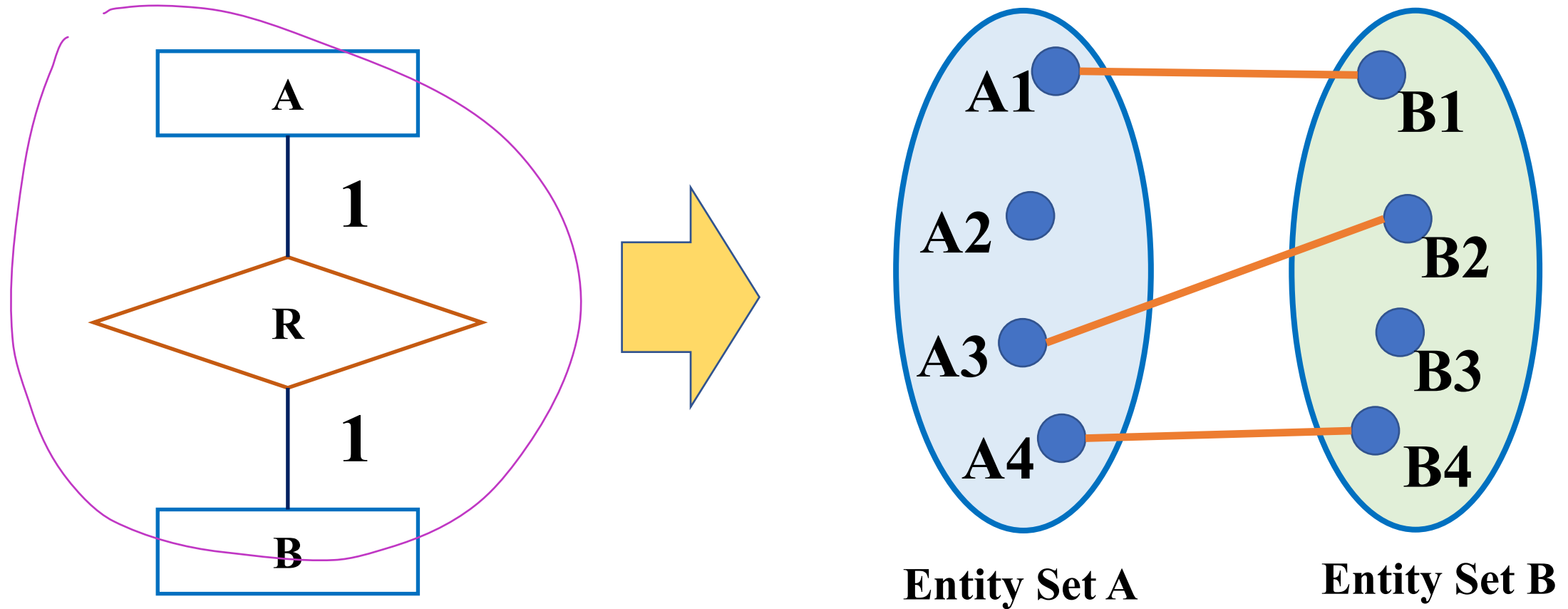
Keep the following ER model example in mind:



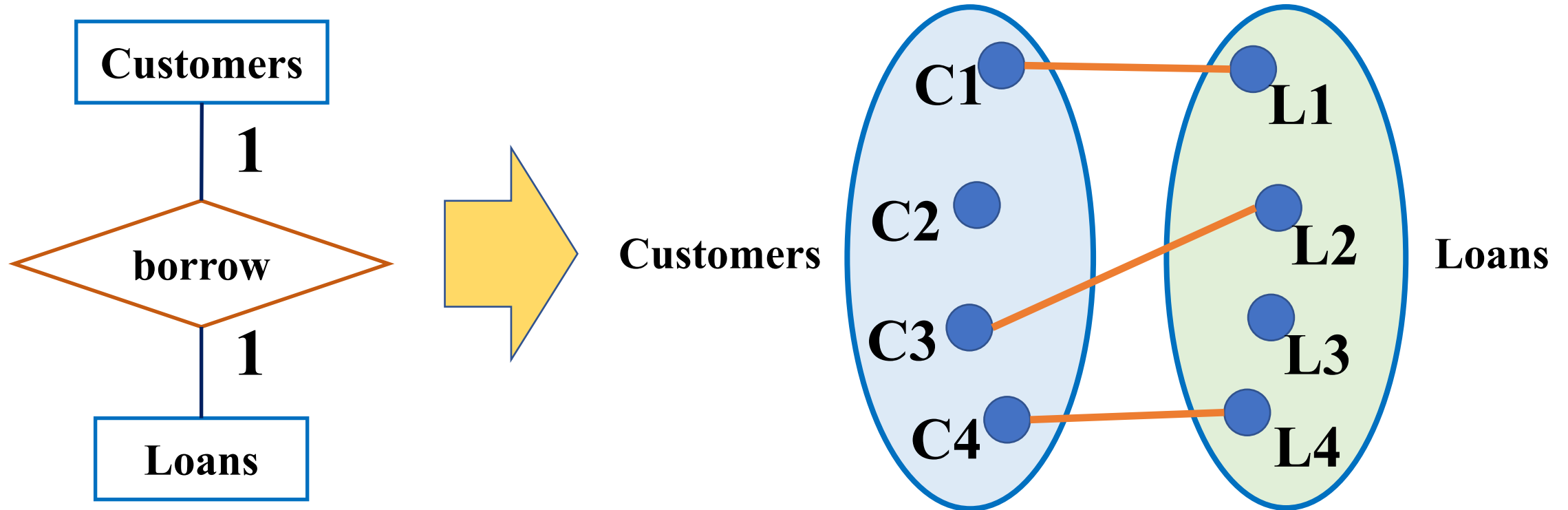
We have **4 possible** mapping cardinalities  
(1:1, 1:M, M:1, N:M)

# Mapping Cardinalities - One-to-One (1:1)

One entity from entity set A can be associated with **at most** one entity from entity set B, and vice versa - (**at most** means minimum of 0 and maximum 1)

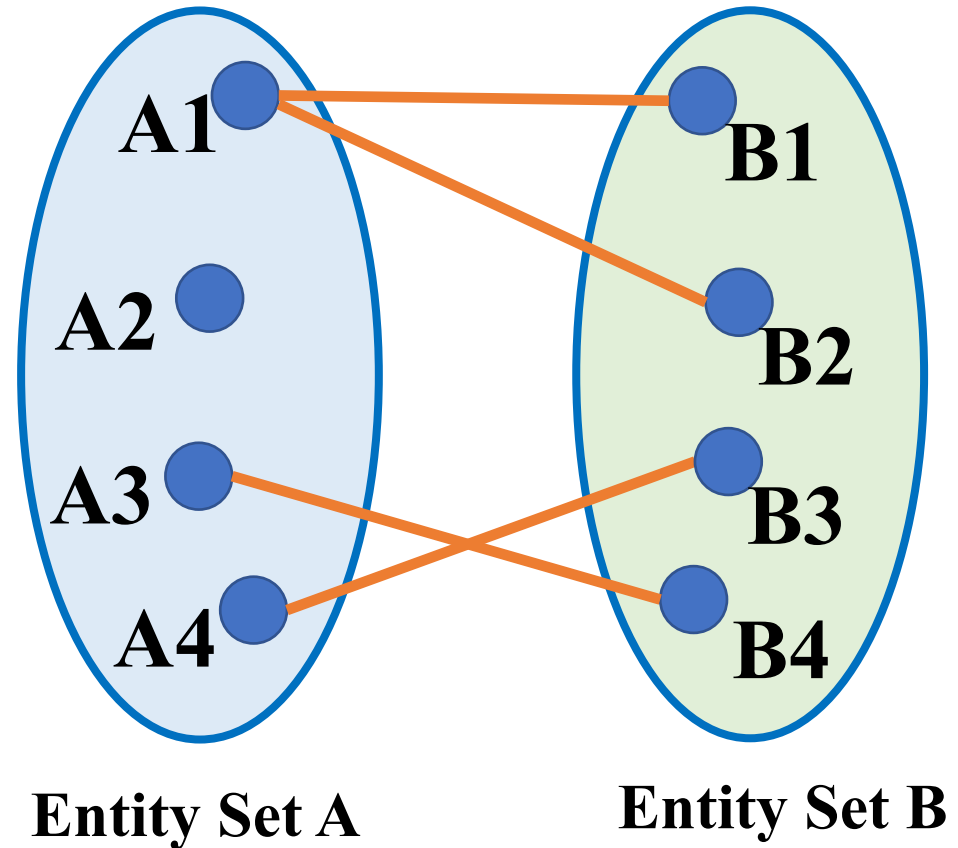
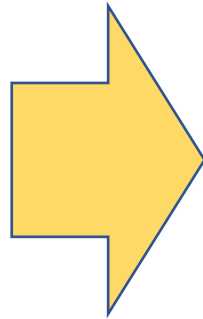
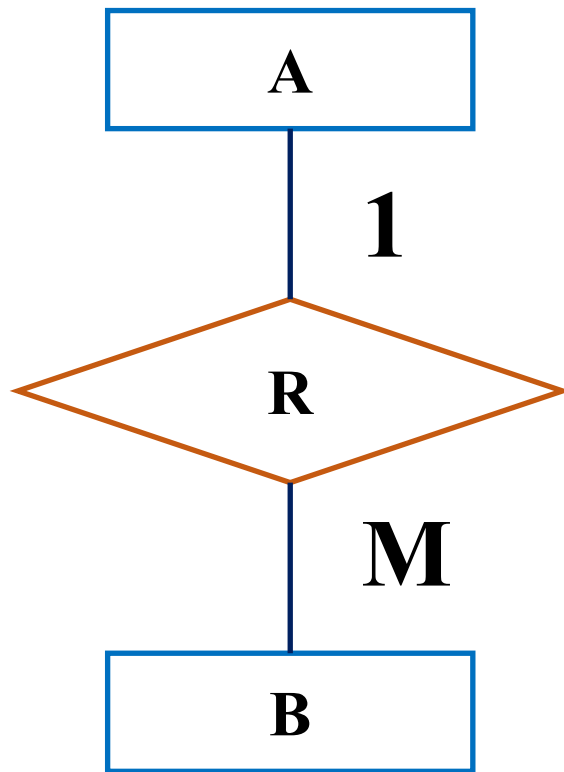


- One loan is associated with *at most* one customer via borrow relationship.
- One customer is associated with *at most* one loan via borrow relationship

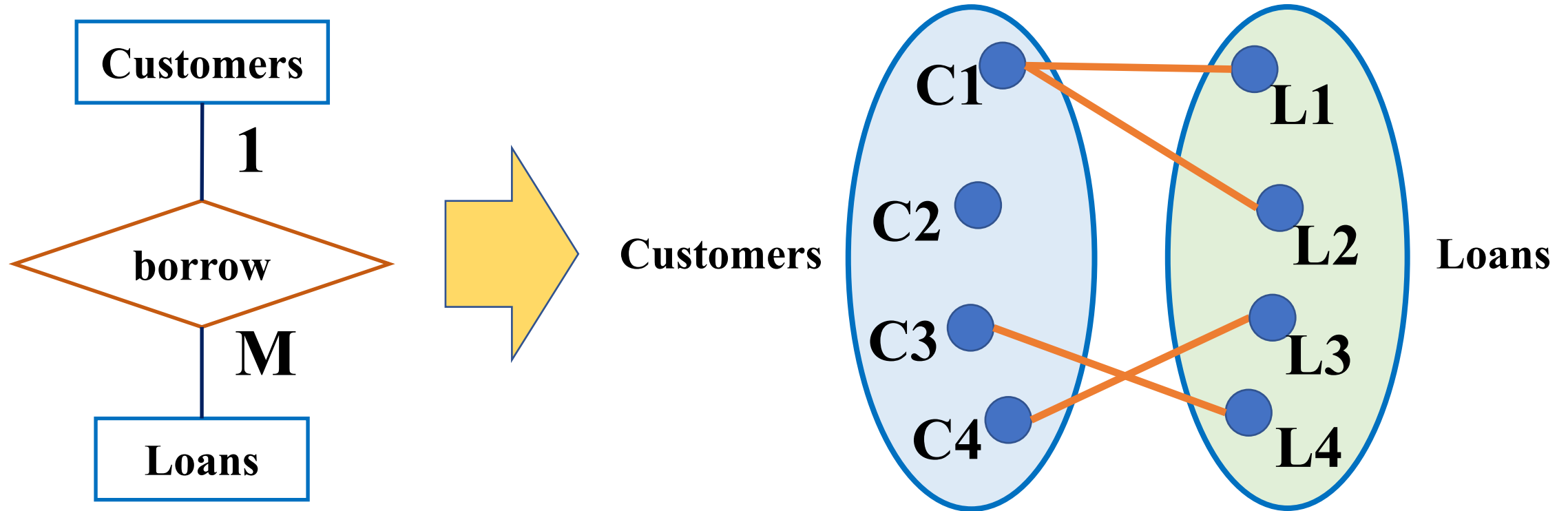


# Mapping Cardinalities - One-to-Many (1:M)

- One entity from entity set A can be associated with *more than (0 is included)* one entity from entity set B.
- However one entity from set B can be associated with *at most* one entity from set A.



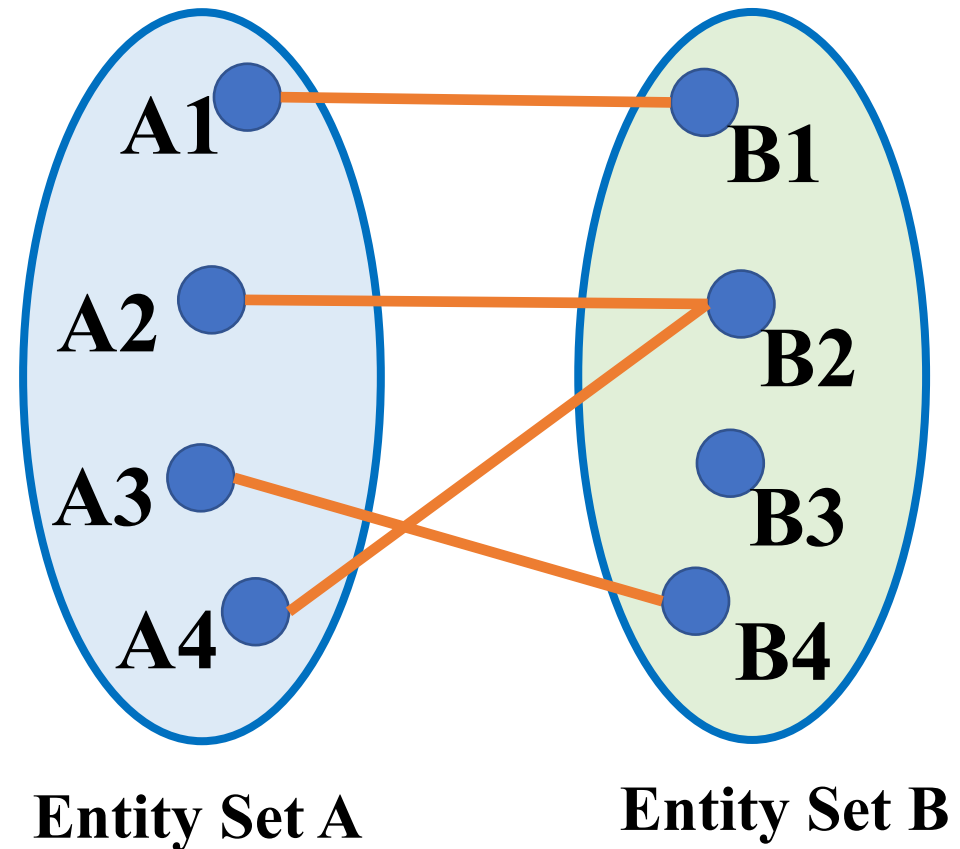
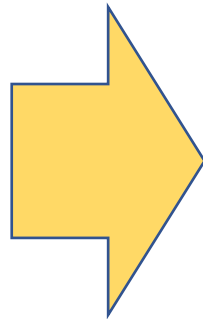
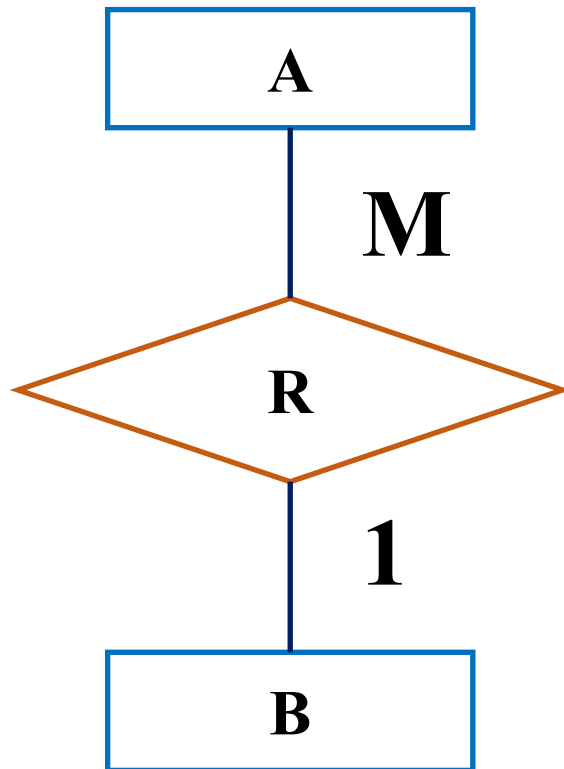
- One loan is associated with *at most* one customer via borrow relationship.
- One customer is associated with *more than (0 is included)* one loan via borrow relationship



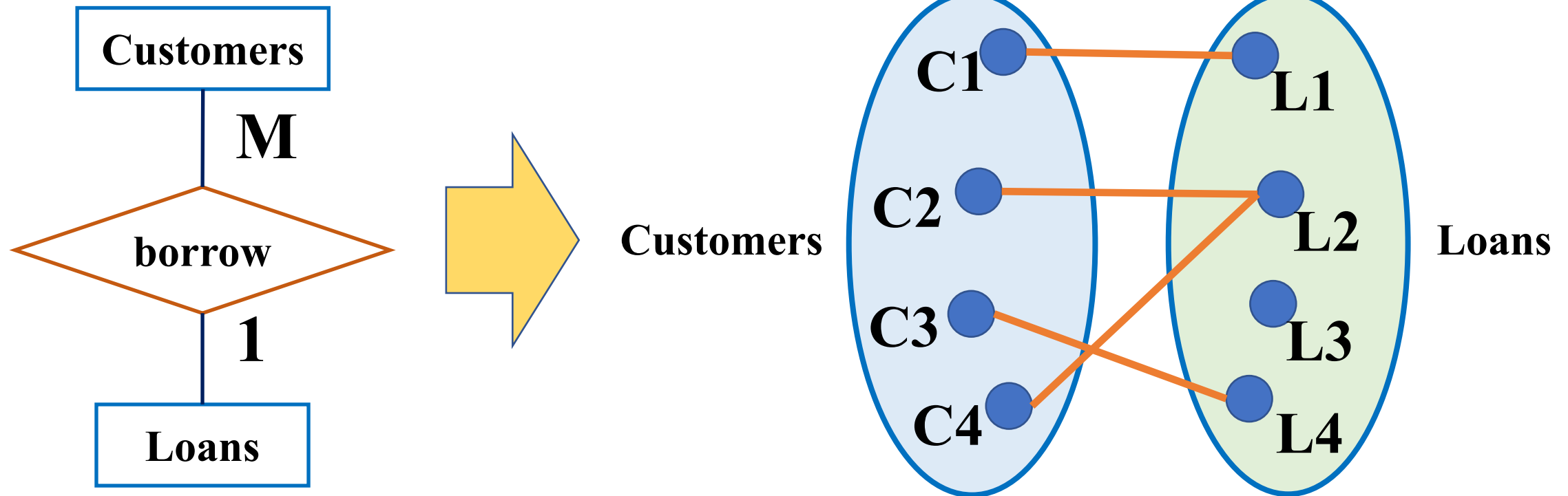


## Mapping Cardinalities - Many-to-One (M:1)

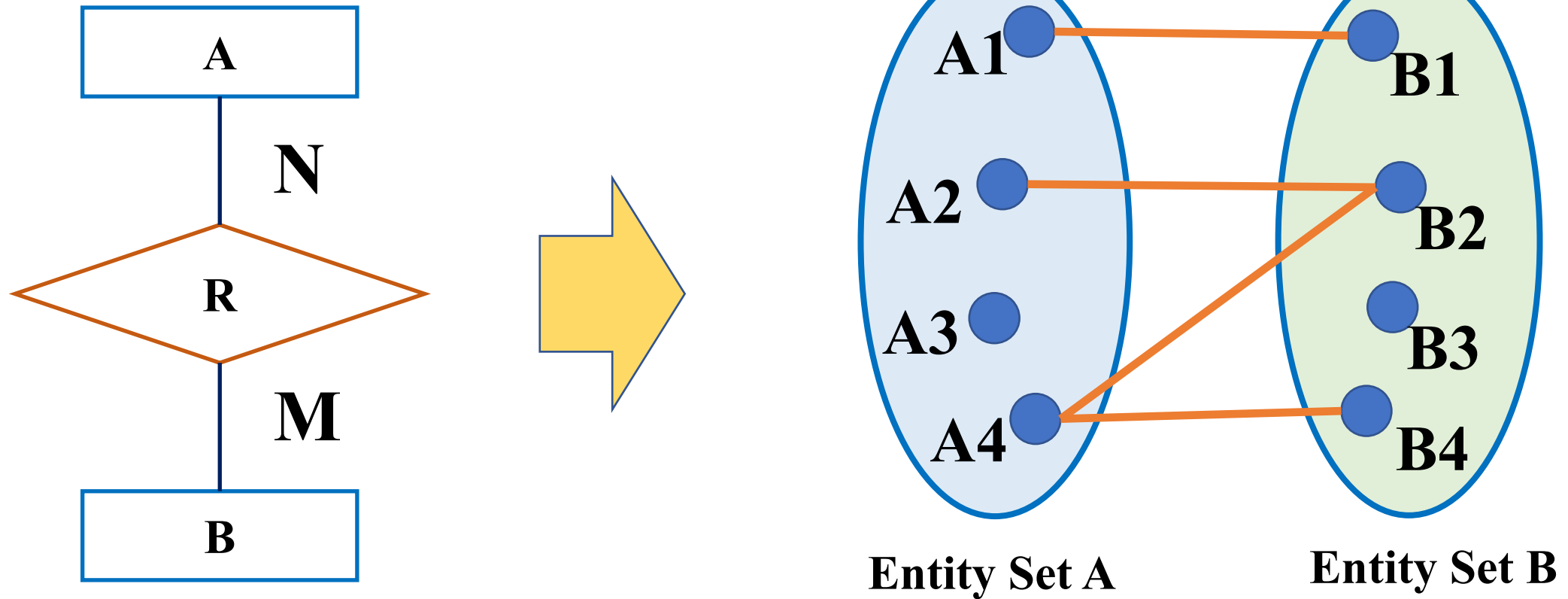
- One entity from entity set A can be associated with *at most* one entity from entity set B.
- However one entity from set B can be associated with *more than (0 included)* one entity from set A.



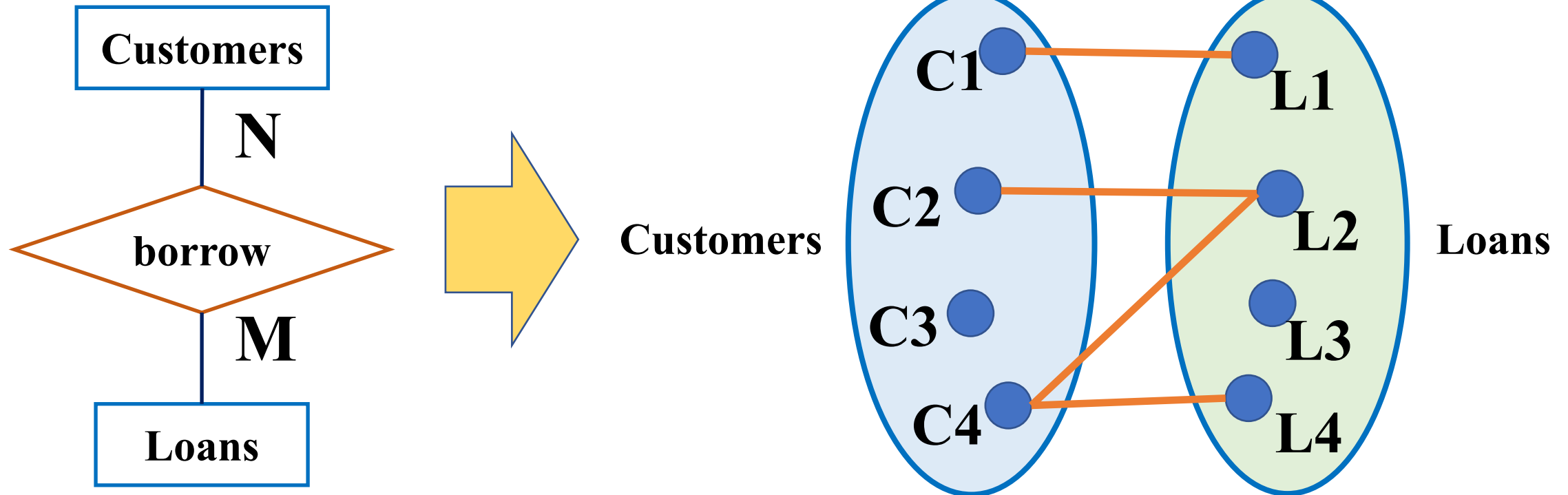
- One loan is associated with *several (0 included)* customers via borrow relationship.
- One customer is associated with *at most* one loan via borrow relationship



- One entity from entity set A can be associated with *more than (0 included)* one entity from entity set B, and vice versa.



- One loan is associated with *several (0 included)* customers via borrow relationship.
- One customer is associated with *several (0 included)* loans via borrow relationship



What is the mapping cardinality for each of the following?

- Users and Users' Profiles
- Employees' and Employees' dependents

You are designing a separate ER diagram for each point, so you can usually place logical assumptions

- Users and Users' Profiles

- (1:1) – if we assume that profile refers to the professional profile (e.g., linked-in), each user has one professional profile and each profile is linked to one user.
- (1:M) – if we assume that profile refers to the social media' profiles (Facebook, Instagram,...), each user can have more than one profile on social media, but each profile is linked to only one user.

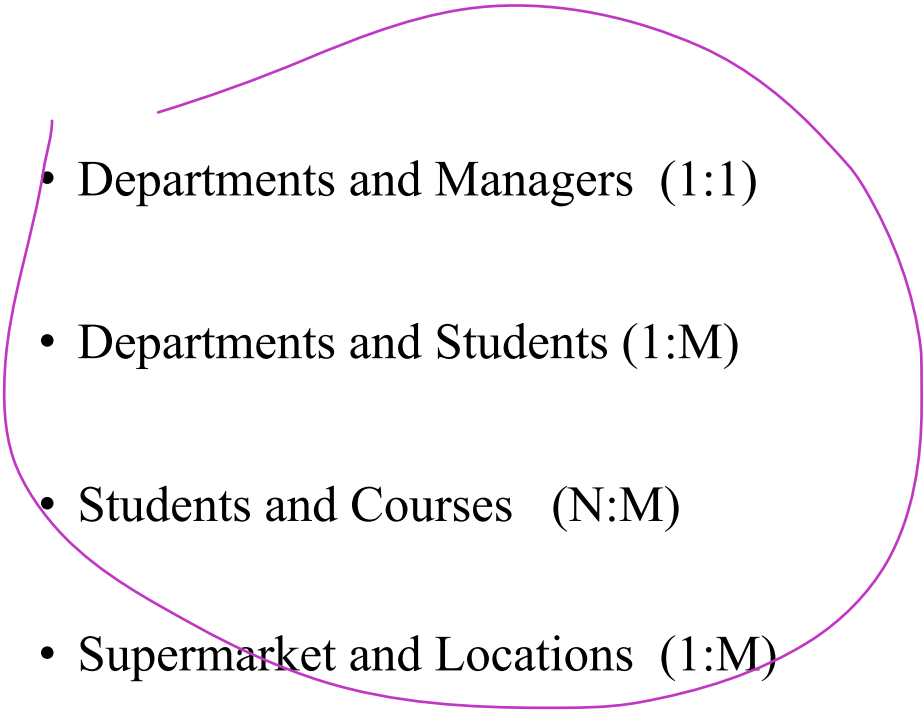
- Employees' and Employees' dependents

- (1:M) – if we assume that either of the parents works in the company, so each dependent is related to only one employee and each employee can have more than one dependent
- (N:M) – if we assume that both parents can work in the company, so each dependent can be related to more than one employee and each employee can have more than one dependent

What is the mapping cardinality for each of the following?

- Departments and Managers
- Departments and Students
- Students and Courses
- Supermarket and Locations

You are designing a separate ER diagram for each point, so you can usually place logical assumptions

- 
- Departments and Managers (1:1)
  - Departments and Students (1:M)
  - Students and Courses (N:M)
  - Supermarket and Locations (1:M)



## *(Min : Max) Annotation*

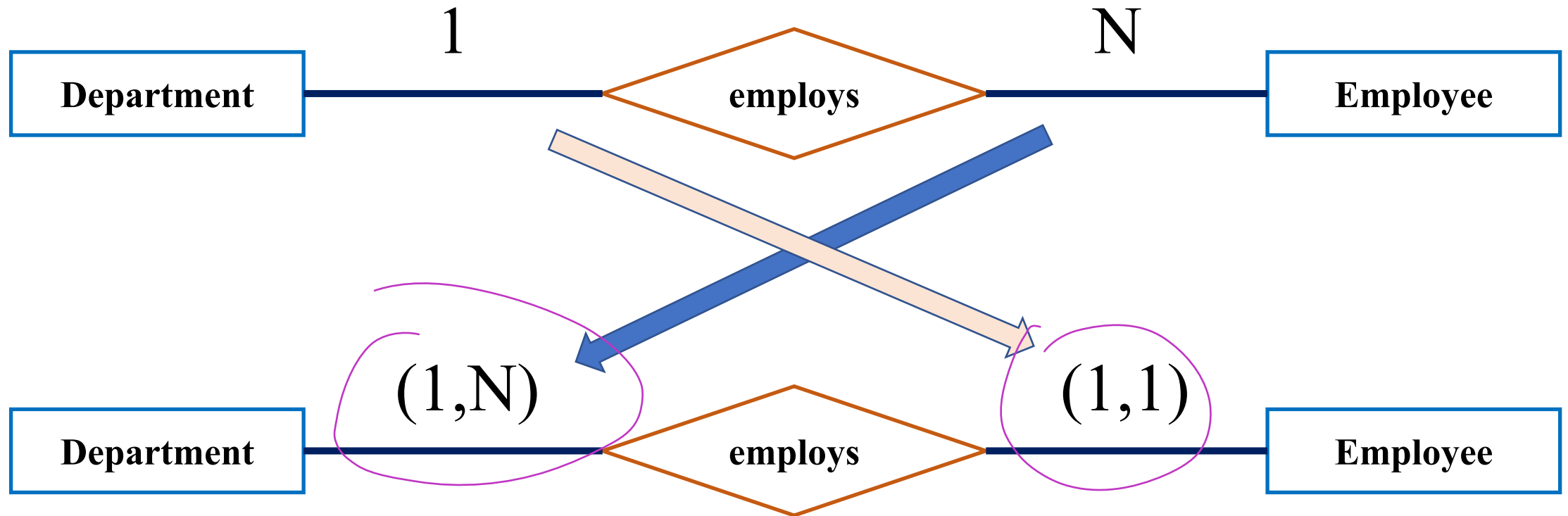
---

- Mapping cardinalities (1:1/1:M/M:M) are very abstract and they capture relationship constraints with two main values:
  - At most one (minimum of zero and maximum of one)
  - Many (minimum of zero and no maximum)
- In order to provide more constraints-related information about the relationships, we can use a (min, max) notation.
- (min, max) means that each entity in the entity set must be linked to *at least min* entries of the relationship, and *at most max*.

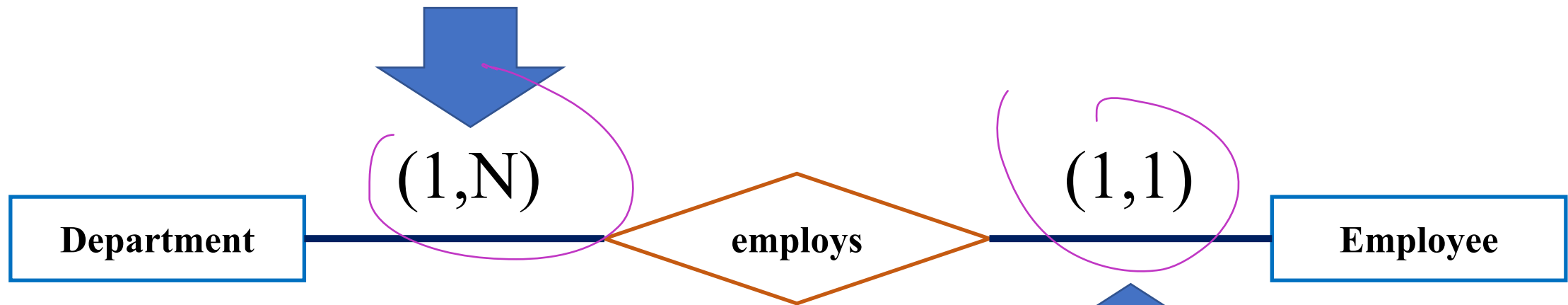
Each employee works for one Department and N employees can work for 1 department



Each employee works for one Department and N employees can work for 1 department

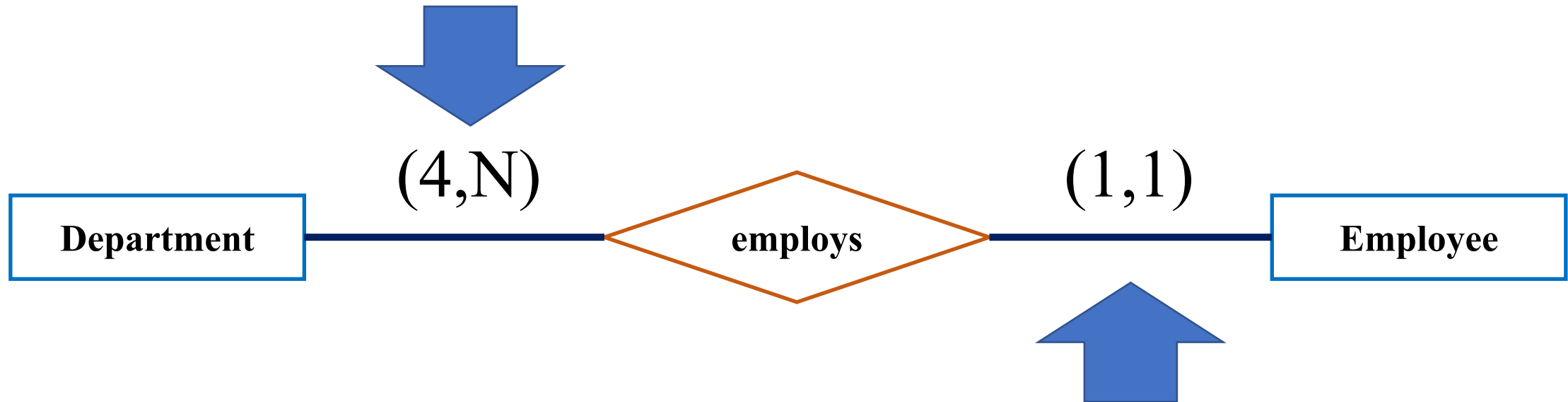


The department can have minimum of 1 employee and maximum of N employees ( $>1$ )



Employee works for minimum of 1 department and maximum of 1 department (exactly 1 department)

The department can have minimum of *4 employees* and maximum of N employees ( $>1$ )



Employee works for minimum of 1 department and maximum of 1 department (exactly 1 department)

Draw an ER diagram for the following requirements, consider using mapping cardinality and min:max annotation

Employee can manage 1 department at most

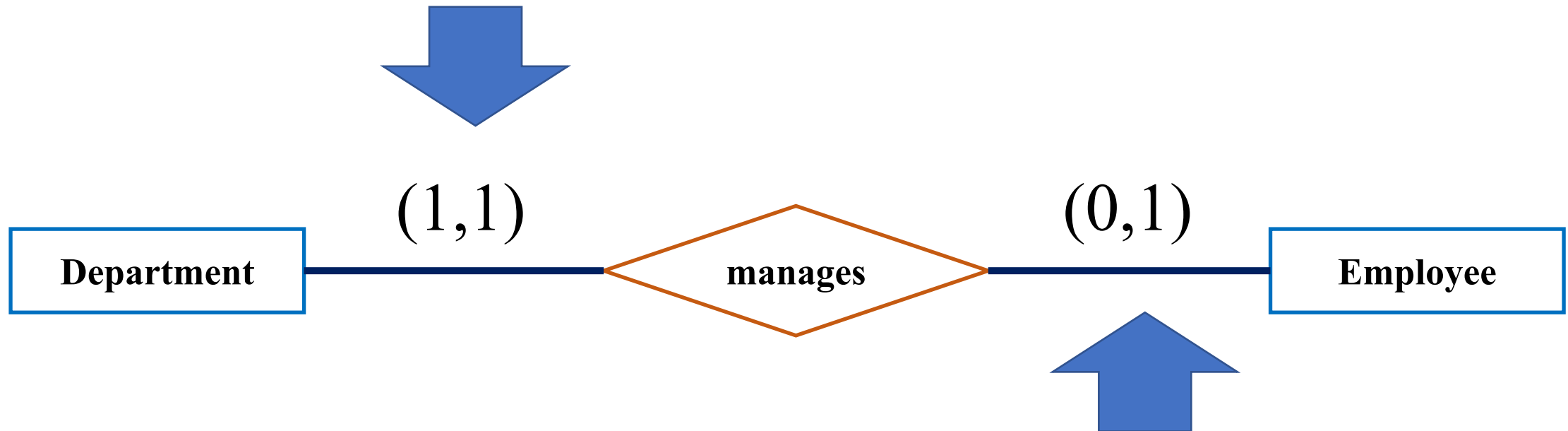
Department must be managed by 1 Employee

Employee can manage 1 department at most  
Department must be managed by 1 Employee






The department must have min of 1 manager and max of 1 manager



The Employee can manage 0 departments or 1 department (1 department at most)

- Draw an ER diagram for two entity sets: Departments and Projects.
  - Use the (min, max) annotation to describe the controls relationship, where:
    - Department can control up to 5 of projects (zero included)
    - Project must be controlled by at most one department (at least one department)
- 

Using mapping cardinalities

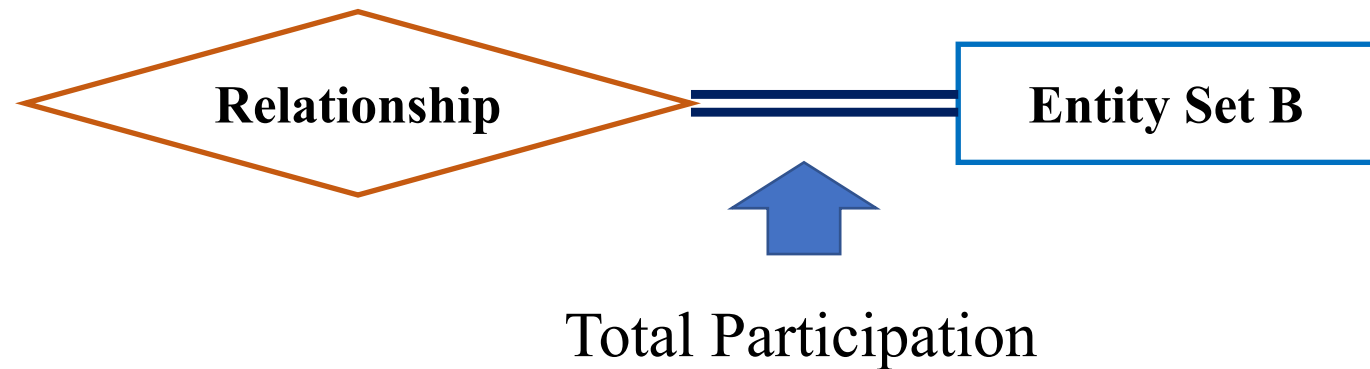


Using (Min, Max) annotation

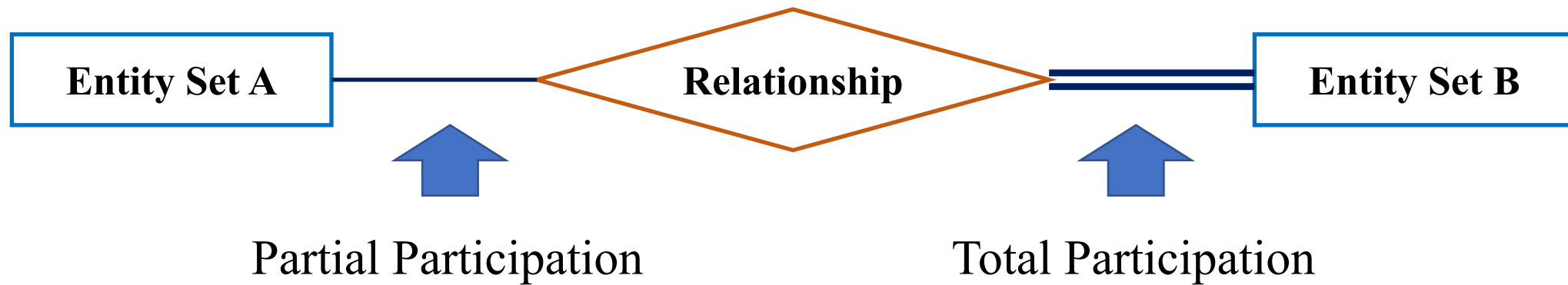


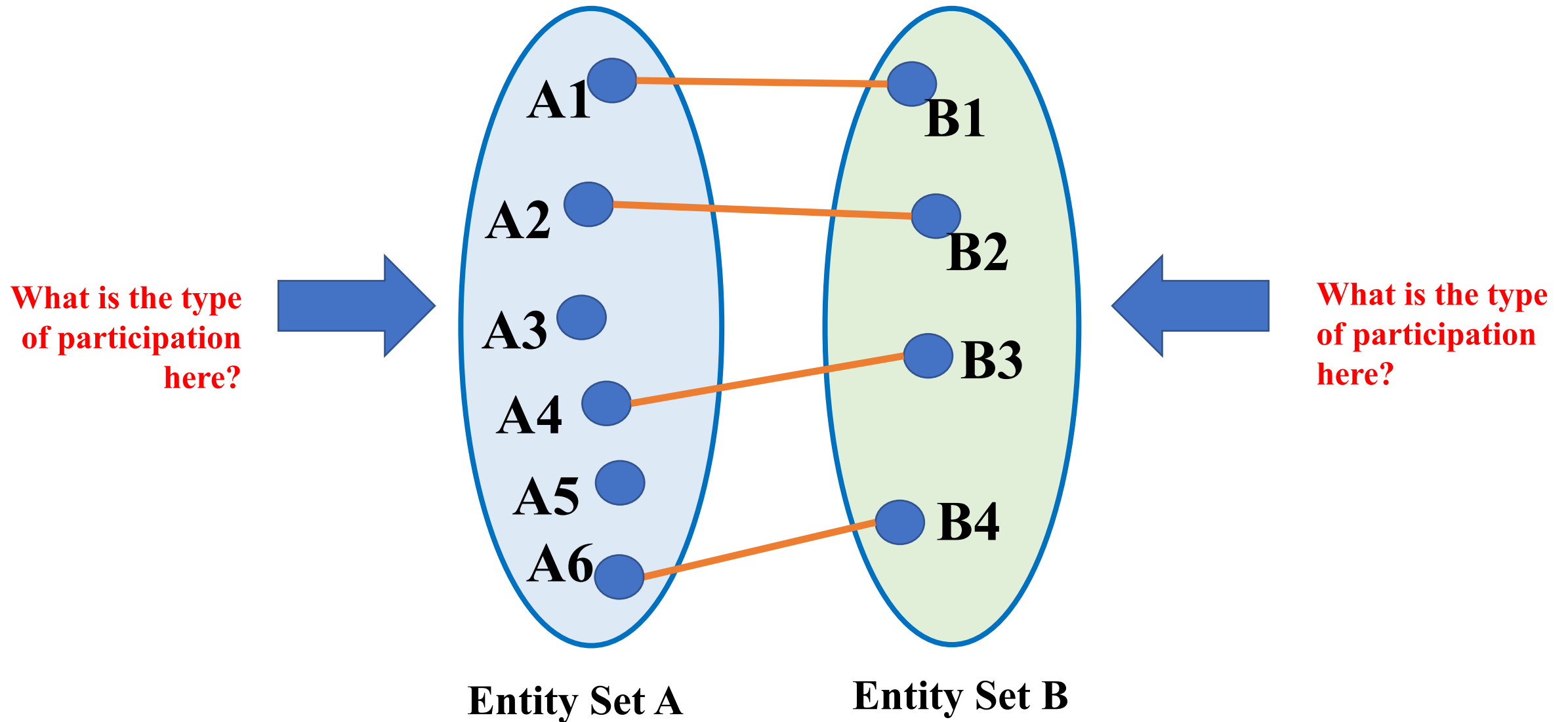
# *Total vs Partial Participation*

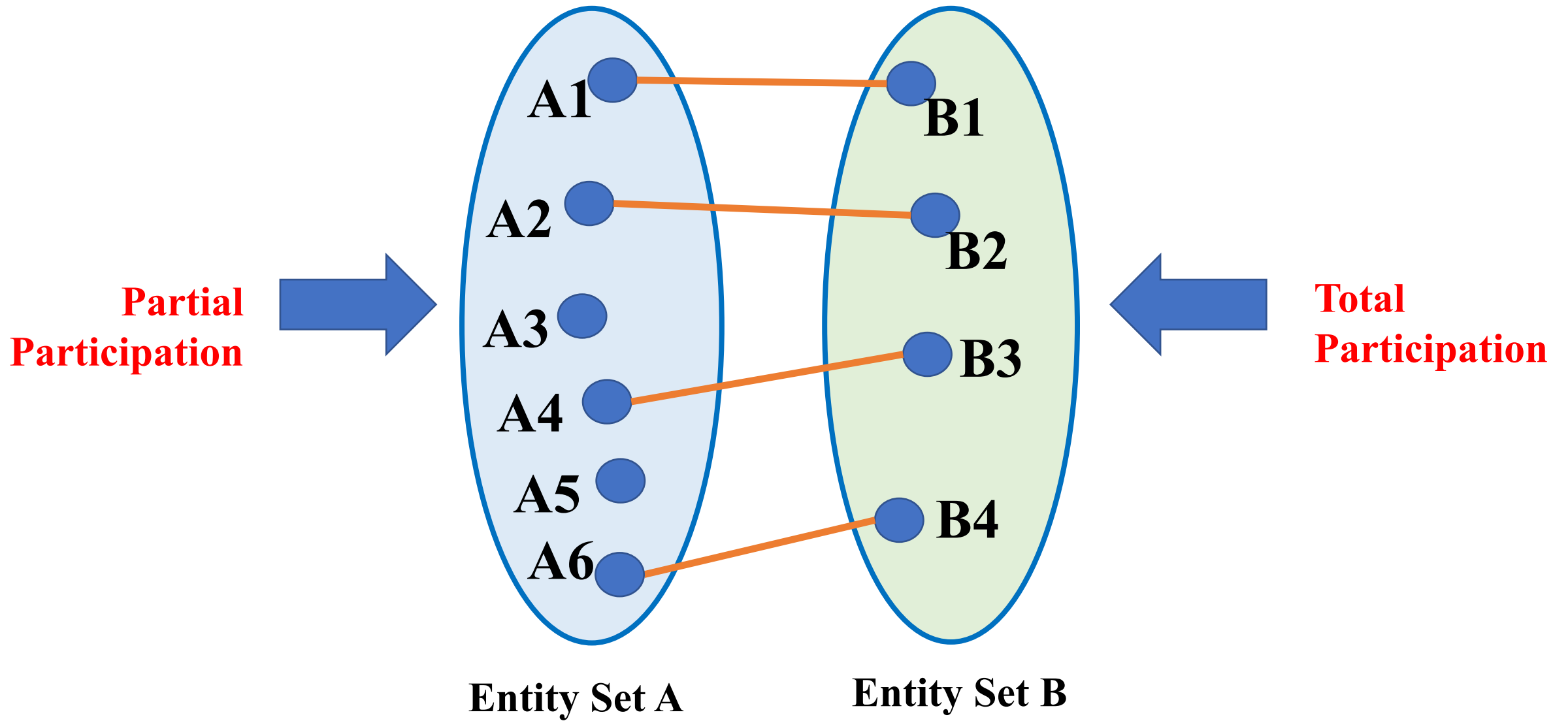
- Total participation: when *each entity* in the entity set is involved in the relationship (we represent it by a *double line*)



- Total participation: when *each entity* in the entity set is involved in the relationship (we represent it by a *double line*)
- Partial participation: when *not all entities* are involved in the relationship (we represent it by a *single line*)

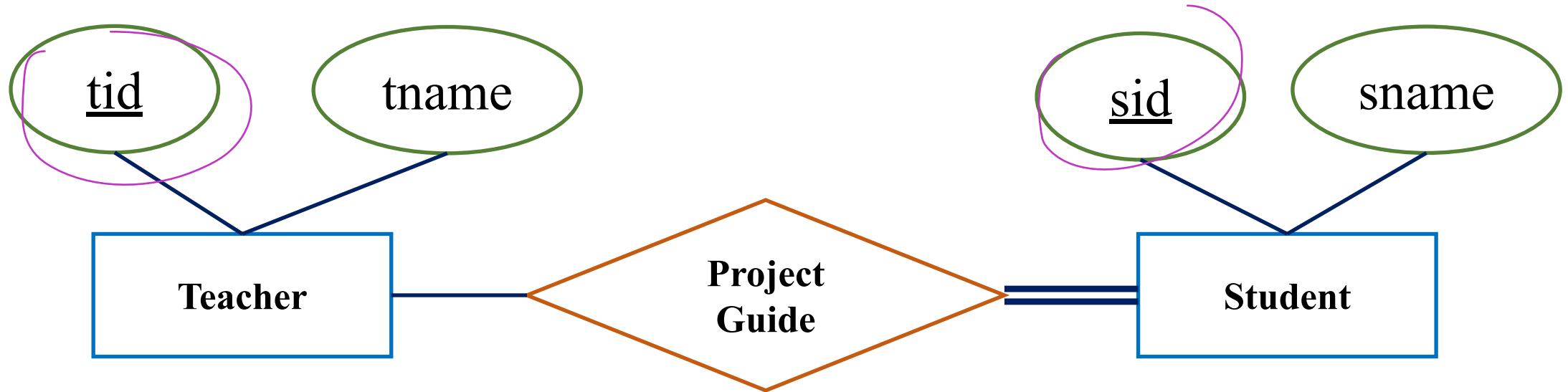




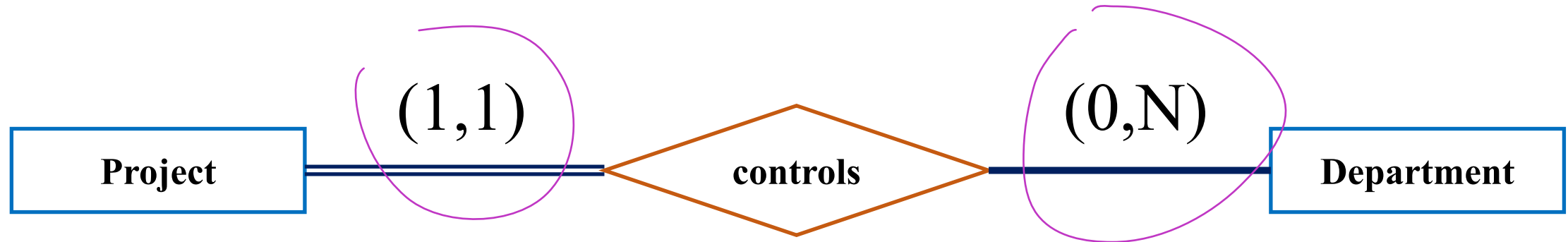




- Total participation of Student entity set means *every entity (student)* must have a teacher who guides them.
- Partial participation of Teacher entity set means *some entities (teachers)* may not guide students.

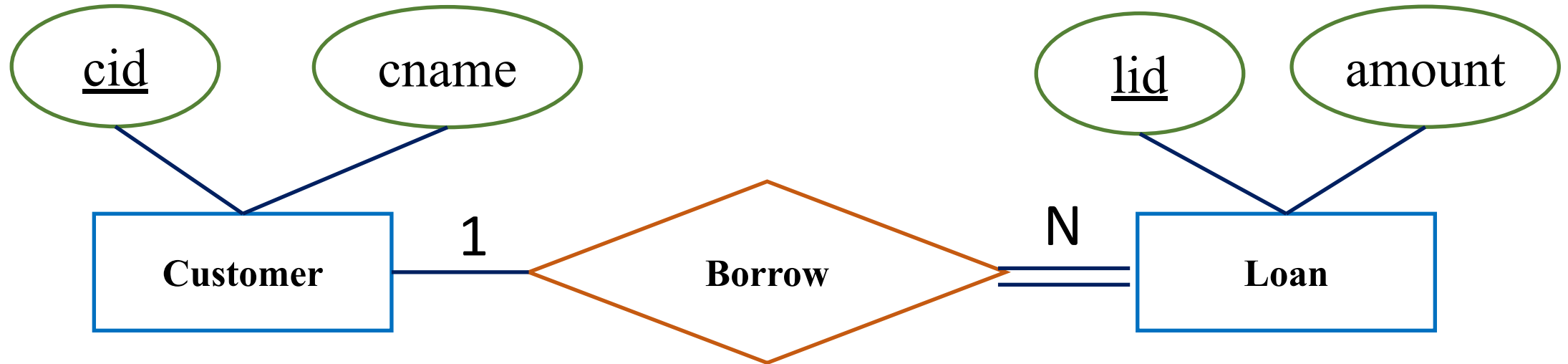


- Using the (min, max) annotation, If ***min*>0**, the participation is ***total***; while if ***min*=0** means ***partial participation***.
- The max is denoted N when we mean it can be ***>1***.
- The participation of weak entity type is always ***total***.



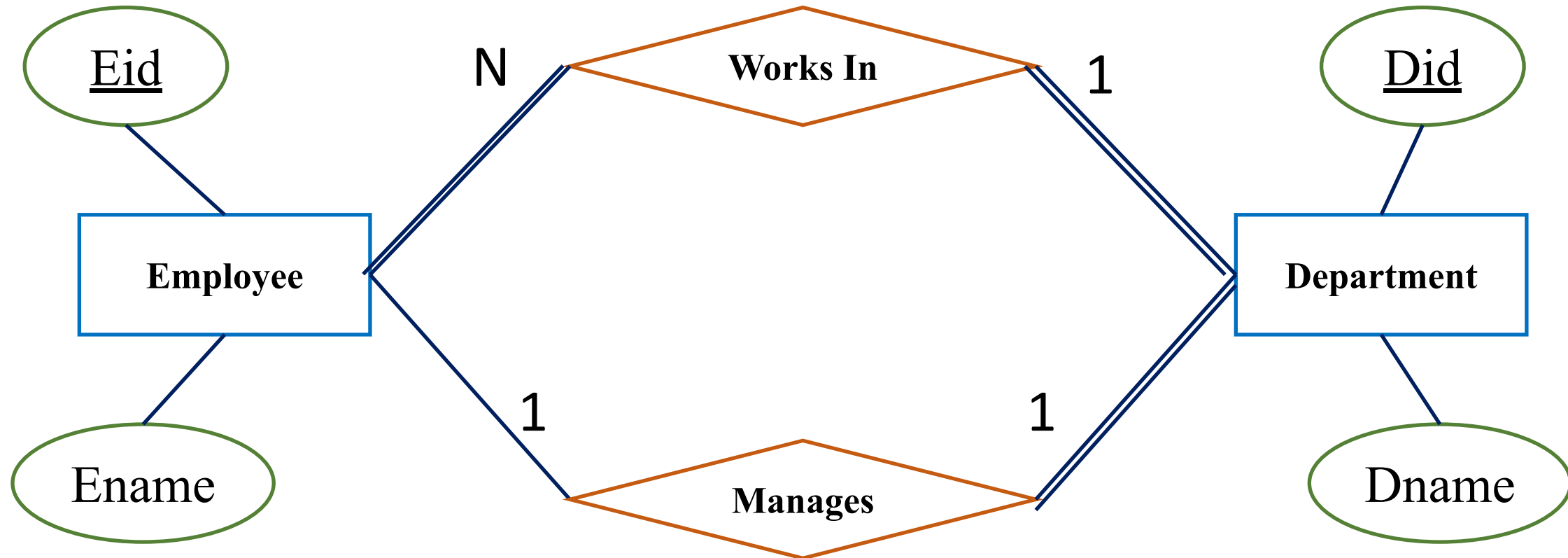
You are asked to design a database for a bank, draw an ER diagram that captures the following requirements:

- The customers are uniquely identified by their customer ID and they also share their names.
- The loans are uniquely identified by the loan ID and they also have dollar amount.
- A customer can borrow loans, where every loan must be linked to one customer, but some customers of that bank may not borrow loans.



You are asked to design a database for a company, draw an ER diagram that captures the following requirements:

- The employees are uniquely identified by the IDs and they also share their names
- The departments are uniquely identified by the IDs and they have names.
- An employee only works for one department, and each department contains one or more employee
- A department is managed by only one employee, and an employee can manage at most one department.



- Relationship vs Relationship set
- Degree of Relationship
- Mapping Cardinalities
- (Min:Max) annotation
- Total and Partial participation

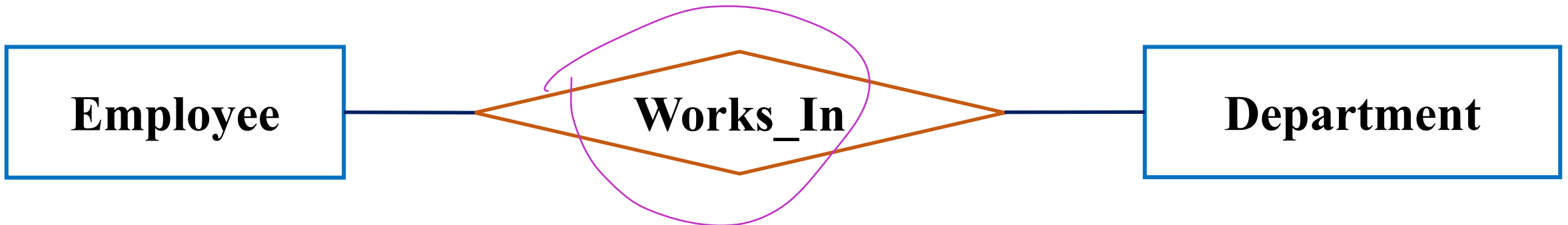
# *Strong vs Weak Entity Set*



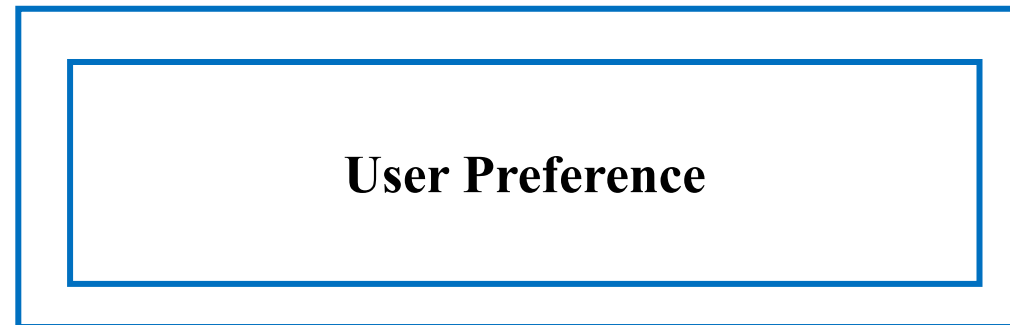
- An entity set is referred to be a **strong** if its existence **does not depend** upon the existence of another entity set.

**Students**

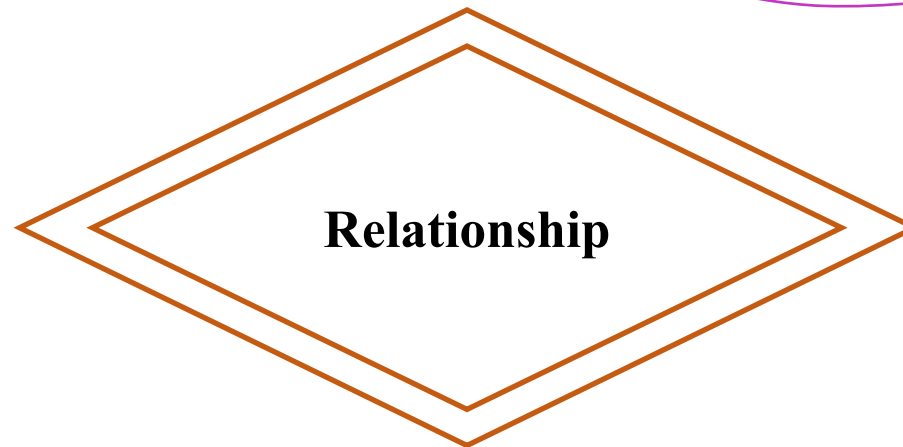
- Strong entity sets are linked together through **regular relationships** - the type of relationships we discussed so far.



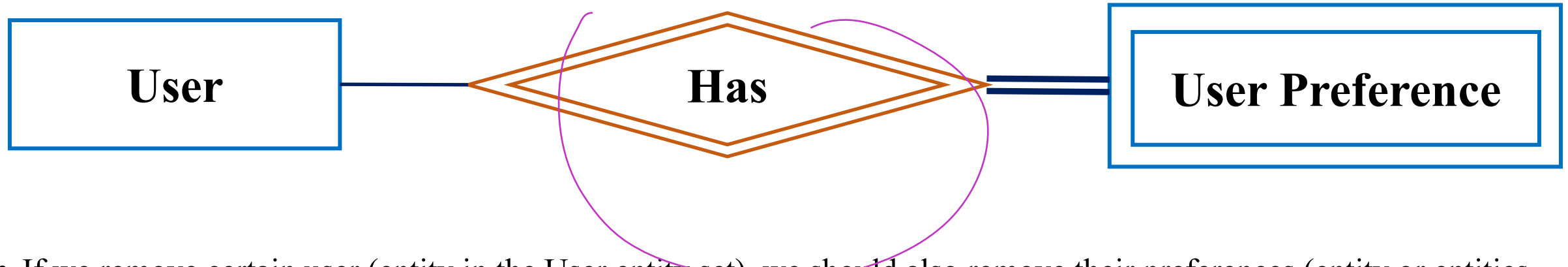
- An entity set is referred to be a *weak* if its existence *depends* upon the existence of another strong entity set.
- User Preference is an example of a weak entity set, as it depends on the existence of another entity set that represents the User.
- The weak entity is represented as follows:



- Weak entity sets are sometimes referred to as *child or dependent* entities and strong entity sets as *parent or dominant* entities
- A weak entity set is linked to its strong entity set through a relationship called *identifying relationship* and is represented as follows:



User (strong entity) has a User Preference (weak entity) can be represented as follows:



- If we remove certain user (entity in the User entity set), we should also remove their preferences (entity or entities in the User Preference entity set).
- If we remove a preference for a user, this doesn't require the removal of the user.

- A company may store the information of dependents (Children) of an Employee and the dependents don't have existence without the employee.
- Employee (strong entity) has dependent (weak entity) can be represented as follows:



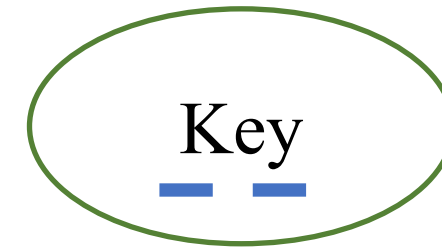


- The Employee (a strong entity) will be *identifying entity* for Dependent (a weak entity).
- Strong entity is uniquely identifiable using *primary key attribute(s)*.
- But we can only uniquely identify each weak entity *through the identifying relationship* that a weak entity has with a strong entity.

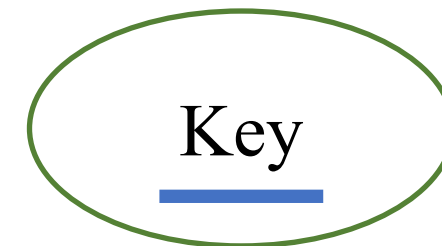
- In another word, there is no *primary key* for the weak entity.
- Weak entity does not have a primary key, instead it has a *partial key* that uniquely differentiates between the weak entities.

The *primary key* of a weak entity is a *composite key* formed from the *primary key of the strong entity* and the *partial key of the weak entity*.

- Partial Key, for a weak entity set, is represented with a dotted line

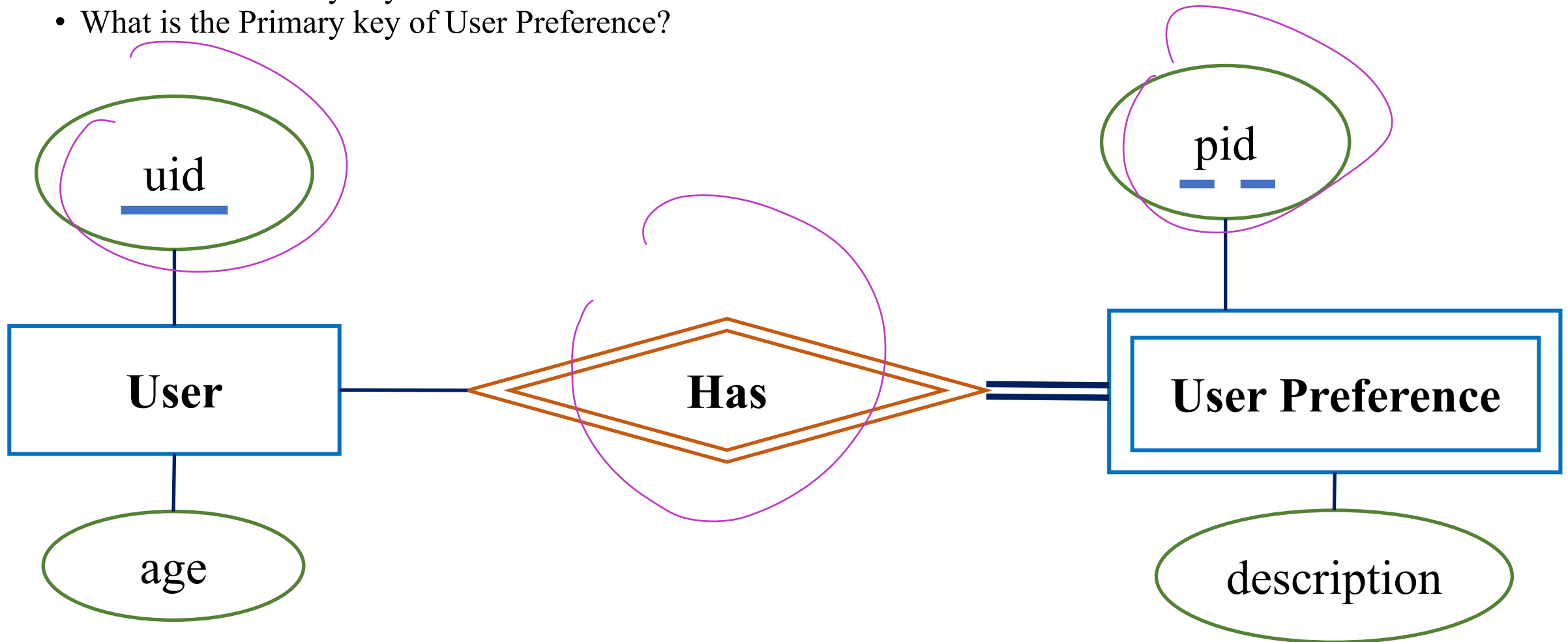


- Primary Key, for a strong entity set, is represented with a line

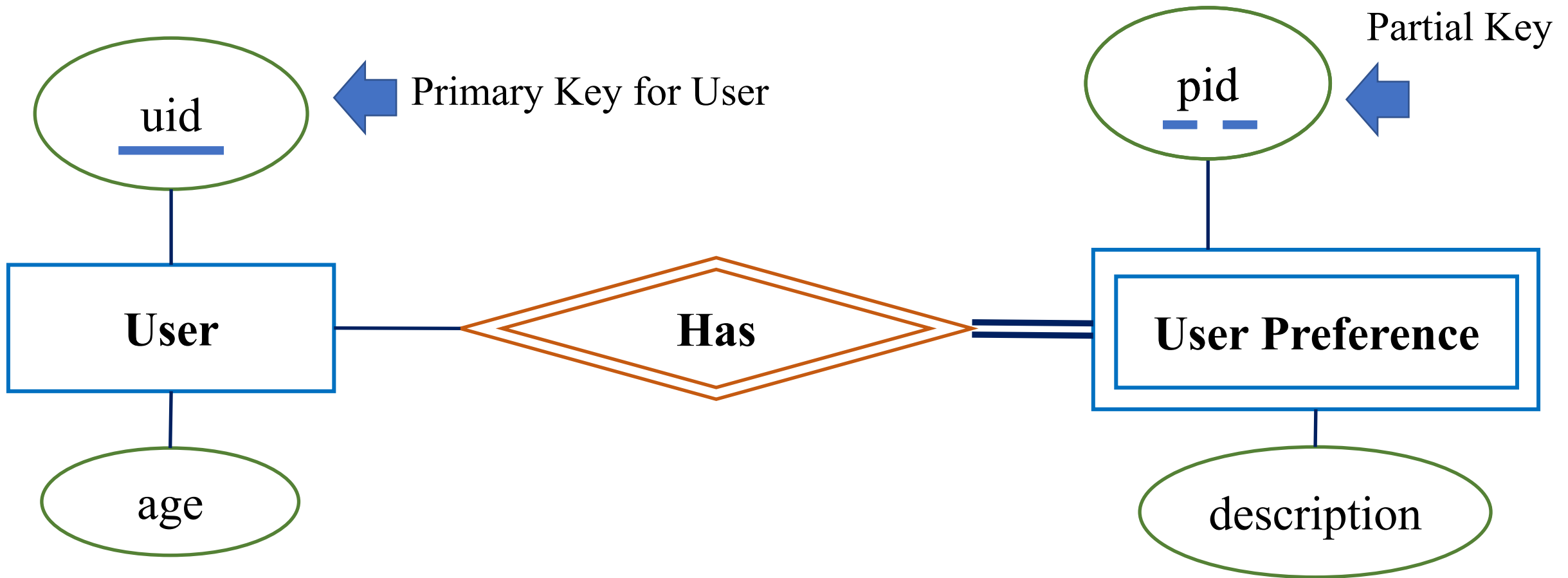




- Image the following scenario:
  - What is the Primary key of User?
  - What is the Primary key of User Preference?

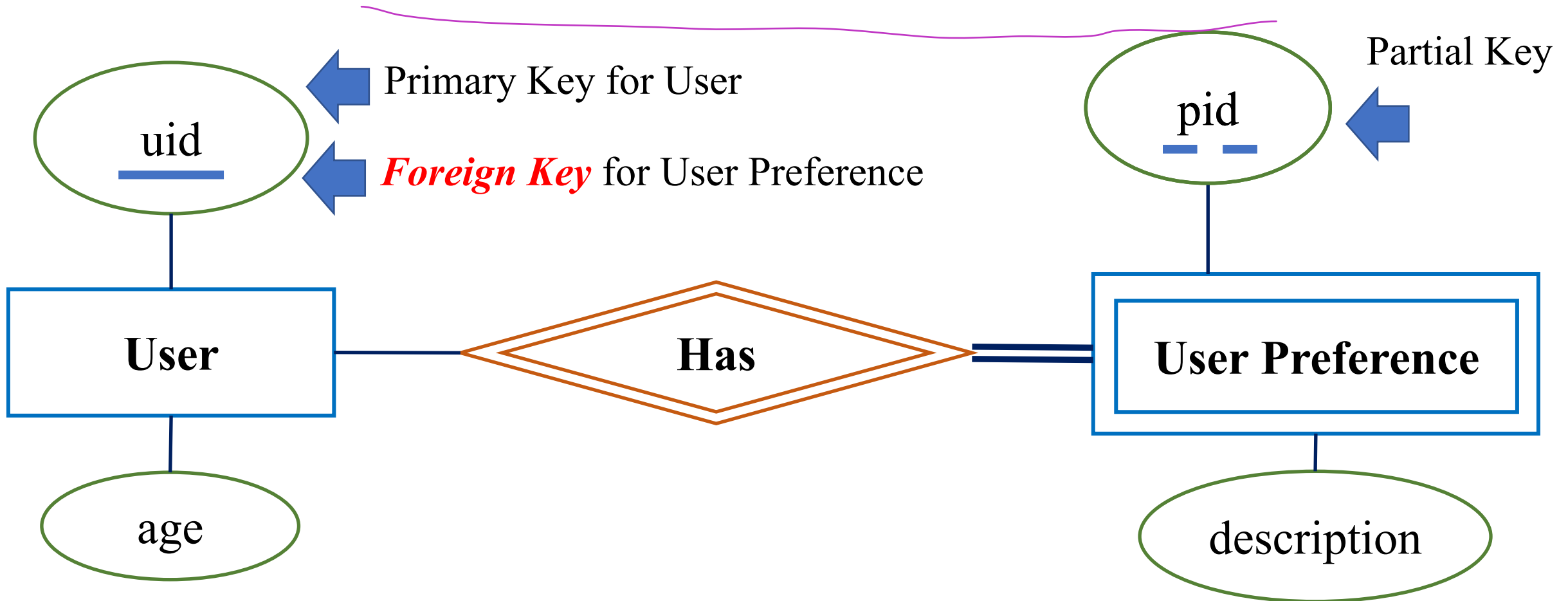


- Primary key of User: **uid**
- Primary key of User Preference: (**uid** of *User* + **pid**)

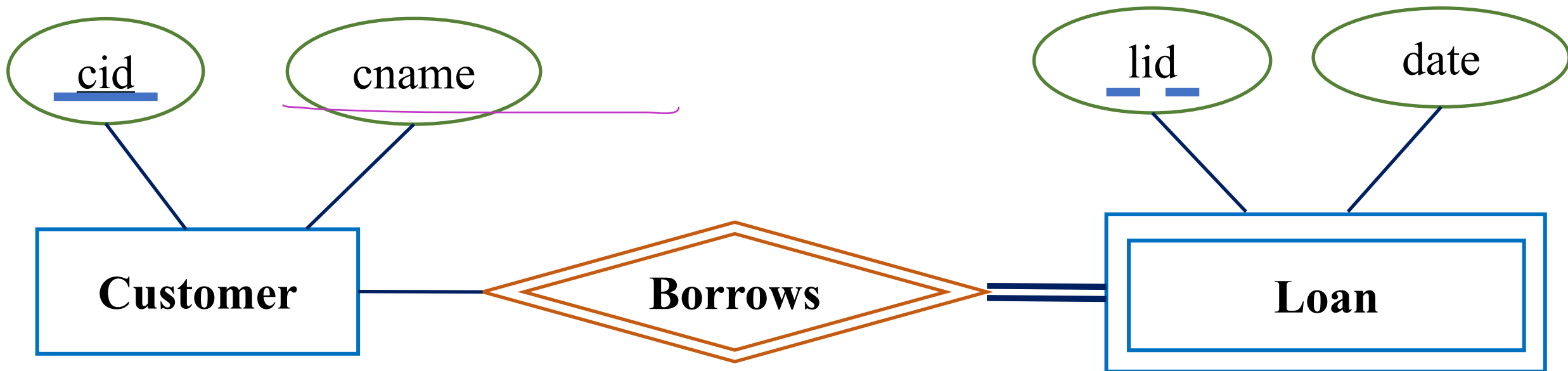


- Since uid is taken/used from one entity set to identify another entity set, we can refer to uid as:

a *primary key* for User and a *foreign key* for User Preference

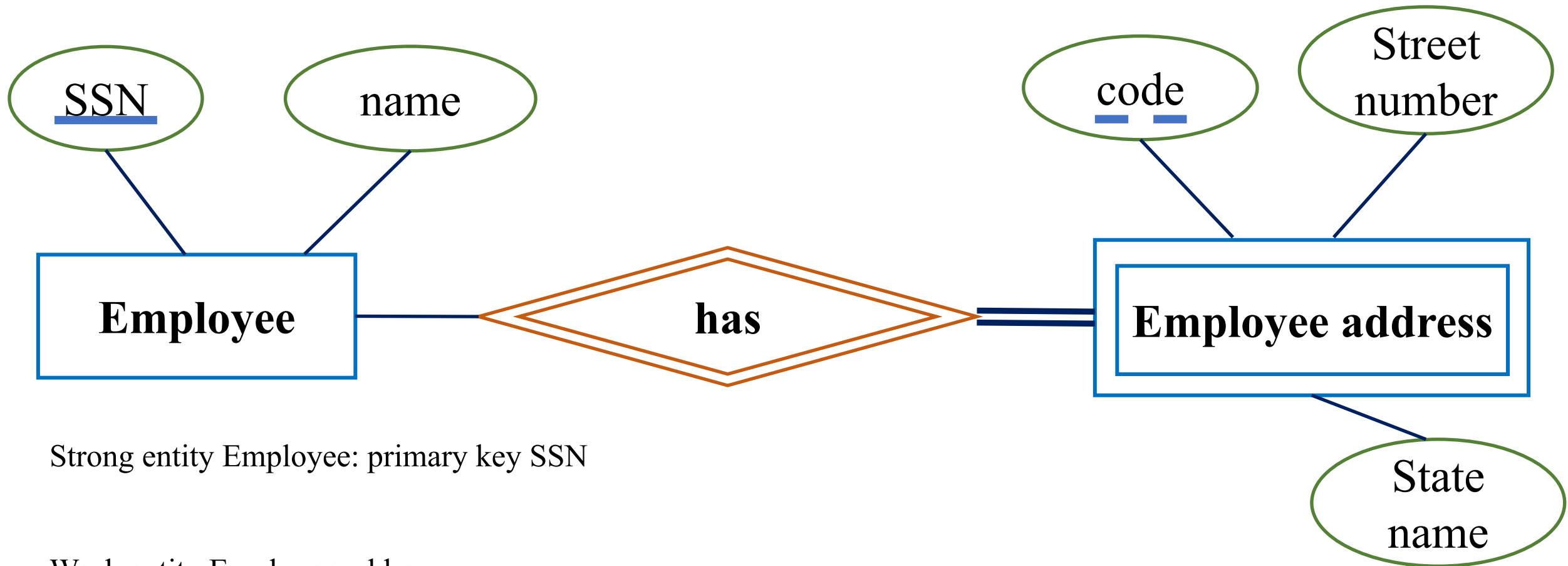


- Image the following scenario:
  - Primary key of Customer: **cid**
  - Primary key of Loan: (**cid** of *Customer* + **lid**)



You are asked to design a database for a company, draw an ER diagram that captures the following requirements:

- The employees are uniquely identified by the SSNs and they also share their names
- Employee addresses are identified by address codes and each address also has street number and state name.
- The Employee is *identifying* the Employee address through *has* relationship, and each employee may have more than one address to be stored in the database.

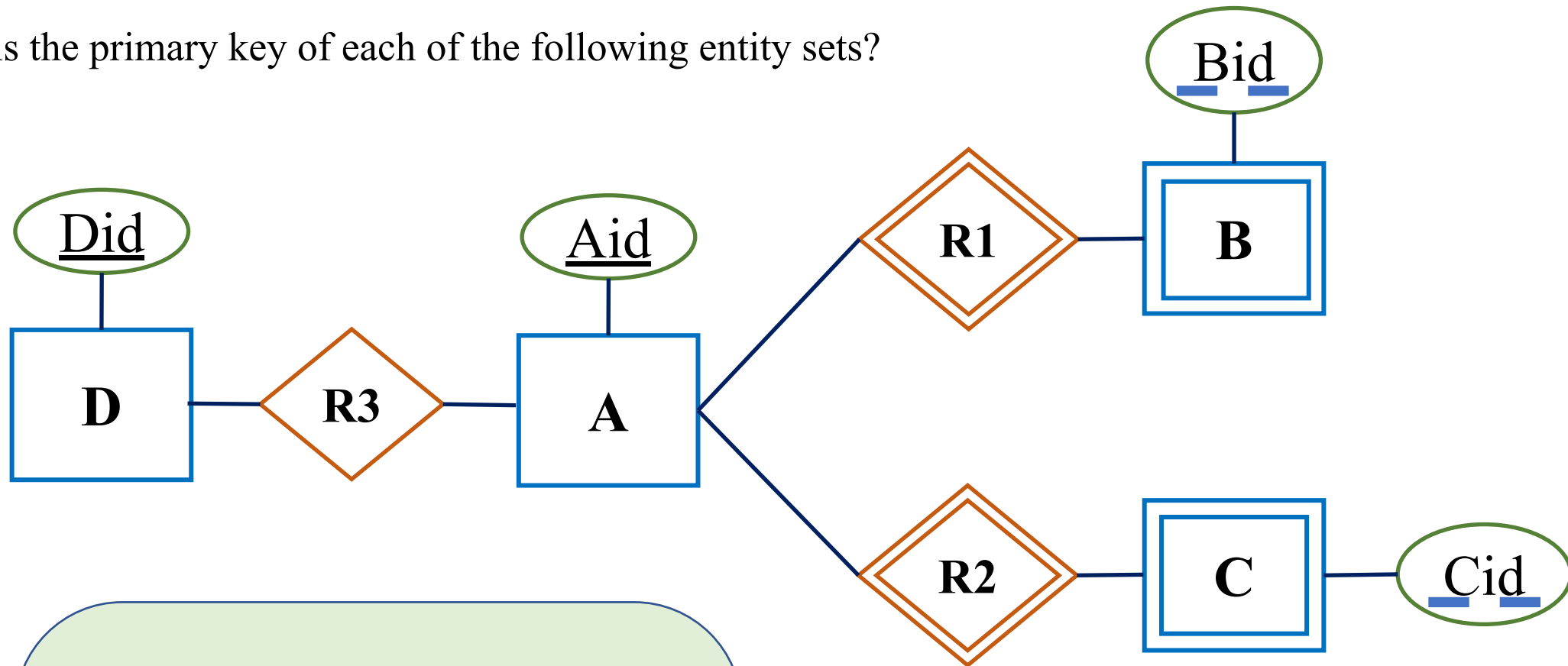


Strong entity Employee: primary key SSN

Weak entity Employee address:

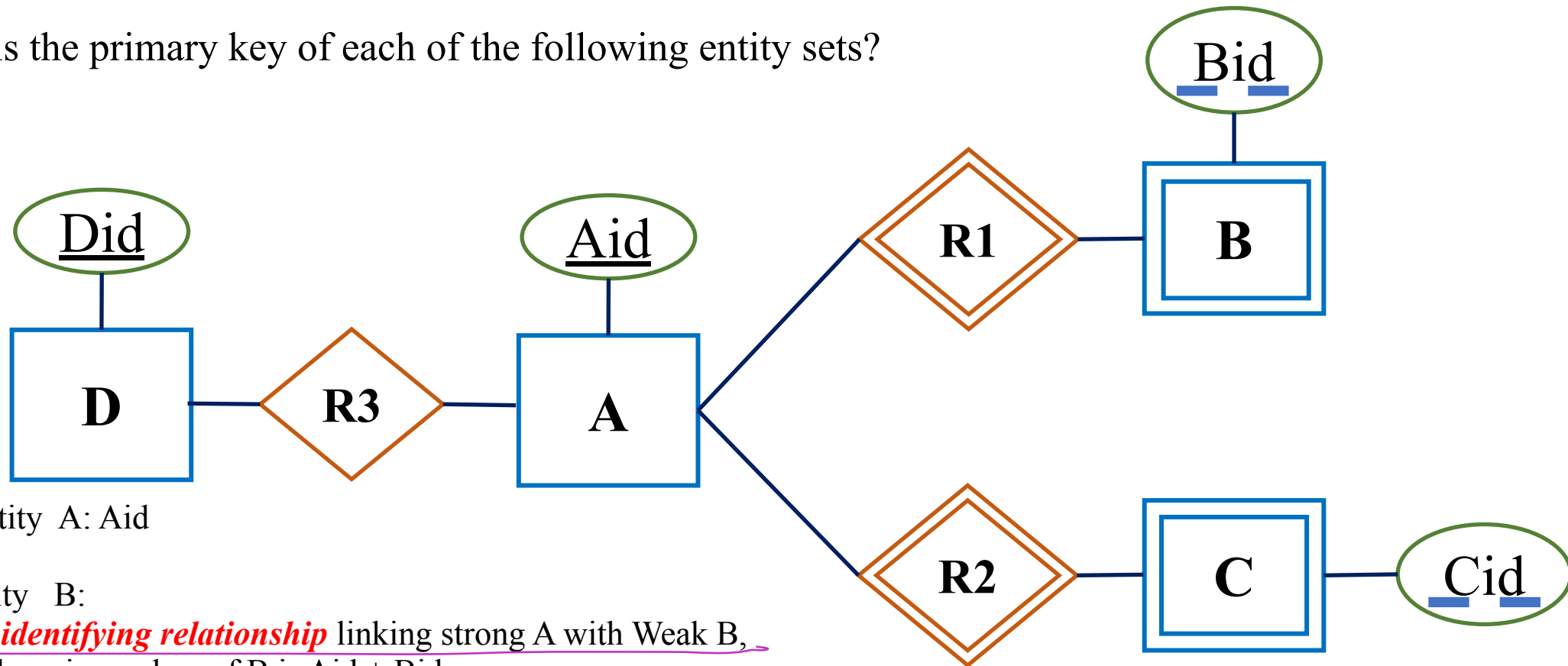
- Has is *identifying relationship* linking strong Employee with Weak Employee address,
- thus the primary key of Employee address is SSN + Code

What is the primary key of each of the following entity sets?



Can one strong entity set identify multiple weak entity sets? - yes

What is the primary key of each of the following entity sets?



Strong entity A: Aid

Weak entity B:

- R1 is *identifying relationship* linking strong A with Weak B,
- thus the primary key of B is Aid + Bid

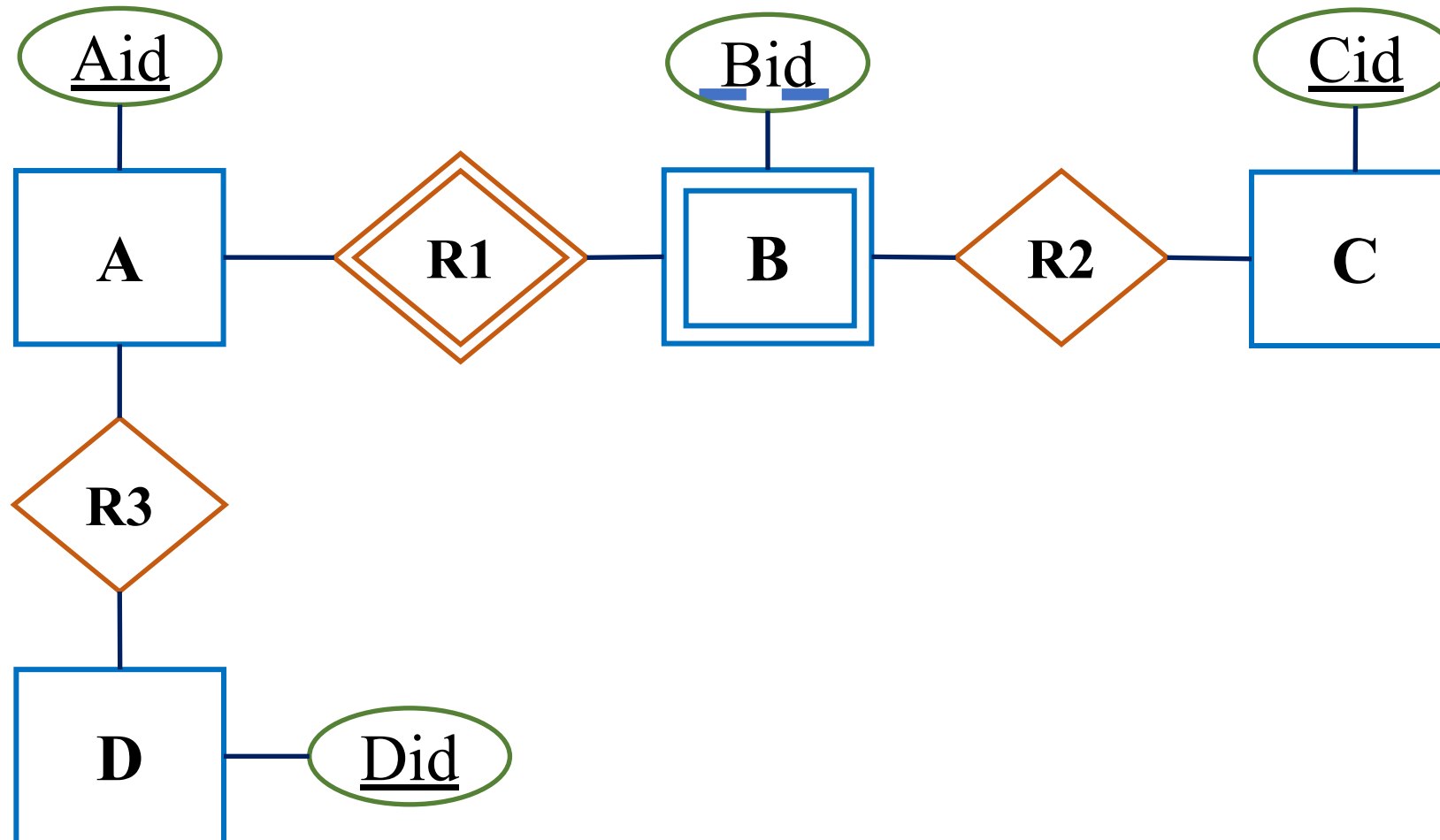
Weak entity C:

- R1 is *identifying relationship* linking strong A with Weak C,
- thus the primary key of C is Aid + Cid

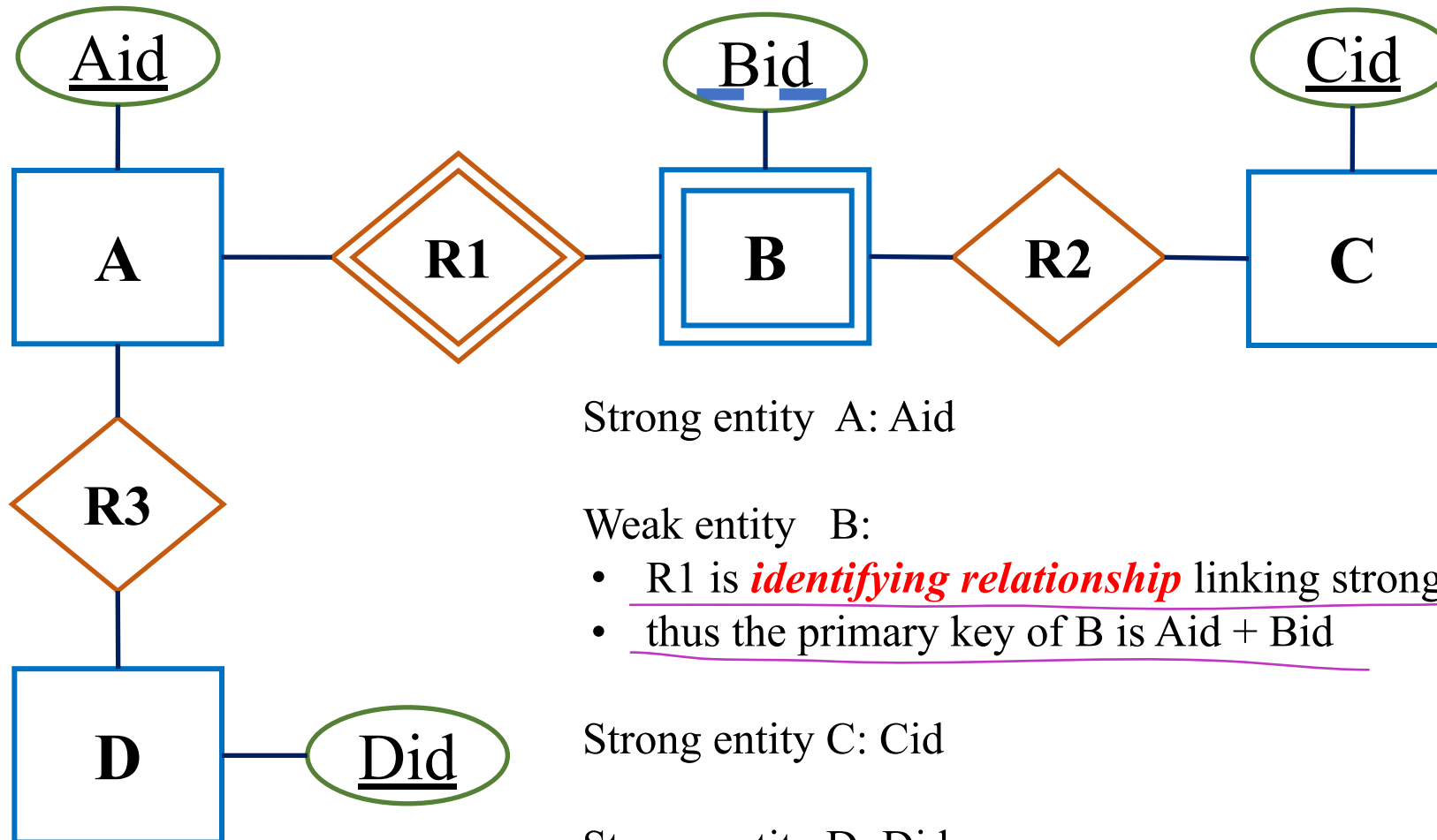
Strong entity D: Did



What is the primary key of each of the following entity sets?



What is the primary key of each of the following entity sets?



Strong entity A: Aid

Weak entity B:

- R1 is *identifying relationship* linking strong A with Weak B,
- thus the primary key of B is Aid + Bid

Strong entity C: Cid

Strong entity D: Did

# *Conceptual Hierarchical Design*

- The ER Model can express objects in a ***conceptual hierarchical manner***.
- Such hierarchical design enables the designer to express:
  - ✓ **Generalization**, where entities are grouped together.
  - ✓ **Specialization**, where entities are divided into different groups.

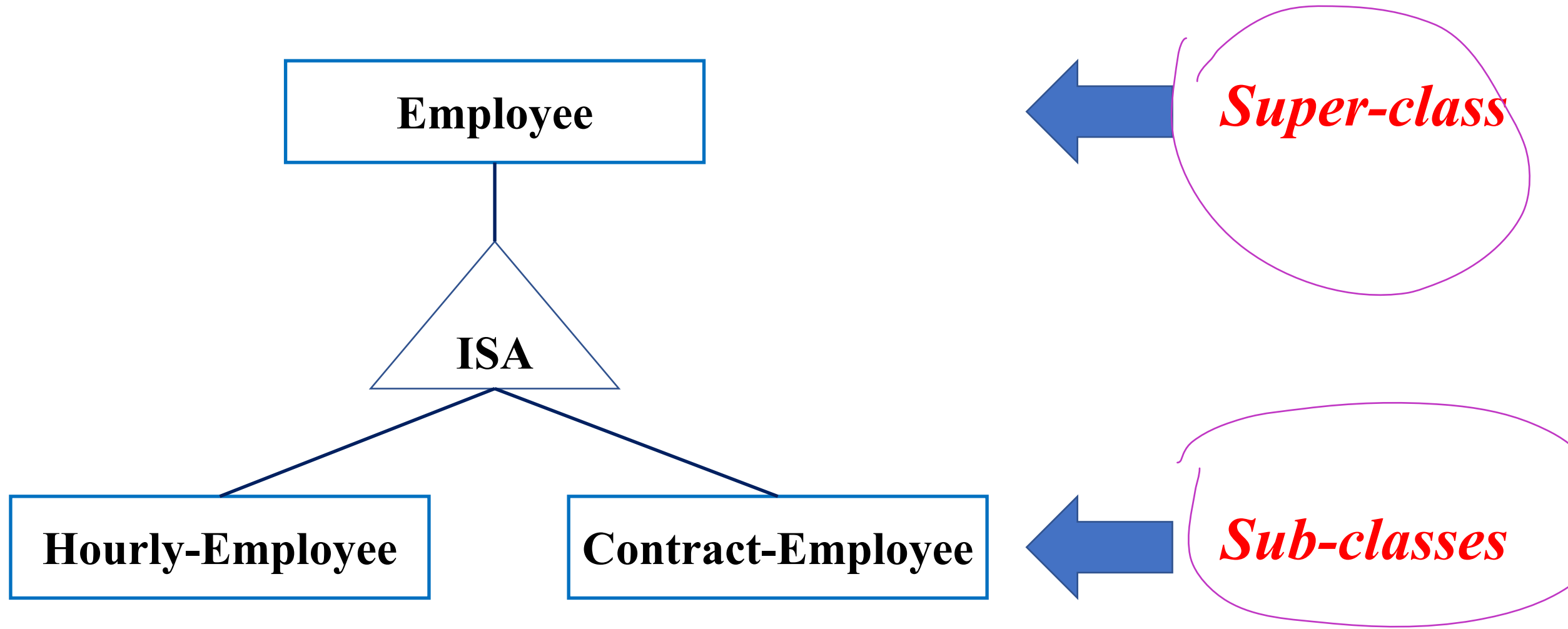
- We would like to include both *Hourly-Employees* and *Contract-Employees* and distinguish them on the basis on which they are paid.

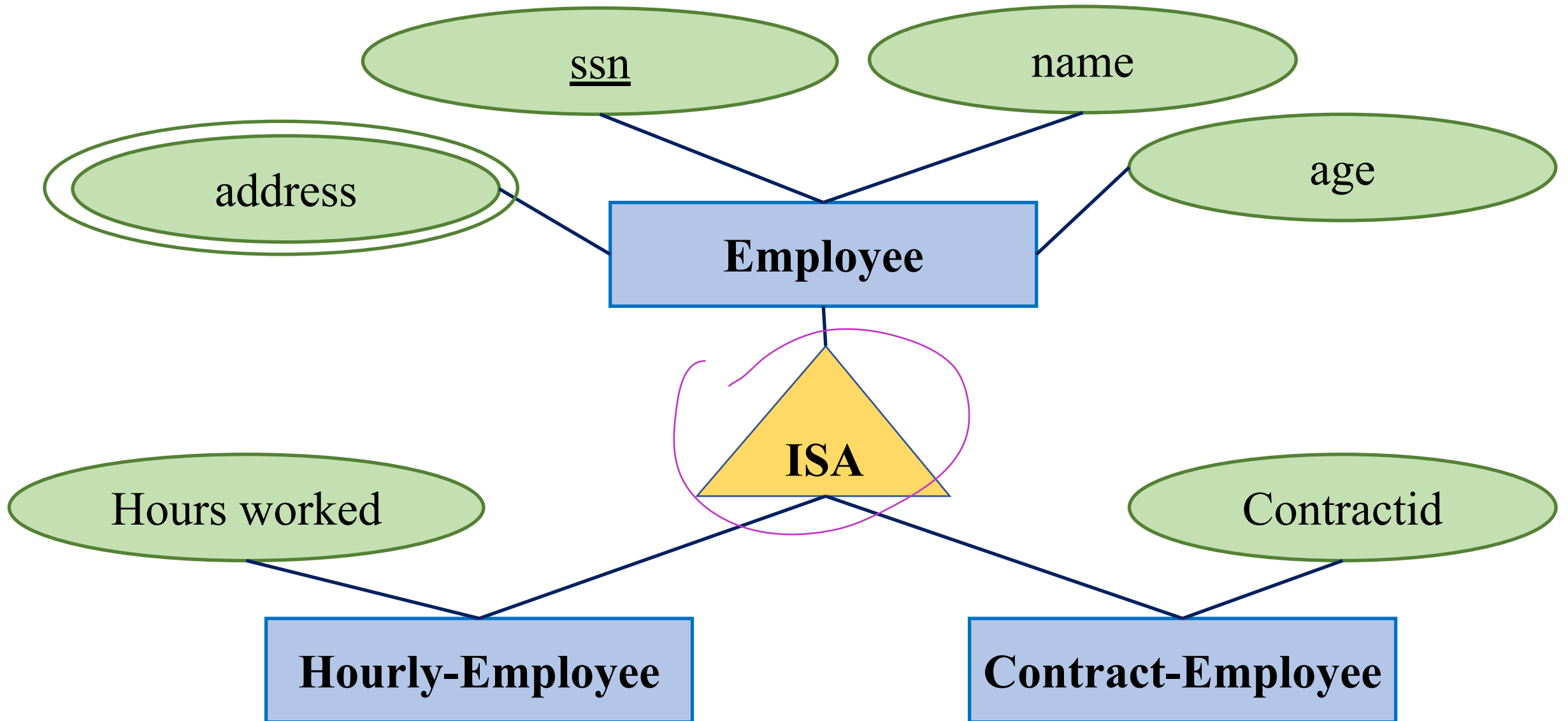
**Hourly-Employees**

**Contract-Employees**

- We might have attribute *hours\_worked* defined for Hourly\_Emps and an attribute *contract\_id* defined for Contract-Emps as special attributes for such entity sets.

- We want the semantics to represent that both are *Employees* and have all the attributes of the employee (name, id, age, address).
- We say that the attributes for Employees are *inherited* by the Hourly\_Emps and that Hourly-Emp *ISA* (read as: *is a*) Employee.
- We say that the attributes for Employees are *inherited* by the Contract\_Emps and that Contract-Emp *ISA* (read as: *is a*) Employee.

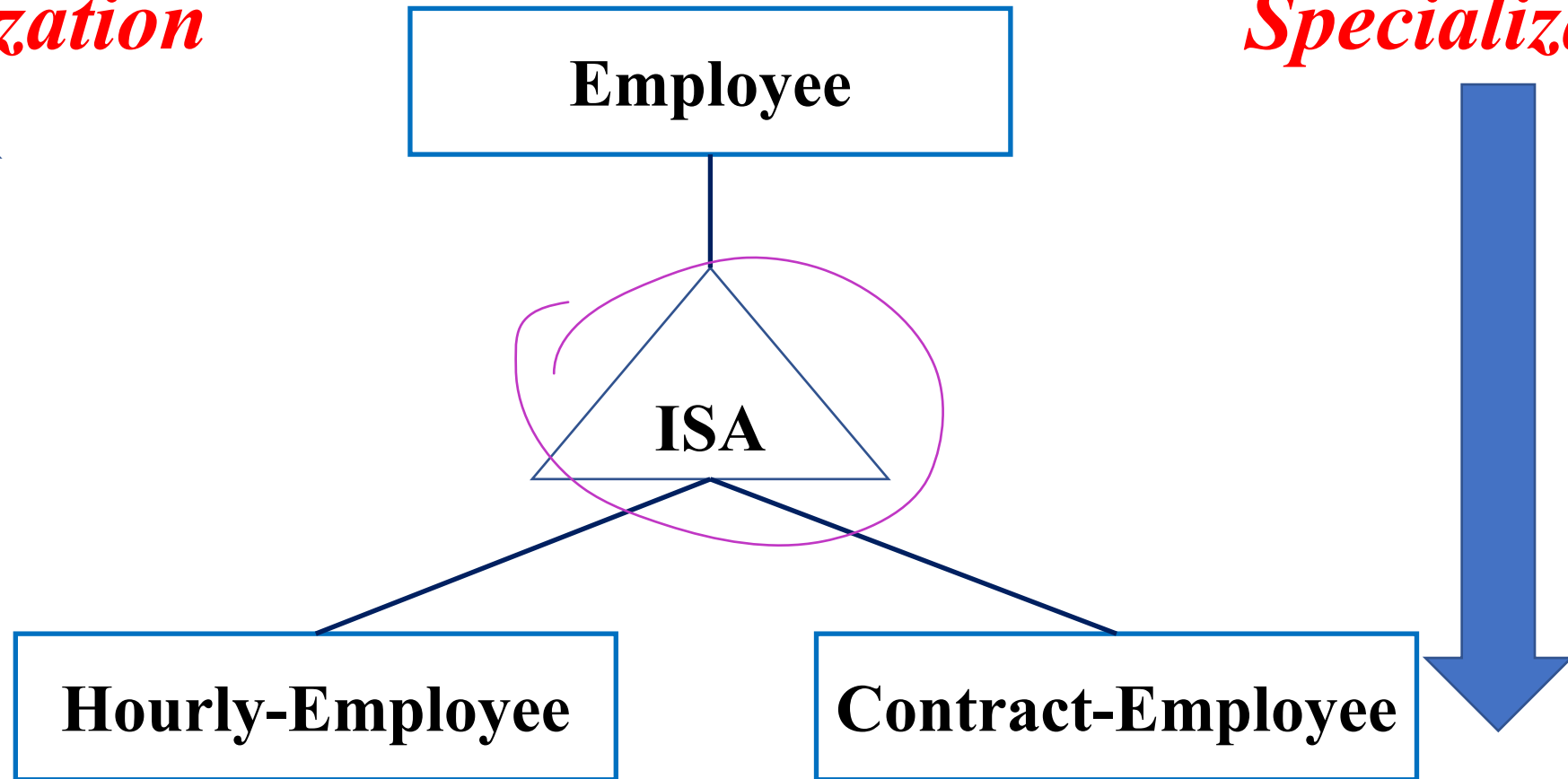


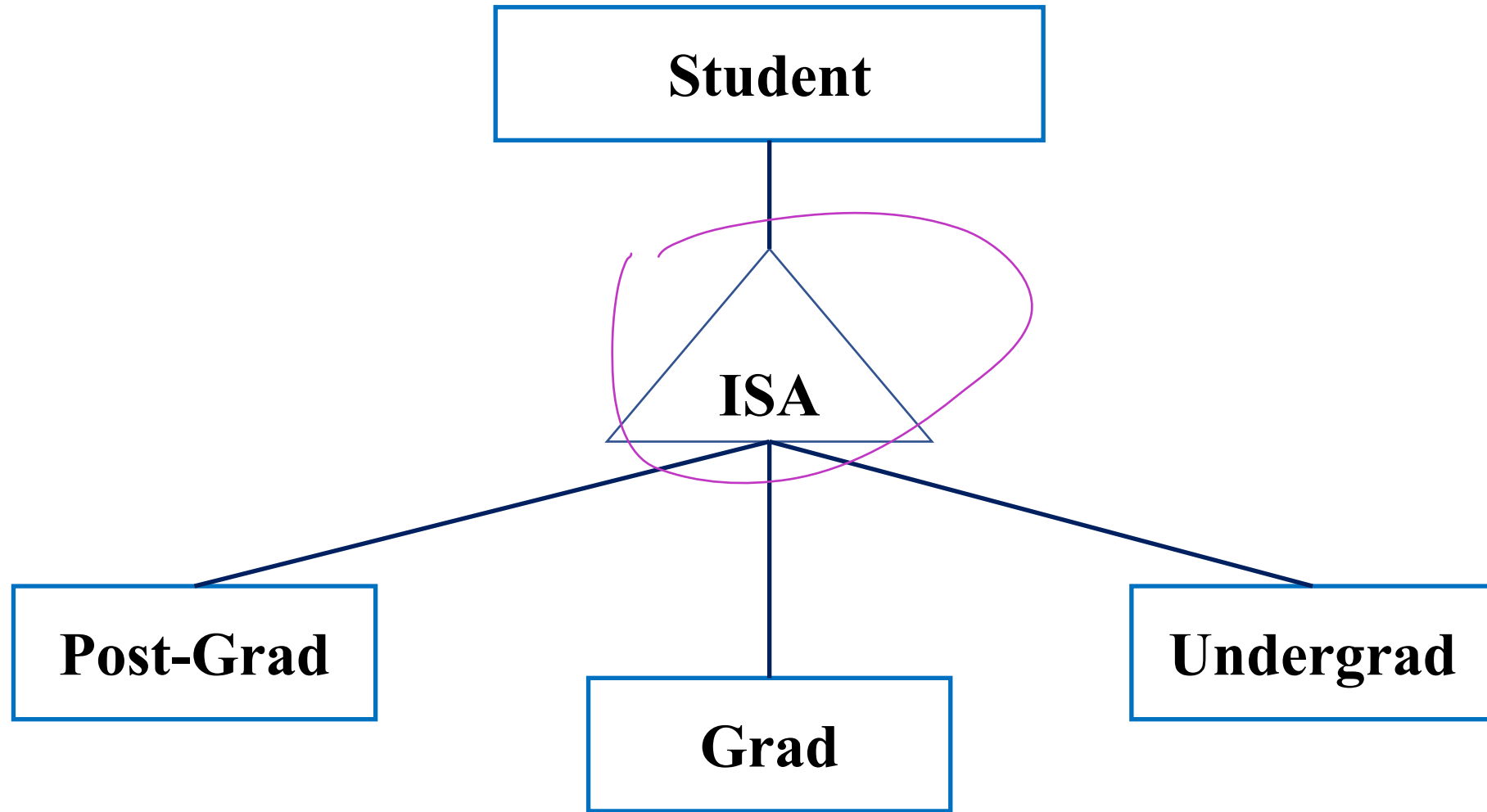




*Generalization*

*Specialization*





- **Generalization**: identifying some common characteristics of a collection of entity sets and creating a new entity set with these common characteristics.
- The different employees are **generalized** to Employee. The subclasses are defined first, then the superclass is defined next.

- ***Specialization***: identifying subsets of an entity set that share some characteristic.
- Employee is ***specialized*** into subclasses. The superclass is defined first, then the subclasses.
- As another example, Motor-boats and Cars may be generalized into an entity set Motor-Vehicles.

# *Review and Revision*

- You are asked to design a database for a company, draw an ER diagram that captures the following requirements:
- The database needs to store information about **employees** (identified by *ssn*, with *salary* and *phone* as attributes), **departments** (identified by *dno*, with *dname* and *budget* as attributes), and **profiles** of employees (identified by *id*, with *name* and *url* as attribute).
- The employee only **works** for one department, and the department contains one or more employee.
- The department is **managed** by only one employee, and an employee can manage at most one department.
- If you remove employee's record, you should remove their profile (we can assume that each employee can have more than one profile).

In order to translate this description (user requirements) to an ER diagram, we can divide this into few steps:

Step 1: identify the entities

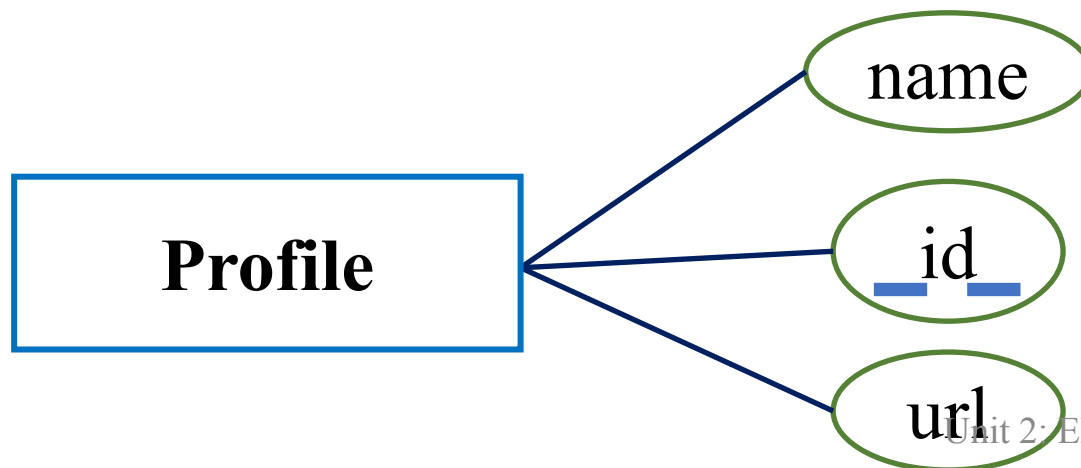
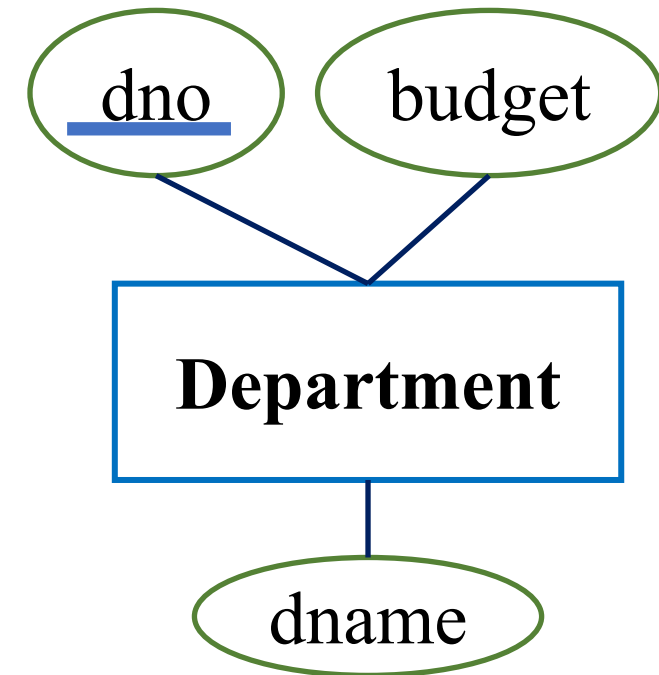
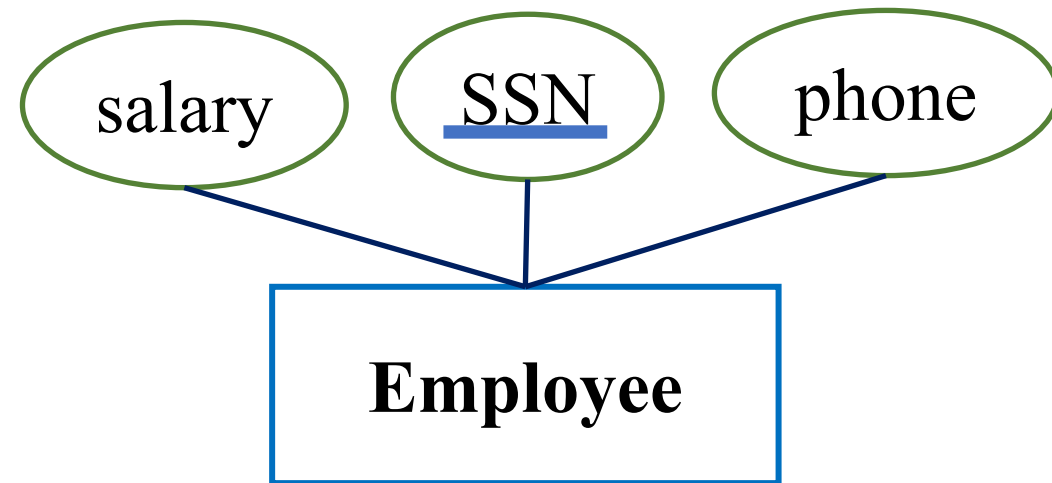
**Employee**

**Department**

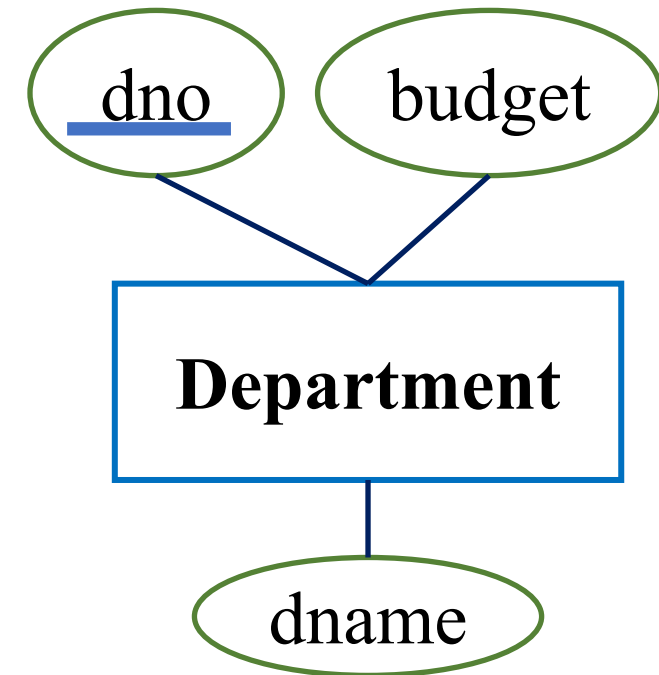
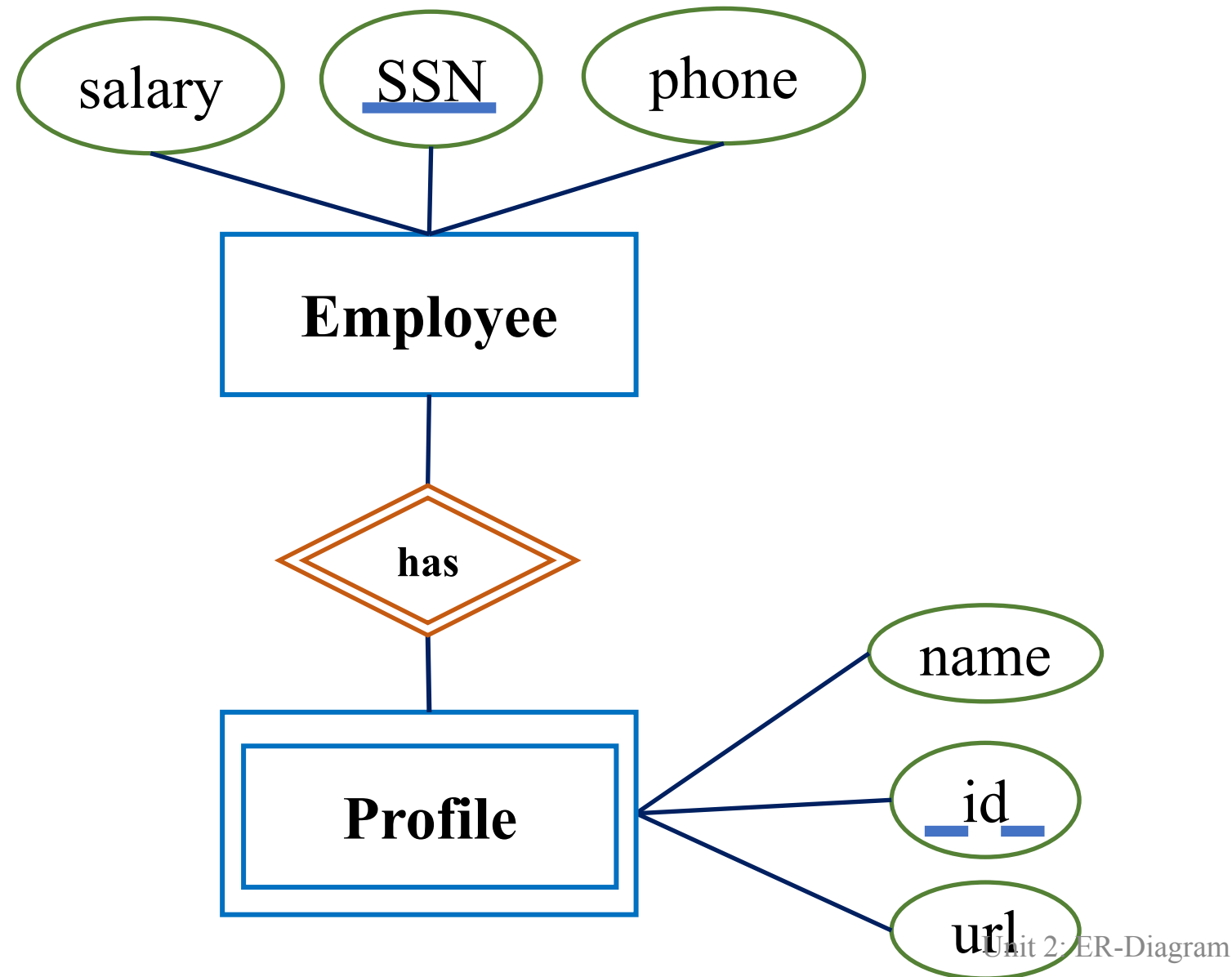
**Profile**



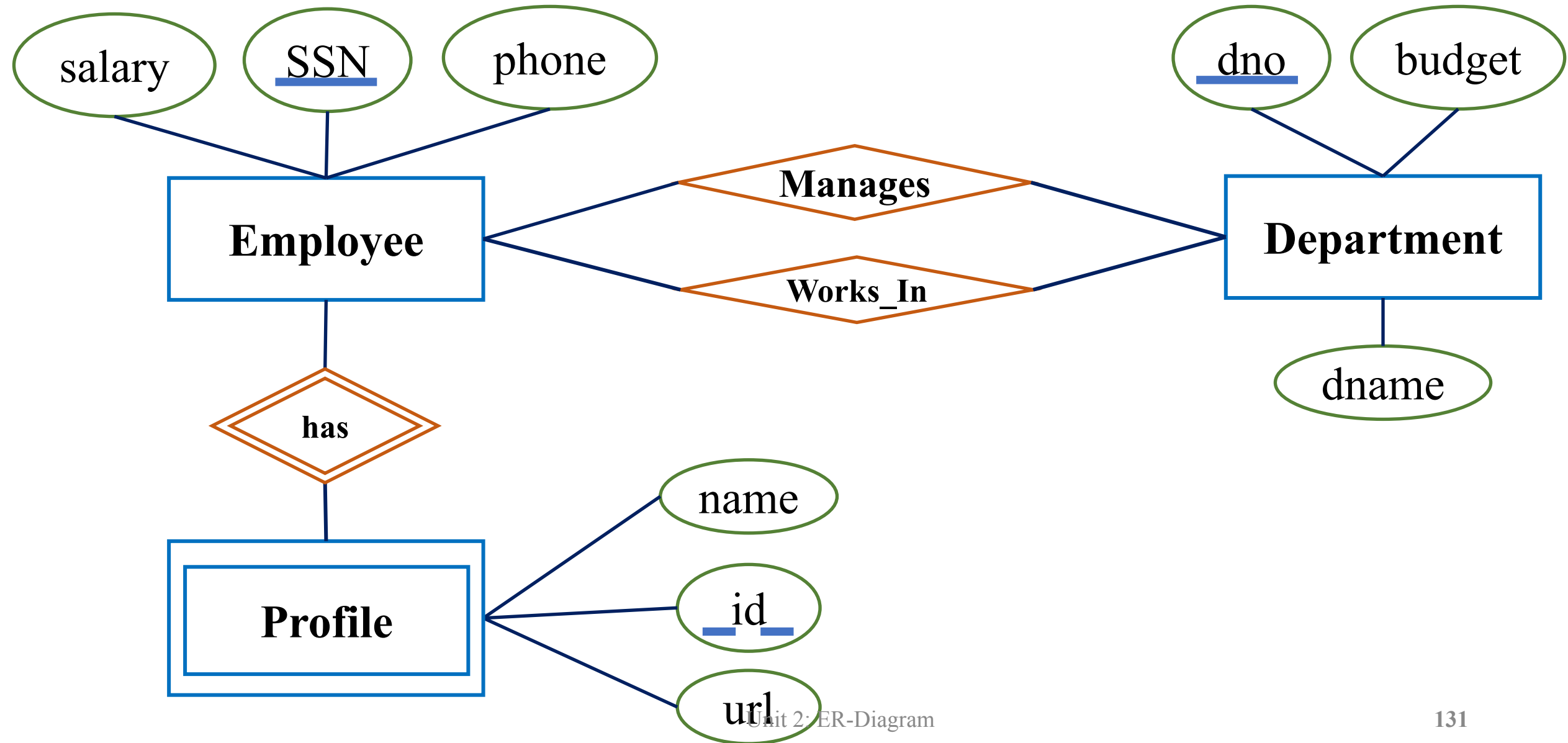
Step 2: identify the attributes



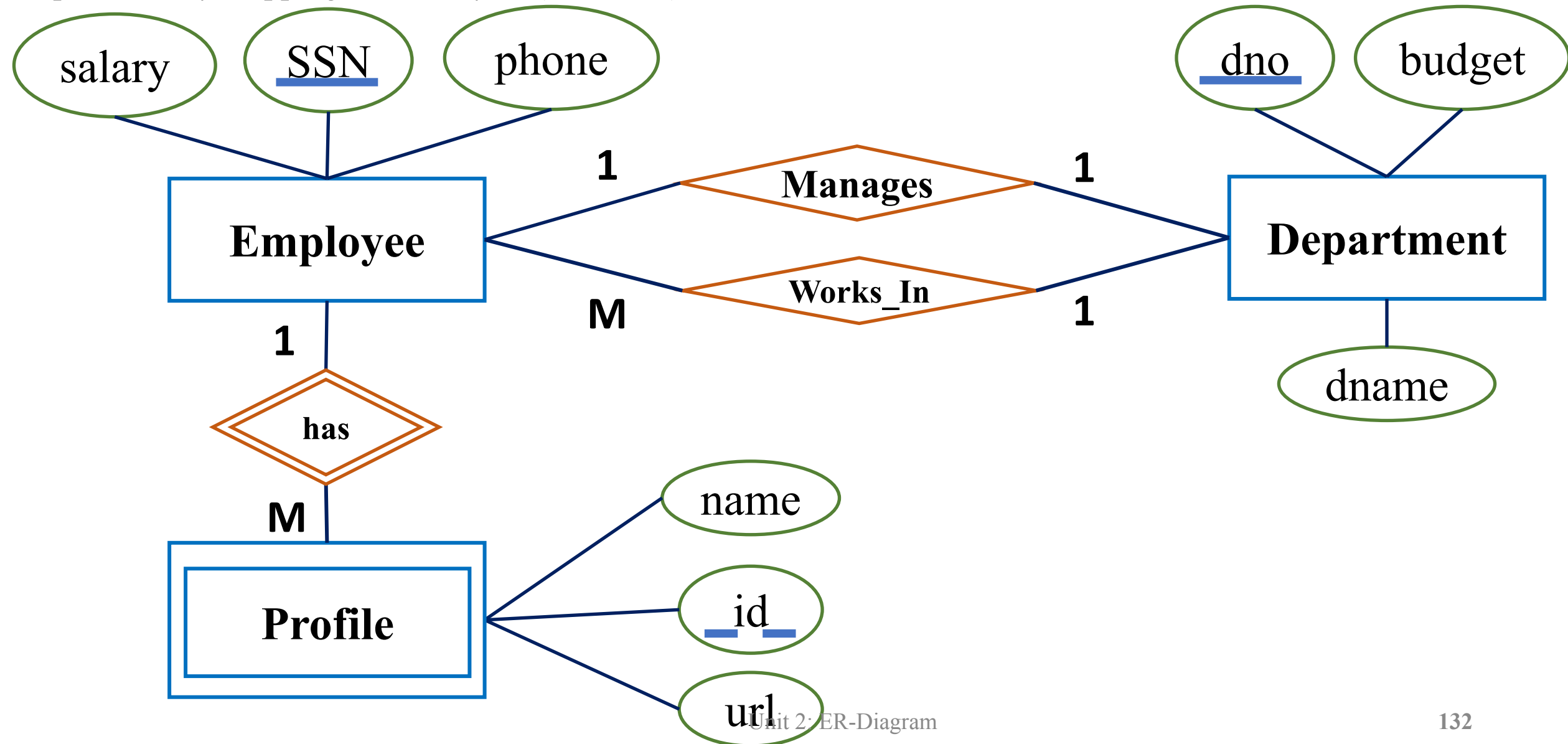
Step 3: identify weak entities and the identifying relationships



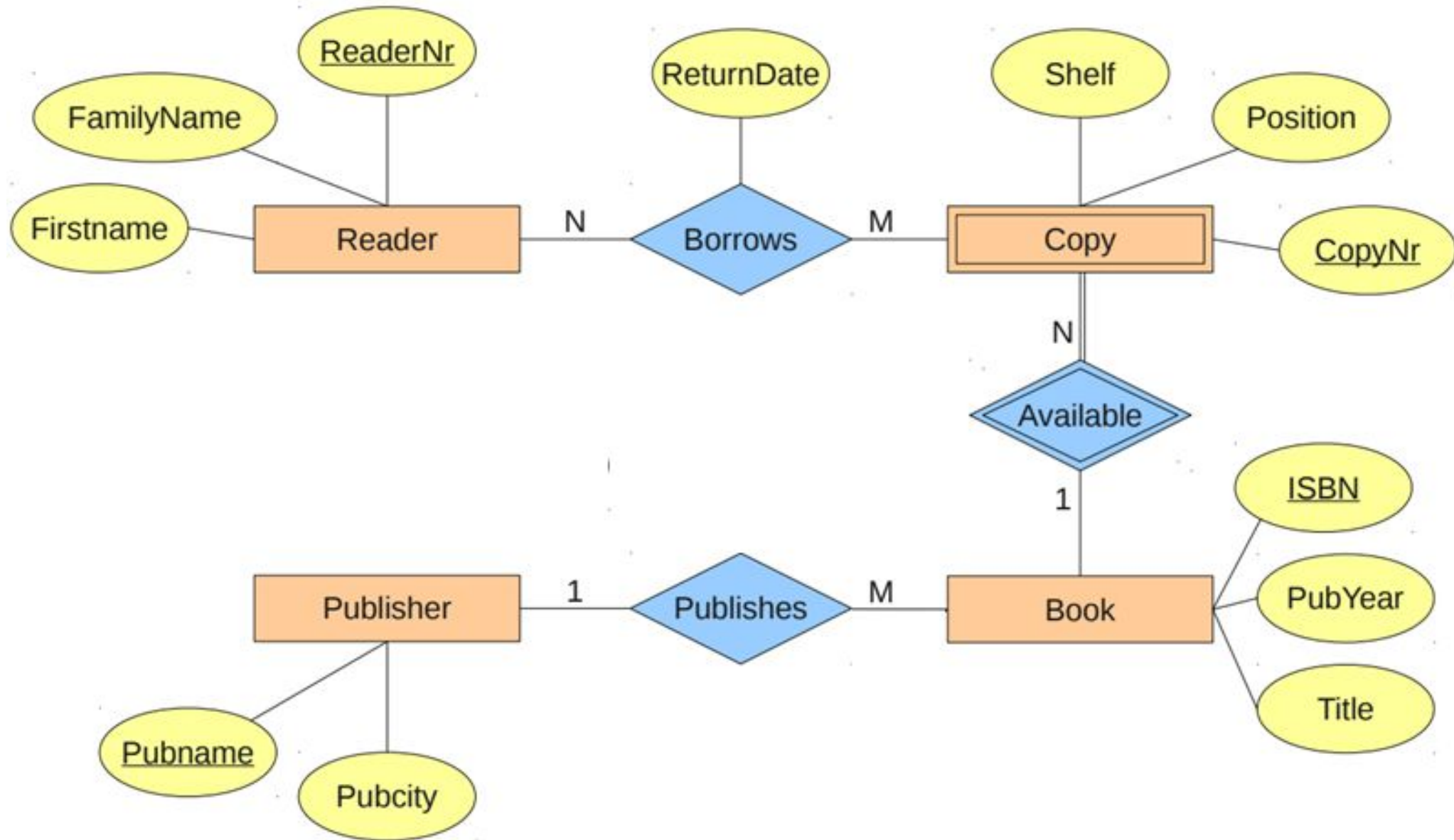
Step 4: identify the normal relationships



Step 5: identify mapping cardinality or (min, max) annotation



- You are asked to design a database for a library, draw an ER diagram that captures the following requirements:
- Each *book* is described by title and year of publication and is also uniquely identified using the ISBN.
- Each *book copy* is described by shelf and position and is identified using the copyNr.
- Deleting the information of a book requires the deletion of the info of the corresponding copies. Each Book can have more than one copy.
- Each *publisher* is described by city and is also uniquely identified using the PubName. Each publisher can publish more than one book.
- Each *reader* is described by first and family name and is uniquely identified using the ReaderNr. Each reader can borrow more than one copy but need to store the return date for each borrow.



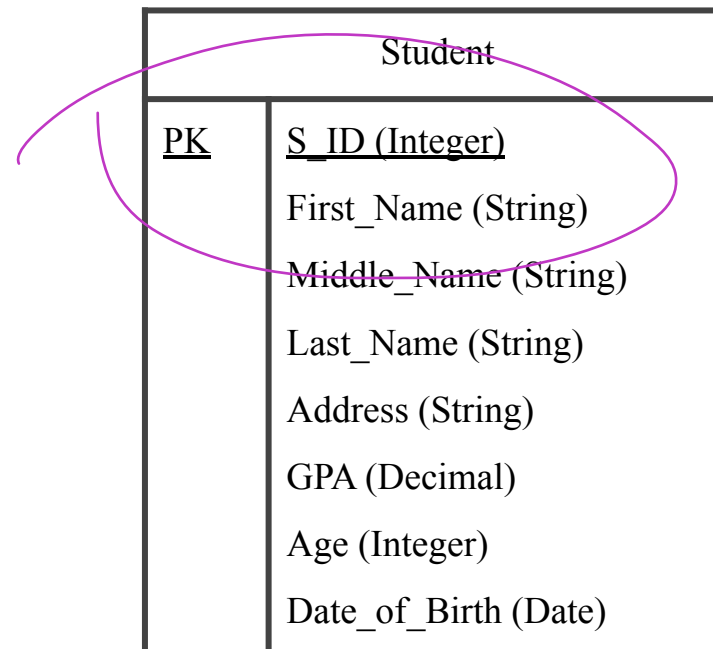
# *Crow's Foot Notation*

We will not use the Crow's Foot  
Notation in the  
assignments/exams of this class

- There is another way to design your ER diagram, which is referred to as the ***Crow's Foot Notation***.
- Crow's Foot Notation -introduced in 1970s- is ***another*** graphical representation for the relationships between entity sets, to focus on illustrating the mapping cardinality in a different way.



An entity set is still represented as a rectangle, with all the attributes listed on separate lines - for example:



Crow's Foot Notation uses a different way to represent the mapping cardinalities of relationships:

- A ring represents "zero"
- A dash represents "one"
- A crow's foot represents "many"



Crow's Foot Notation uses a different way to represent the mapping cardinalities of relationships:

- Minimum zero, maximum one



- Minimum one, maximum one



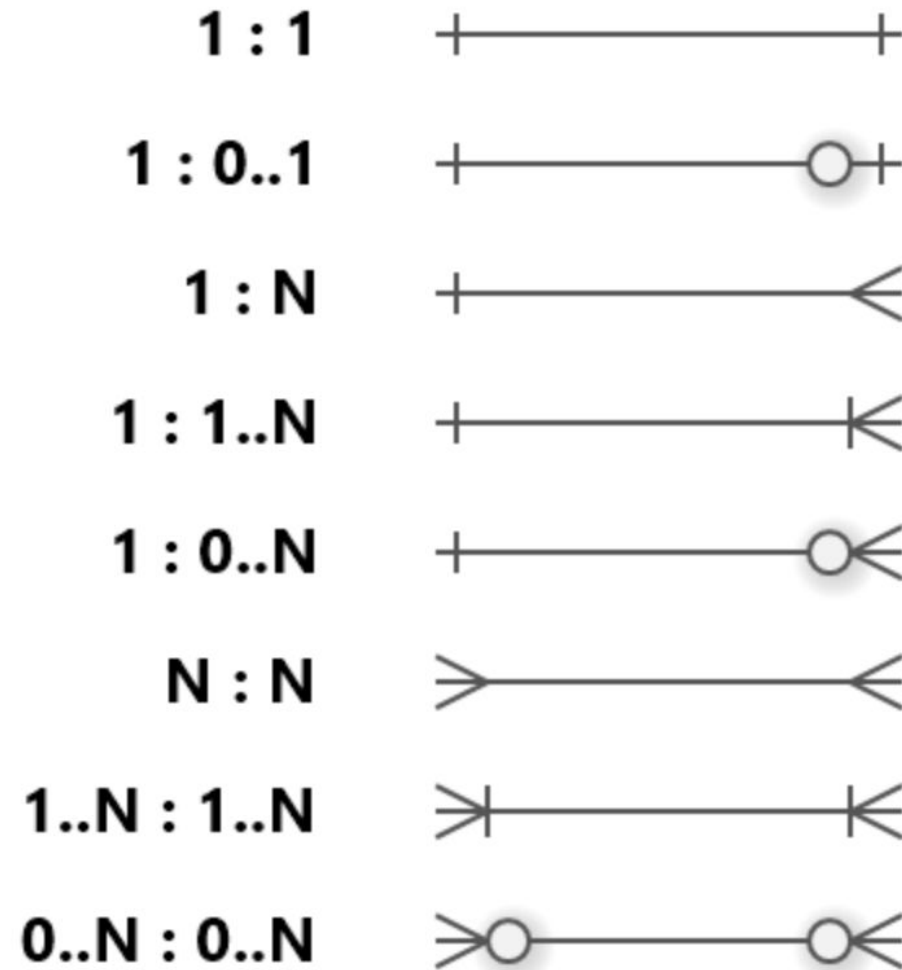
- Minimum zero, maximum many



- Minimum one, maximum many



As a summary:



... for example, If every employee works for one department, and every department has from 1 to many employees - that can be represented as:

