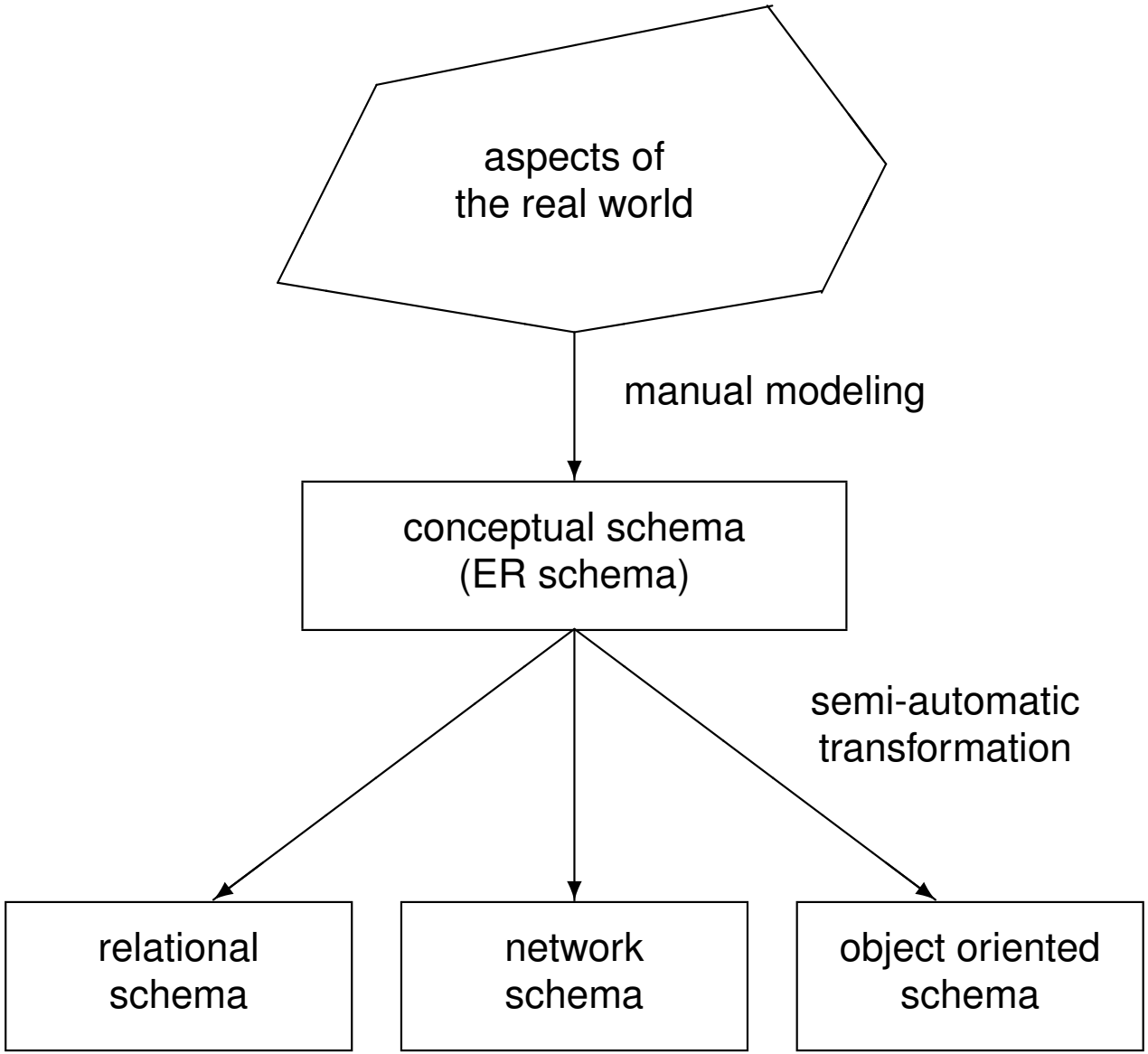
The Entity Relationship Model

- An example of a conceptual (high-level) data model
- Useful for design before moving to a lower level model (e.g. relational)
- Alternatives include
  - UML (Unified Modelling Language) and
  - ODL (Object Description Language)

Modelling Steps



The Components of the ER Model

- Structural part
  - entity types
  - attributes
  - relationship types

The Components of the ER Model

- Structural part
  - entity types
  - attributes
  - relationship types
- Integrity constraints
  - primary keys for entity types and relationship types, and
  - multiplicity (cardinality) constraints for relationship types

The Components of the ER Model

- Structural part
  - entity types
  - attributes
  - relationship types
- Integrity constraints
  - primary keys for entity types and relationship types, and
  - multiplicity (cardinality) constraints for relationship types
- The ER model is only a partial data model, since it has no standard manipulative part

The Main Advantages of the ER Model

The diagrams (ERDs) associated with entity-relationship models

- are relatively simple
- are user-friendly
- can provide a unified view of data, which is independent of any *implemented* data model

The Building Blocks of an ERD

- Entities and entity types (entity sets)
- Relationships and relationship types
- Attributes, keys and roles

Entities and Entity Types

An *entity* is a “thing” that exists and can be uniquely identified, e.g., an individual person.

An *entity type* (or *entity set*) is a collection of similar entities, e.g., a collection of people.

An entity type

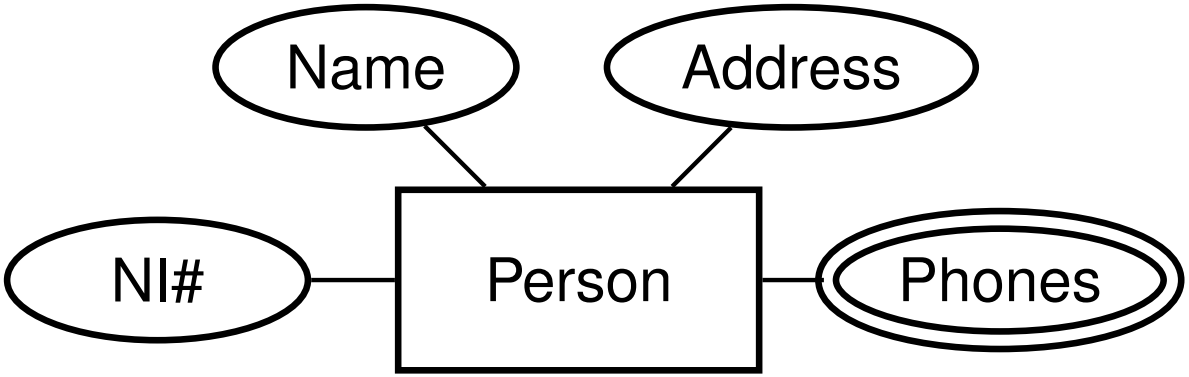
- is similar to a class in object-oriented languages
- has associated *attributes*, which represent properties of the entities comprising the entity type
- is represented in an ERD by a *rectangle*

Attributes

Attribute names (or simply *attributes*) are properties of entity types.

- Each attribute usually has only *simple* values (e.g. integers or character strings)
  - represented by an *oval* in an ERD, with a line to the rectangle representing its entity type
- Sometimes *multi-valued* attributes (e.g. Phones) are allowed
  - represented by a *double oval* in an ERD

Example of Entity Type and Attributes



- Each Person has a single Name, Address and National Insurance number (NI#)
- Each Person can have many Phones

Domain

The *domain* of an attribute of an entity type is the set of constant values associated with that attribute.

Domains can be

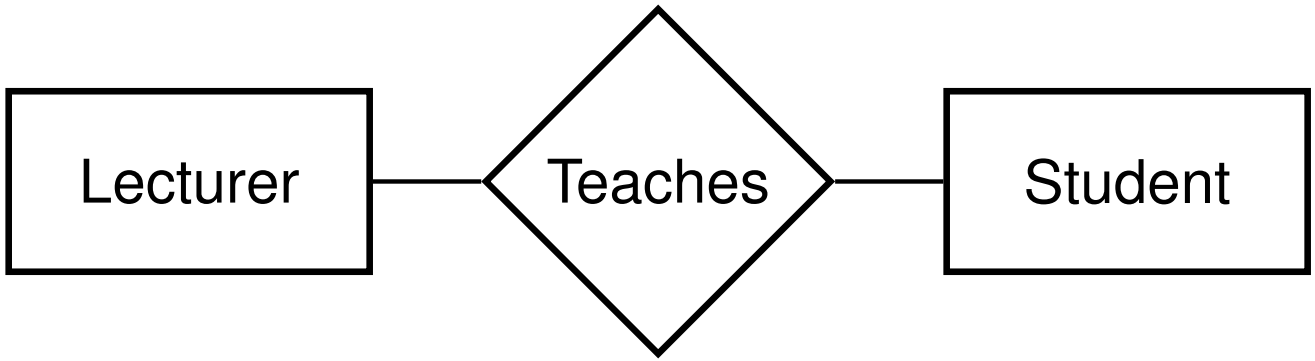
- **atomic** such as the domain of integers or the domain of strings, or
- **set-valued** such as the domain of finite sets of integers or of finite sets of strings.

Relationship Types

A *relationship type* is an association between two or more entity types.

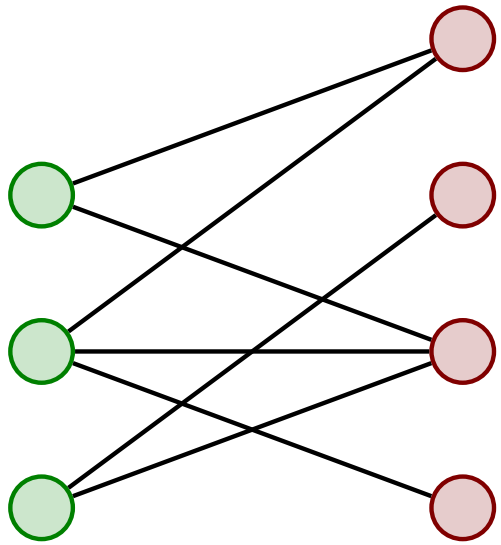
- Relationship types are represented in an ERD by diamond shapes,
- with lines to each of the rectangles representing entity types involved in the association

Example of a relationship type *Teaches* between entity type *Lecturer* and entity type *Student*:



Relationships

A *relationship* is an instance of a relationship type, i.e., it is the set of associations between the *entities* comprising the entity types associated by the relationship type.

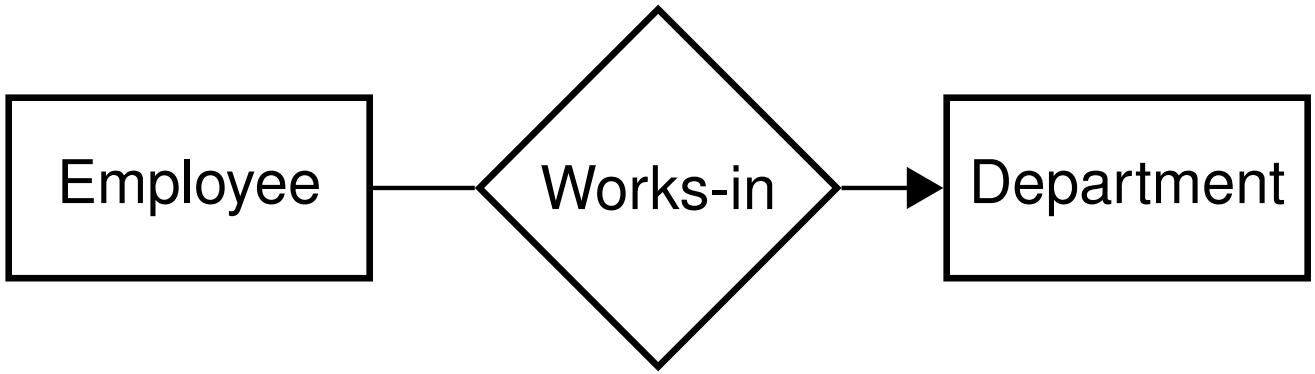


- Above diagram is *not* an ERD
- Entities for one entity type on left (green)
- Entities for another entity type on right (red)
- Lines between entities show the relationship

Multiplicity Constraints in Relationship Types

- many-to-one (or one-to-many)
  An Employee Works in one Department or a Department has many Employees.
- one-to-one
  A Manager Heads one Department and vice versa.
- many-to-many
  A Lecturer Teaches many Students and a Student is Taught by many Lecturers.

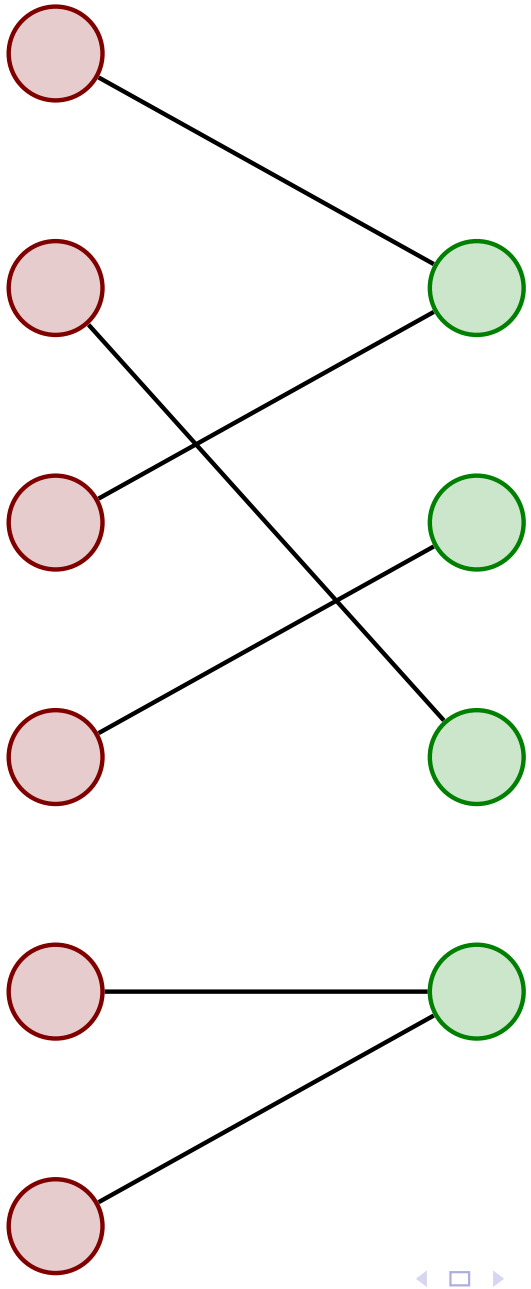
Example of many-to-one relationship type



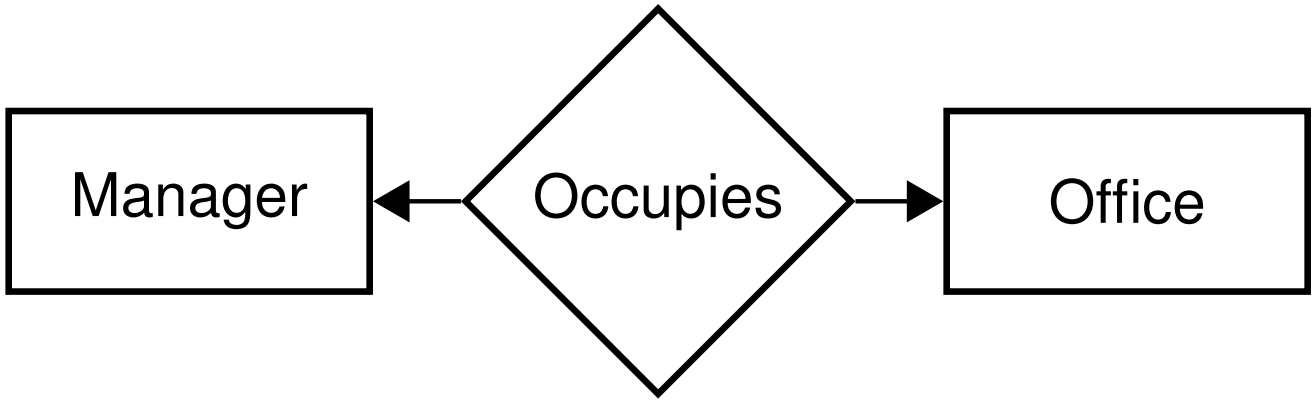
The arrowhead is drawn at the “one” end of the relationship type

- Each Employee Works-in one Department
- Each Department has many Employees working in it

Example of many-to-one relationship



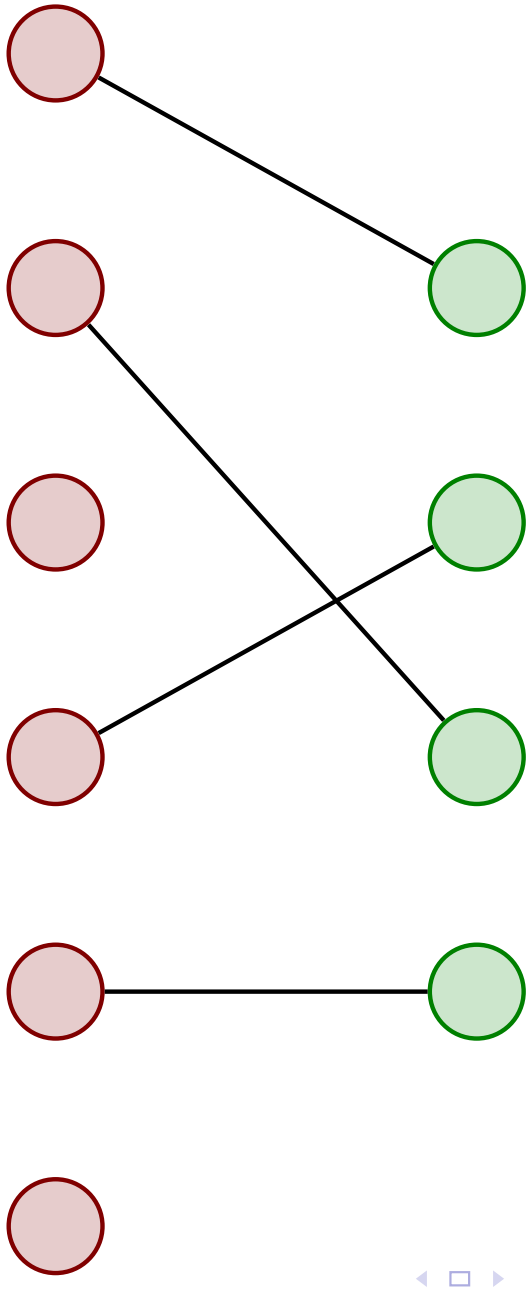
Example of one-to-one relationship type



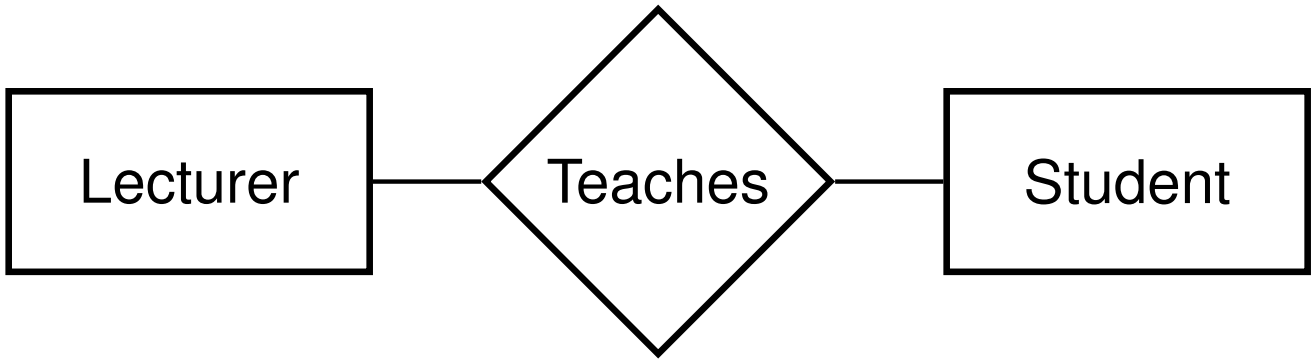
The arrowhead is drawn at both ends of the relationship type

- Each Manager Occupies one Office
- Each Office has one Manager occupying it

Example of one-to-one relationship



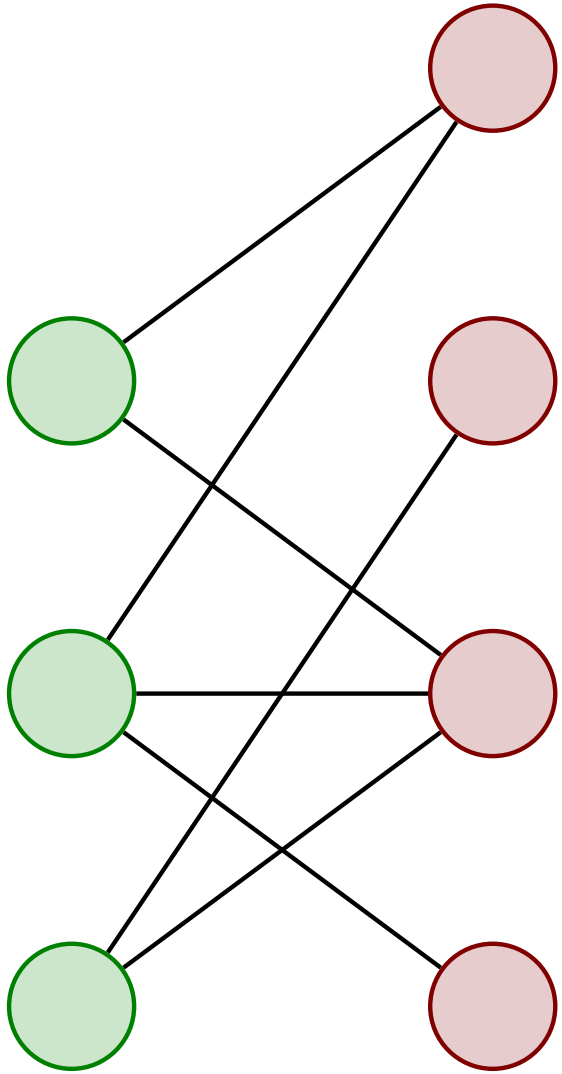
Example of many-to-many relationship type



No arrowheads are drawn

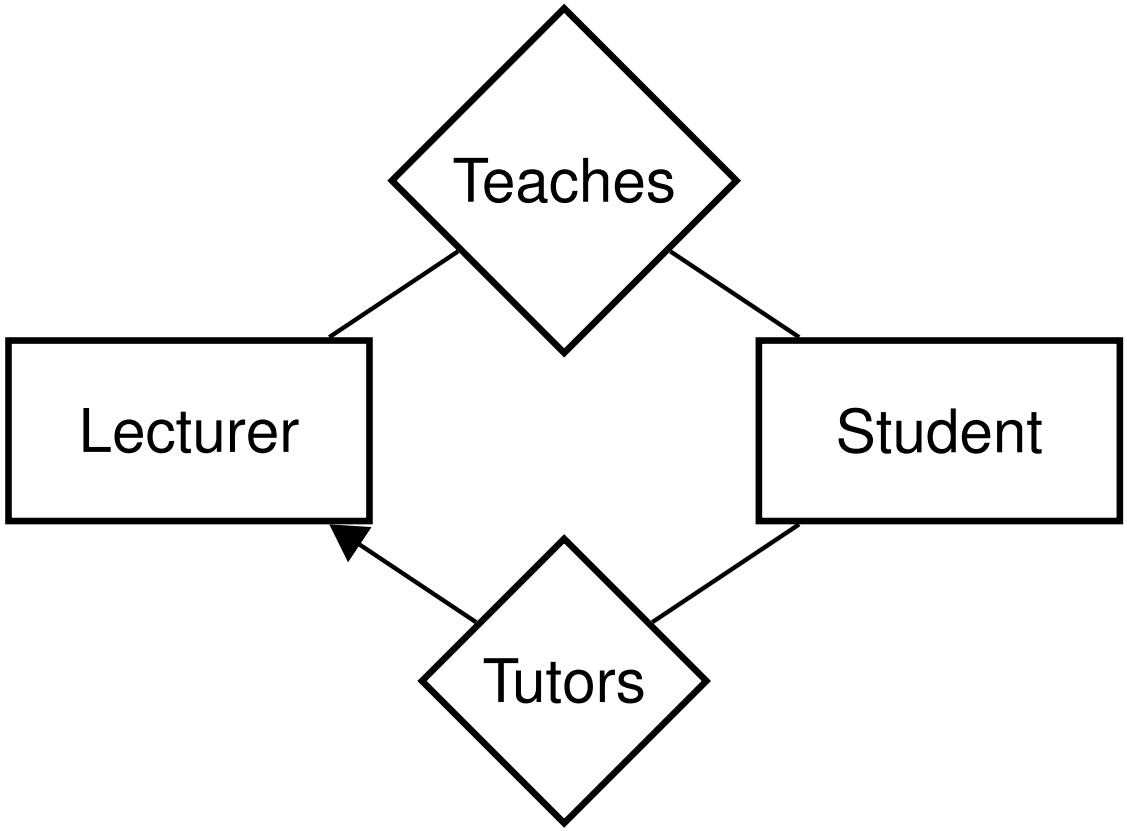
- Each Lecturer Teaches many Students
- Each Student is taught by many Lecturers

Example of many-to-many relationship



Multiple relationship types

There may be more than one relationship type between two entity types.

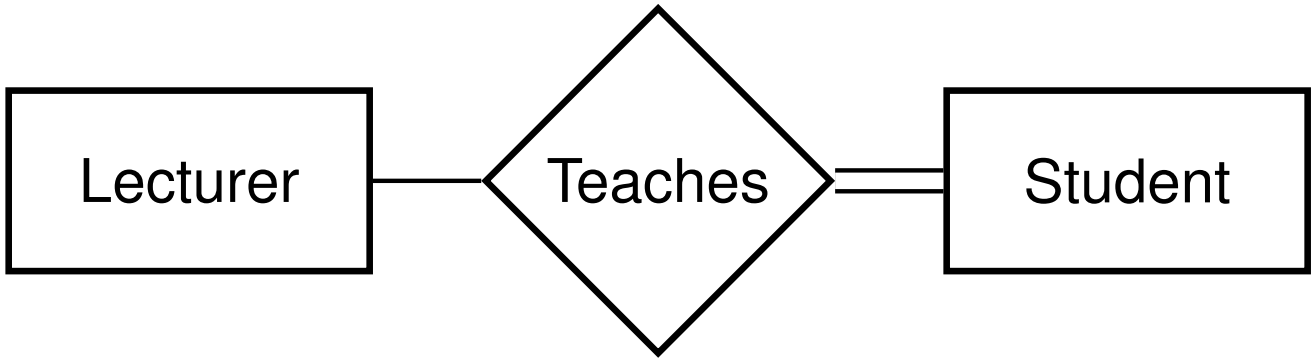


- Teaches relationship type is many-to-many
- Tutors relationship type is one-to-many (from Lecturer to Student)

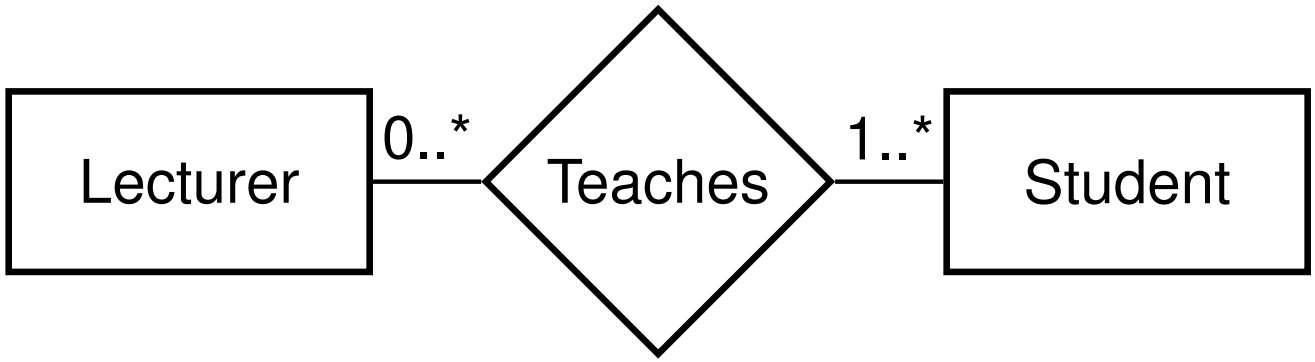
Participation Constraints in Relationships

- **optional** relationship
  - e.g., an Employee may or may not be assigned to a Department
  - Participation of entity (e.g., Employee) in relationship is said to be *partial*
  - This is the default for relationships in our notation
  - (Sometimes indicated by cardinality constraint of 0..*)
- **mandatory** relationship
  - e.g., every Course must be Taught by at least one Lecturer
  - Participation of entity (e.g., Course) in relationship said to be *total*
  - Indicated by double lines in our notation
  - (Sometimes indicated by cardinality constraint of 1..*)

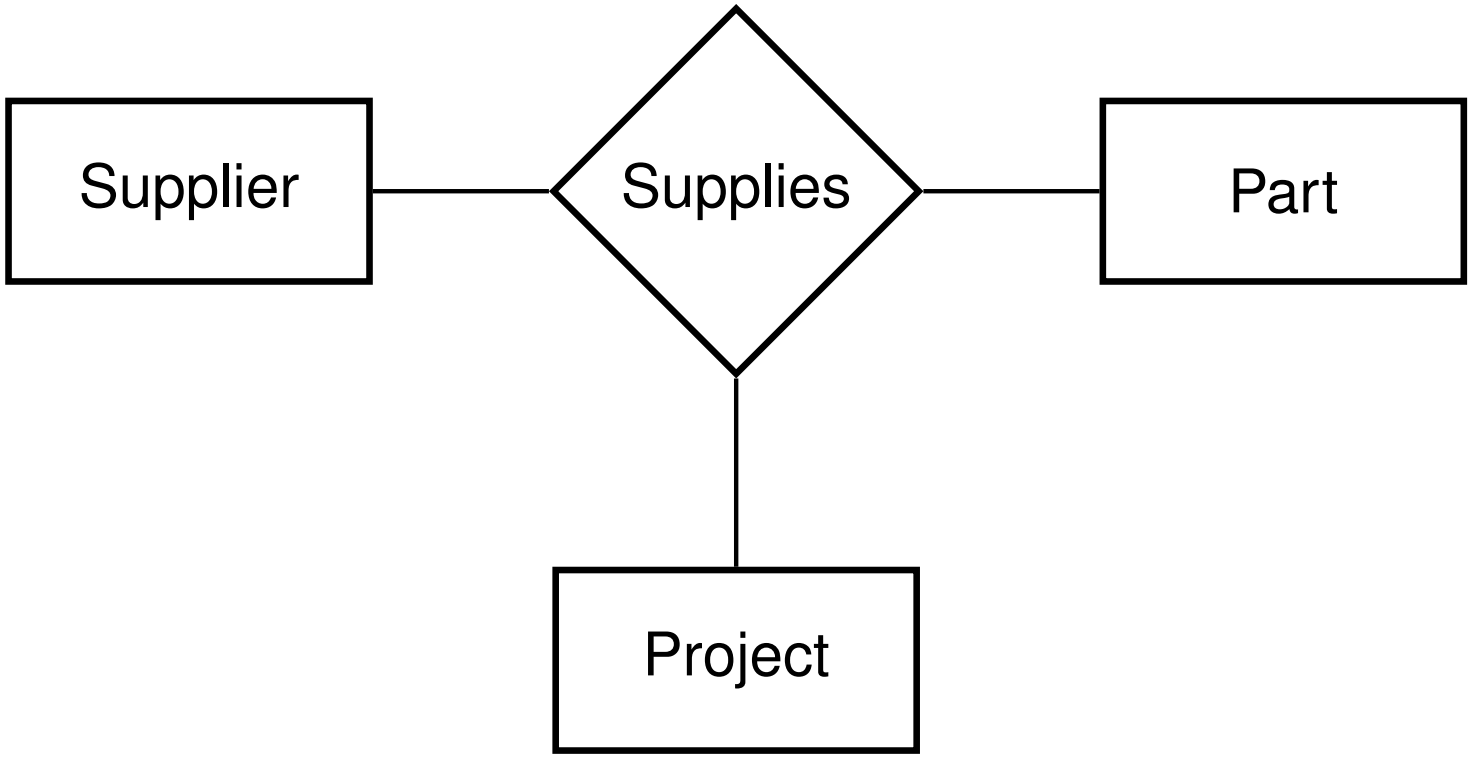
Example of Participation Constraints



- Some Lecturers may not Teach any Students
- Each Student *must* be taught by at least one Lecturer



Binary Versus Multiway Relationship Types



- each supplier may supply different parts to different projects
- Supplies is an example of a *ternary* relationship type

Keys and Superkeys

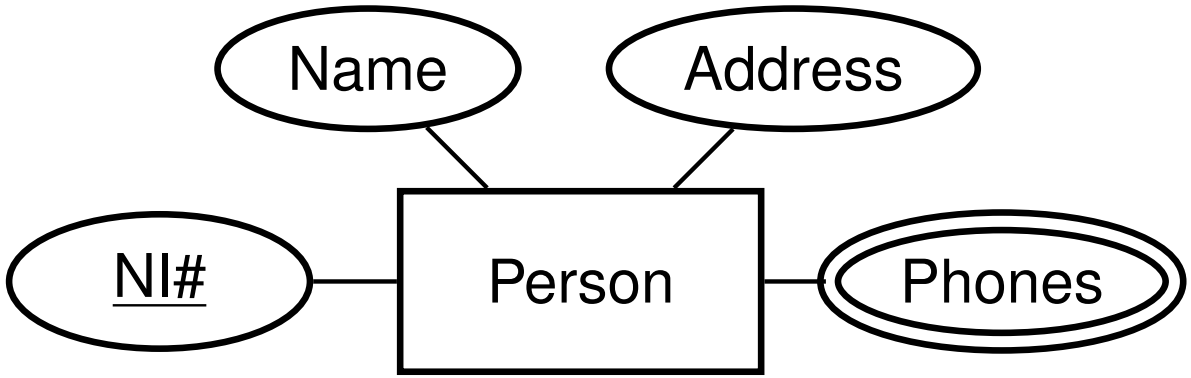
Definition of a superkey. A set of attributes of an entity type is a *superkey* if for each entity, say *e*, over that type, the set of values of the attributes in the superkey *uniquely* identifies *e*.
e.g., a Person may be uniquely identified by {Name, NI#}

Definition of a key. A *key* for an entity type is a superkey which is *minimal*.
e.g., to uniquely identify a Person, NI# is sufficient

- **simple** keys are single attribute keys, such as Emp# and NI#.
- **composite** keys are keys having more than one attribute, such as {Department, University} and {Name, Address}.

Definition of primary key of an entity type. A *primary key* is a key that has been designated as such by the database designer.

- The primary key guarantees logical access to every entity.
- Attributes that are members of the primary key of an entity type are *underlined* in an ERD.



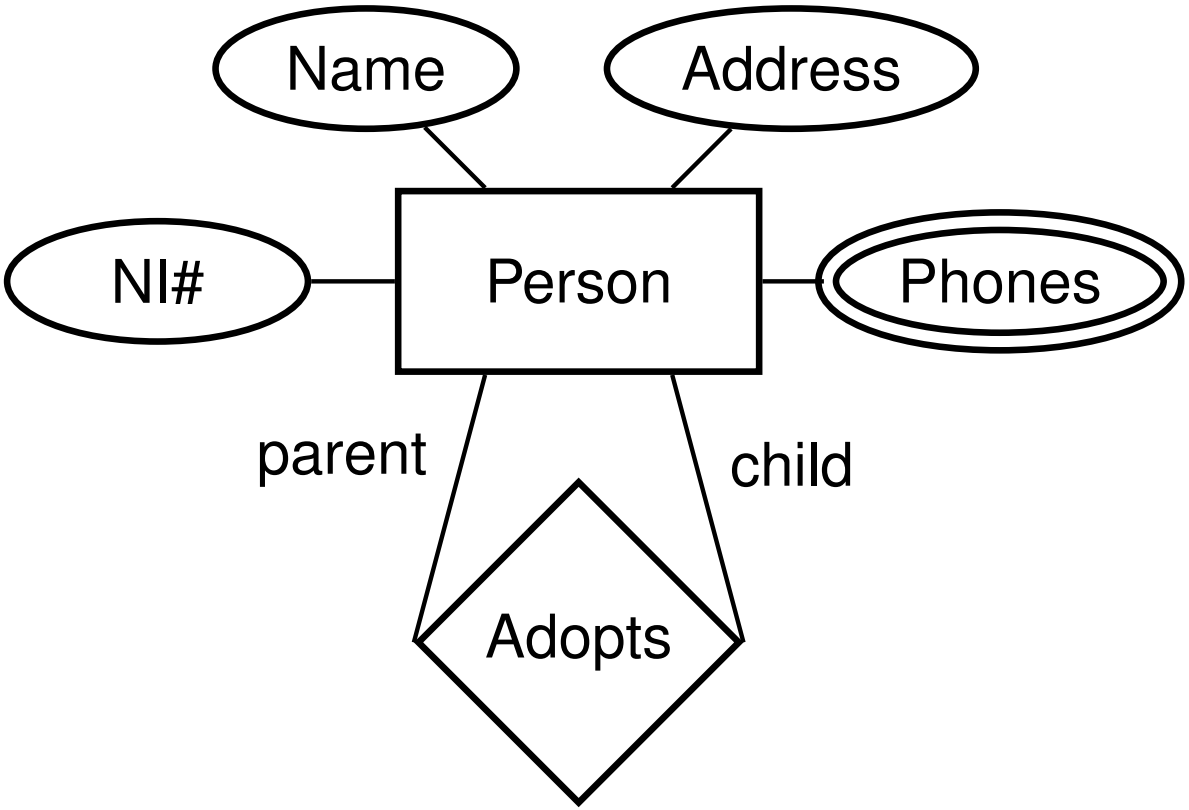
Cyclic Relationship Types

Definition. A *cyclic relationship* type (also called *recursive*) is a relationship type between two occurrences of the same entity type.
Examples:

- Marriage between Person and itself.
- Parent-Child between Person and itself.
- Part-Sub-Part between Part and itself.

With each entity type in a cyclic relationship type we associate a *role*, represented by labels on lines in an ERD.

Example of a Cyclic Relationship Type with Roles

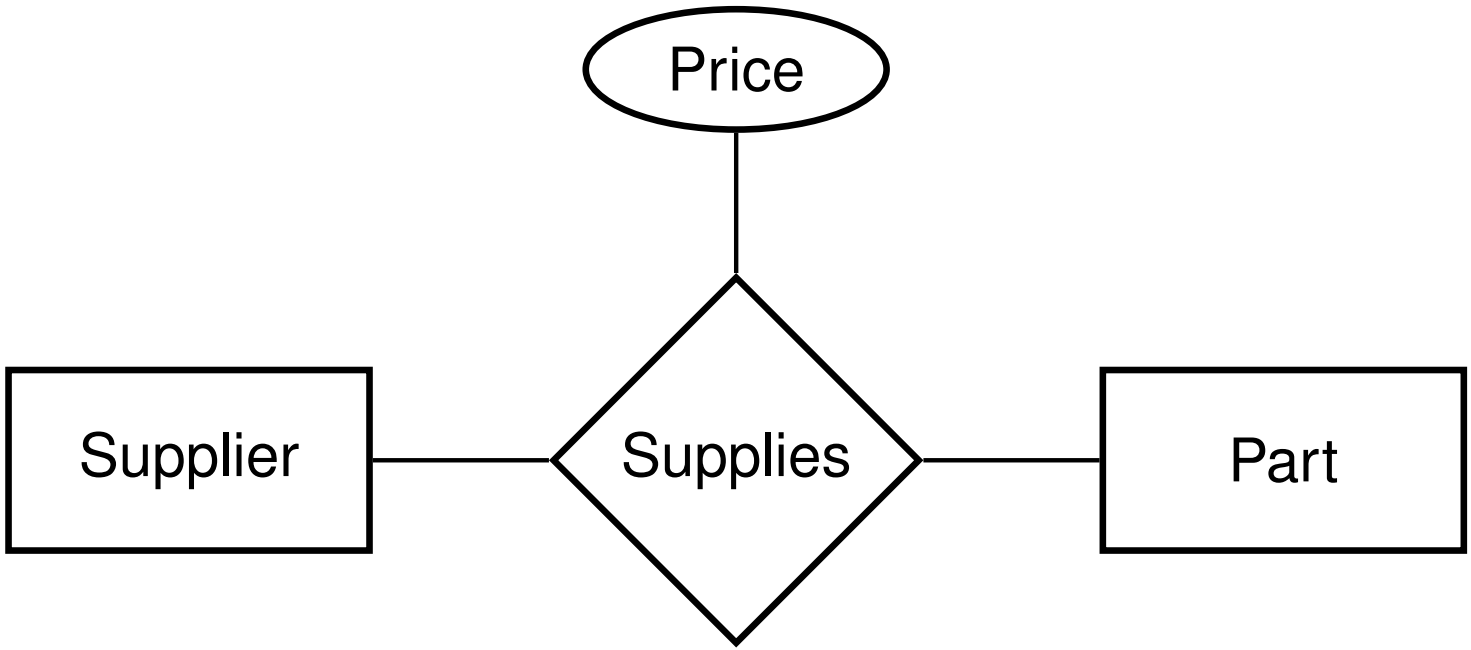


Roles are specified by “parent” and “child” labels.

Relationship Types with Attributes

Examples:

- attribute Date associated with the relationship type Married
- attribute Price associated with the relationship type Supplies



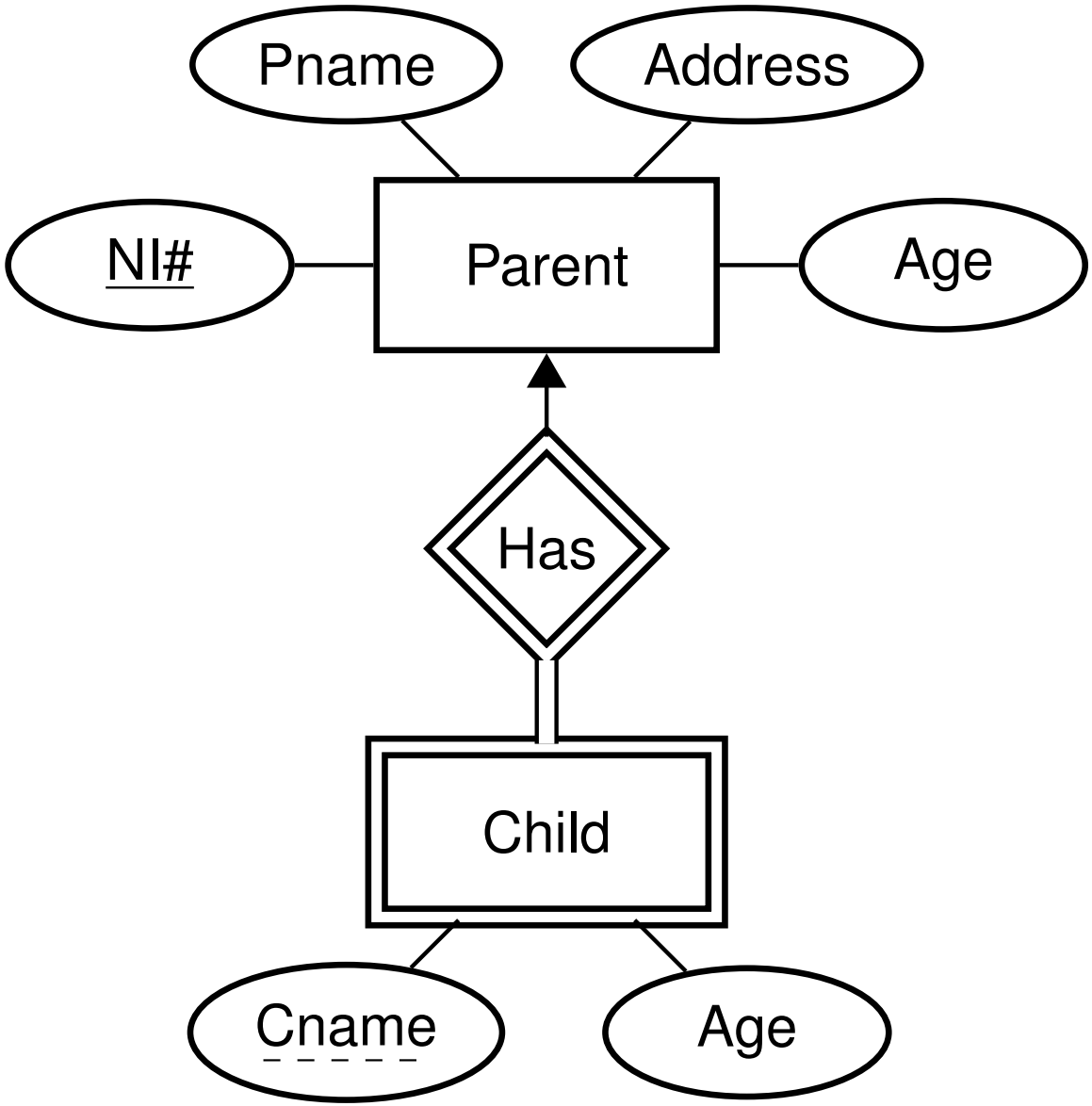
Each Supplier Supplies a Part at a certain Price.

Weak Entity Types

Definition. A Weak Entity Type is an entity type that does not have sufficient attributes to form a primary key.

- The existence of a weak entity depends on the existence of an *identifying* or *owner* entity type.
- The relationship between them is called an *identifying* (ID) relationship.
- The identifying relationship type is always many-to-one from the weak entity type to the identifying entity type.
- The weak entity type must have a *discriminator* (one or more attributes) for distinguishing among its entities.
- For example, in an employees database, Child entities *exist* only if their corresponding Parent employee entity exists.

Weak Entity Type and ID Relationship Type



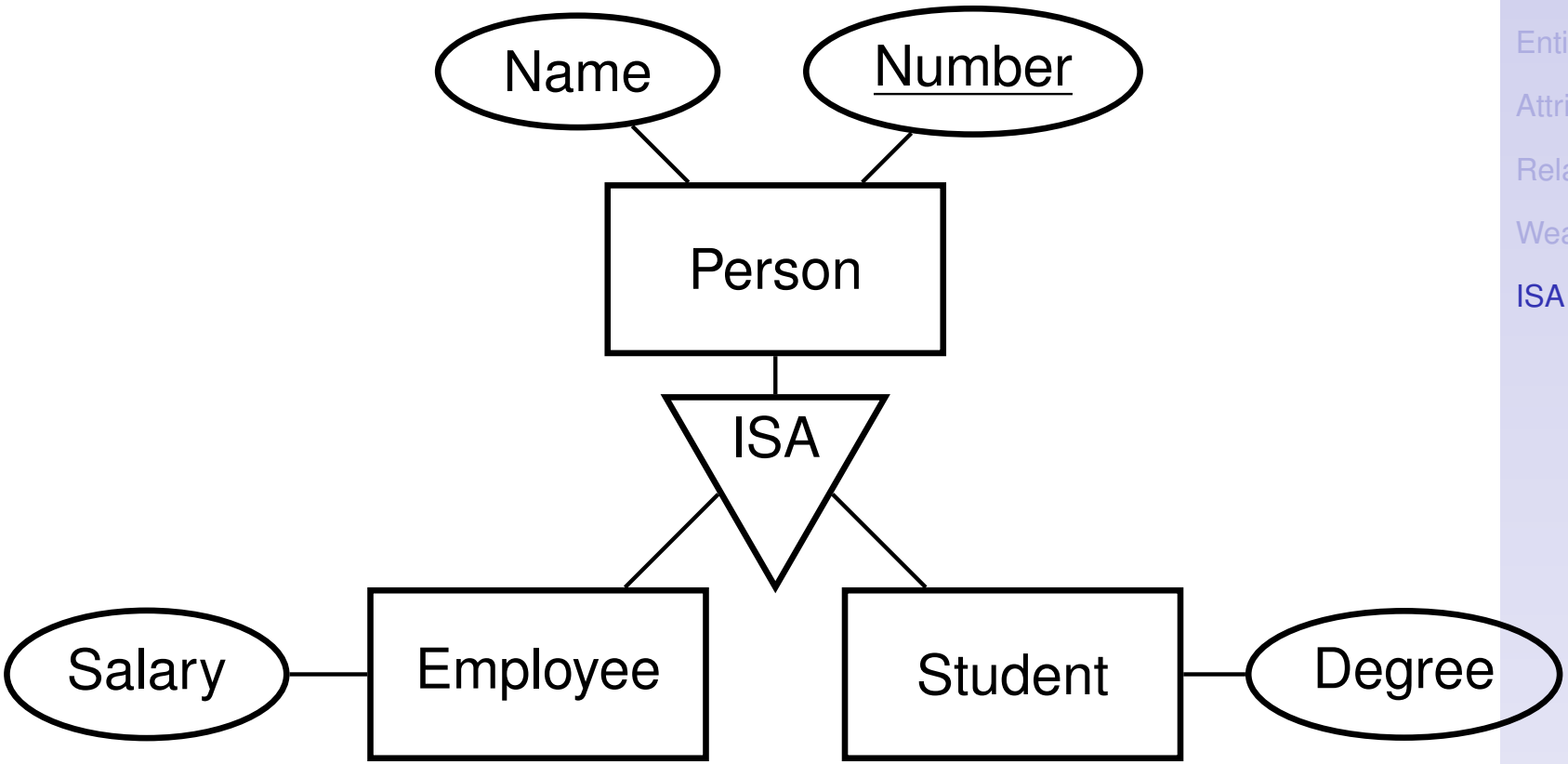
Weak Entity Types in an ERD

- A weak entity type is identified by a double rectangle.
- The discriminator is underlined by a dashed line.
- An identifying relationship is identified by a double diamond.
- The fact that the existence of the weak entity requires the existence of an owner entity is captured by the total participation of the weak entity type in the relationship (double line).
- The *primary key* of a weak entity type is the combination of the primary key of its owner type and its discriminator, e.g., (NI#, Cname) for Child.

ISA Relationship Types

- When an entity type contains certain entities that have special properties not shared by all entities, this suggests two entity types should be created with an ISA relationship type between them.
- Also known as generalisation/specialisation.
- E.g. An Employee ISA Person and a Student ISA Person (see next slide)
- If Employee ISA Person then Employee *inherits* all the attributes of Person.

ISA Relationships

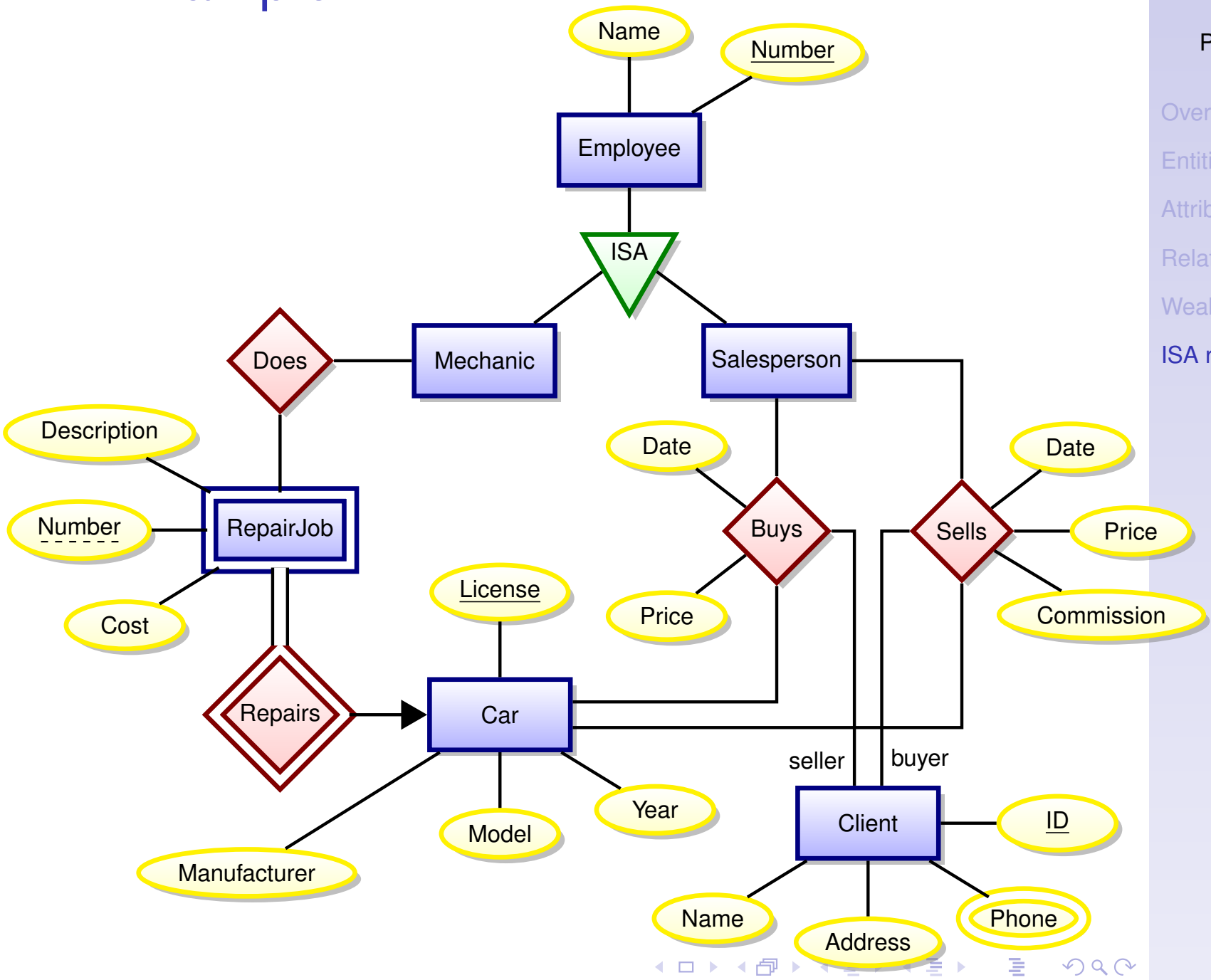


- Attributes of Employee: Name, Number and Salary.
- Attributes of Student: Name, Number and Degree.

Informal Method for Constructing an ERD

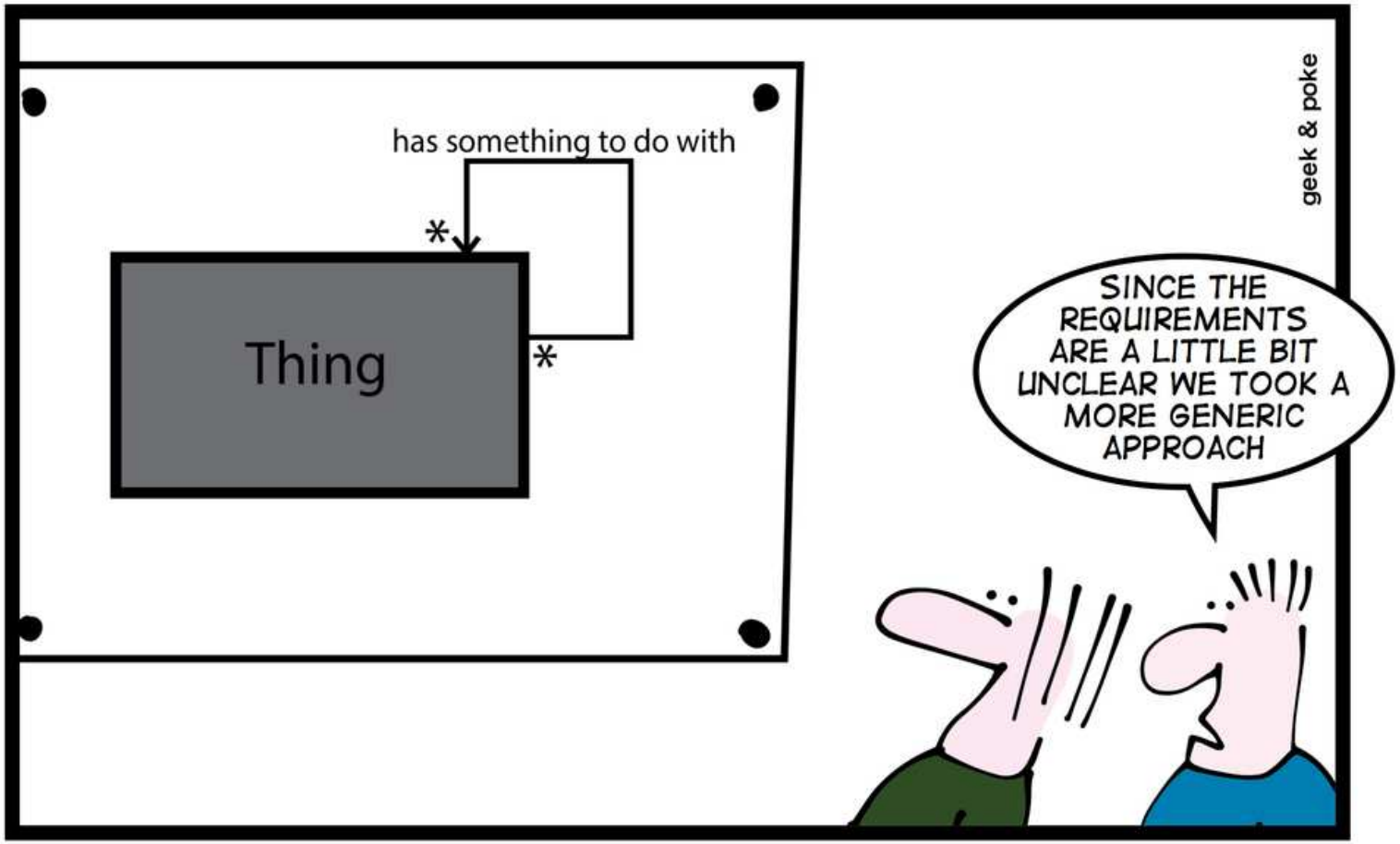
1. Identify the entity types (including weak entity types) of the application.
2. Identify the relationship (including ISA and ID) types.
3. Classify each relationship type identified in step 2 according to its multiplicity, i.e. if it is a one-to-one, many-to-one or many-to-many.
4. Determine the participation constraints for each entity type in each relationship type.
5. Draw an ERD with the entity types and the relationship types between them.
6. Identify the attributes of entity and relationship types and their underlying domains
7. Identify a primary key for each entity type.
8. Add the attributes and primary keys to the ERD drawn in step 5.

ERD Example


Name
Number
Employee
ISA
Does
Mechanic
Salesperson
Description
Date
Date
Number
RepairJob
Buys
Sells
Price
Cost
License
Price
Commission
Repairs
Car
seller
buyer
Year
Model
Client
ID
Manufacturer
Name
Address
Phone

Generic model

- And then there's always the generic approach:



For more information

See, e.g.,

- Chapter 12 of Connolly and Begg.
- Chapter 7 of Silberschatz et al.
- Chapter 4 of Ullman and Widom.