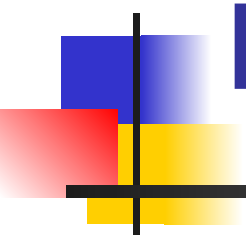
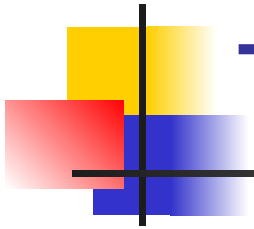


LECTURE 1:
Entity Relationship MODEL

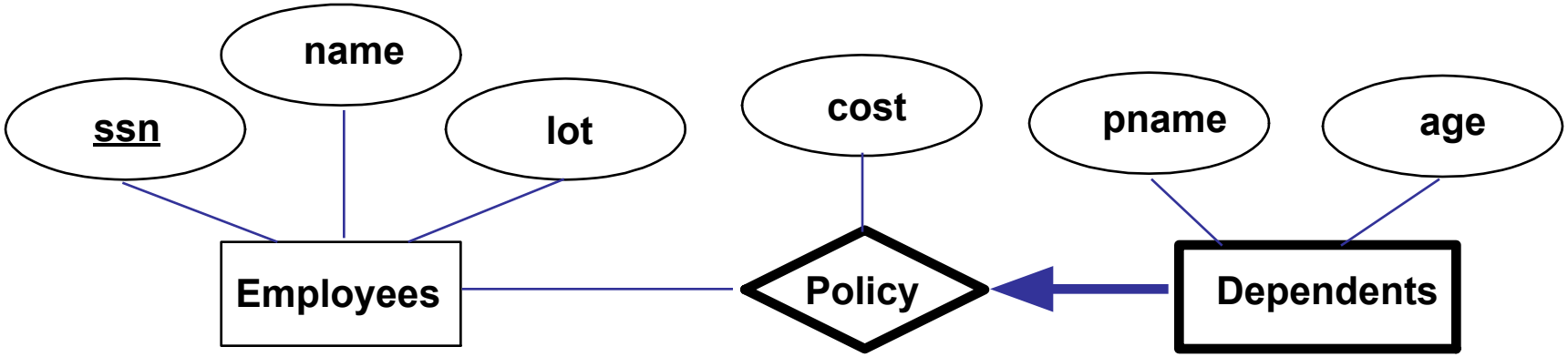
Dr. Samson

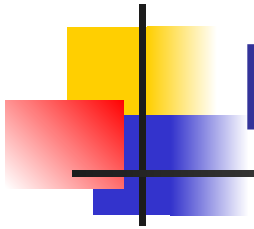


Think before doing it!

- Like most of the software projects, you need to think before you do something.
- Before developing your database application, you need to collect the requirements, and build a conceptual model.
- ER model is a widely accepted standard for conceptual DB design.

AN Entity Relationship (ER) Diagram Looks Like This





ER Model

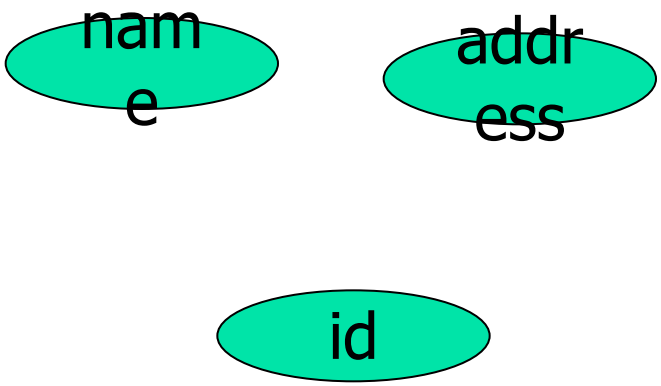
- Key concepts of ER model
  - Entities
  - Relationships
- Entity:
  - Is an object that exists and that can be distinguished from other objects

Samson Daniel

CS306

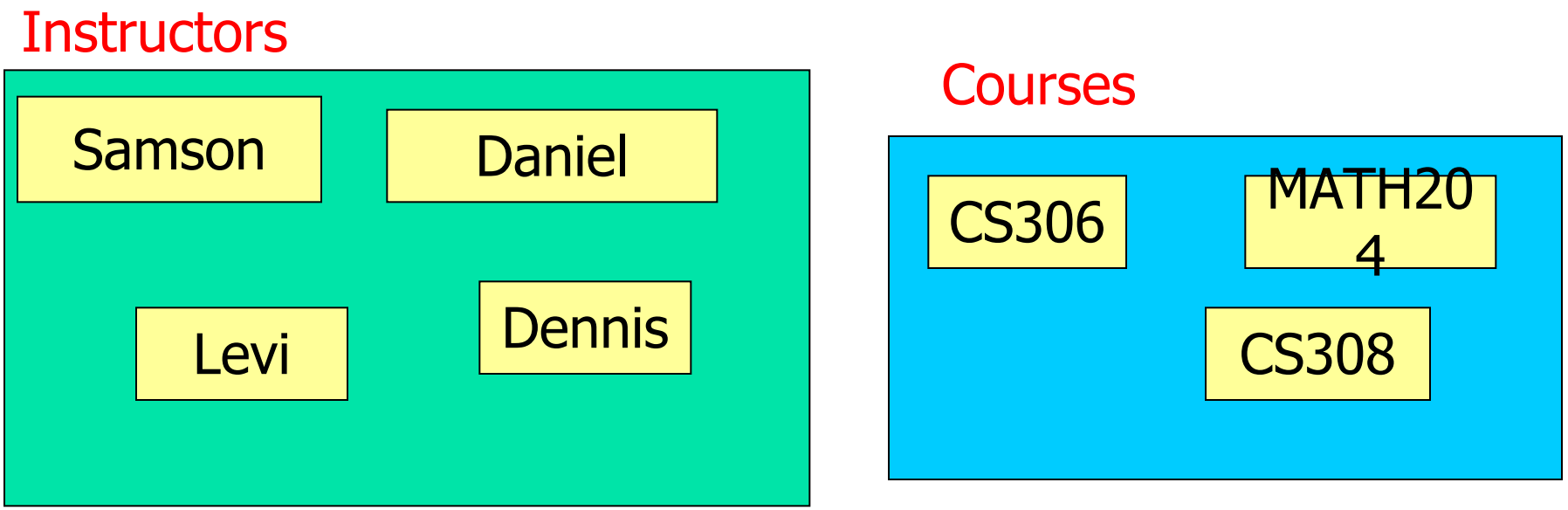
ER Model

- Entity Has attributes that describe it



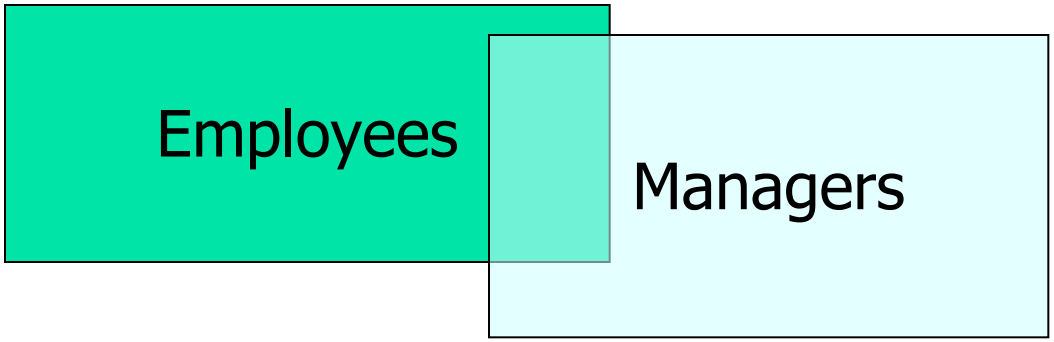
ER Model

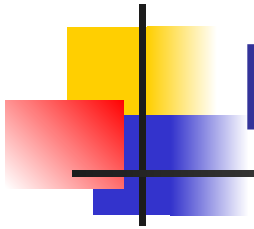
- Entity set:
  - Is the set of entities that share the same properties



ER Model

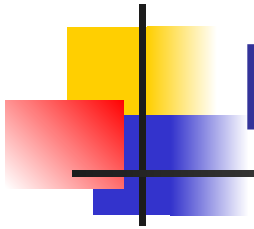
- Entity sets may overlap
- Example?





ER Model

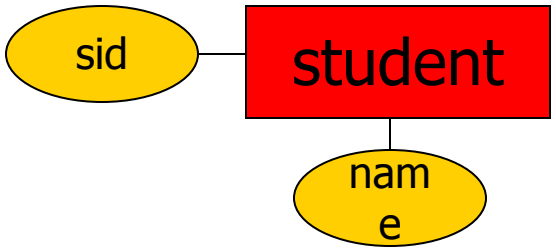
- Relationships:
  - Relate two or more entities (such as Ali is enrolled in CS306)



ER Model

- Relationships:
  - Relate two or more entities (such as Serafettin is enrolled in CS306)
- Relationship sets:
  - Collection of all relationship sets with the same properties (all student enrollments)
- Relationships may also have attributes

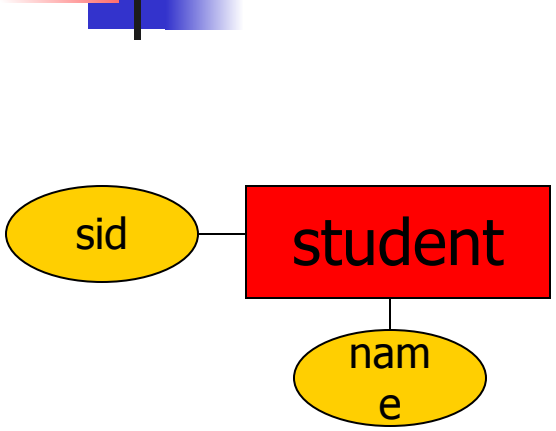
ER Model

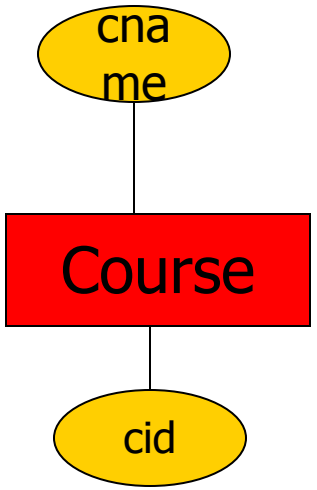


Rectangles : Entity sets

Ellipses : attributes

ER Model

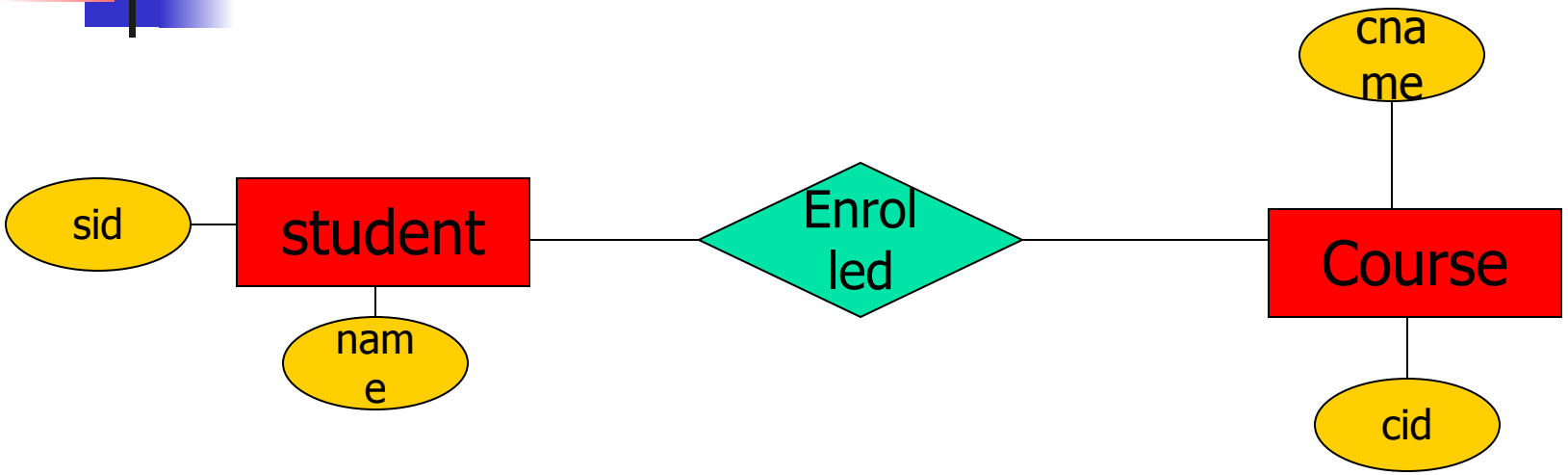




Rectangles : Entity sets

Ellipses : attributes

ER Model



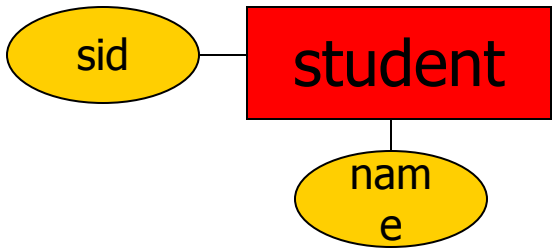
Rectangles : Entity sets

Diamonds : Relationship Sets

Ellipses : attributes

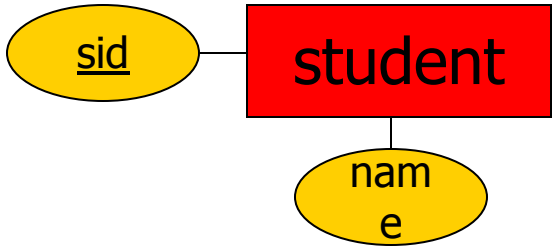
ER Model

- Each entity set has attributes
- Each attribute has a domain (domain is the set of permitted values)

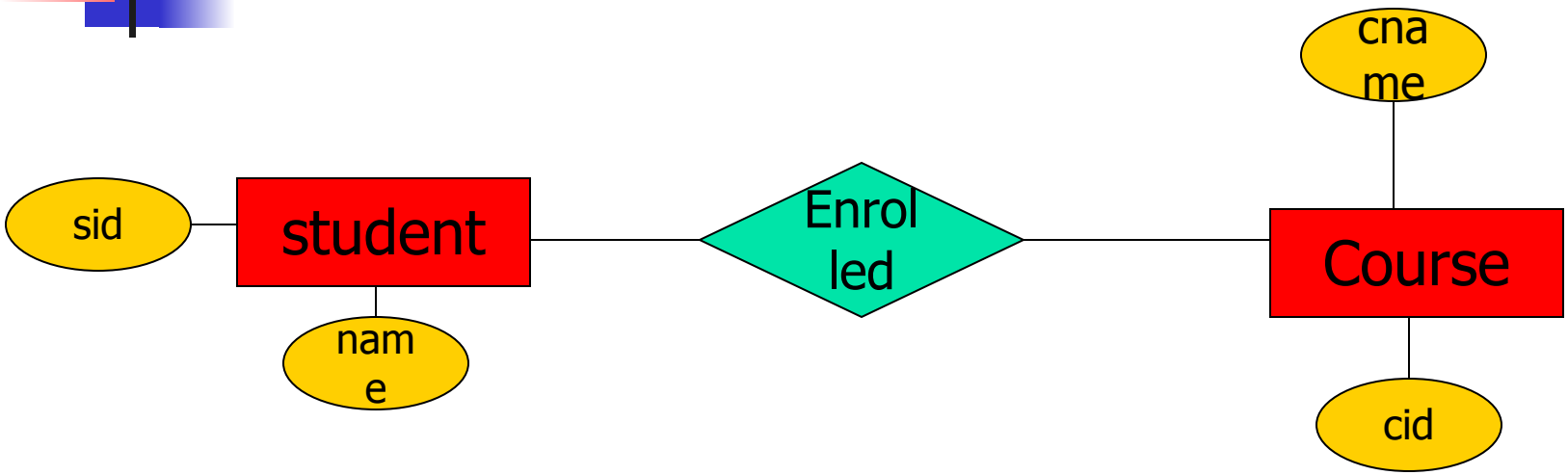


ER Model

- Each entity set has attributes
- Each attribute has a domain (domain is the set of permitted values)
- Each entity set has a key
- Keys are denoted by underlining the attribute name in the ER diagram

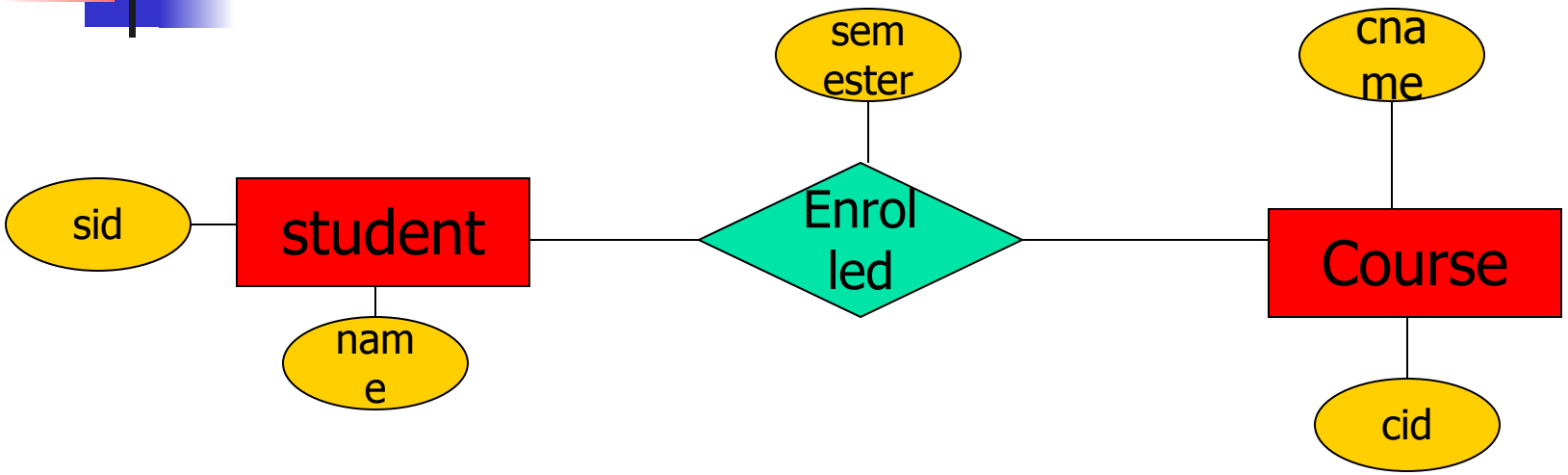


ER Model



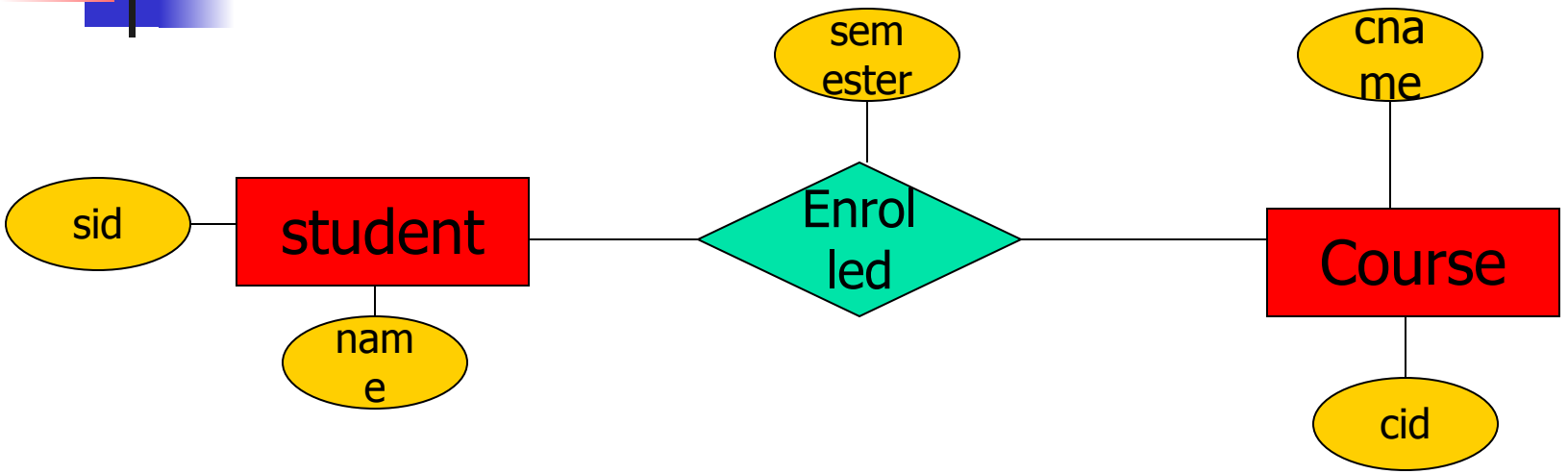
- Relationship sets also have attributes

ER Model



- Relationship sets also have attributes
- We are going to talk about the key in a relationship set later on

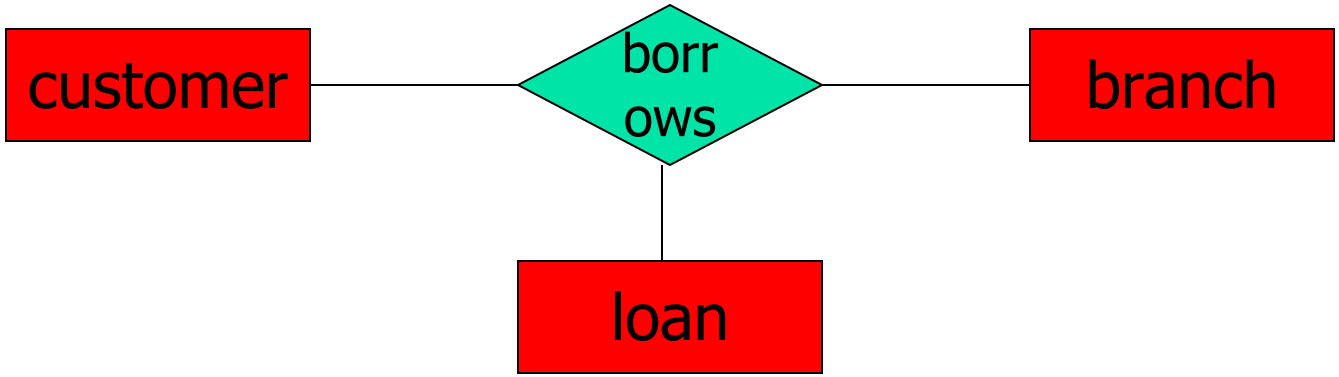
ER Model



- Degree of a relationship set is the number of entity sets that participate in a relationship
- Binary relationship sets involve two entity sets

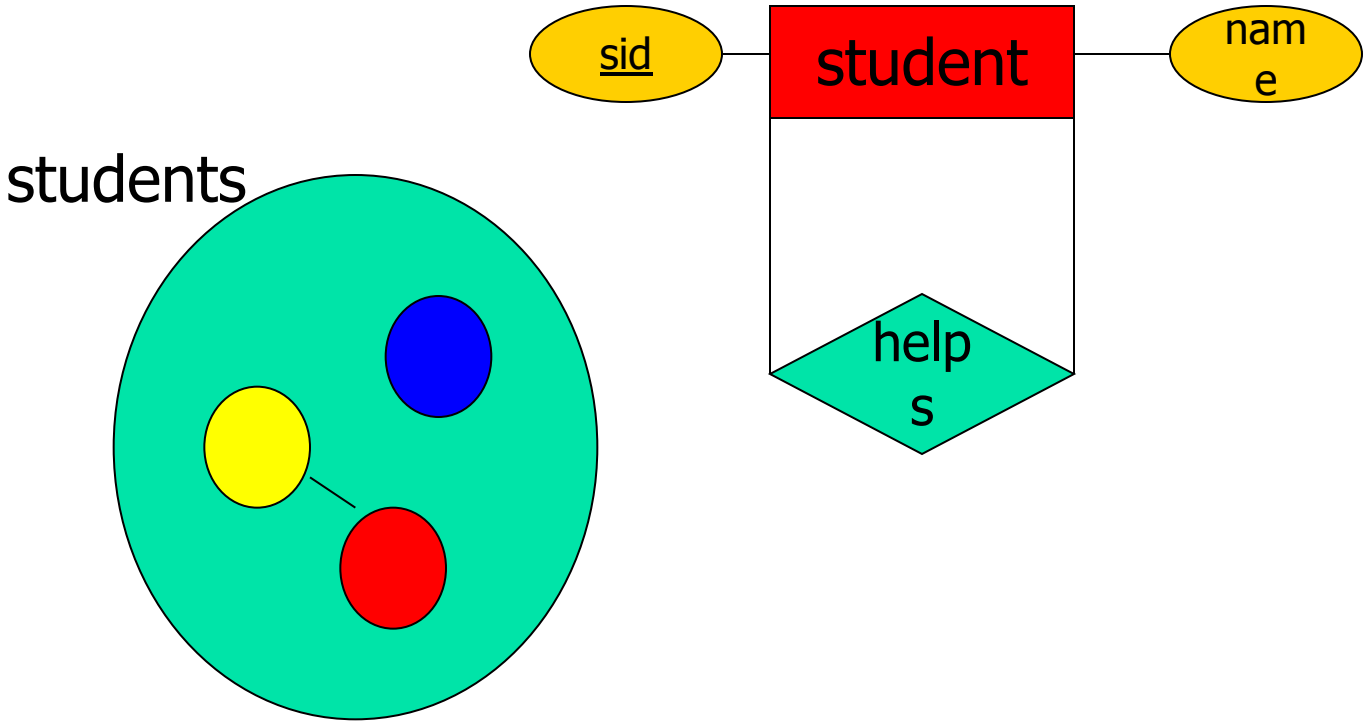
ER Model

- Ternary relationship sets involve three entity sets



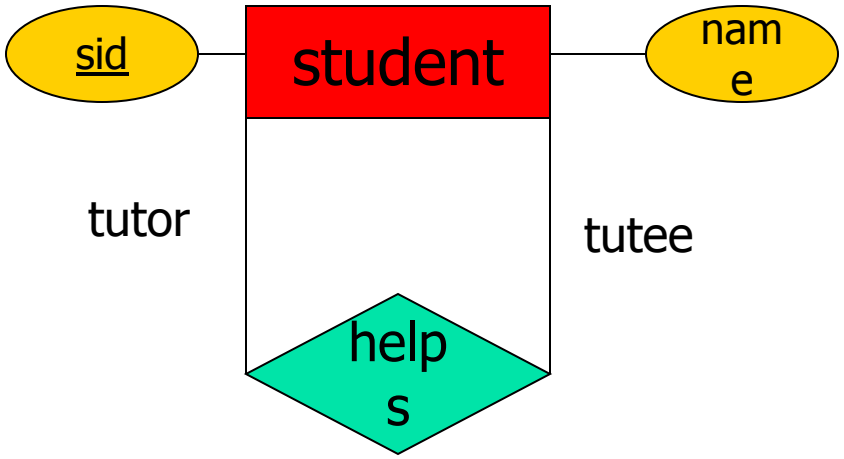
ER Model

- We may have relationships among the entities that belong to the same entity set
- each entity has a role in such a relationship



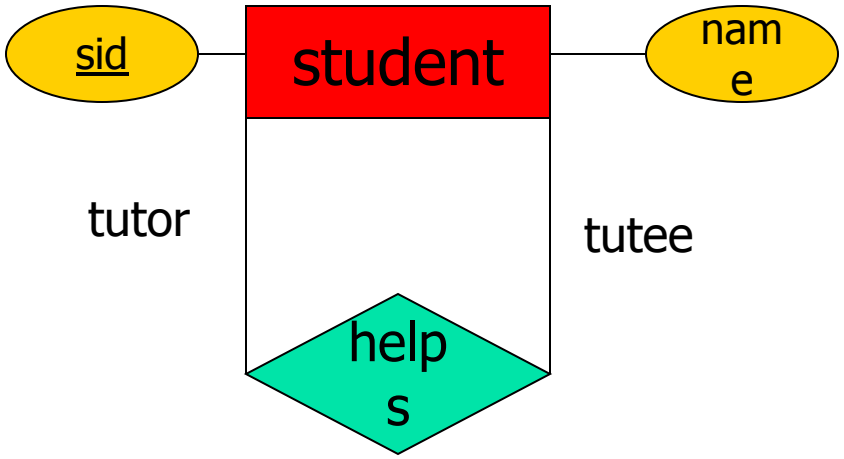
ER Model

- We may have relationships among the entities that belong to the same entity set
- each entity has a role in such a relationship



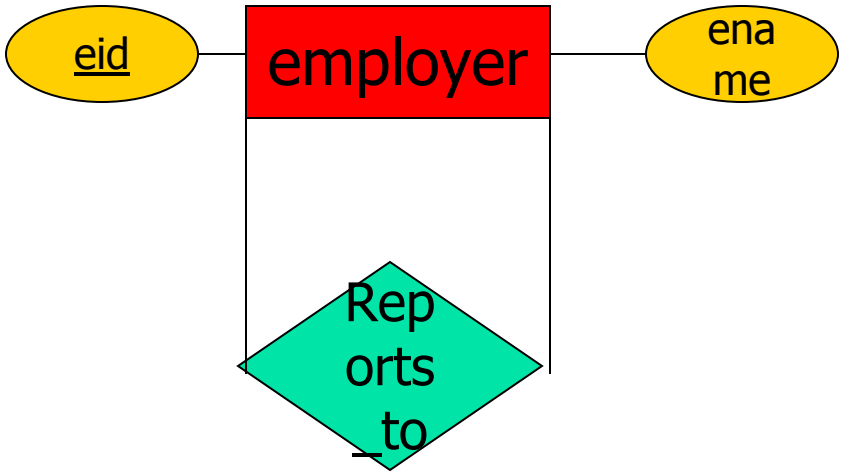
ER Model

- We may have relationships among the entities that belong to the same entity set (each entity has a role in such a relationship)
- What is the degree of the following relationship set (2 or 1)?

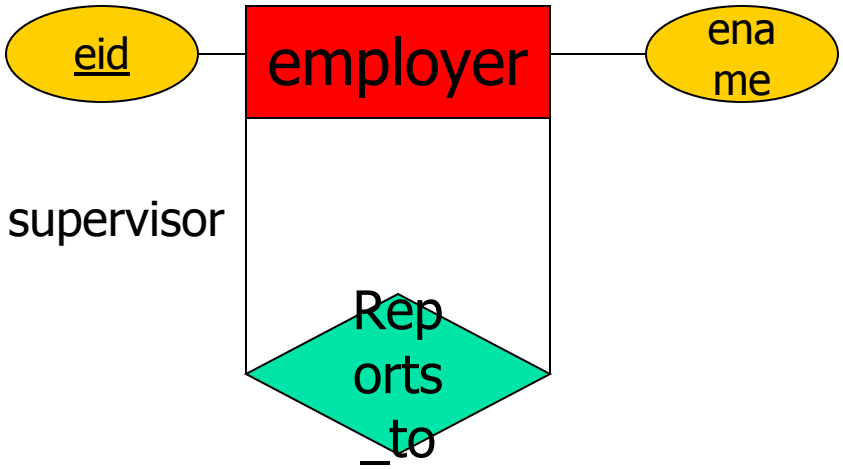


ER Model

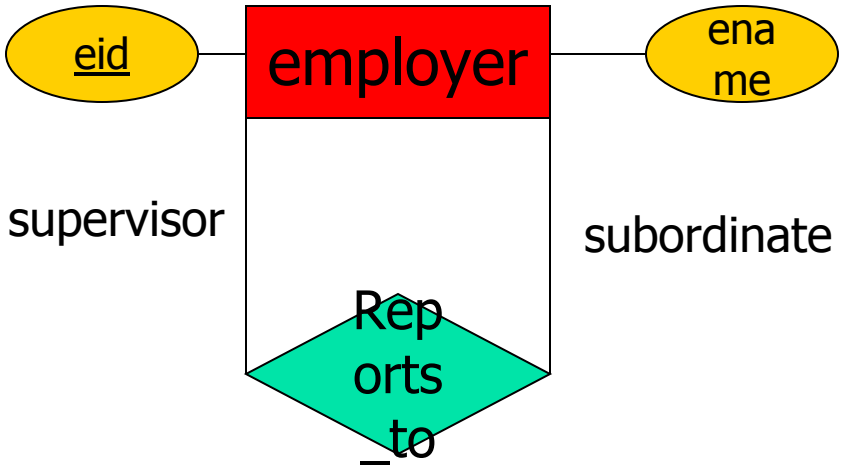
ER Model



ER Model

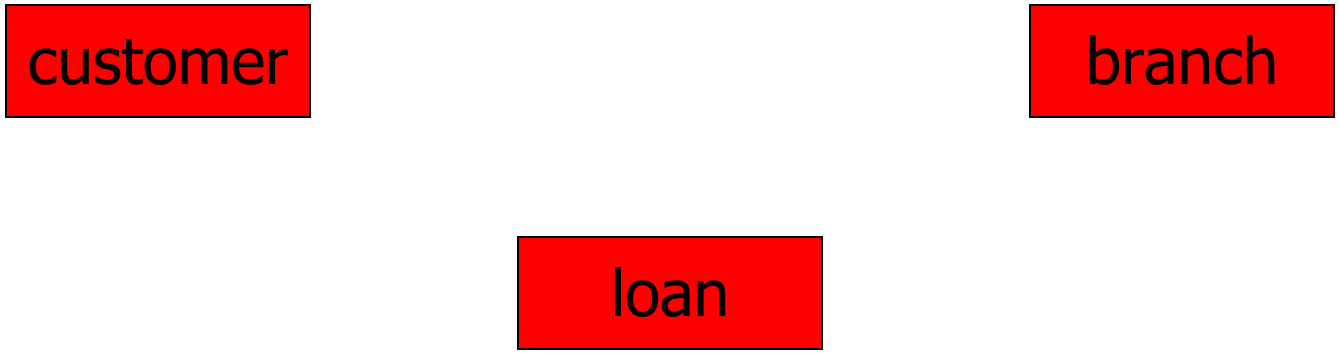


ER Model



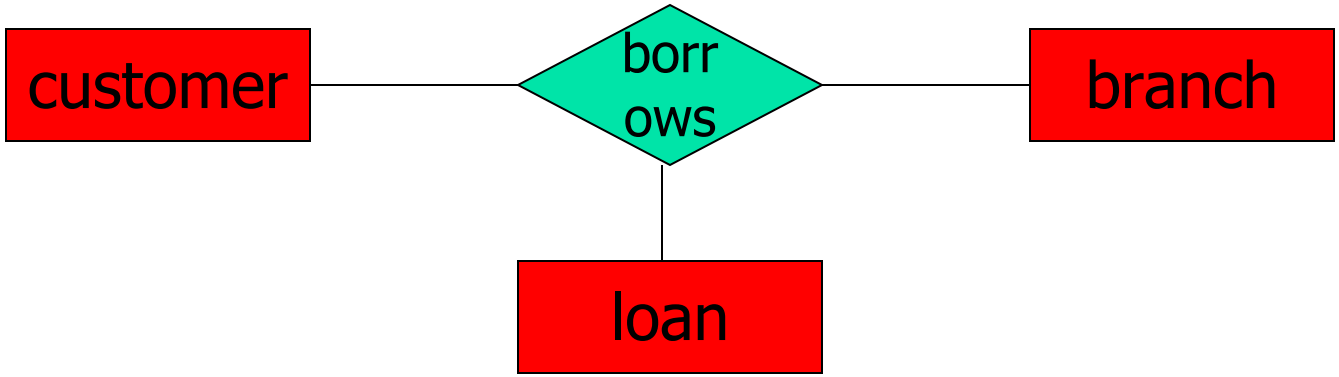
ER Model

- Ternary relationship sets



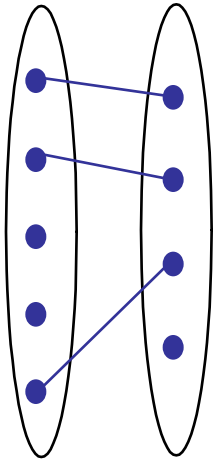
ER Model

- Ternary relationship sets



Mapping cardinalities

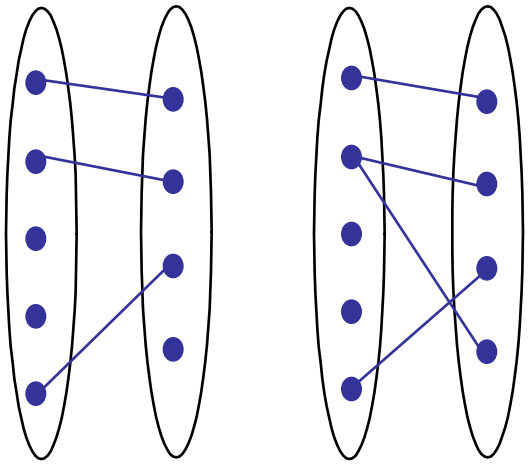
- One-to-One relationship (ex: marriage relationship set between husbands and wives)



1-to-1

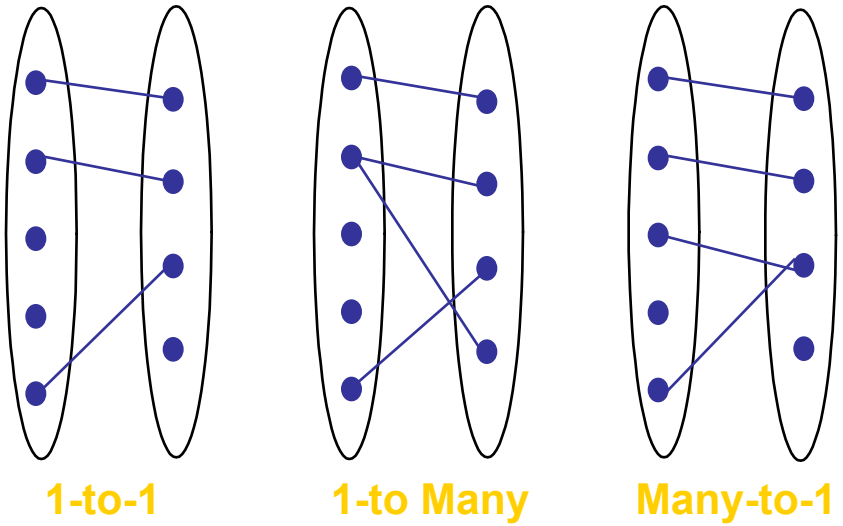
Mapping cardinalities

- One-to-One (ex: marriage relationship set between husbands and wifes)
- One-to-Many (example?)



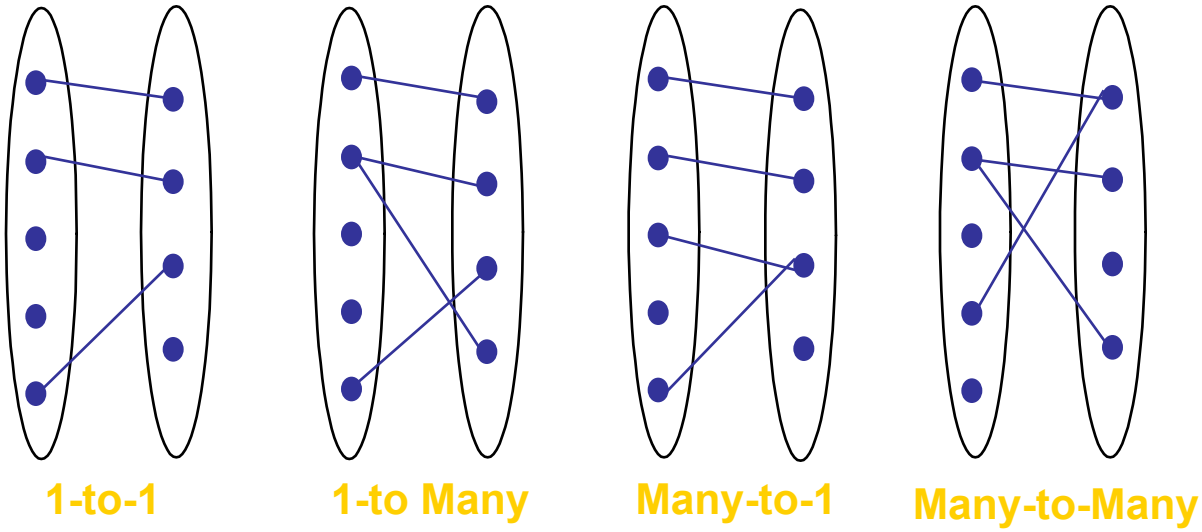
Mapping cardinalities

- One-to-One (ex: marriage relationship set between husbands and wifes)
- One-to-Many
- Many-to-One

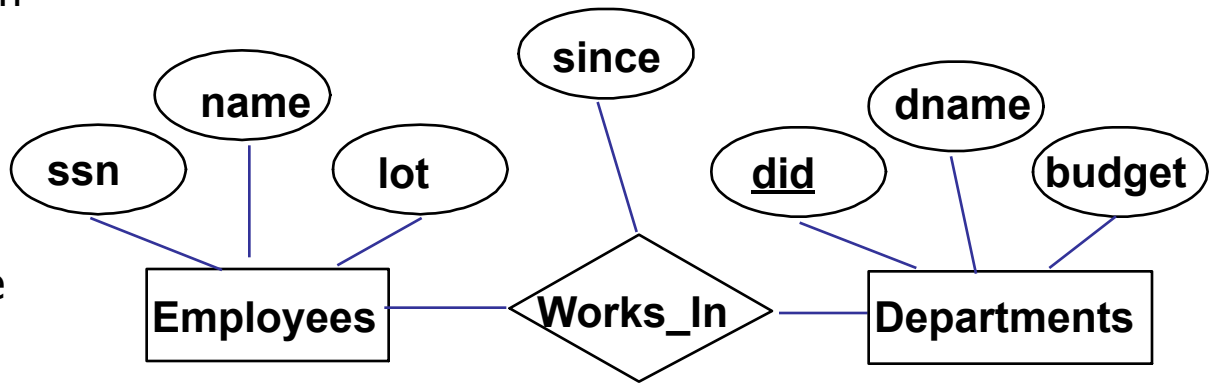


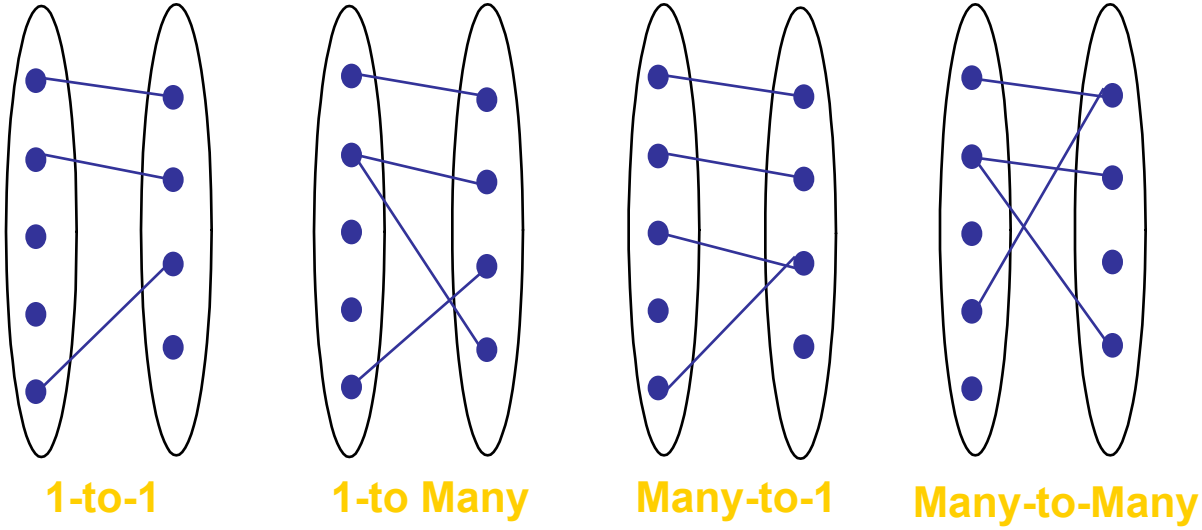
Mapping cardinalities

- One-to-One (ex: marriage relationship set between husbands and wifes)
- One-to-Many
- Many-to-One
- Many-to-Many

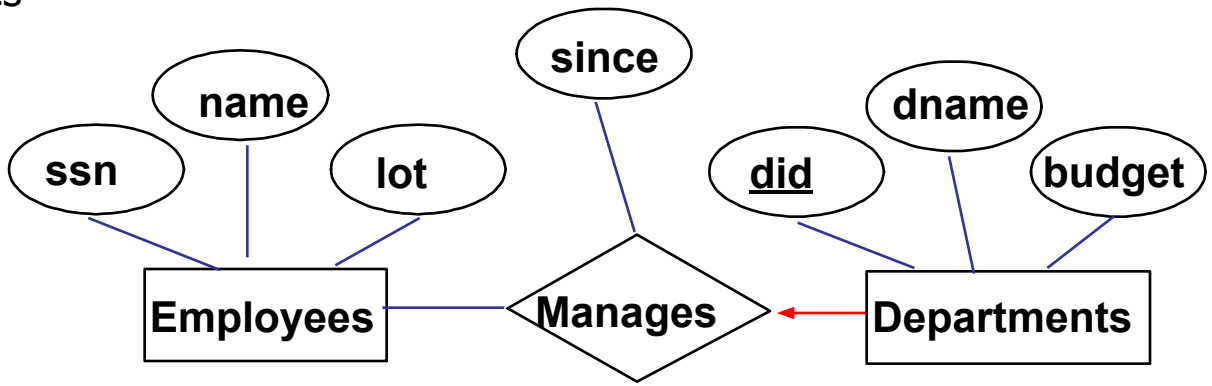


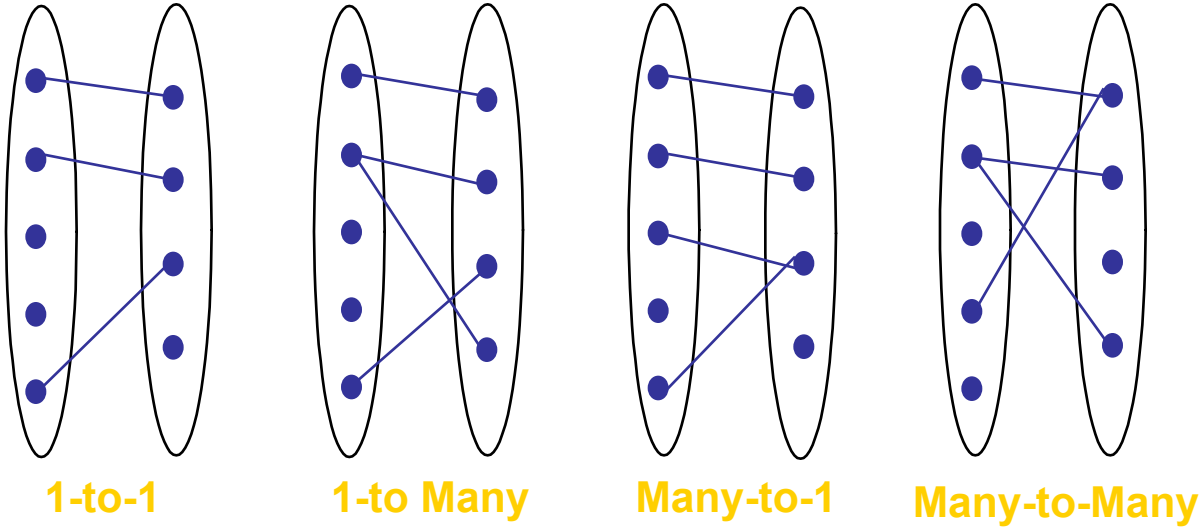
- Consider the works_in relationship
- If an employee can work in multiple departments and a department can have multiple employees
- What type of relationship is that?





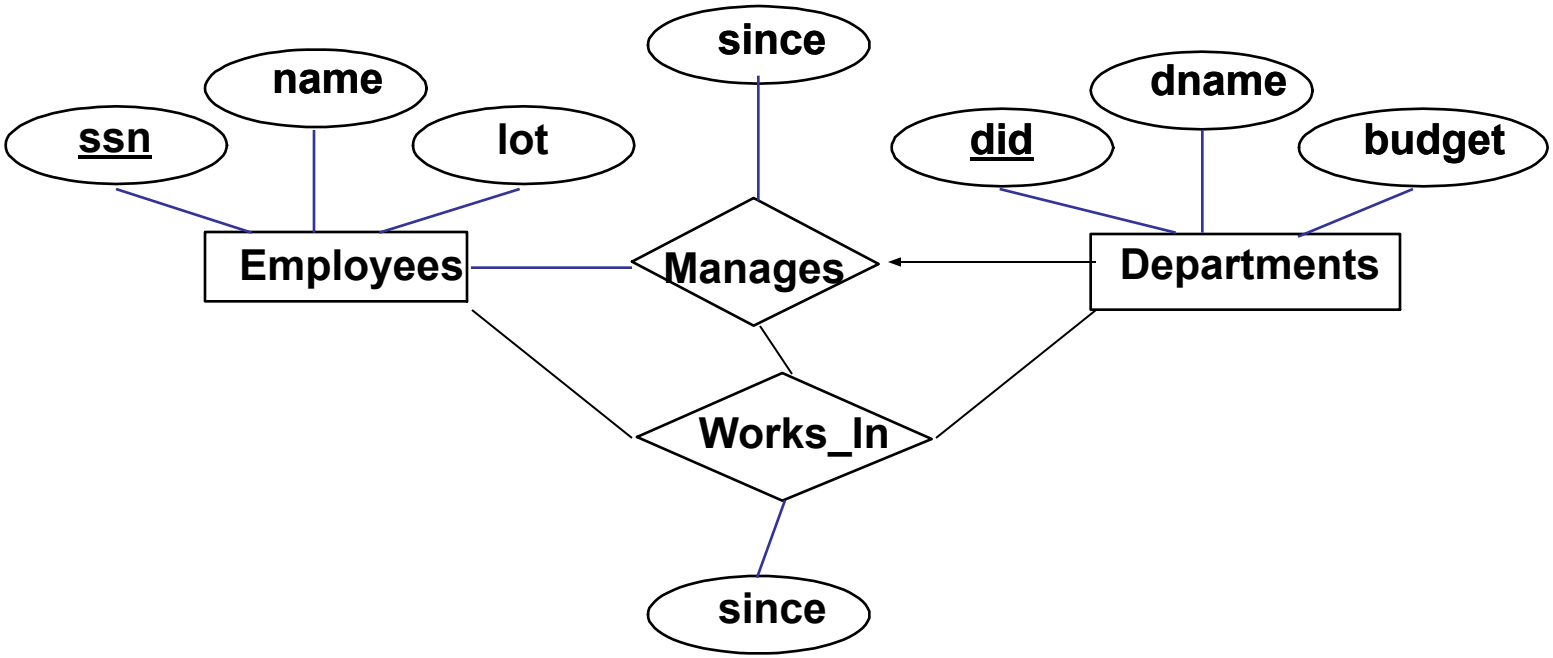
- Consider the manages relationship
- If an employee can manage multiple departments but a department has only one manager
- What type of relationship is that?
- This is called a key constraint (denoted with an arrow)





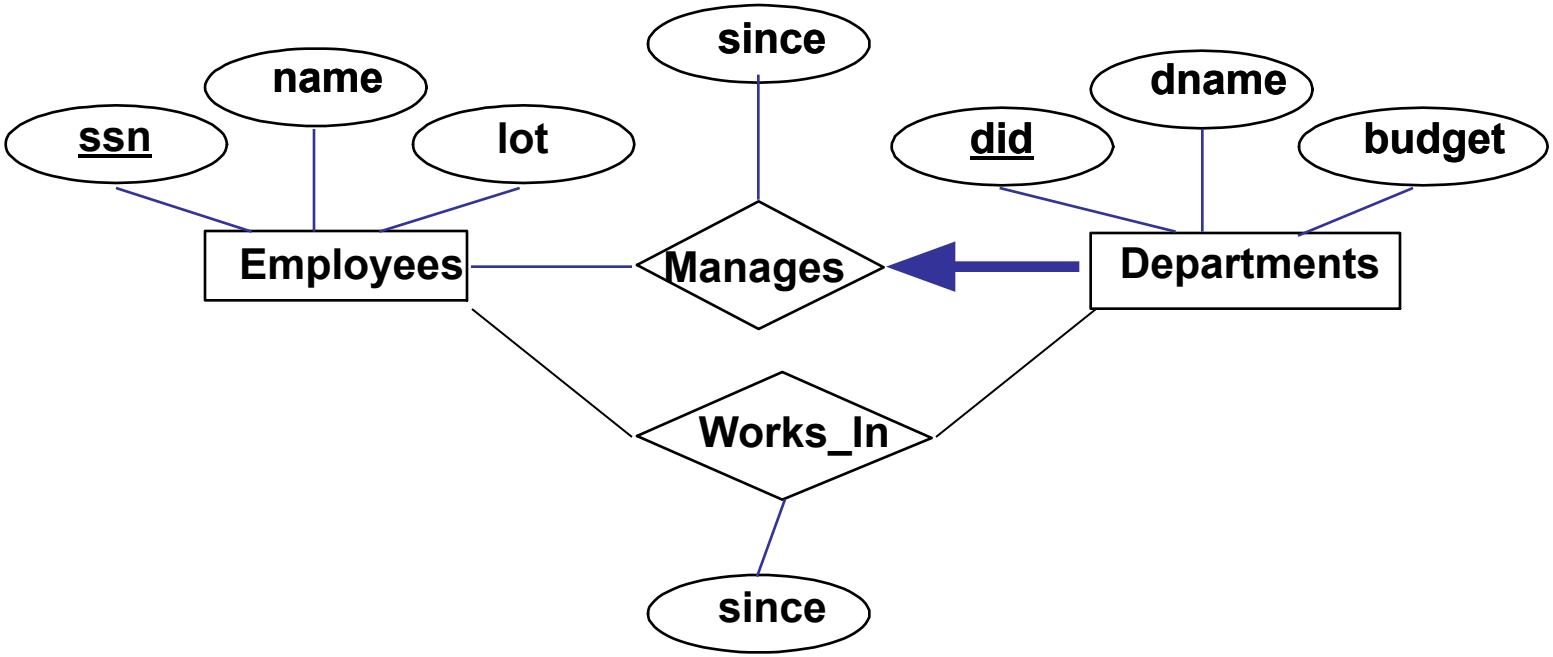
Participation Constraints

- If every department MUST have a manager, then there is a participation constraint
- The participation of Departments in Manages is total (otherwise it is partial).



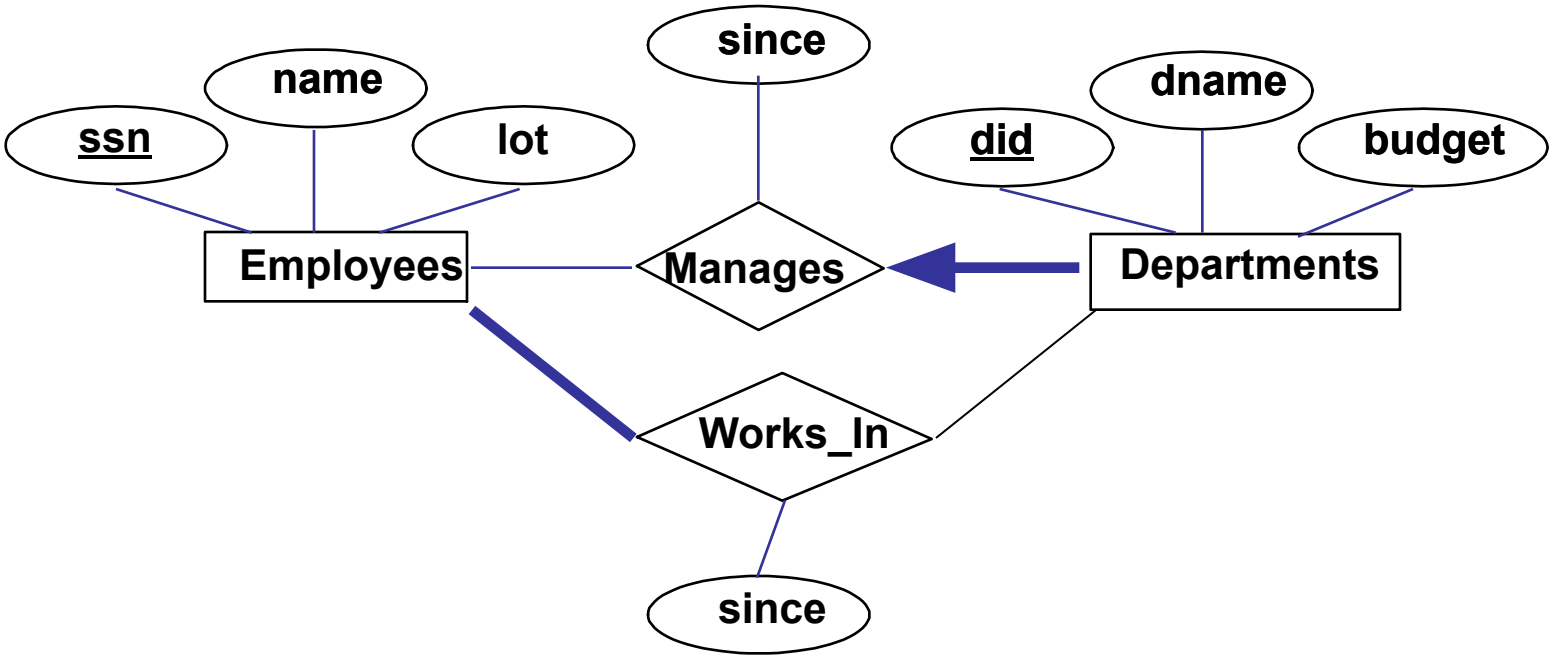
Participation Constraints

- If every department MUST have a manager, then there is a participation constraint
- The participation of Departments in Manages is total (otherwise it is partial).
- Participation constraints are denoted with a thick line (for example each department must participate in the manages relationship, therefore this is denoted with a thick line in the relationship)



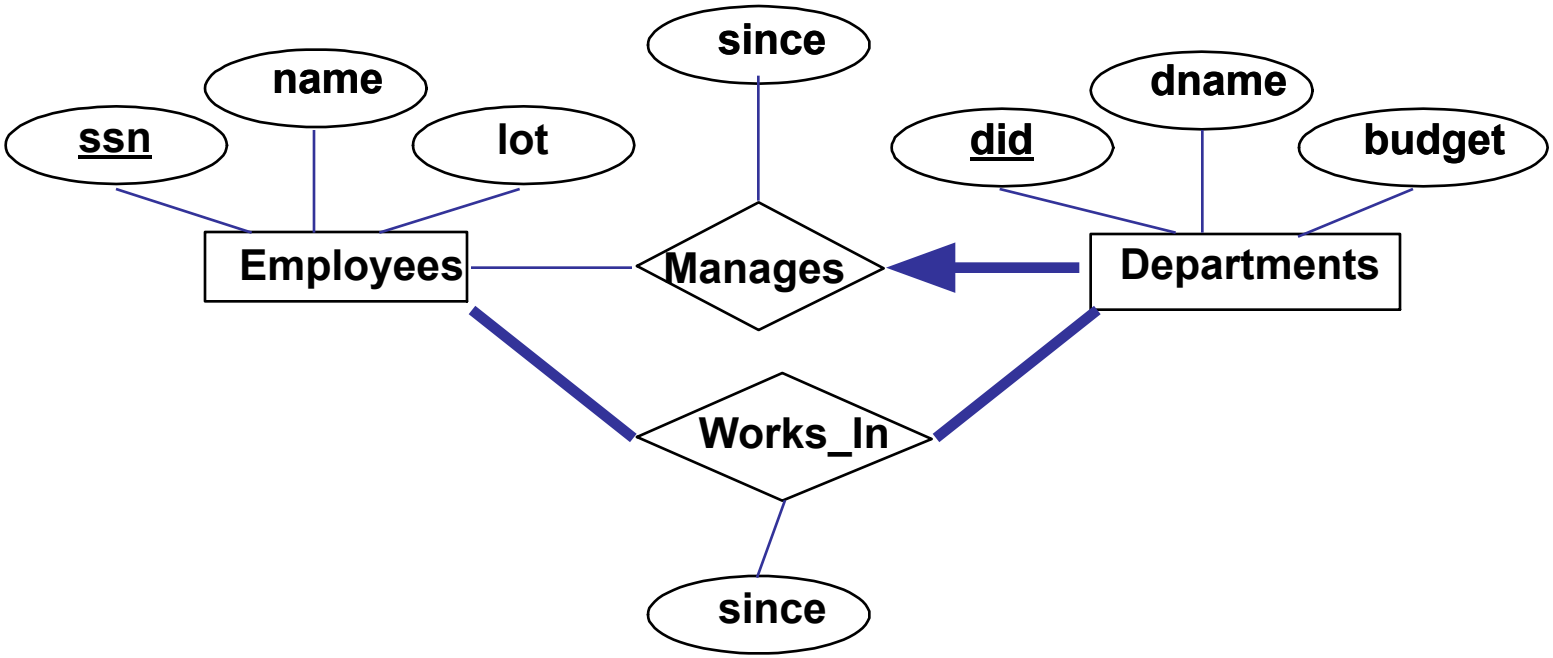
Participation Constraints

- If every employee MUST work in a department, then there is a participation constraint on employee entity set

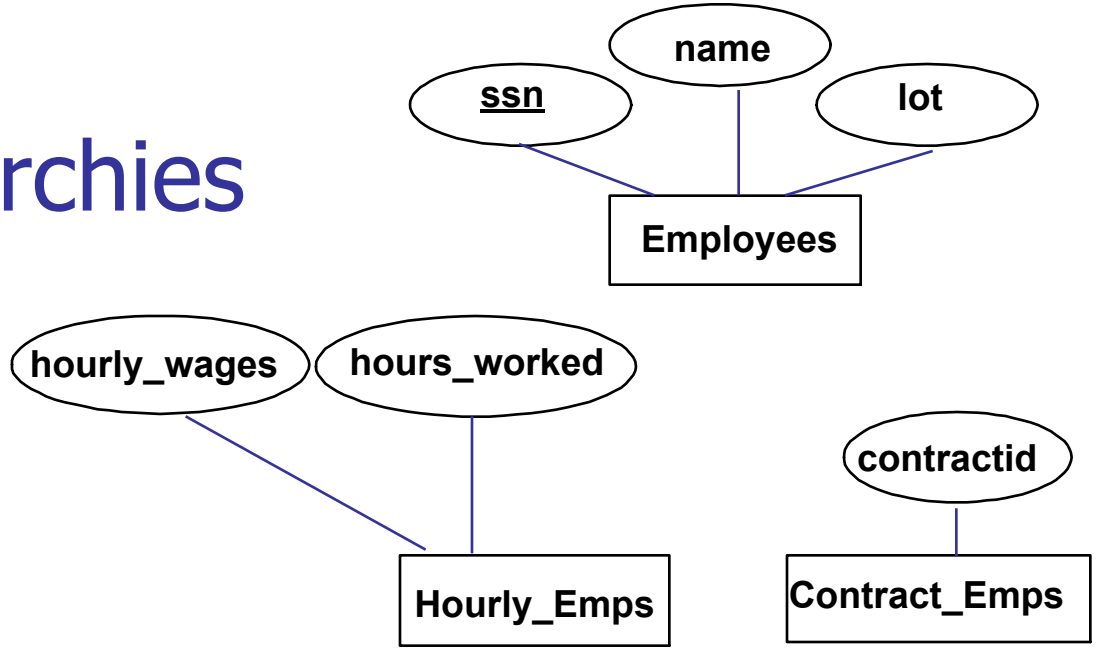


Participation Constraints

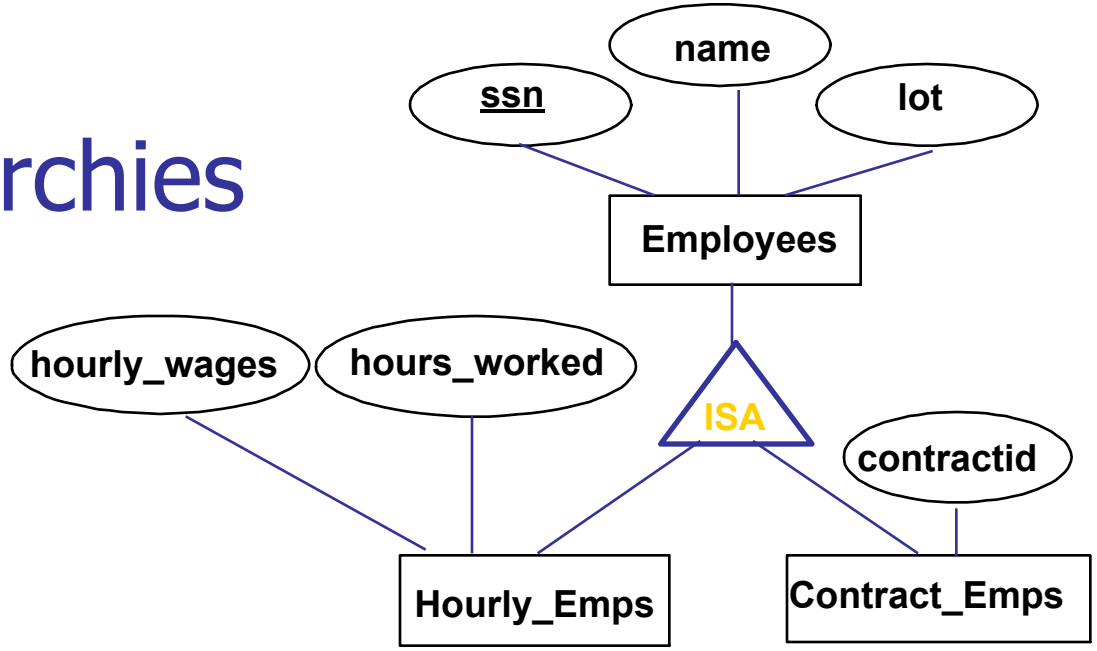
- Plus, if every department MUST have employee(s) working in that department, then there is a participation constraint on department entity set



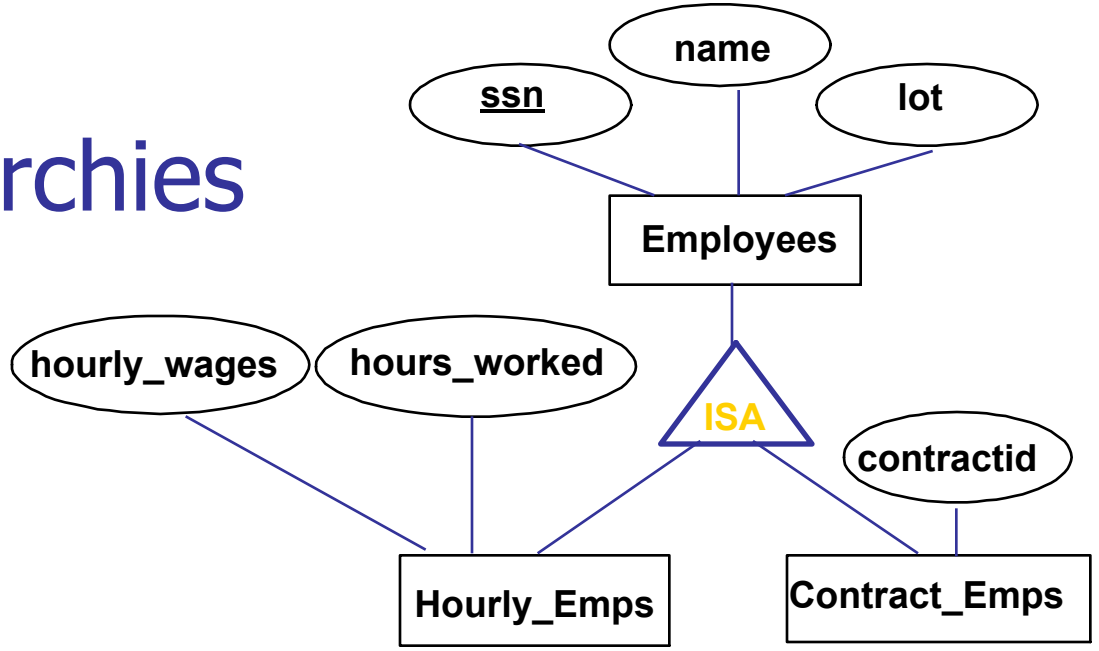
ISA (`is a') Hierarchies



ISA (`is a') Hierarchies



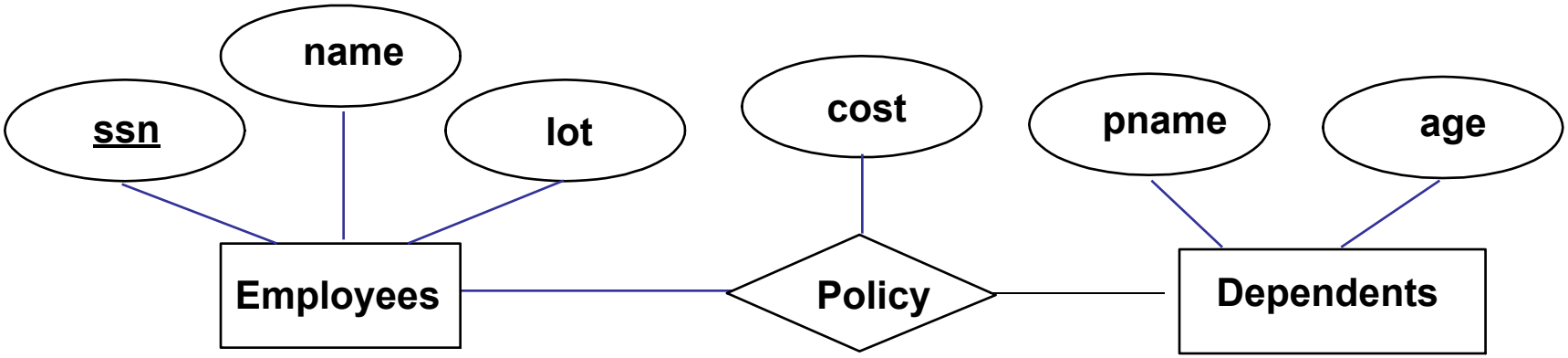
ISA (`is a') Hierarchies



- *Overlap constraints*: Can Serafettin be an Hourly Employee as well as a Contract Employee?
- *Covering constraints*: Does every Employee also have to be an Hourly Employee or a Contract Employee?
- Reasons for using ISA:
  - To add descriptive attributes specific to a subclass.
  - To identify entities that participate in a relationship.
- Specialization vs. generalization

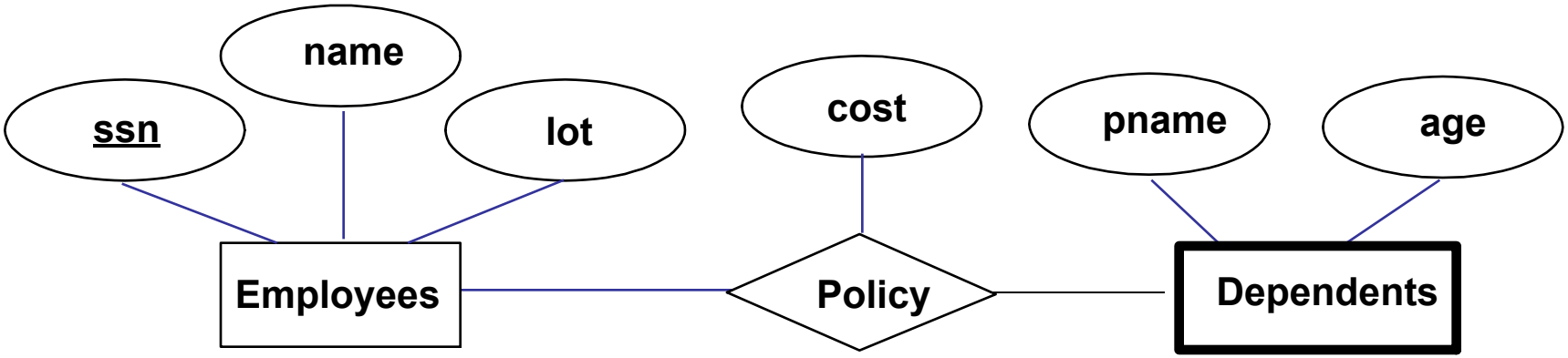
Weak Entities

- A *weak entity* can be identified uniquely only by considering the primary key of another (*owner*) entity.



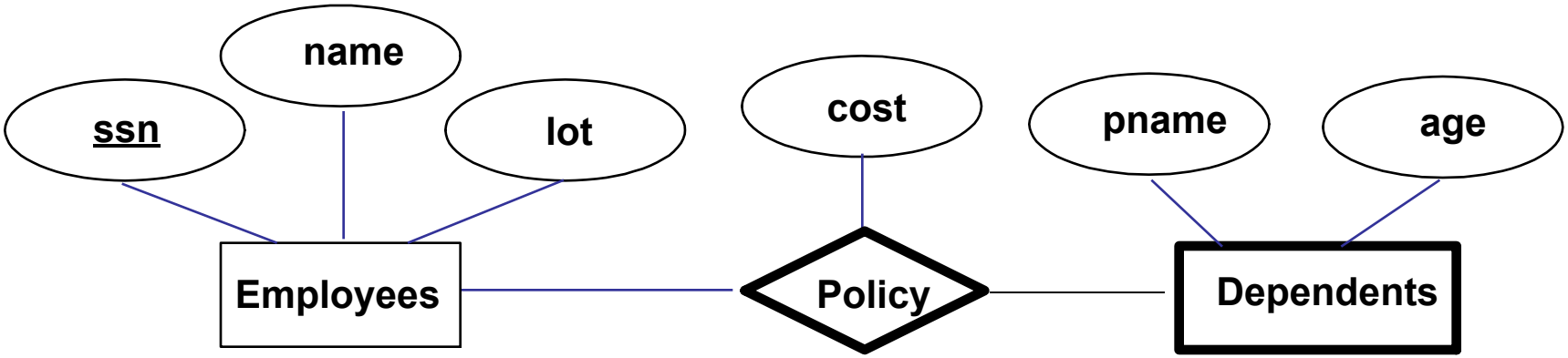
Weak Entities

- A *weak entity* can be identified uniquely only by considering the primary key of another (*owner*) entity.
- A weak entity set is denoted by a rectangle with thick lines



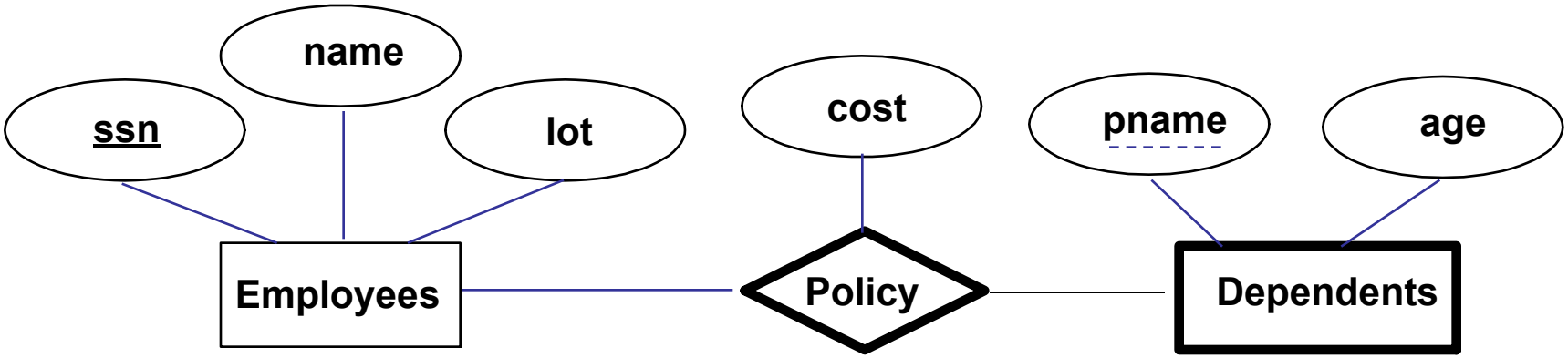
Weak Entities

- A *weak entity* can be identified uniquely only by considering the primary key of another (*owner*) entity.
- A weak entity set is denoted by a rectangle with thick lines
- The relationship between a week entity and the owner entity is denoted by a diamond with thick lines.



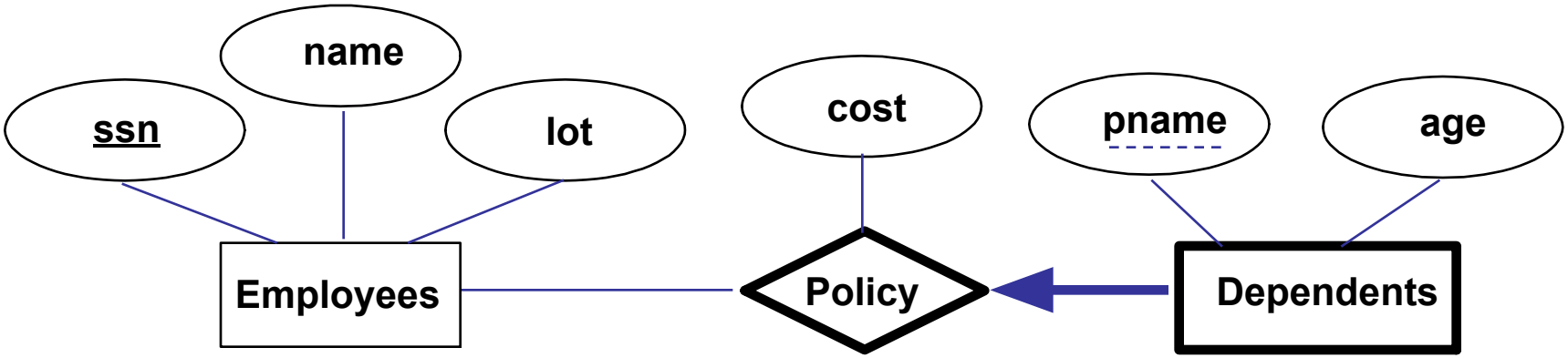
Weak Entities

- A *weak entity* can be identified uniquely only by considering the primary key of another (*owner*) entity.
- What can you say about the constraints on the indentifying relationship? (i.e., participation and key constraints)



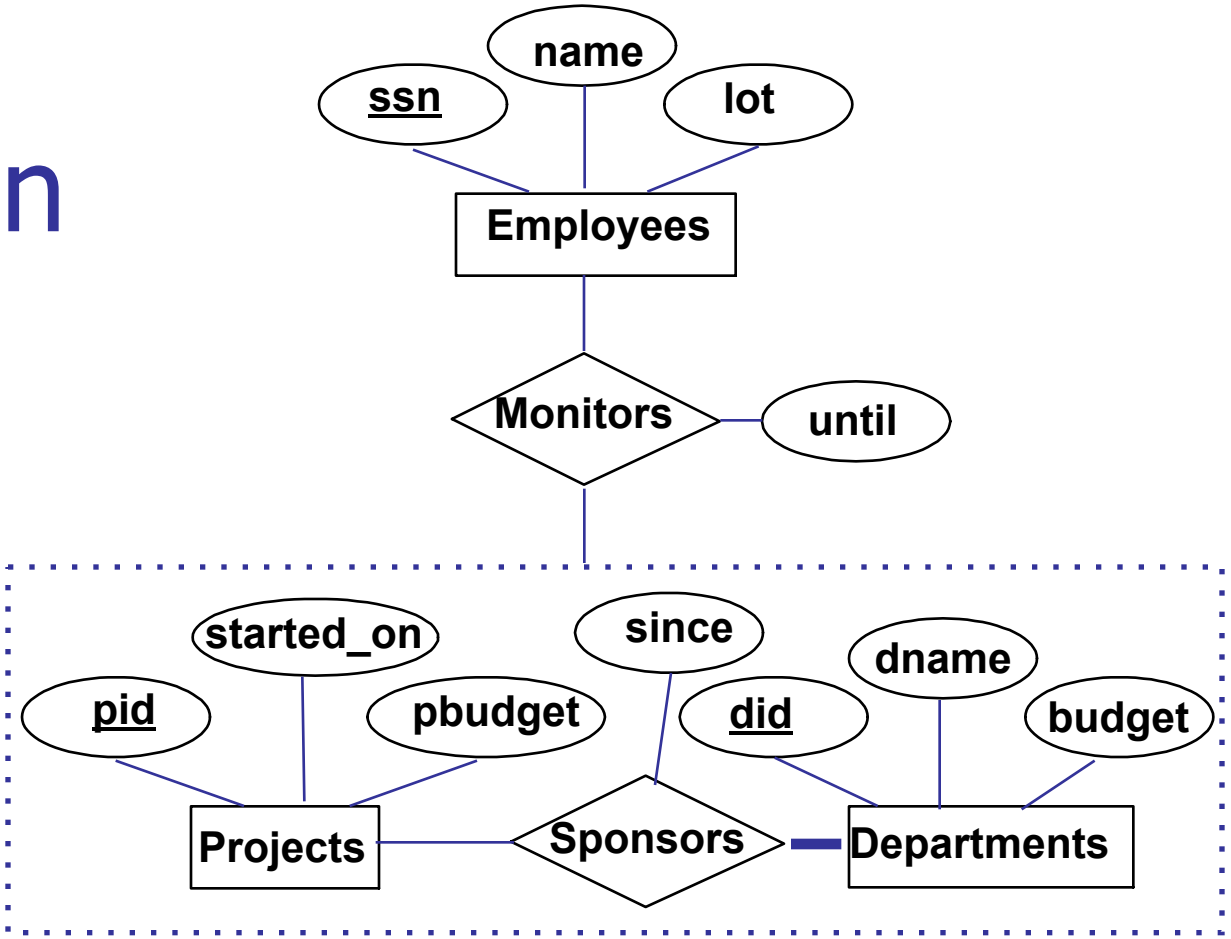
Weak Entities

- A *weak entity* can be identified uniquely only by considering the primary key of another (*owner*) entity.
  - Owner entity set and weak entity set must participate in a one-to-many relationship set (one owner, many weak entities).
  - Weak entity set must have total participation in this *identifying* relationship set.



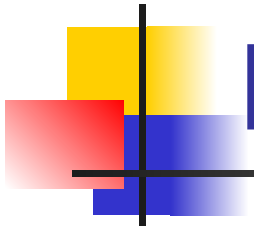
Aggregation

- Used when we have to model a relationship involving (entitity sets and) a *relationship set*.
  - *Aggregation* allows us to treat a relationship set as an entity set for purposes of participation in (other) relationships.



- *Aggregation vs. ternary relationship*:
  Monitors is a distinct relationship, with a descriptive attribute.

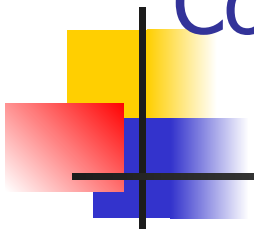
  Also, can say that each sponsorship is monitored by at most one employee.



Example:

- Draw the ER diagram for the following specifications: There are conferences, universities, and professors. Conferences have names (such as VLDB, ICDE, SIGMOD), and years they are organized. A conference can be organized in different years but a conference can not be organized more than once in a certain year. For example SIGMOD is organized in 2001, 2002, etc, but SIGMOD can not be organized twice in 2001.Universities have names and cities they are located, such as Sabanci Universiy located in Istanbul. Each conference at a specific year is organized by one university, but a university can organize many conferences. Each conference organized at a specific year has a list of PC (Program Committee) members which consists of professors associated with universities. Professors have names and SSNs. A professor is associated with one university, but a university may have many professors.

Conceptual Design Using the ER Model

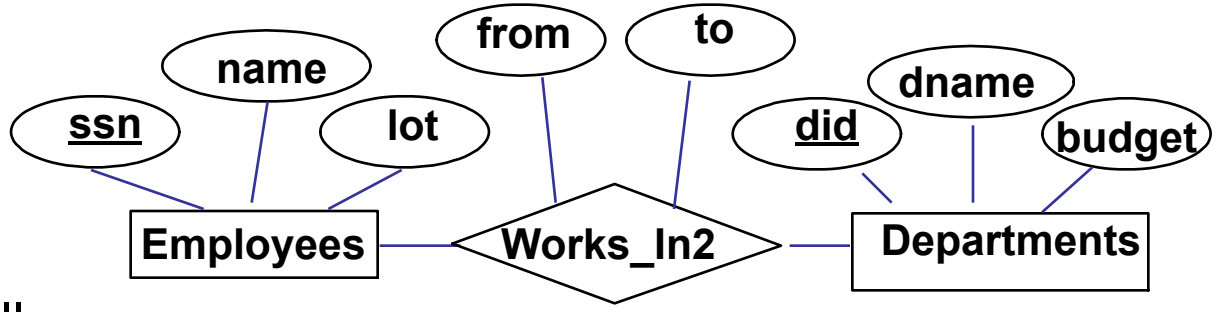


- Design choices:
  - Should a concept be modeled as an entity or an attribute?
  - Should a concept be modeled as an entity or a relationship?
  - Identifying relationships: Binary or ternary? Aggregation?
- Constraints in the ER Model:
  - A lot of data semantics can (and should) be captured.
  - But some constraints cannot be captured in ER diagrams.

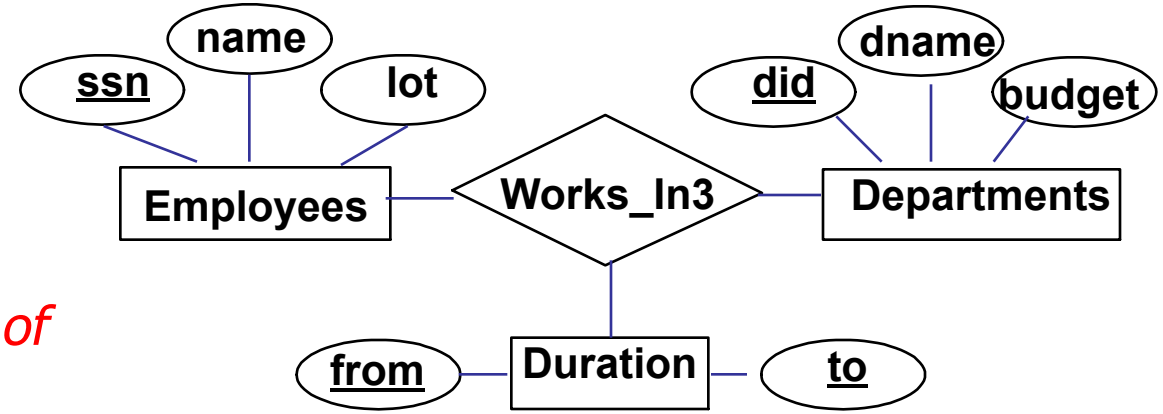
Entity vs. Attribute

- Should *address* be an attribute of Employees or an entity (connected to Employees by a relationship)?
- Depends upon the use we want to make of address information, and the semantics of the data:
  - If we have several addresses per employee, *address* must be an entity (since 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 (since attribute values are atomic).

Entity vs. Attribute (Contd.)

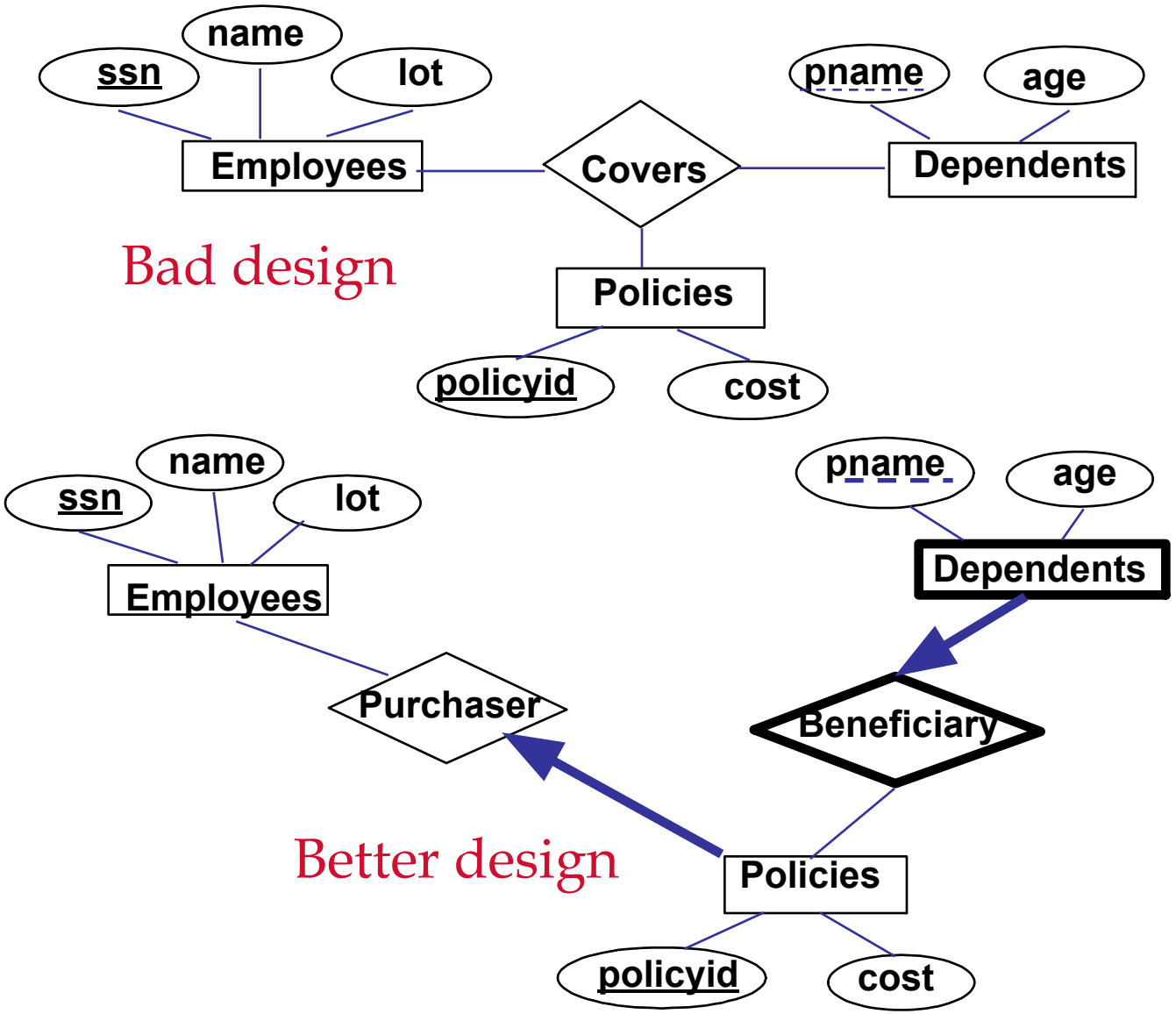


- Works_In2 does not allow an employee to work in a department for two or more periods.
- Similar to the problem of wanting to record several addresses for an employee: we want to record *several values of the descriptive attributes for each instance of this relationship.*



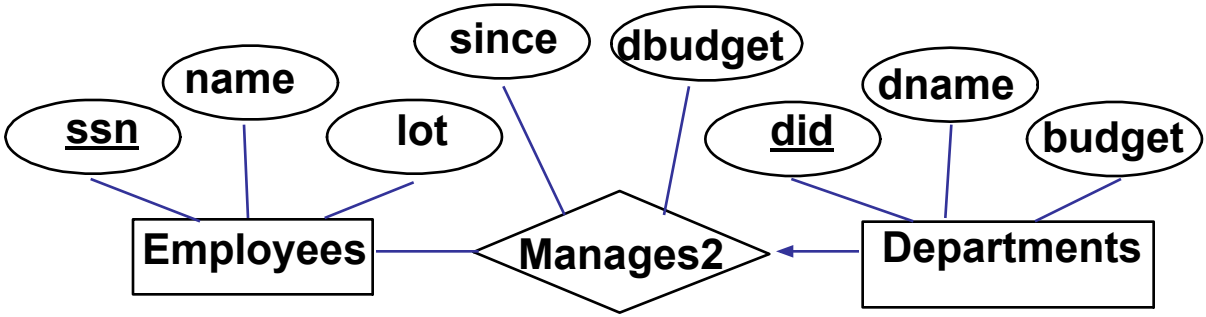
Binary vs. Ternary Relationships

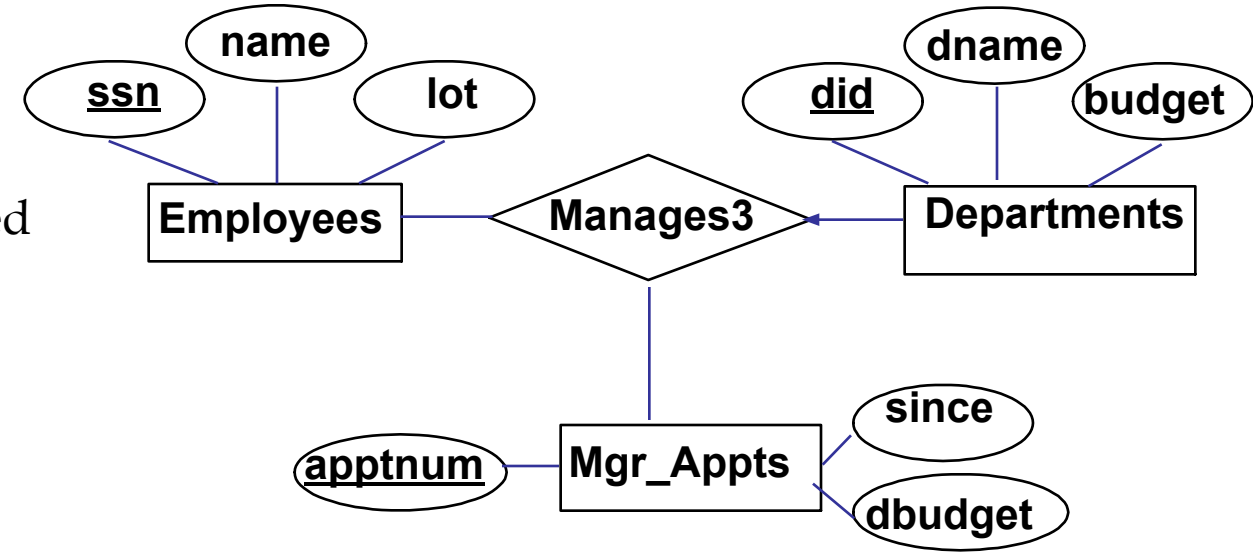
- If each policy is owned by just 1 employee:
  - Key constraint on Policies would mean policy can only cover 1 dependent!



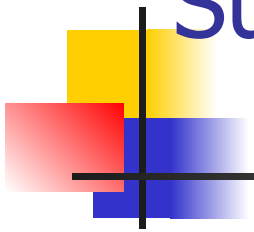
Entity vs. Relationship

- First ER diagram OK if a manager gets a separate discretionary budget for each dept.
- What if a manager gets a discretionary budget that covers *all* managed depts?
  - Redundancy of *dbudget,* which is stored for each dept managed by the manager.
  - Misleading: suggests *dbudget* tied to managed dept.





Summary of Conceptual Design



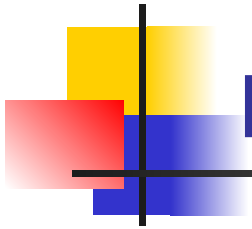
- *Conceptual design* follows *requirements analysis*,
  - Yields a high-level description of data to be stored
- ER model popular for conceptual design
  - Constructs are expressive, close to the way people think about their applications.
- Basic constructs: *entities*, *relationships*, and *attributes* (of entities and relationships).
- Some additional constructs: *weak entities*, *ISA hierarchies*, and *aggregation*.
- Note: There are many variations on ER model.

Summary of ER (Contd.)

- Several kinds of integrity constraints can be expressed in the ER model: *key constraints*, *participation constraints*, and *overlap/covering constraints* for ISA hierarchies. Some *foreign key constraints* are also implicit in the definition of a relationship set.
  - Some constraints (notably, *functional dependencies*) cannot be expressed in the ER model.
  - Constraints play an important role in determining the best database design for an enterprise.

Summary of ER (Contd.)

- ER design is *subjective*. There are often many ways to model a given scenario! Analyzing alternatives can be tricky, especially for a large enterprise. Common choices include:
  - Entity vs. attribute, entity vs. relationship, binary or n-ary relationship, whether or not to use ISA hierarchies, and whether or not to use aggregation.
- Ensuring good database design: resulting relational schema should be analyzed and refined further. FD information and normalization techniques are especially useful.



Banks Database in North Cyprus

You are asked to design a database of banks in North Cyprus .

- Now Lets think about the requirements
- What are the entities in our database?
- What are their attributes?
- Draw the ER diagram!