E-R Modeling and E-R Diagram

E-R Modeling is used under
Conceptual Design of your Database.

During Conceptual Design of database we decide

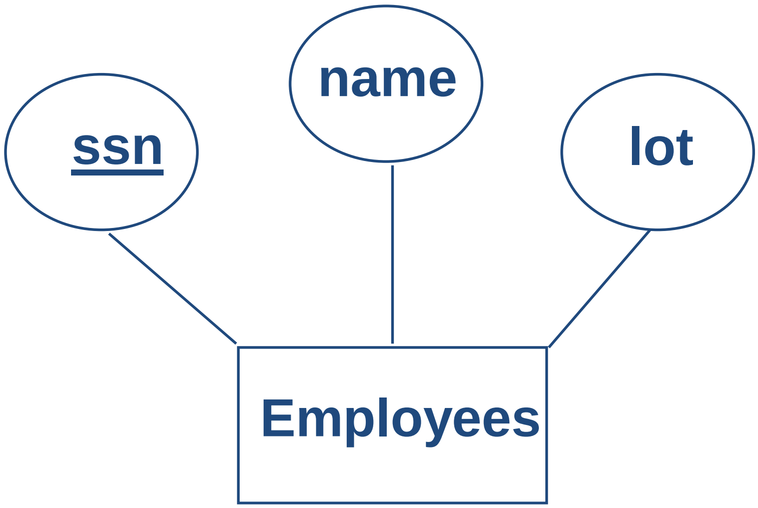
- What are the *entities* and *relationships* in the enterprise?
- What information about these entities and relationships should we store in the database?
- What are the *integrity constraints* or *business rules* that hold?

- A database `schema’ in the ER Model can be represented pictorially (*ER diagrams*).

- Can map an ER diagram into a relational schema.

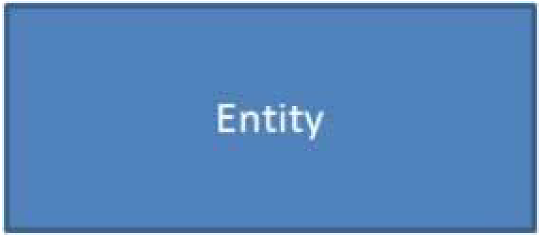
- The ER diagram is just an approximate description of the data, constructed through a subjective evaluation of the information collected during requirements analysis.

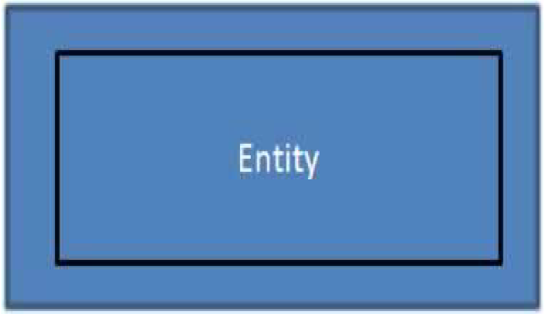
ER Model Basics



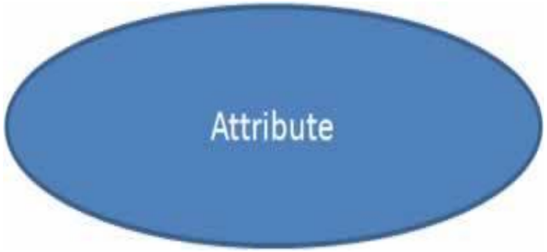
- *Entity:* Real-world object distinguishable from other objects. An entity is described using a set of *attributes*. Each attribute has a *domain*.
- *Entity Set*: A collection of similar entities. E.g., all employees.
  - All entities in an entity set have the same set of attributes. (Until we consider ISA hierarchies, anyway!)
  - Each entity set has a *key*.

- Two classes of entities are there:
  - Regular Entity or Strong Entity
  - Weak Entity



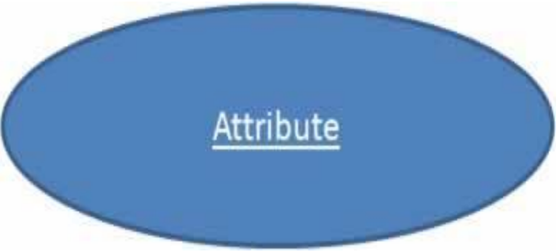


Attribute

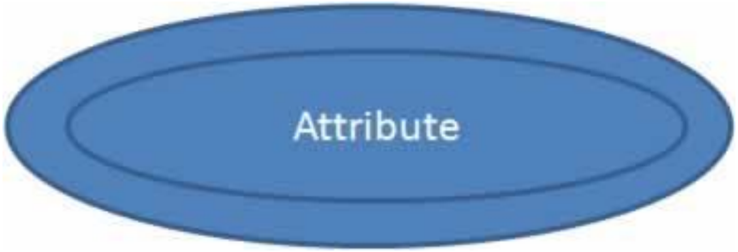


Properties/characteristics which describe entities are called attributes.

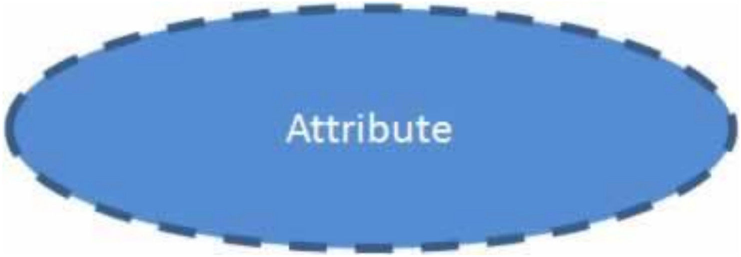
Key Attribute

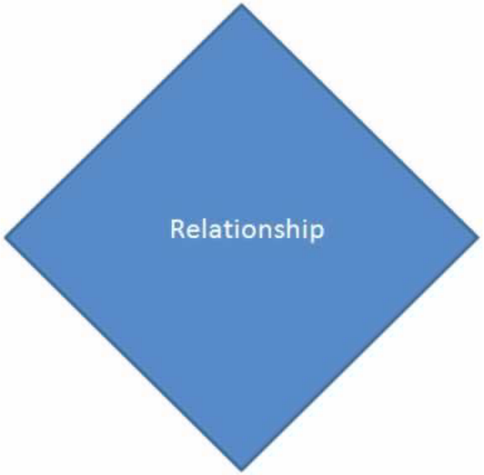


Multivalued Attributes



Derived Attribute: which can be derived from a given value



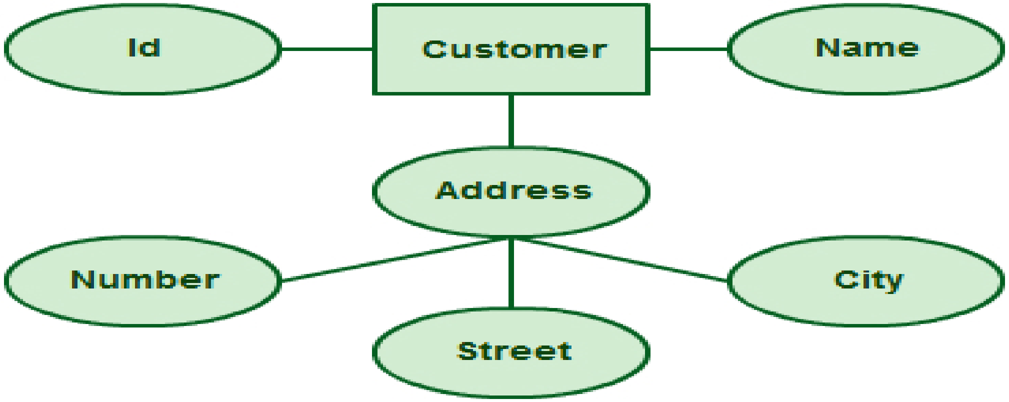


Relationships

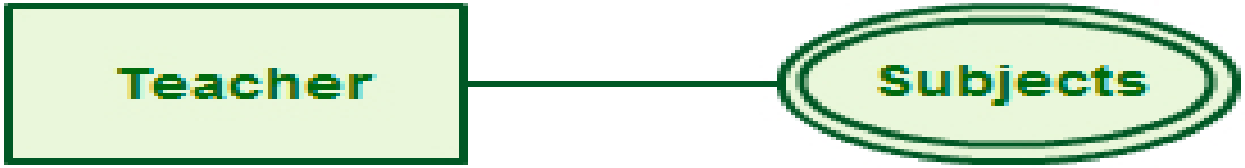
Associations between entities are called relationships

Examples of attributes

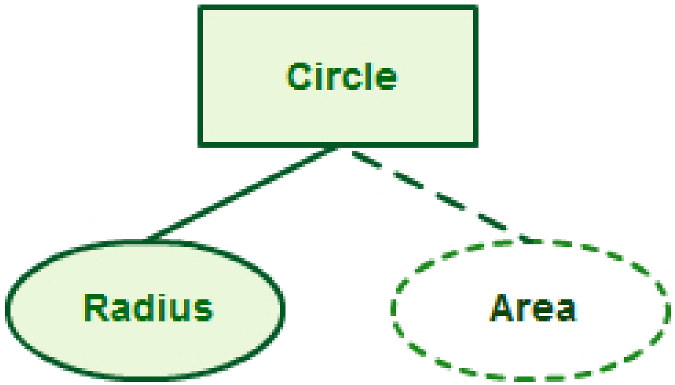
- Regular attributes of an entity



- Multivalued attributes

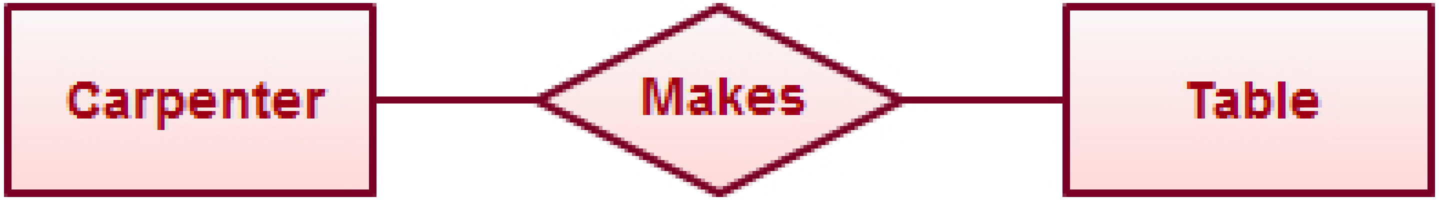


- Derived attribute

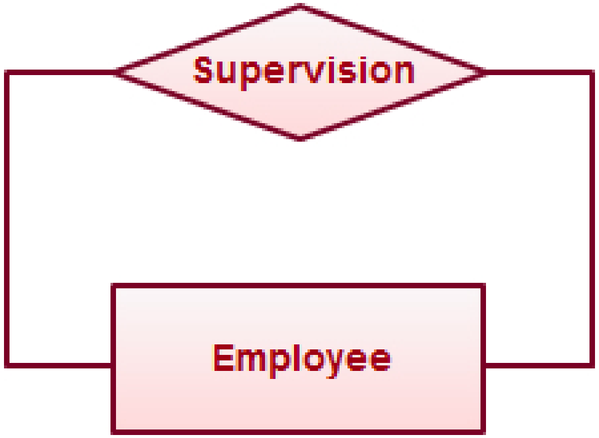


Examples of Relationship

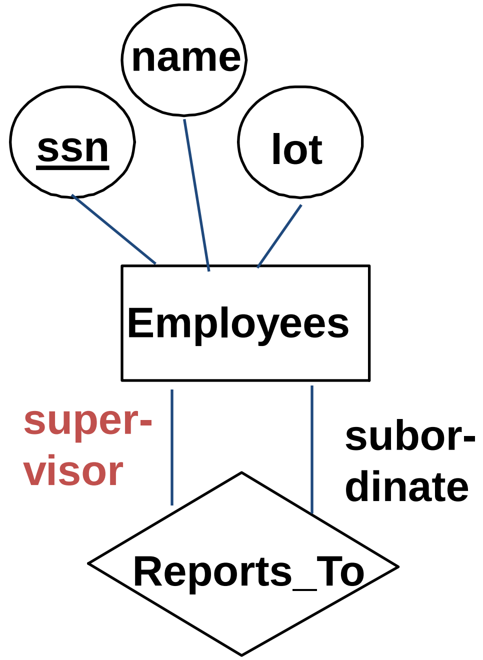
- Regular relationship

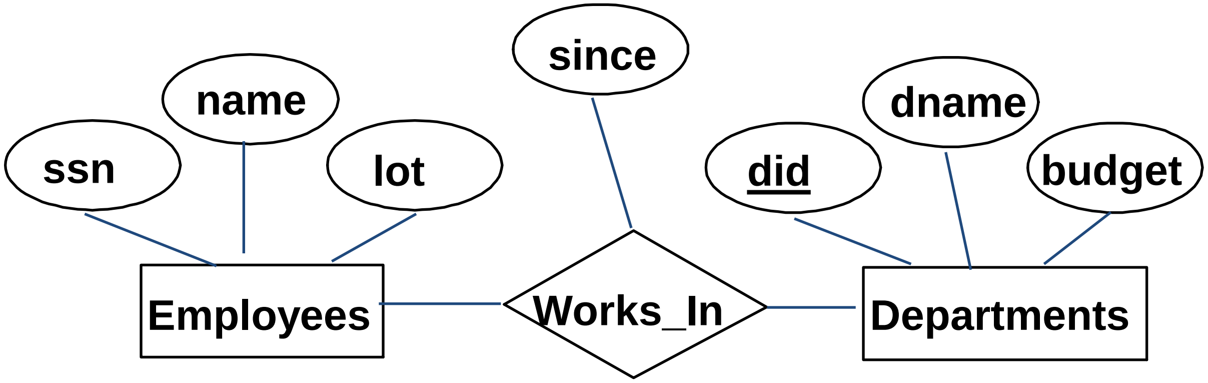


- Recursive Relationship



ER Model Basics (Contd.)



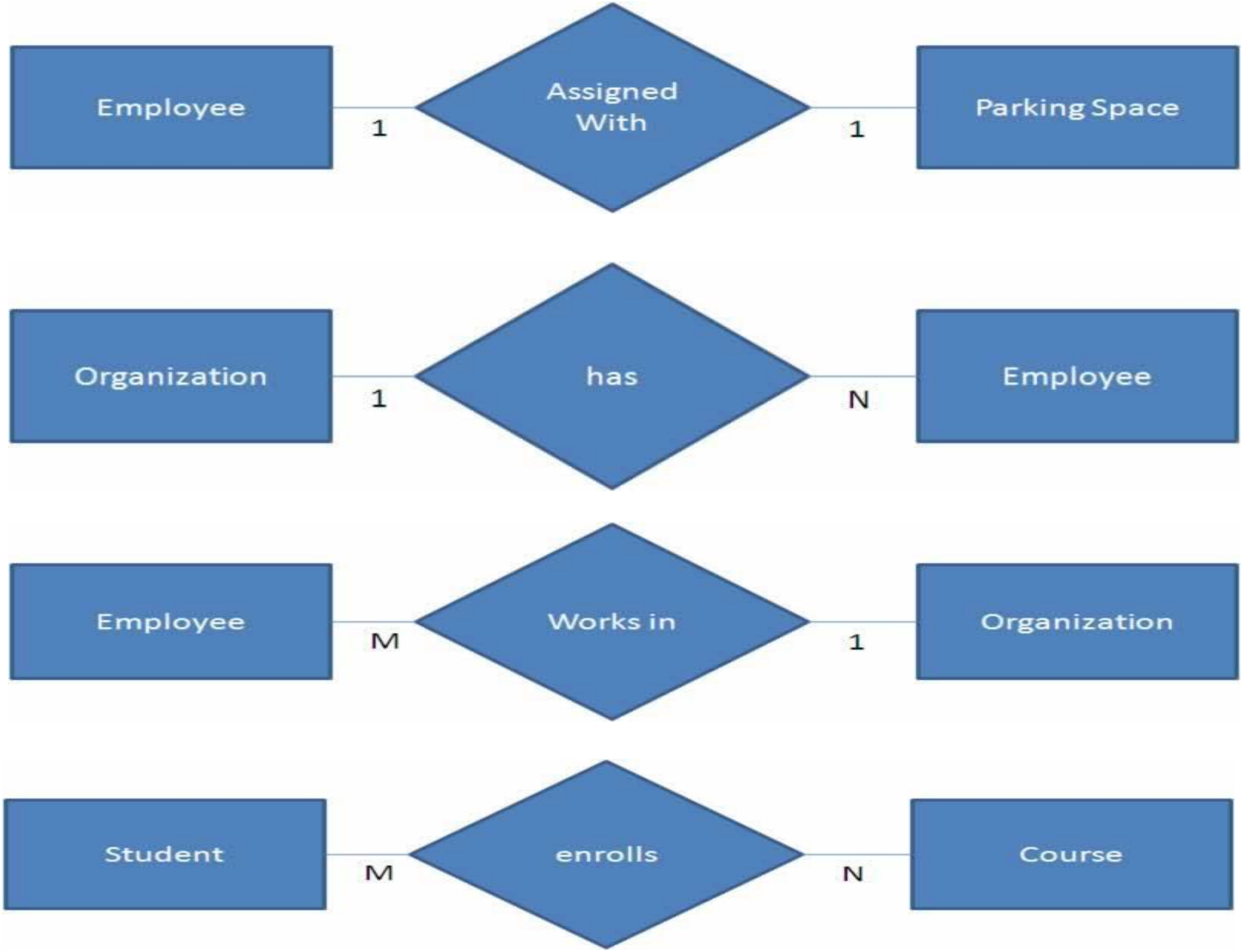


- *Relationship*: Association among two or more entities. E.g., Employee Atish works in Pharmacy department.
- *Relationship Set*: Collection of similar relationships.
  - An n-ary relationship set R relates n entity sets E1 ... En; each relationship in R involves entities e1 E1, ..., en En
  - Same entity set could participate in different relationship sets, or in different "roles" in same set.

Cardinality of a Relationship

- Relationship cardinalities specify how many of each entity type is allowed. Relationships can have four possible connectivity's as given below.
- 1. One to one (1:1) relationship
- 2. One to many (1:N) relationship
- 3. Many to one (M:1) relationship
- 4. Many to many (M:N) relationship

Examples


Employee
1
Assigned With
1
Parking Space
Organization
1
has
N
Employee
Employee
M
Works in
1
Organization
Student
M
enrolls
N
Course

Degree of a Relationship

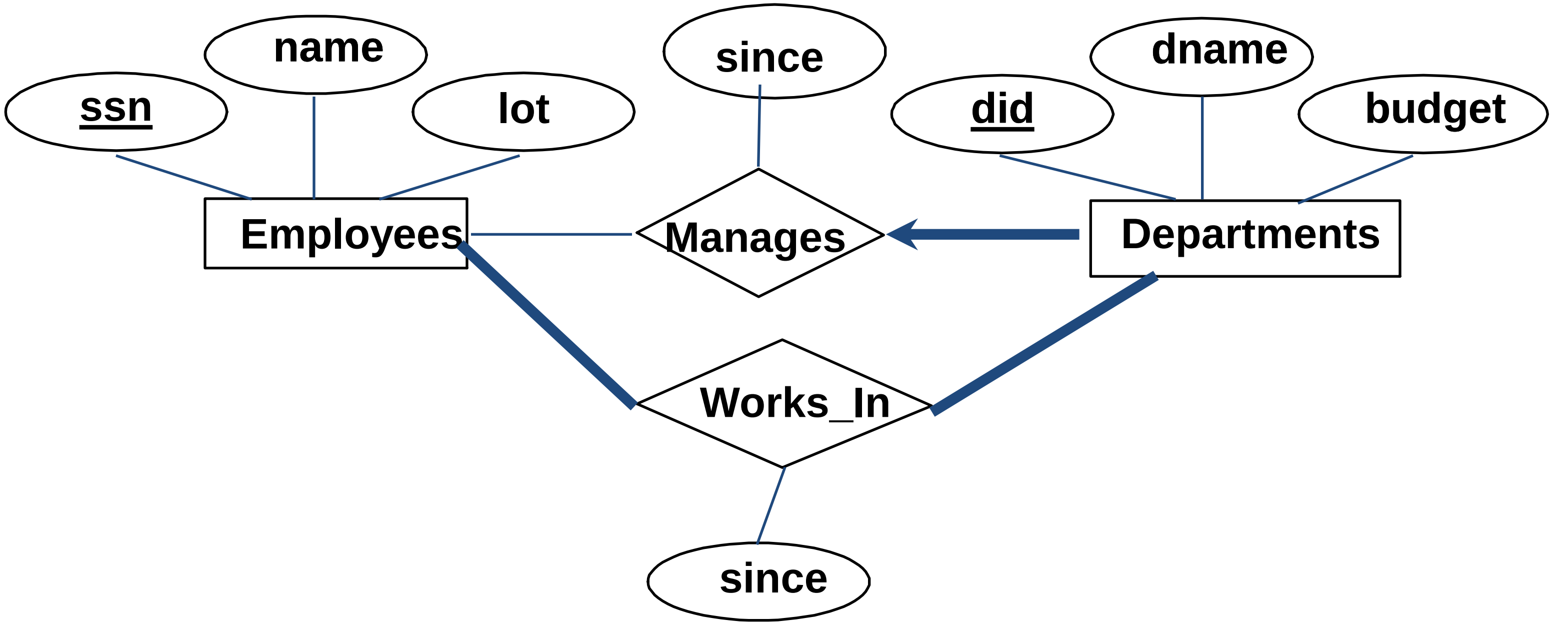
- Degree of a relationship is the number of entity types involved. The n-ary relationship is the general form for degree n.
- Special cases are unary, binary, and ternary where the degree is 1, 2, and 3, respectively.
- Example for unary relationship : An employee is a manager of another employee
- Example for binary relationship : An employee works-for department.
- Example for ternary relationship : customer purchase item from a shop keeper

Participation Constraints

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

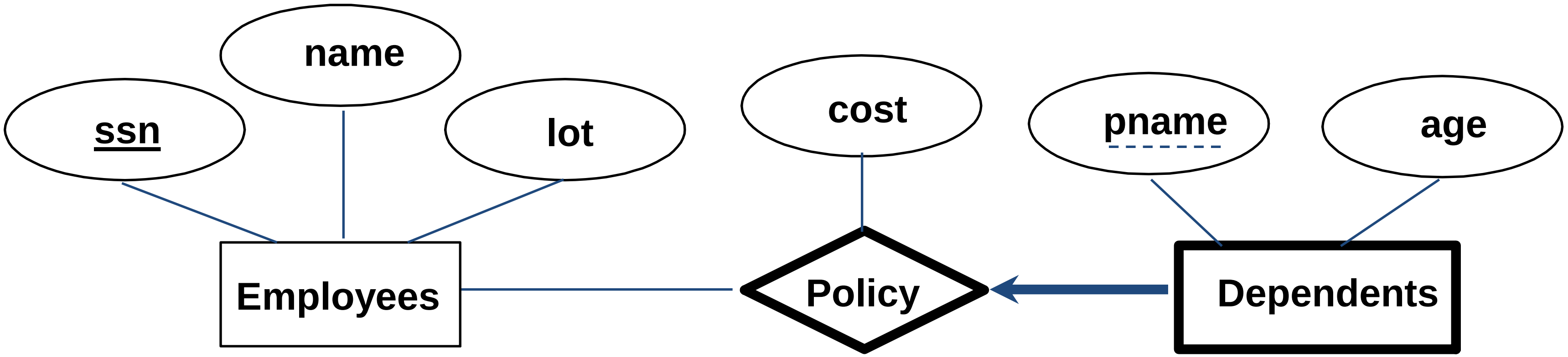
- Every Department entity must appear in an instance of the relationship Works_In (have an employee) and every Employee must be in a Department

- Both Employees and Departments participate totally in Works_In

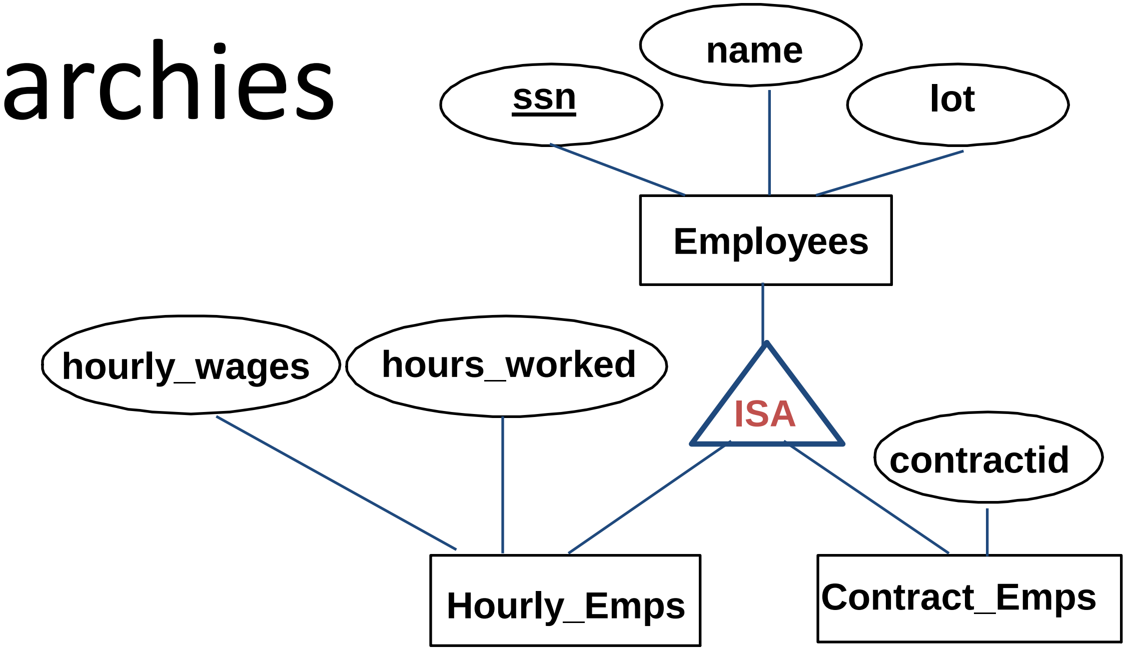


Weak Entities

- 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.



ISA (`is a’) Hierarchies



- ❖ As in C++, attributes can be inherited.
- ❖ If we declare A **ISA** B, every A entity is also considered to be a B entity.
- ❖Upwards is generalization. Down is specialization

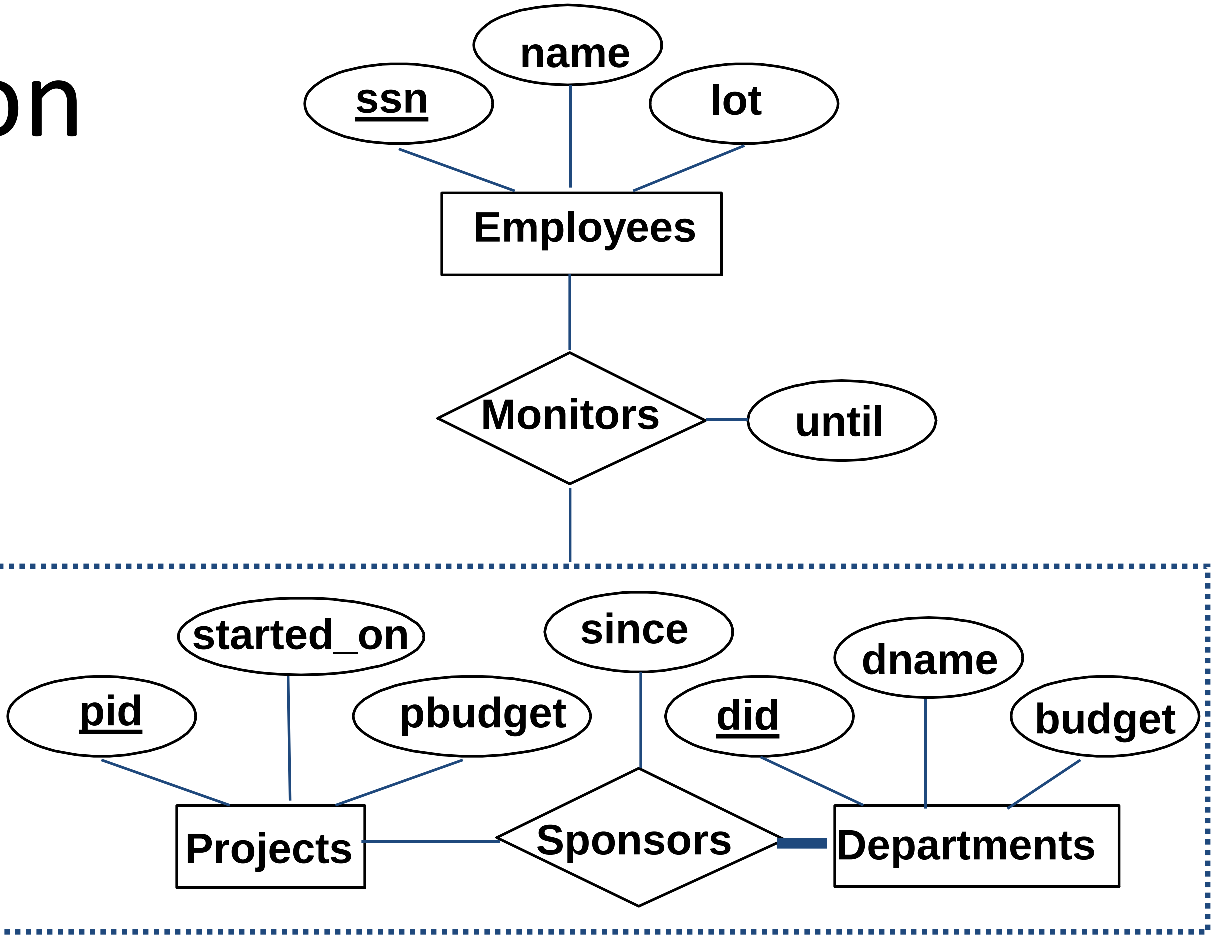
Constraints in ISA relation

- *Overlap constraints*: Can Joe be an Hourly_Emps as well as a Contract_Emps entity? (*Allowed/disallowed*)

- *Covering constraints*: Does every Employees entity also have to be an Hourly_Emps or a Contract_Emps entity? *(Yes/no)*

- Reasons for using ISA:
  - To add descriptive attributes specific to a subclass.
  - To identify entitities that participate in a relationship.

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 in last example is a distinct relationship, with a descriptive attribute.
- Also, can say that each sponsorship is monitored by at most one employee.

Conceptual Design Using the ER Model

- Design choices:
  - 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.

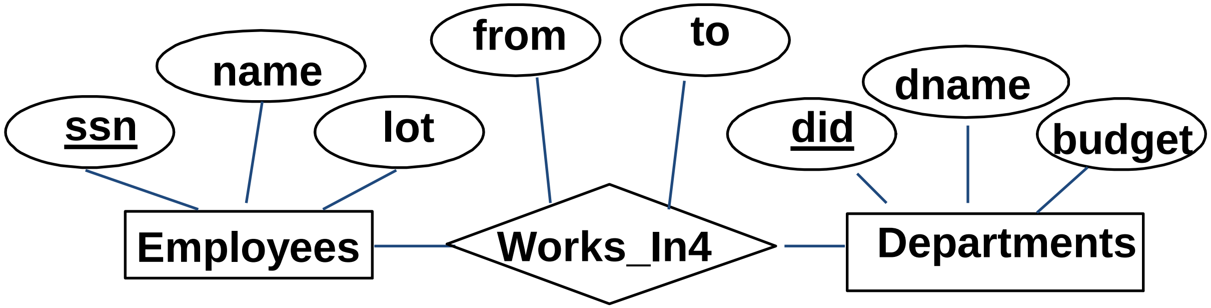
Entity vs. Attribute

- Should *address* be an attribute of Employees or an entity (connected to Employees by a relationship)?

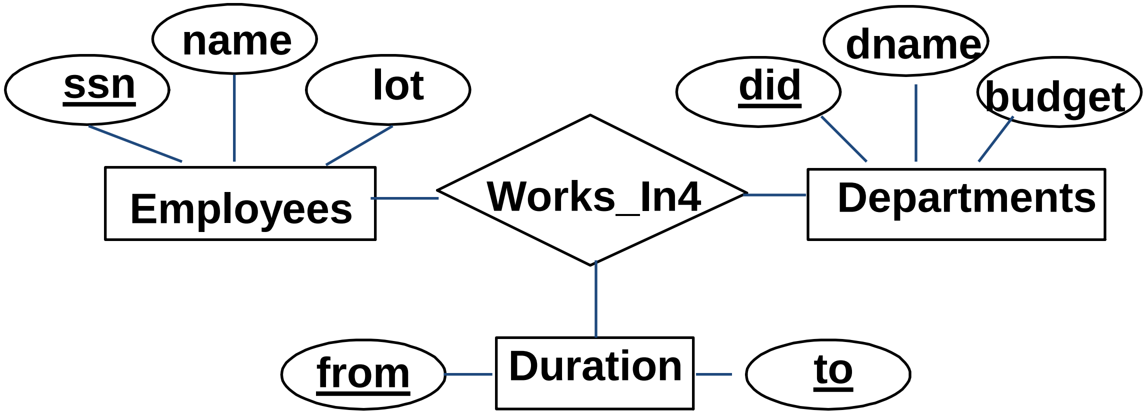
- Depends upon the use we want to make of address information, and the semantics of the data:
    - If we have several addresses per employee, *address* must be an entity (since attributes cannot be set-valued).
    - If the structure (city, street, etc.) is important, e.g., we want to retrieve employees in a given city, *address* must be modeled as an entity (since attribute values are atomic).

Entity vs. Attribute (Contd.)

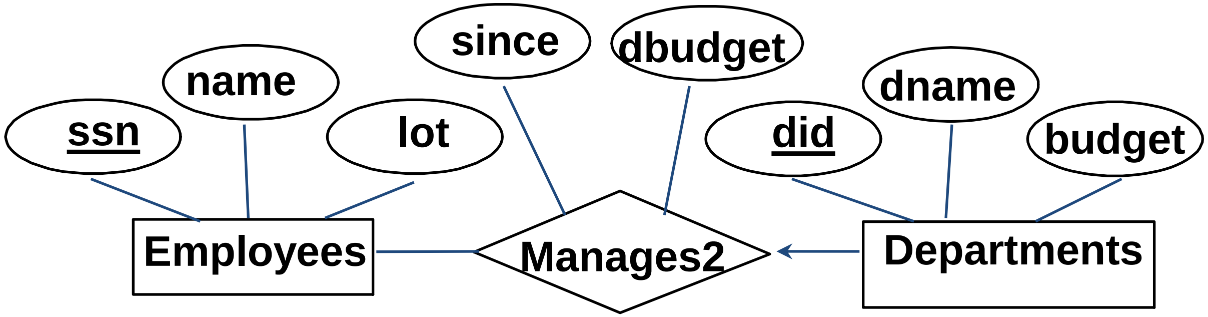
- Works_In4 does not allow an employee to work in a department for two or more periods.



- 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.* Accomplished by introducing new entity set, Duration.

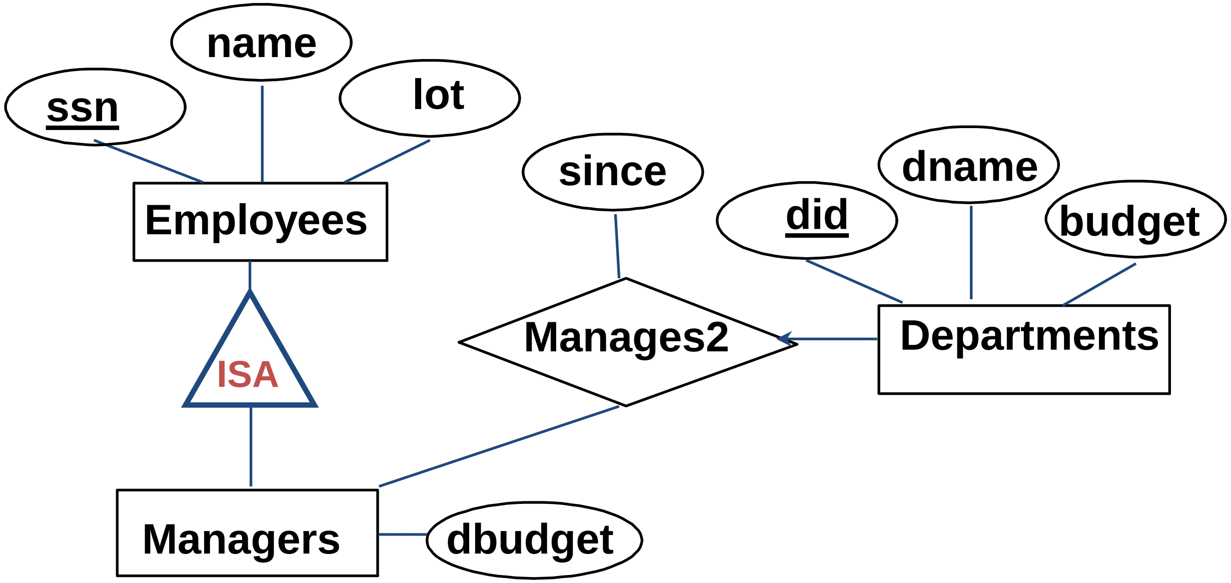


Entity vs. Relationship

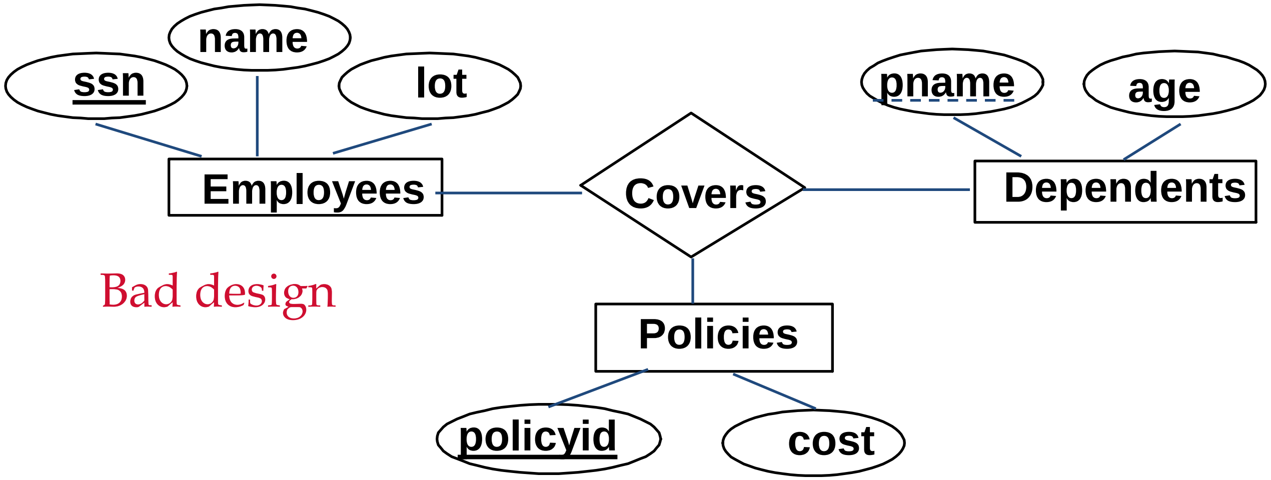


- ER diagram OK if a manager gets a separate discretionary budget for each dept.
- What if a manager gets a discretionary budget that covers *all* managed depts?
  - Redundancy: *dbudget* stored for each dept managed by manager.
  - Misleading: Suggests *dbudget* associated with department-mgr combination.

This fixes the problem!



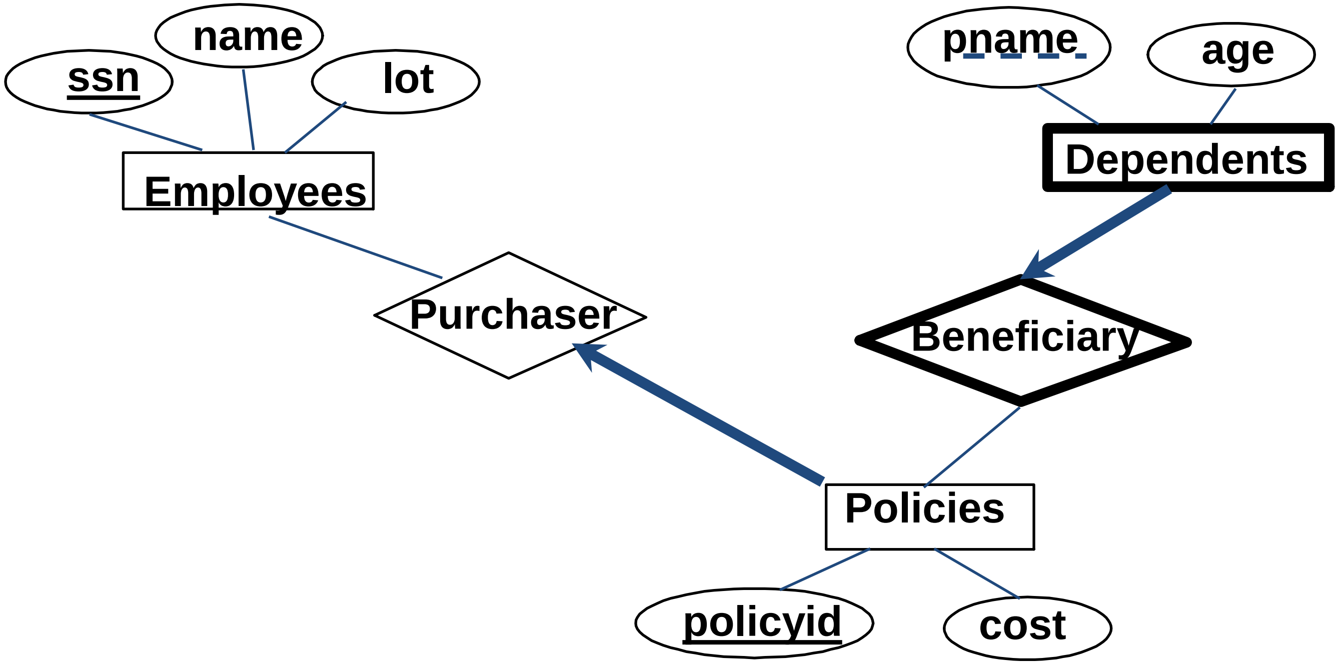
Binary vs. Ternary Relationships



Bad design

- If each policy is owned by just 1 employee, and each dependent is tied to the covering policy, first diagram is inaccurate.
- What are the additional constraints do we need?

Better design



- Define the Key constraint
- Total participation of policies in purchaser relationship
- The ternary relationships are break down into binary relationships again.

Summary of ER (Contd.)

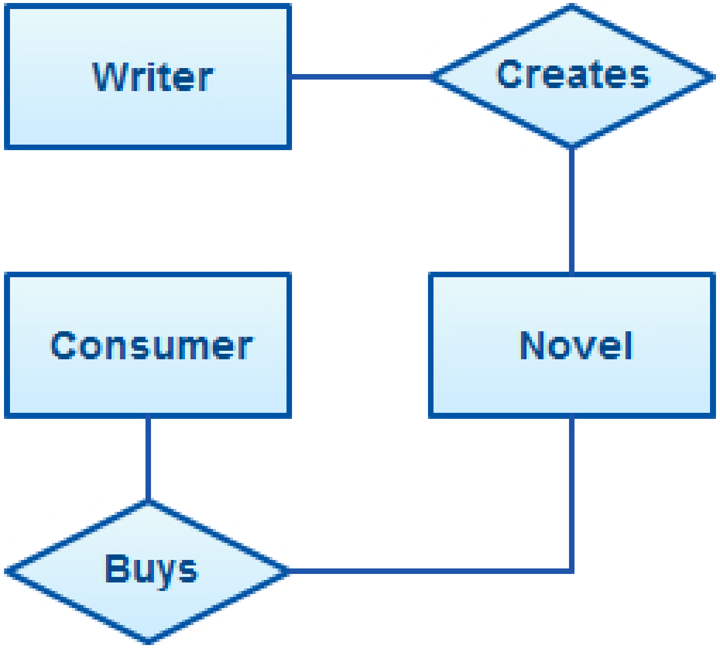
- Several kinds of integrity constraints can be expressed in the ER model: *key constraints*, *participation constraints*, and *overlap/covering constraints* for ISA hierarchies. Some *foreign key constraints* are also implicit in the definition of a relationship set.
  - Some constraints (notably, *functional 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.)

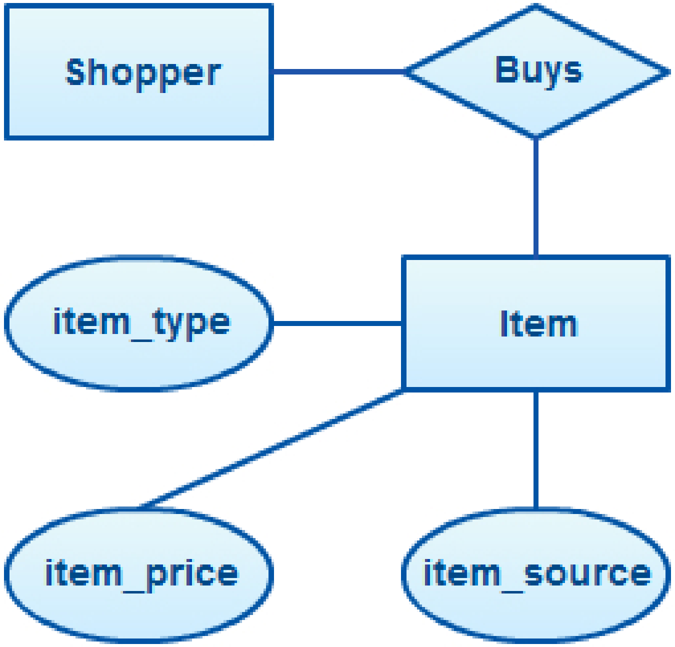
- ER design is *subjective*. There are often many ways to model a given scenario! Analyzing alternatives can be tricky, especially for a large enterprise. Common choices include:
  - Entity vs. attribute, entity vs. relationship, binary or n-ary relationship, whether or not to use ISA hierarchies, and 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.

Some more examples of ER Diagrams

- An Entity Relationship Diagram (ERD) is a visual representation of **different entities within a system and how they relate to each other**. For example, the elements writer, novel, and a consumer may be described using ER diagrams the following way:



- For example, an inventory software used in a retail shop will have a database that monitors elements such as purchases, item, item type, item source and item price. Rendering this information through an ER diagram would be something like this:



How to Draw ER Diagrams

- **Identify all the entities** in the system. An entity should appear only once in a particular diagram. Create rectangles for all entities and name them properly.
- **Identify relationships** between entities. Connect them using a line and add a diamond in the middle describing the relationship.
- **Add attributes** for entities. Give meaningful attribute names so they can be understood easily.

ER Diagram best practices

- Provide a precise and appropriate name for each entity, attribute, and relationship in the diagram. Terms that are simple and familiar always beats vague, technical-sounding words.

- In naming entities, remember to use singular nouns. However, adjectives may be used to distinguish entities belonging to the same class (part-time employee and full-time employee, for example). Meanwhile attribute names must be meaningful, unique, system-independent, and easily understandable.

- Remove vague, redundant or unnecessary relationships between entities.

- Never connect a relationship to another relationship.

- Make effective use of colors. You can use colors to classify similar entities or to highlight key areas in your diagrams.

ERD Case Study :

- In a University , there are several departments and each department has a head of department who belongs to Faculty. Department have a name , phone extension , specific mailing address and Students that belong to the department. Students can belong to only one Department at a time and Department can have more than one or no Student

- Students and faculty have names and unique identification numbers , with address , age , gender and other information. Student studies different Courses offered by University .

- Faculty teaches these *Courses* . In each semester one student can take more than one course and Faculty can teach more than one courses . Faculty members can teach in multiple Departments. Each course can be taught by many faculty members or no one

ERD Case Study :

- Faculty members are also working on multiple research projects. These projects are funded by government and university.

- One project can have more than one faculty member and one faculty member can work on more than one project

- Huff , Looks like a long task.

- Lets apply our four steps on this requirement. Think of them and study this requirement again.

- In a **University** , there are several **departments** and each **department** has a head of department who belongs to **Faculty**. **Department** have a name , phone extension , specific mailing address and Students that belong to the department. **Students** can belong to only one **Department** at a time and Department can have more than one or no Student.

- **Students** and **faculty** have names and unique identification numbers , with address , age , gender and other information. Student studies different
- Courses offered by **University** .

- Faculty teaches these ***Courses*** . In each semester one student can take more than one course and Faculty can teach more than one courses . Faculty members can teach in multiple **Departments**. Each course can be taught by many faculty members or no one

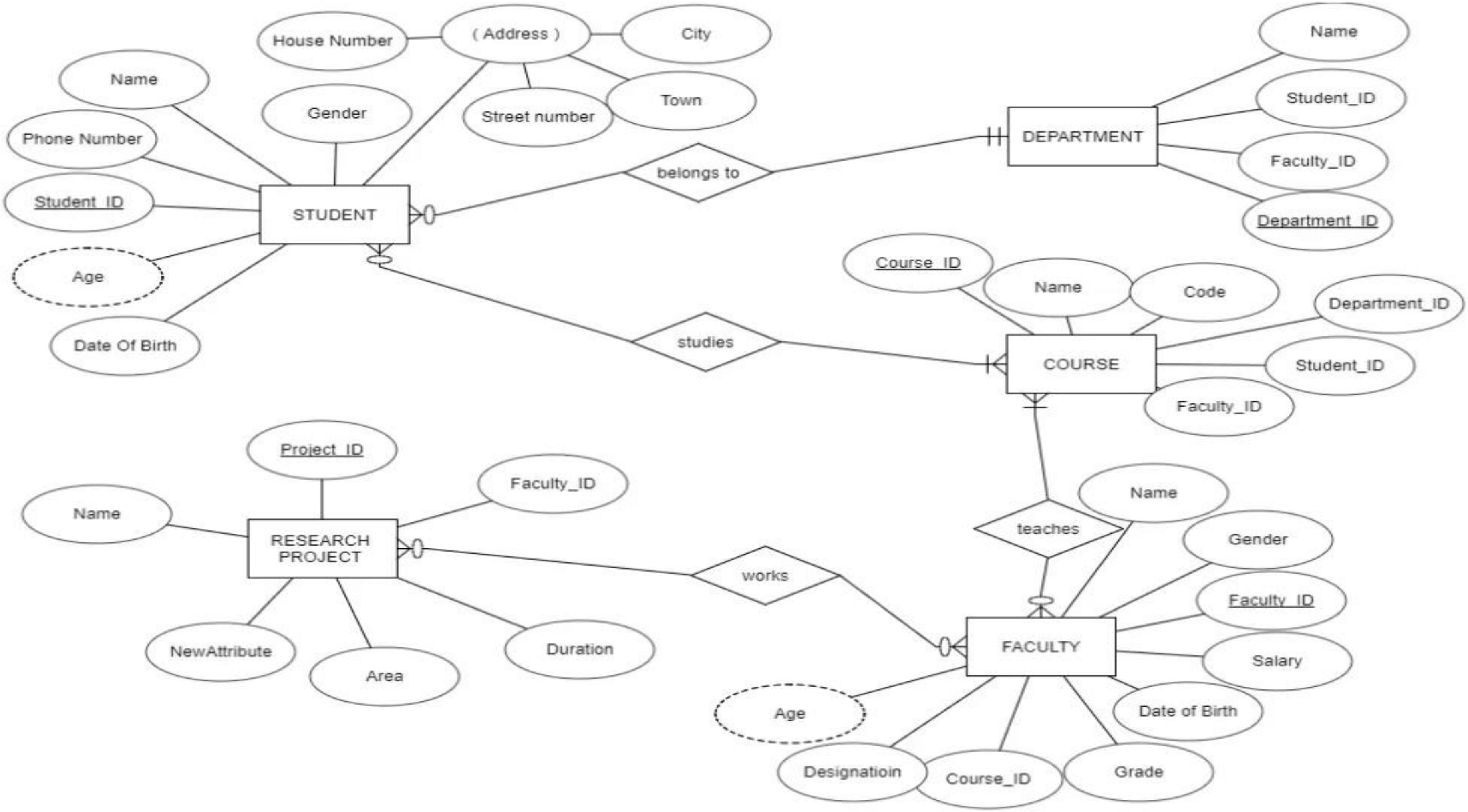
- Faculty members are also working on multiple **research projects**. These projects are funded by government and university. One project can have more than one faculty member and one faculty member can work on more than one project

- Decide the relationships, cardinality and degree of relationships. (Different software packages denote various notations of ER diagrams)

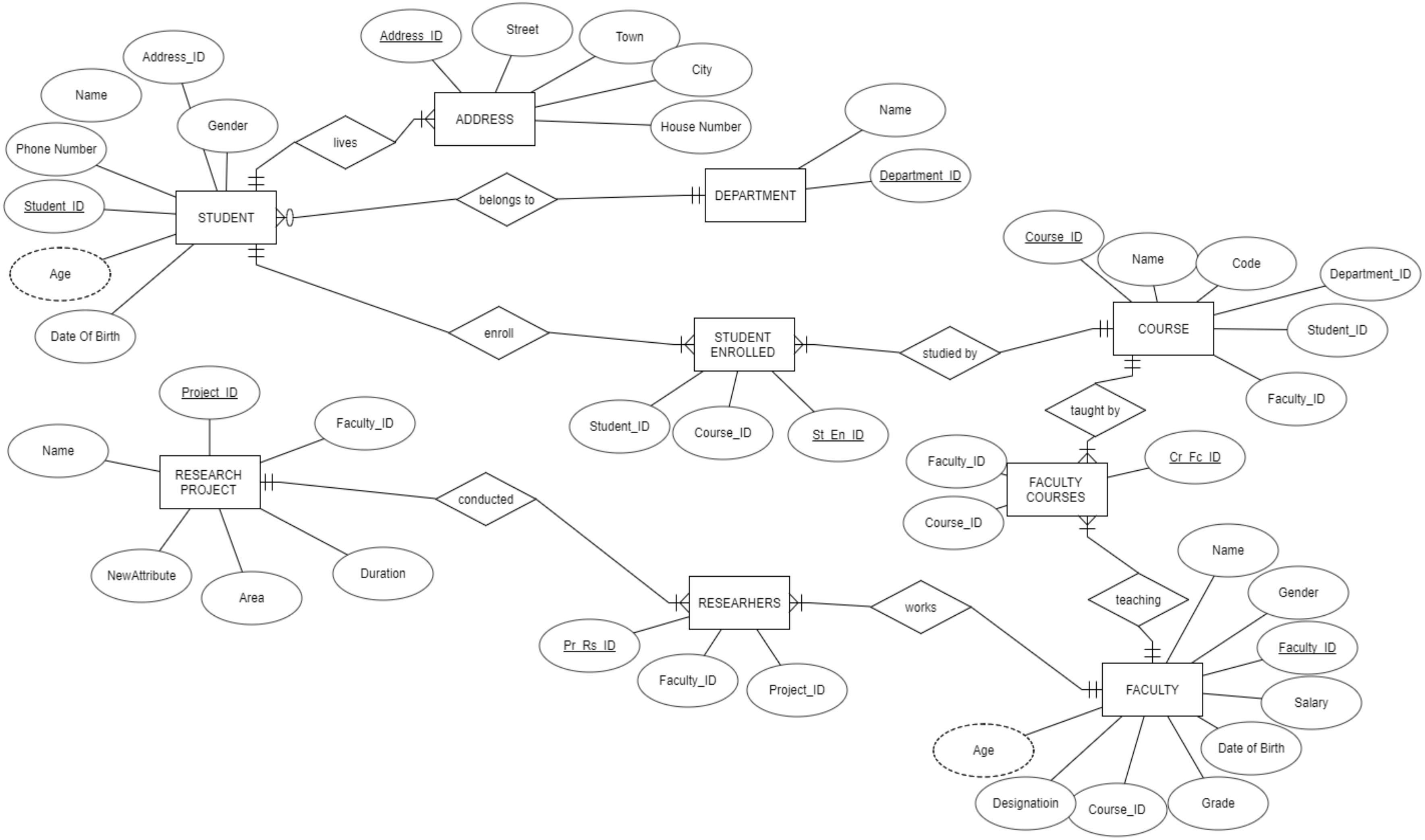
- Draw entities and attributes.

- A primary key is a must attribute for every entity .
- **Student** have Name , age , gender , address , phone Number , Roll Number , Semester , Course_ID and Student_ID.
- **Faculty** have Name , age , gender , address , phone Number , Semester , Course_ID, Grade , Salary , Faculty_ID and designation.
- **Course** have Name , Code , Student_ID , Faculty_ID , Department_ID and Course_ID.
- **Department** have Name , Student_ID , Faculty_ID and Department_ID.
- **Research Project** – Project_ID, Faculty_ID , Name , Duration.

Final ER diagram may look like this:



If we add student enrolled and Researchers , faculty courses information the final ER diagram may be



Address_ID
Street
Town
City
House Number
Address_ID
Name
Gender
Phone Number
Student_ID
Age
Date Of Birth
STUDENT
lives
ADDRESS
belongs to
DEPARTMENT
Name
Department_ID
enroll
STUDENT ENROLLED
Student_ID
Course_ID
St_En_ID
studied by
COURSE
Course_ID
Name
Code
Department_ID
Student_ID
Faculty_ID
taught by
FACULTY COURSES
Faculty_ID
Course_ID
Cr_Fc_ID
teaching
FACULTY
Name
Gender
Faculty_ID
Salary
Date of Birth
Grade
Course_ID
Designatioin
Age
RESEARCH PROJECT
Project_ID
Name
Faculty_ID
NewAttribute
Area
Duration
conducted
RESEARHERS
Pr_Rs_ID
Faculty_ID
Project_ID
works