Database Design Using The Entity-Relationship Model

Outline

1. Database Design
2. ER Basics: Entities & Relations
3. ER Design considerations
4. Advanced ER Concepts

Design Phases

- Initial phase -- characterize fully the data needs of the prospective database users.

- Second phase -- choosing a data model
  - Applying the concepts of the chosen data model
  - Translating these requirements into a conceptual schema of the database.
  - A fully developed conceptual schema indicates the functional requirements of the enterprise.
    - Describe the kinds of operations (or transactions) that will be performed on the data.

Design Phases (Cont.)

- Final Phase -- Moving from an abstract data model to the implementation of the database
  - Logical Design – Deciding on the database schema.
    - Database design requires that we find a "good" collection of relation schemas.
    - Business decision – What attributes should we record in the database?
    - Computer Science decision – What relation schemas should we have and how should the attributes be distributed among the various relation schemas?
  - Physical Design – Deciding on the physical layout of the database

Design Alternatives

- In designing a database schema, we must ensure that we avoid two major pitfalls:
  - **Redundancy**: a bad design may result in repeat information.
    - Redundant representation of information may lead to data inconsistency among the various copies of information
  - **Incompleteness**: a bad design may make certain aspects of the enterprise difficult or impossible to model.

- Avoiding bad designs is not enough. There may be a large number of good designs from which we must choose.

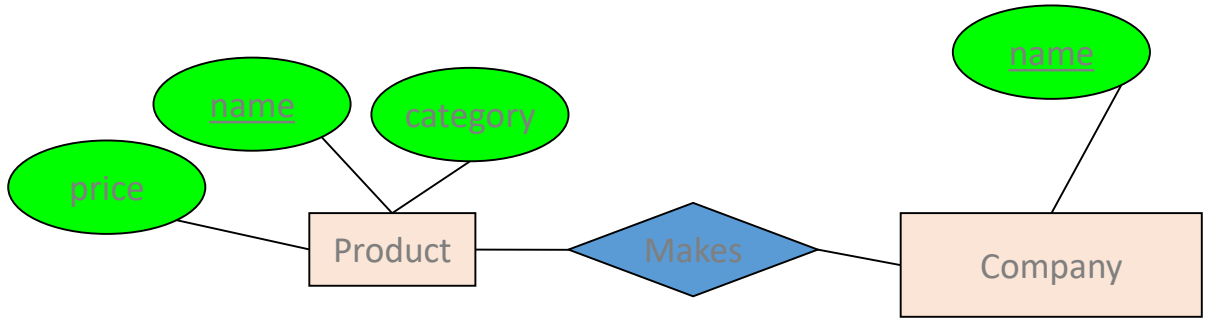
Design Approaches

- Entity Relationship (ER) Model
  - Models an enterprise as a collection of *entities* and *relationships*
    - Entity: a “thing” or “object” in the enterprise that is distinguishable from other objects
      - Described by a set of *attributes*
    - Relationship: an association among several entities
  - Represented diagrammatically by an *entity-relationship diagram:*
- Normalization Theory (will be discussed in BBM471)
  - Formalize what designs are bad, and test for them

Database Design Process

1. Requirements Analysis → 2. Conceptual Design → 3. Logical, Physical, Security, etc.

ER Model & Diagrams used



This process is iterated **many** times

ER is a *visual syntax* for DB design which is ***precise enough*** for technical points, but ***abstracted enough*** for non-technical people

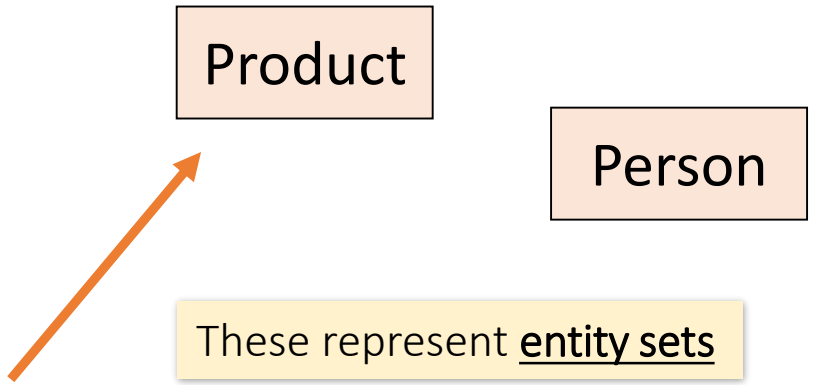
Interlude: Impact of the ER model

- The ER model is one of the most cited articles in Computer Science
  - *"The Entity-Relationship model – toward a unified view of data"* Peter Chen, 1976

- Used by companies big and small
  - You'll know it soon enough

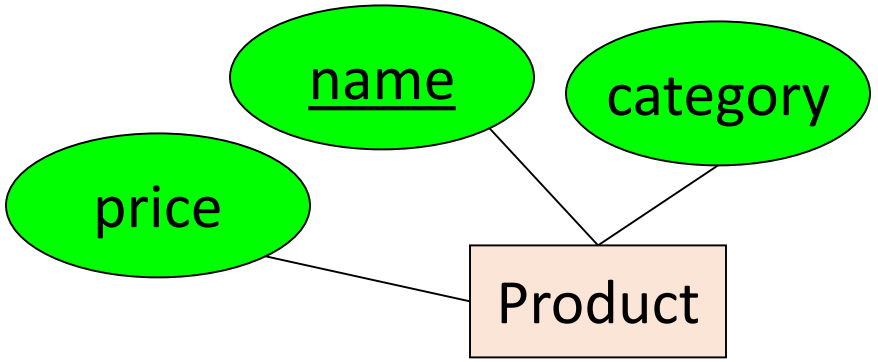
Entities and Entity Sets

- **Entities** & **entity sets** are the primitive unit of the ER model
  - Entities are the individual objects, which are members of entity sets
    - Ex: A specific person or product
  - Entity sets are the *classes* or *types* of objects in our model
    - Ex: Person, Product
    - *These are what is shown in E/R diagrams - as rectangles*
    - *Entity sets represent the sets of all possible entities*



Entities and Entity Sets

- An entity set has **attributes**
  - Represented by ovals attached to an entity set

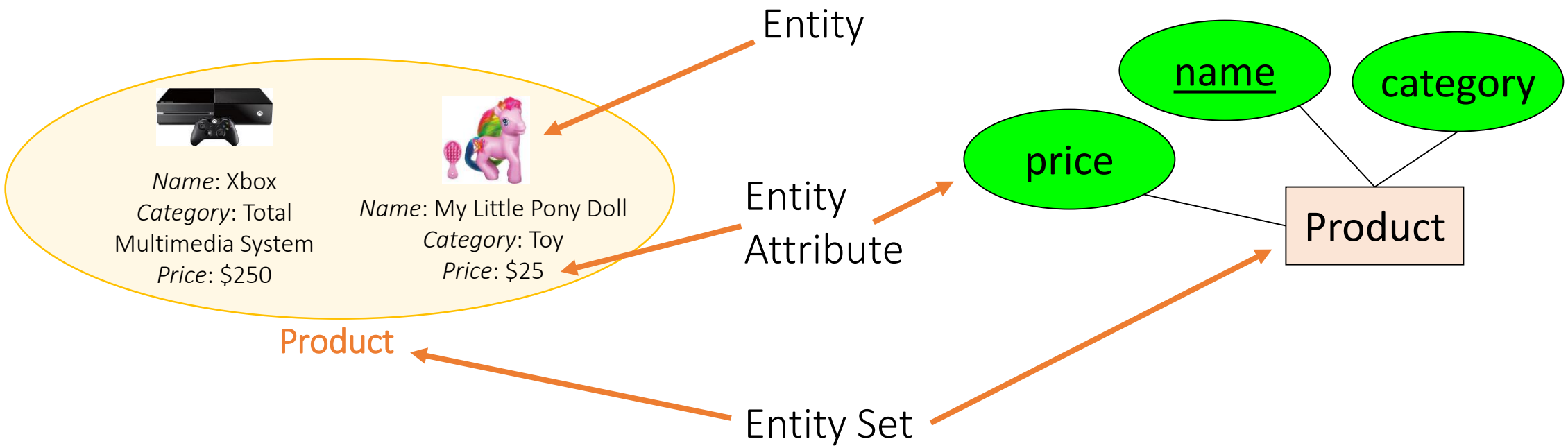


Shapes **are** important.
Colors **are not**.

Entities vs. Entity Sets

Example:

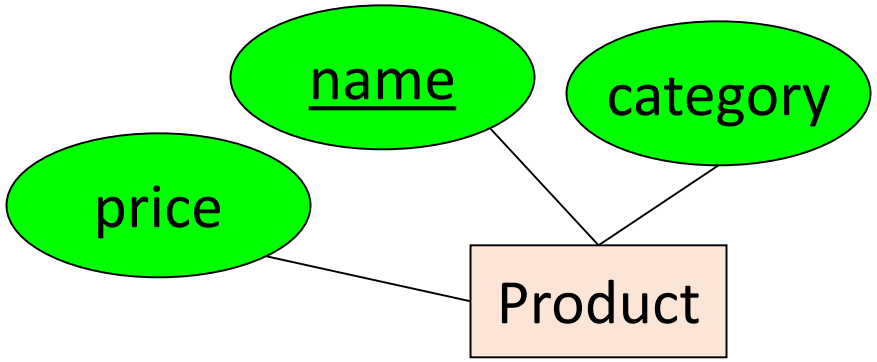
Entities are **not** explicitly represented in ER diagrams!



Keys

- A *key* is a **minimal** set of attributes that uniquely identifies an entity.

Denote elements of the primary key by underlining.



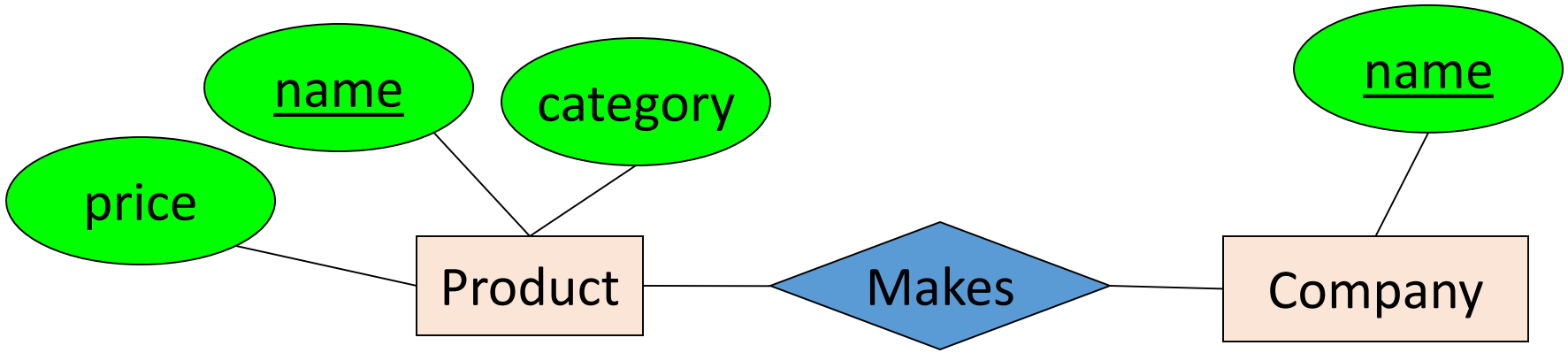
Here, {price, category} is **not** a key.

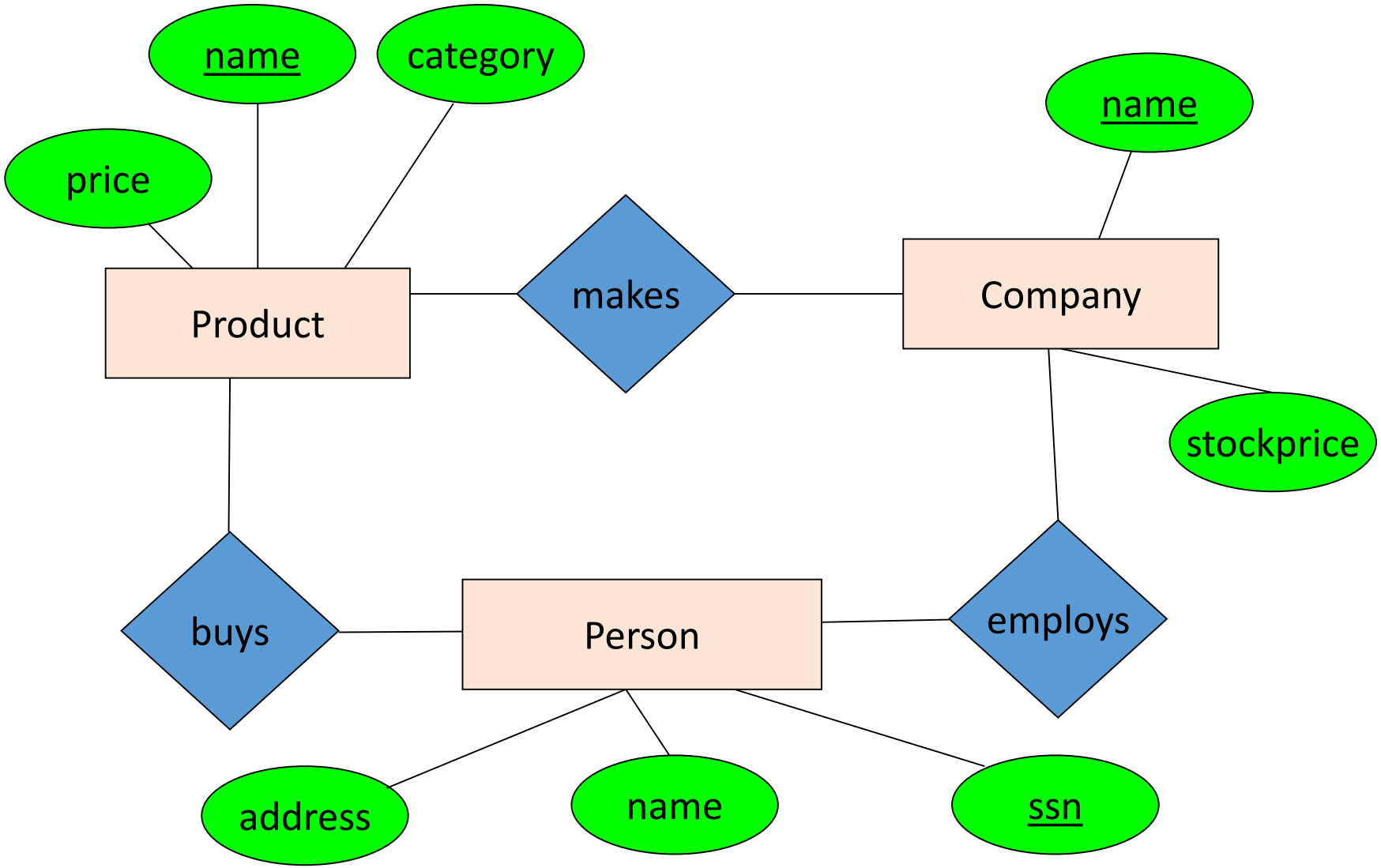
If it were, what would it mean?

The ER model forces us to designate a single **primary** key, though there may be multiple candidate keys

The R in ER: **Relationships**

- A **relationship** is between two entities

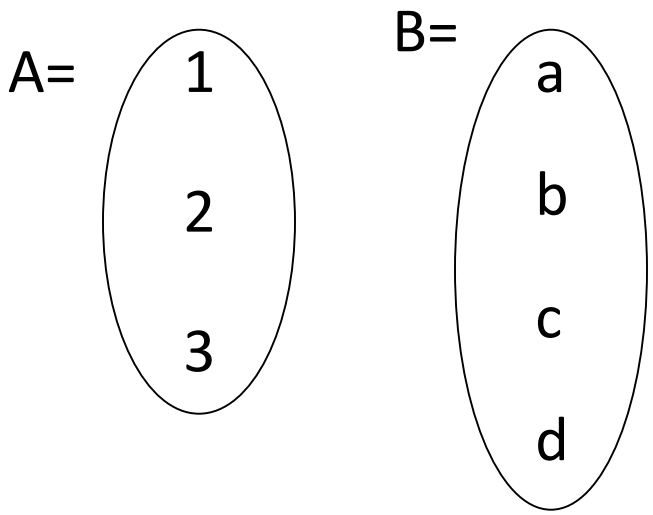




Company makes one product, employs one person.
Person buys one product.

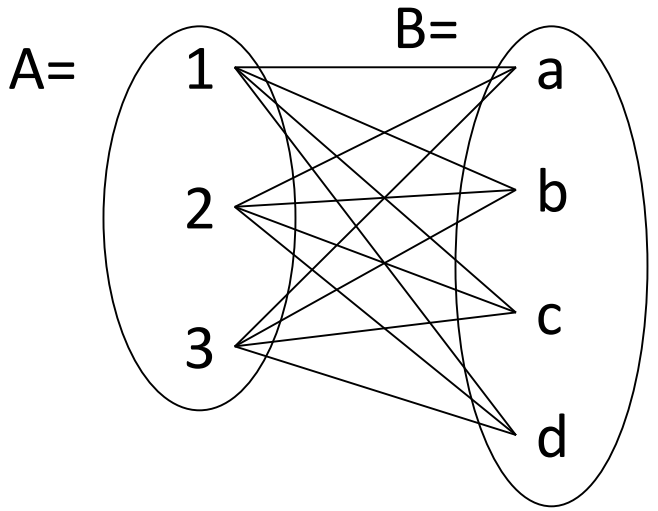
What is a Relationship?

- ***A mathematical definition:***
  - Let A, B be sets
    - *A={1,2,3}, B={a,b,c,d}*



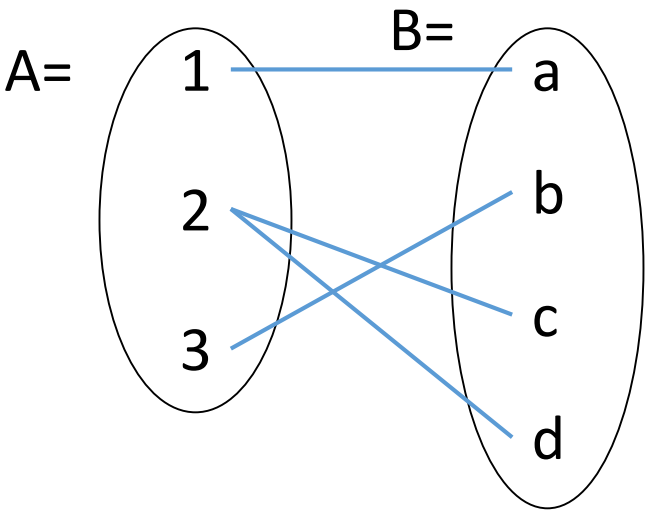
What is a Relationship?

- ***A mathematical definition:***
  - Let A, B be sets
    - *A={1,2,3}, B={a,b,c,d}*
  - A x B (the ***cross-product***) is the set of all pairs (a,b)
    - *$A \times B$ = {(1,a), (1,b), (1,c), (1,d), (2,a), (2,b), (2,c), (2,d), (3,a), (3,b), (3,c), (3,d)}*



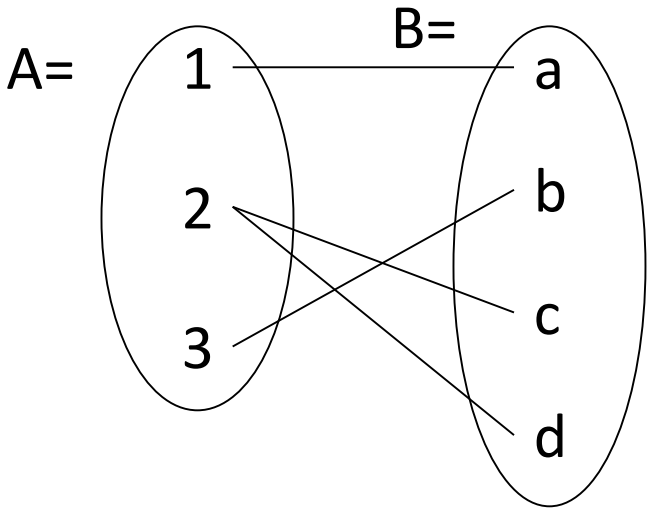
What is a Relationship?

- ***A mathematical definition:***
  - Let A, B be sets
    - *A={1,2,3}, B={a,b,c,d},*
  - A x B (the ***cross-product***) is the set of all pairs (a,b)
    - *$A \times B$ = {(1,a), (1,b), (1,c), (1,d), (2,a), (2,b), (2,c), (2,d), (3,a), (3,b), (3,c), (3,d)}*
  - **We define a <u>relationship</u> to be a subset of A x B**
    - *R = {(1,a), (2,c), (2,d), (3,b)}*



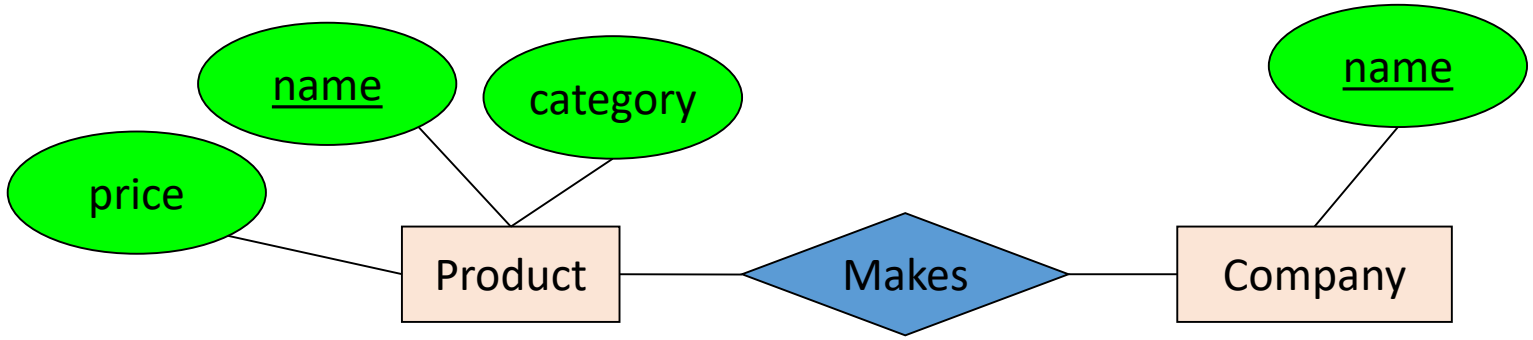
What is a Relationship?

- ***A mathematical definition:***
  - Let A, B be sets
  - A x B (the ***cross-product***) is the set of all pairs
  - A relationship is a subset of A x B



- **Makes** is relationship- it is a ***subset*** of **Product** × **Company**:

What is a Relationship?



A **relationship** between **entity sets P and C** is a ***subset of all possible pairs of entities in P and C***, with tuples uniquely identified by ***P and C's keys***

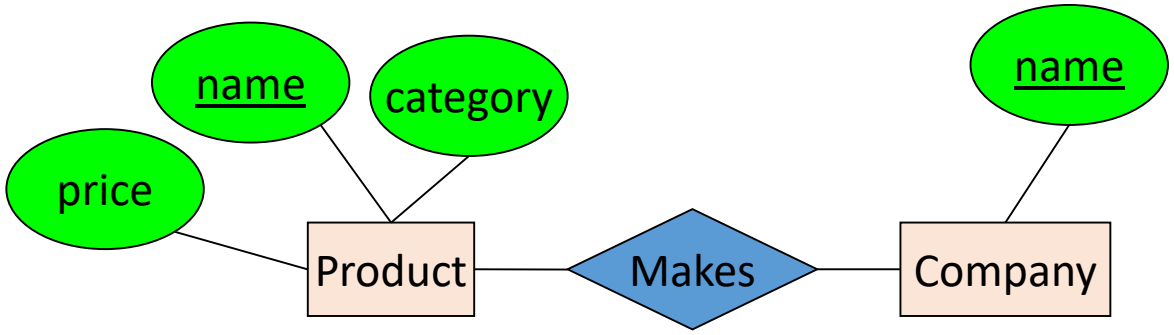
What is a Relationship?

Company

name
GizmoWorks
GadgetCorp

Product

name	category	price
Gizmo	Electronics	$9.99
GizmoLite	Electronics	$7.50
Gadget	Toys	$5.50



A **relationship** between **entity sets P and C** is a ***subset of all possible pairs of entities in P and C***, with tuples uniquely identified by ***P and C's keys***

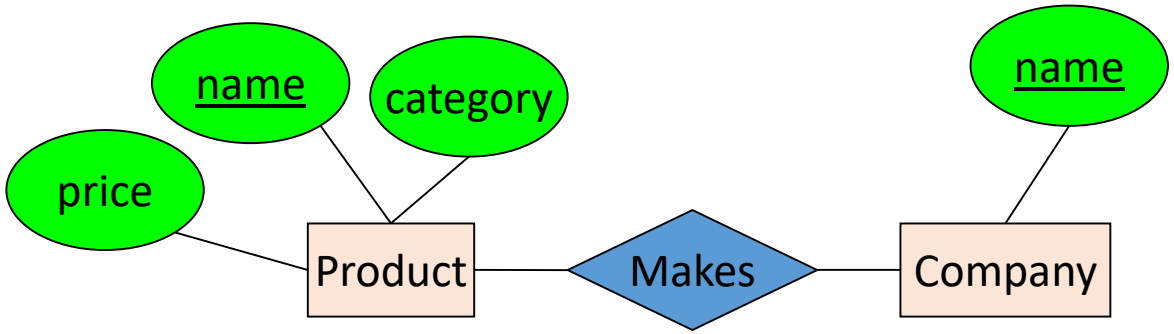
What is a Relationship?

Company

name
GizmoWorks
GadgetCorp

Product

name	category	price
Gizmo	Electronics	$9.99
GizmoLite	Electronics	$7.50
Gadget	Toys	$5.50



Company C × Product P

C.name	P.name	P.category	P.price
GizmoWorks	Gizmo	Electronics	$9.99
GizmoWorks	GizmoLite	Electronics	$7.50
GizmoWorks	Gadget	Toys	$5.50
GadgetCorp	Gizmo	Electronics	$9.99
GadgetCorp	GizmoLite	Electronics	$7.50
GadgetCorp	Gadget	Toys	$5.50

A **relationship** between **entity sets P and C** is a ***subset of all possible pairs of entities in P and C***, with tuples uniquely identified by ***P and C's keys***

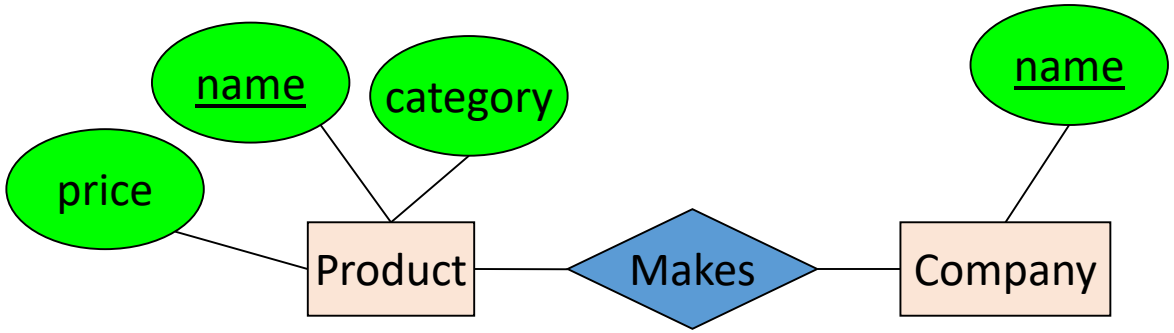
What is a Relationship?

Company

name
GizmoWorks
GadgetCorp

Product

name	category	price
Gizmo	Electronics	$9.99
GizmoLite	Electronics	$7.50
Gadget	Toys	$5.50



Company C × Product P

C.name	P.name	P.category	P.price
GizmoWorks	Gizmo	Electronics	$9.99
GizmoWorks	GizmoLite	Electronics	$7.50
GizmoWorks	Gadget	Toys	$5.50
GadgetCorp	Gizmo	Electronics	$9.99
GadgetCorp	GizmoLite	Electronics	$7.50
GadgetCorp	Gadget	Toys	$5.50

Makes

C.name	P.name
GizmoWorks	Gizmo
GizmoWorks	GizmoLite
GadgetCorp	Gadget

A relationship between **entity sets P and C** is a ***subset of all possible pairs of entities in P and C***, with tuples uniquely identified by ***P and C's keys***

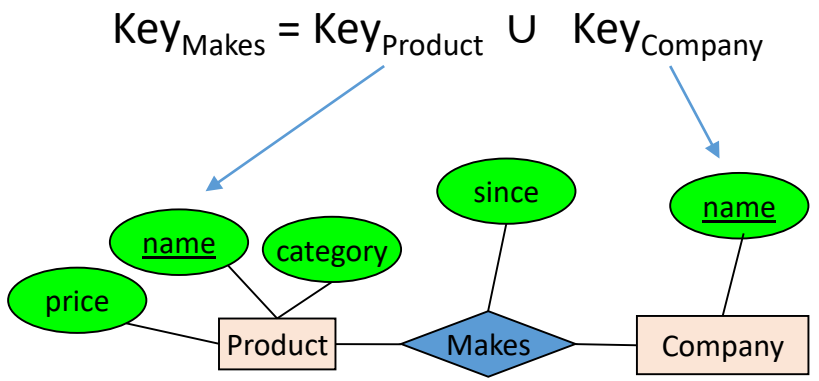
What is a Relationship?

- There can only be **one relationship for every unique combination of entities**

This follows from our mathematical definition of a relationship- it's a SET!

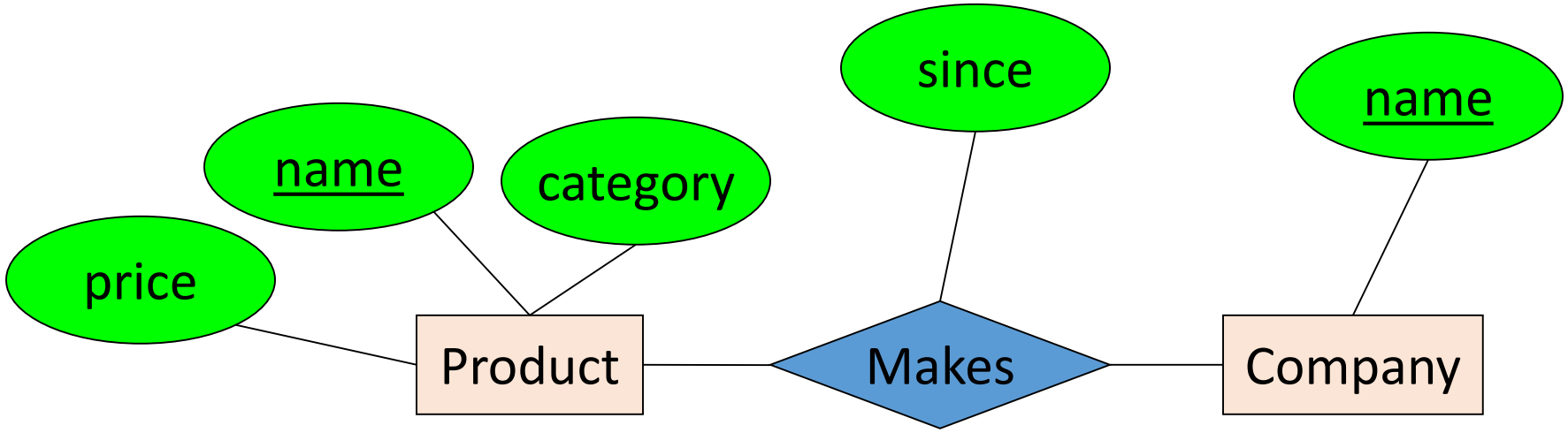
- This also means that **the relationship is uniquely determined by the keys of its entities**

- *Example: the "key" for Makes (to right) is {Product.name, Company.name}*



Relationships and Attributes

- Relationships may have attributes as well.



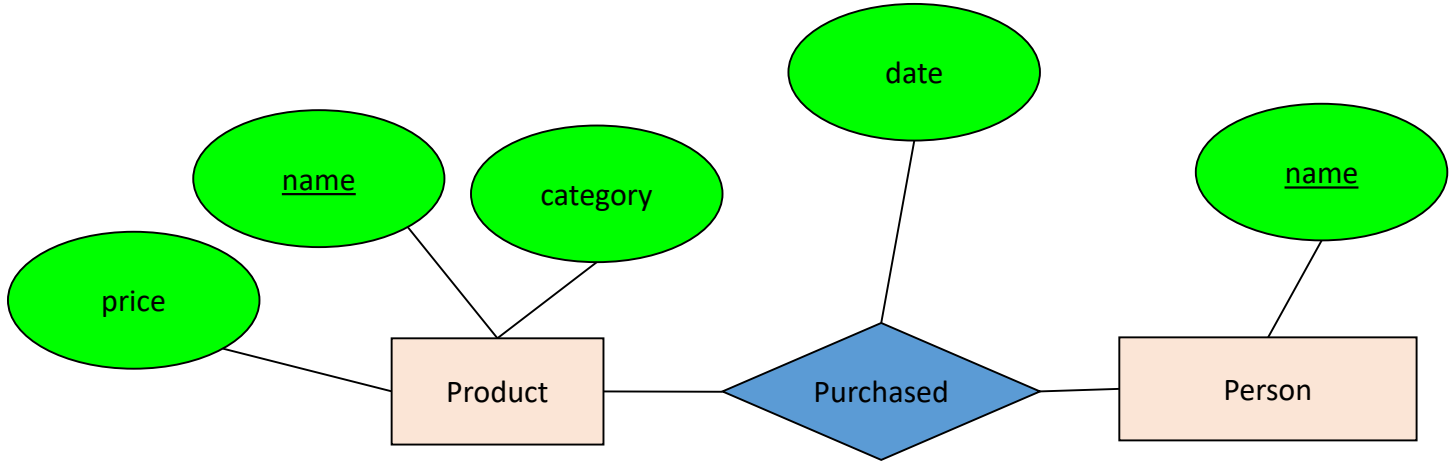
For example: "since" records when company started making a product

Note: *"since"* is implicitly unique per pair here! Why?

Note #2: Why not "how long"?

Decision: Relationship vs. Entity?

- **Q:** What does this say?

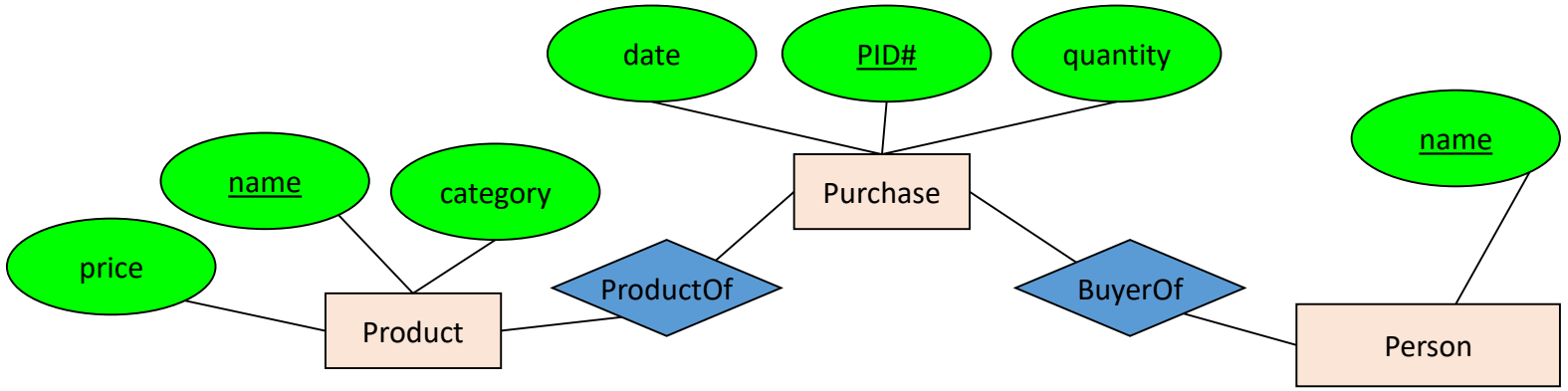


- **A:** A person can only buy a specific product once (on one date)

Modeling something as a relationship makes it unique; what if not appropriate?

Decision: Relationship vs. Entity?

- What about this way?



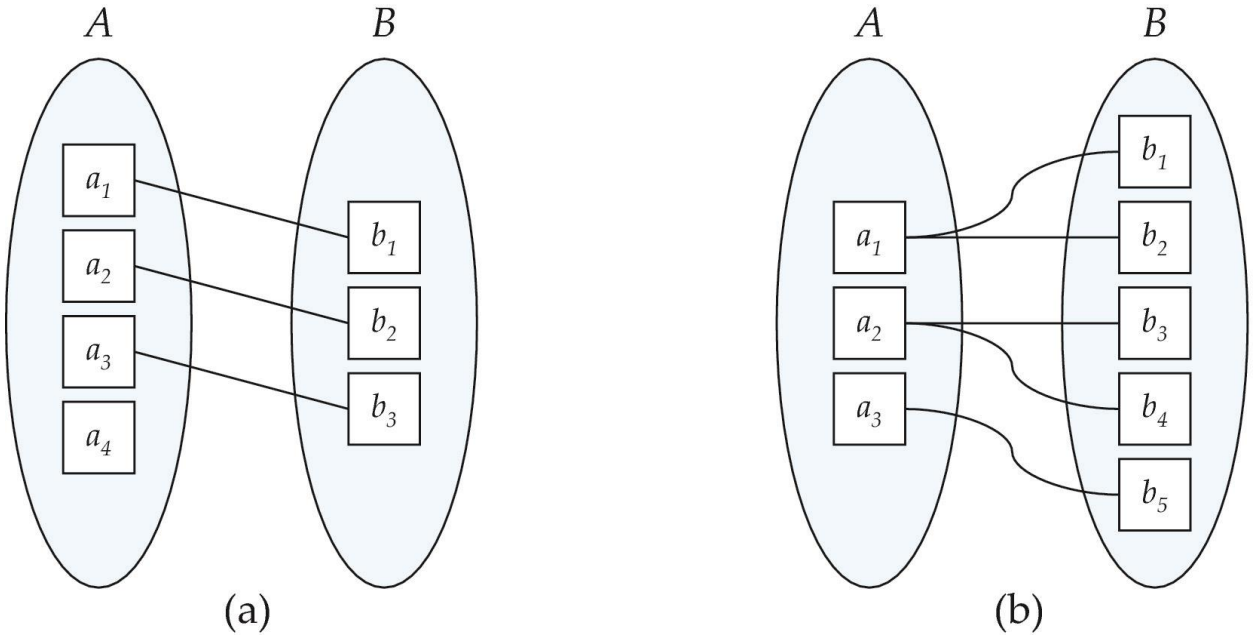
- *Now we can have multiple purchases per product, person pair!*

We can always use **a new entity** instead of a relationship. For example, to permit multiple instances of each entity combination!

Mapping Cardinality Constraints

- Express the number of entities to which another entity can be associated via a relationship set.
- Most useful in describing binary relationship sets.
- For a binary relationship set the mapping cardinality must be one of the following types:
  - One to one
  - One to many
  - Many to one
  - Many to many

Mapping Cardinalities

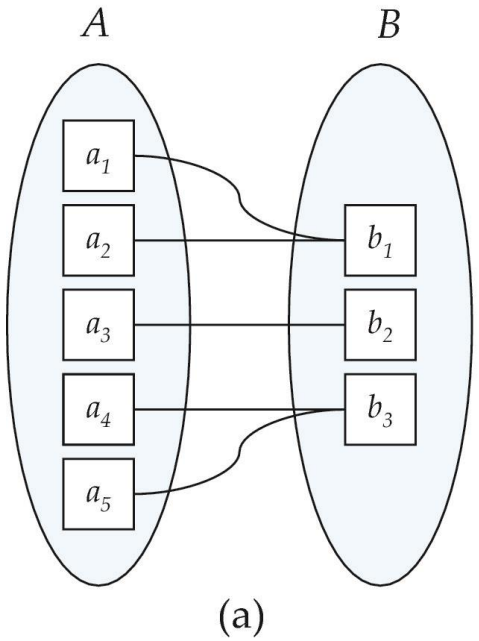


One to one

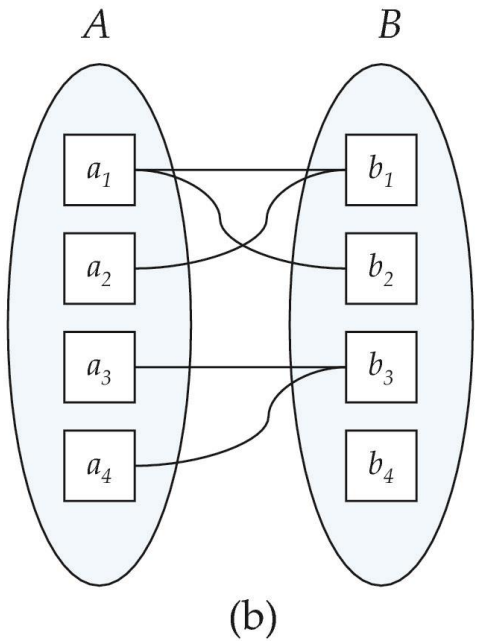
One to many

Note: Some elements in *A* and *B* may not be mapped to any elements in the other set

Mapping Cardinalities



Many to one



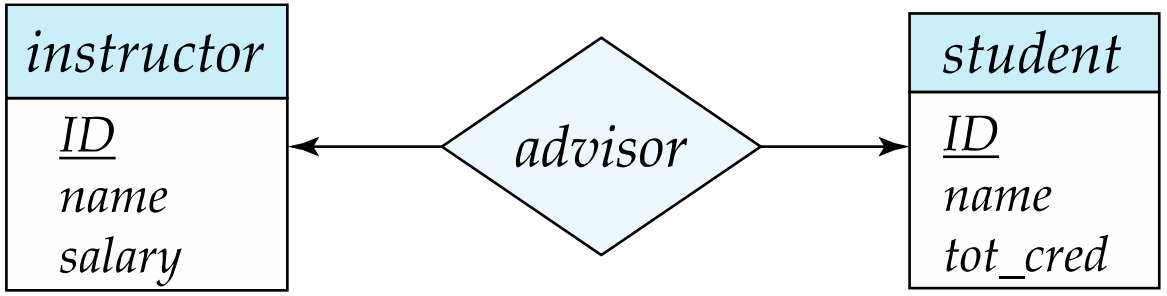
Many to many

Note: Some elements in A and B may not be mapped to any elements in the other set

Representing Cardinality Constraints in ER Diagram

- We express cardinality constraints by drawing either a directed line (→), signifying “one,” or an undirected line (—), signifying “many,” between the relationship set and the entity set.

- One-to-one relationship between an *instructor* and a *student* :
  - A student is associated with at most one *instructor* via the relationship *advisor*
  - A *student* is associated with at most one *department* via *stud_dept*



One-to-Many Relationship

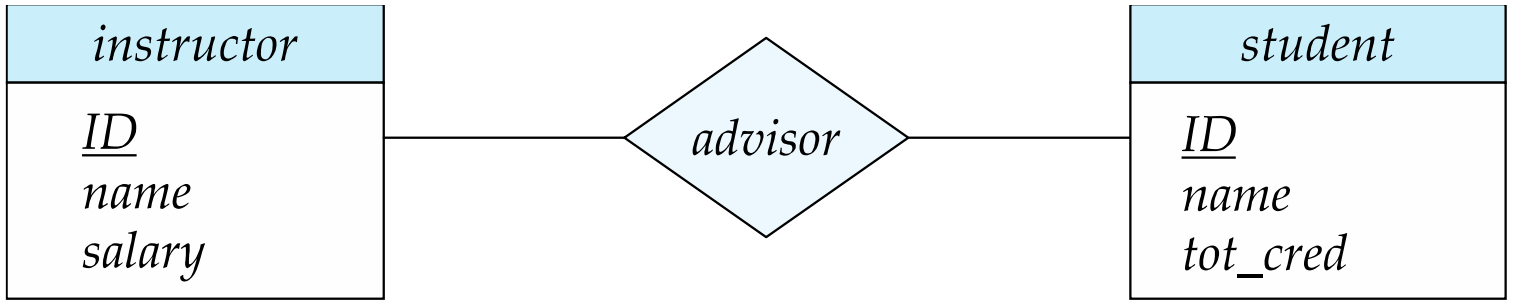
- one-to-many relationship between an *instructor* and a *student*
  - an instructor is associated with several (including 0) students via *advisor*
  - a student is associated with at most one instructor via advisor,

Many-to-One Relationships

- In a many-to-one relationship between an *instructor* and a *student,*
  - an instructor is associated with at most one student via *advisor*,
  - and a student is associated with several (including 0) instructors via *advisor*

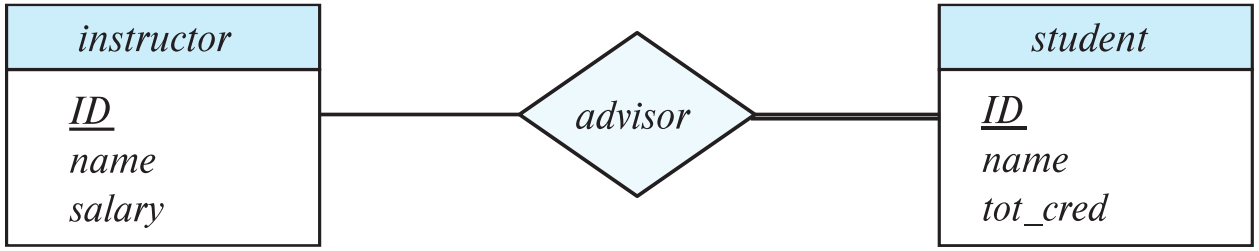
Many-to-Many Relationship

- An instructor is associated with several (possibly 0) students via *advisor*
- A student is associated with several (possibly 0) instructors via *advisor*



Total and Partial Participation

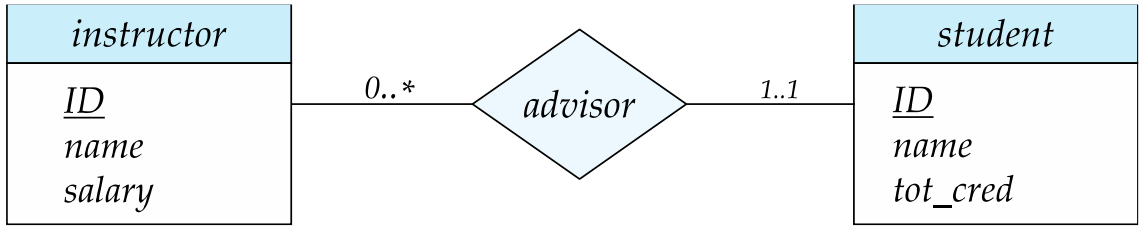
- **Total participation** (indicated by double/bold line): every entity in the entity set participates in at least one relationship in the relationship set



  - participation of *student* in *advisor* relation is total
    - every *student* must have an associated instructor
- **Partial participation**: some entities may not participate in any relationship in the relationship set
  - Example: participation of *instructor* in *advisor* is partial

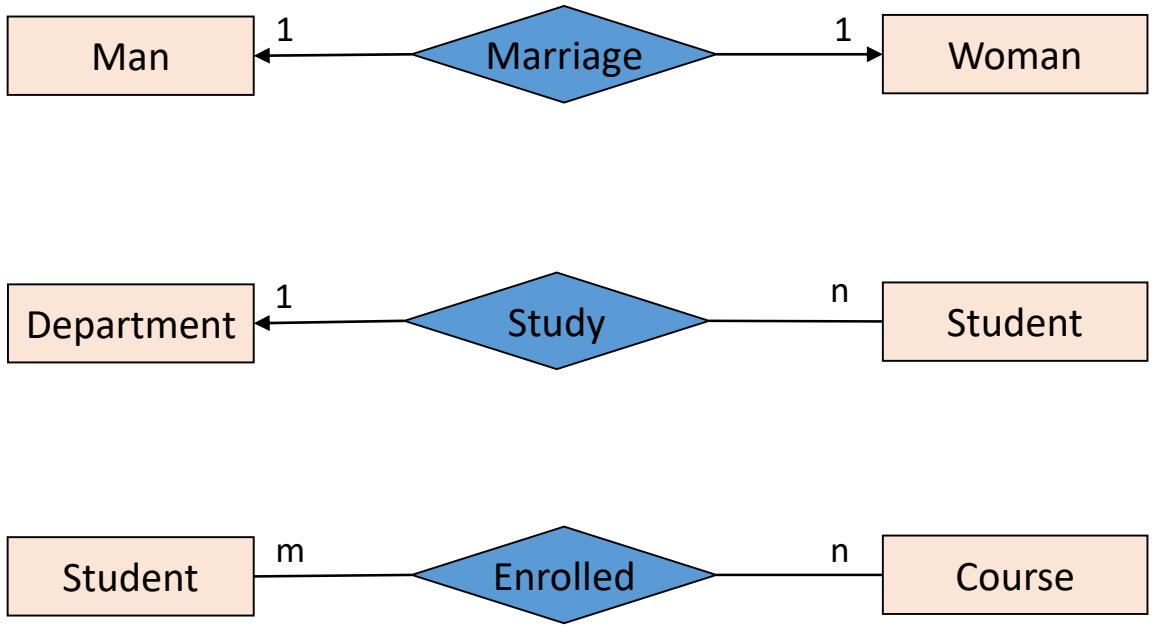
Notation for Expressing More Complex Constraints

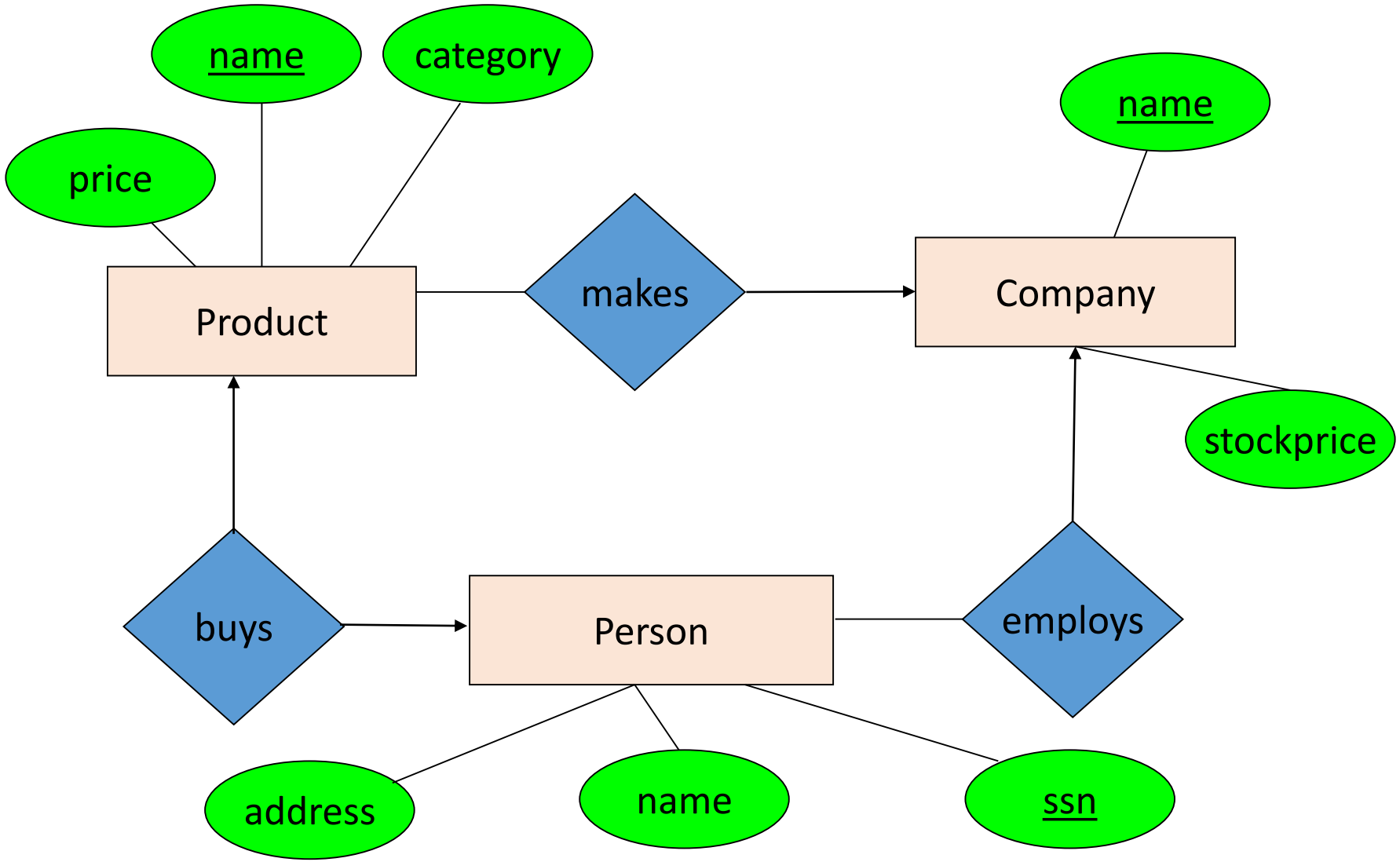
- A line may have an associated minimum and maximum cardinality, shown in the form *l..h*, where *l* is the minimum and *h* the maximum cardinality
  - A minimum value of 1 indicates total participation.
  - A maximum value of 1 indicates that the entity participates in at most one relationship
  - A maximum value of * indicates no limit.
- Example



  - Instructor can advise 0 or more students. A student must have 1 advisor; cannot have multiple advisors

Multiplicity of ER Relationships

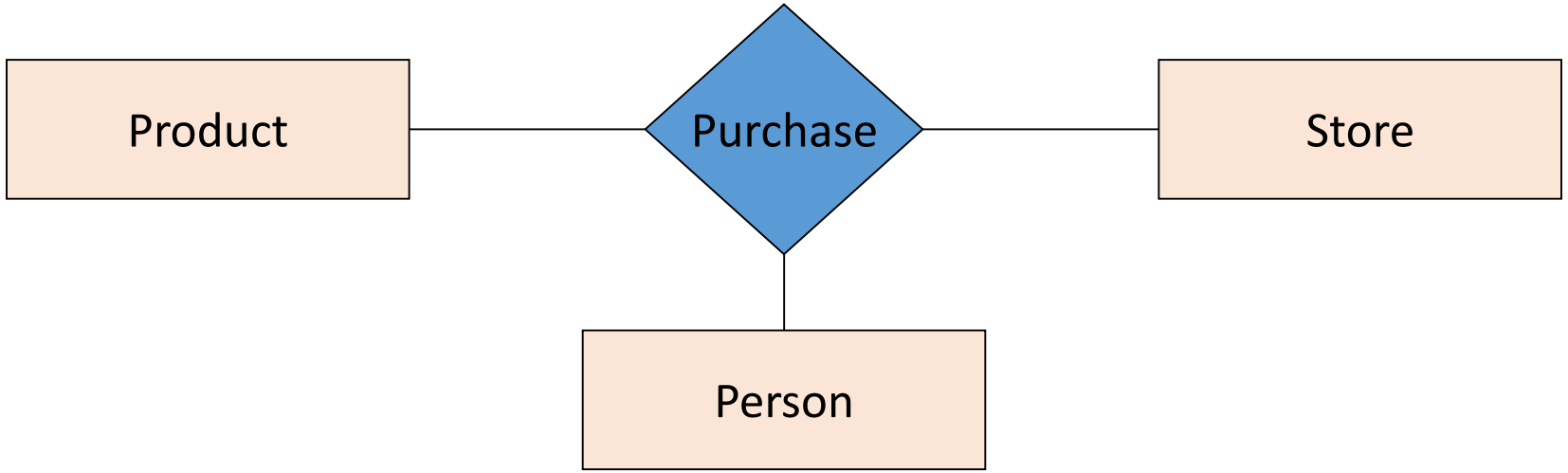




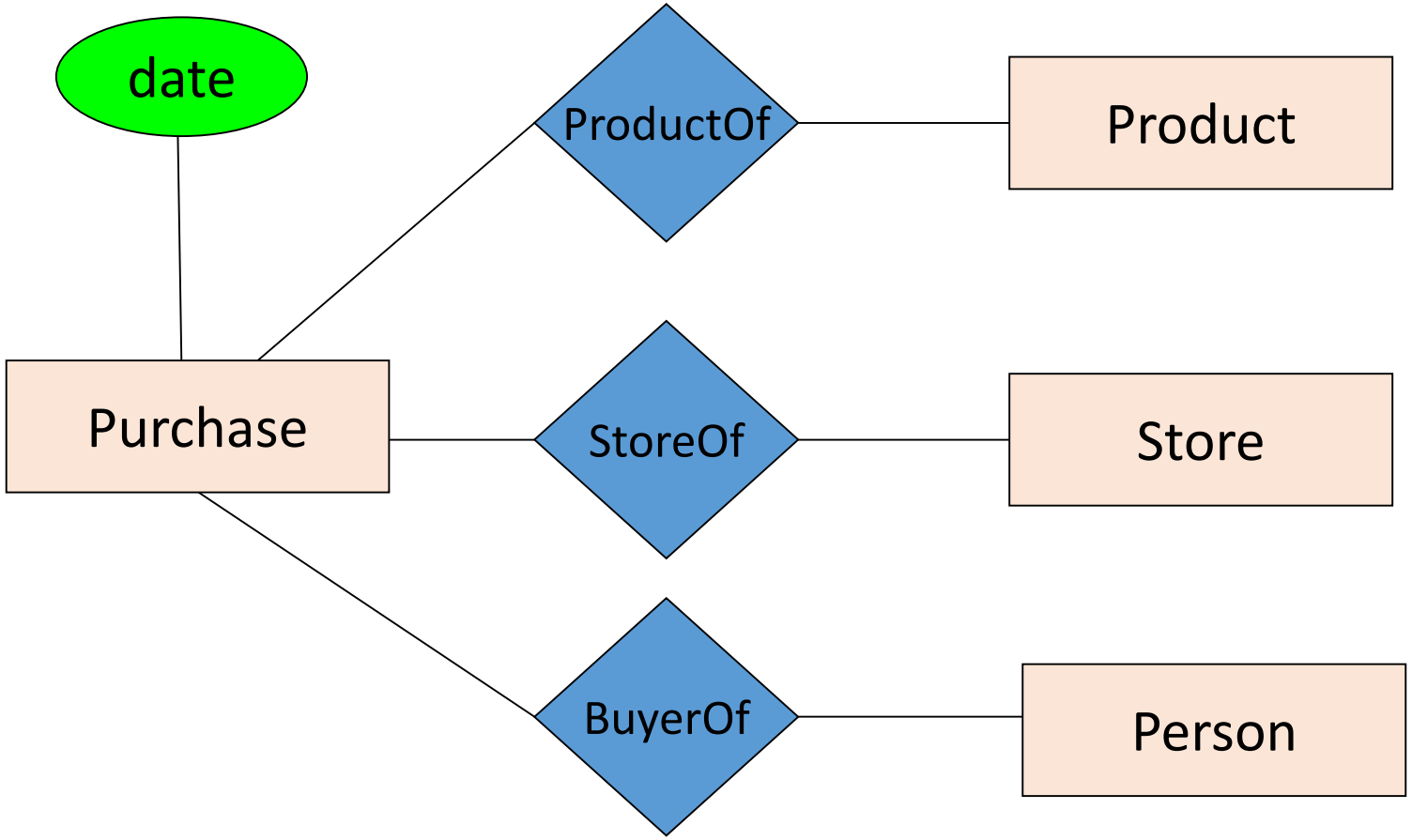
Company can make many product, can employ many person.
Person buys still one product.

Multi-way Relationships

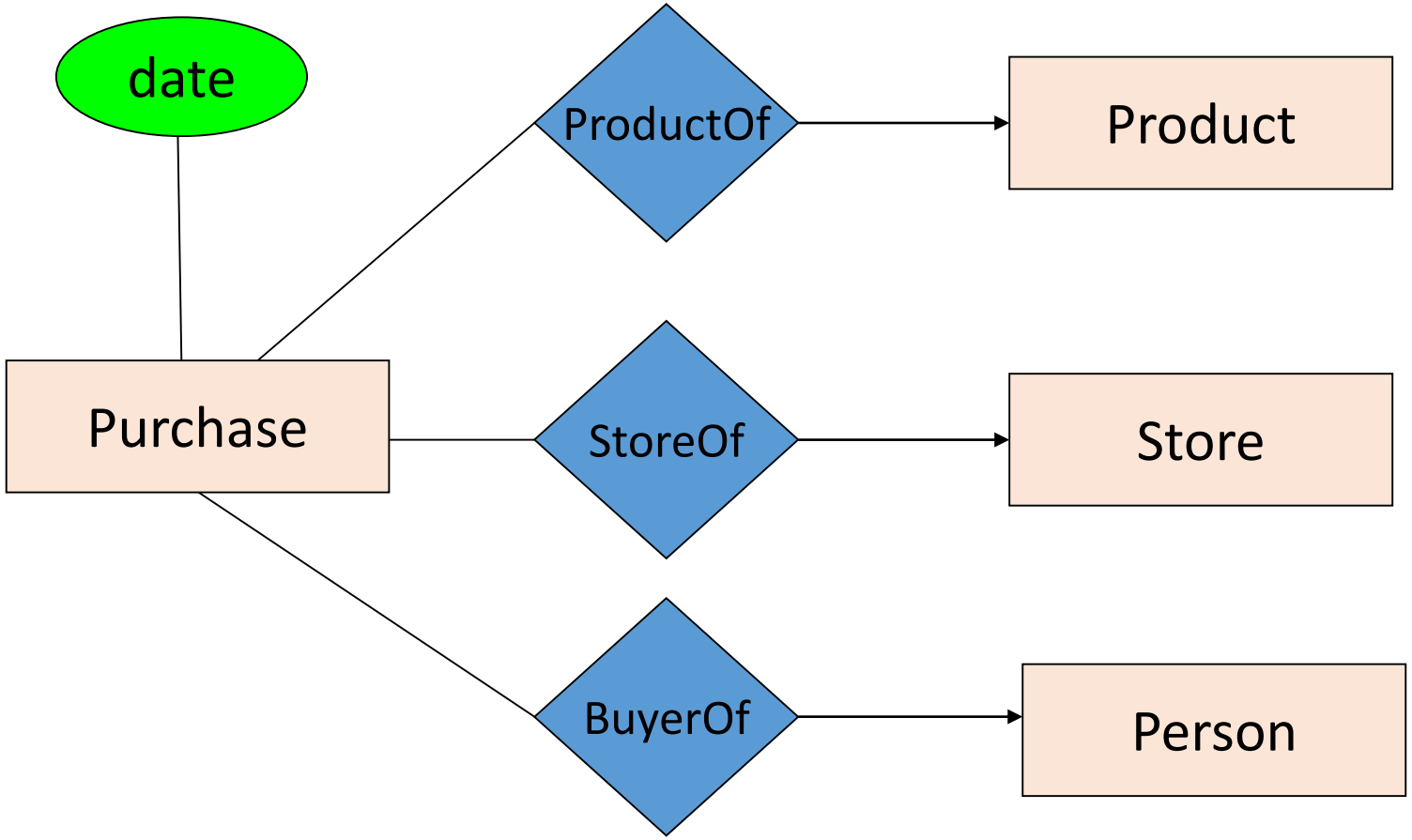
How do we model a purchase relationship between buyers, products and stores?



Converting Multi-way Relationships to Binary

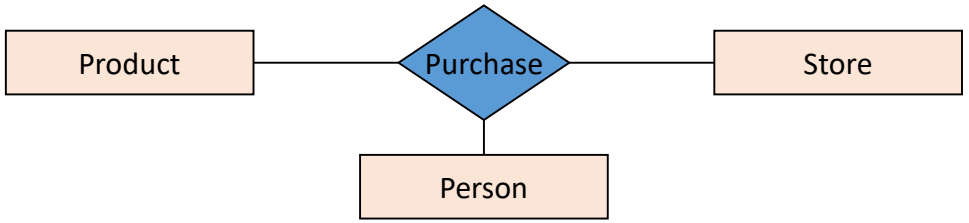


Converting Multi-way Relationships to New Entity + Binary Relationships

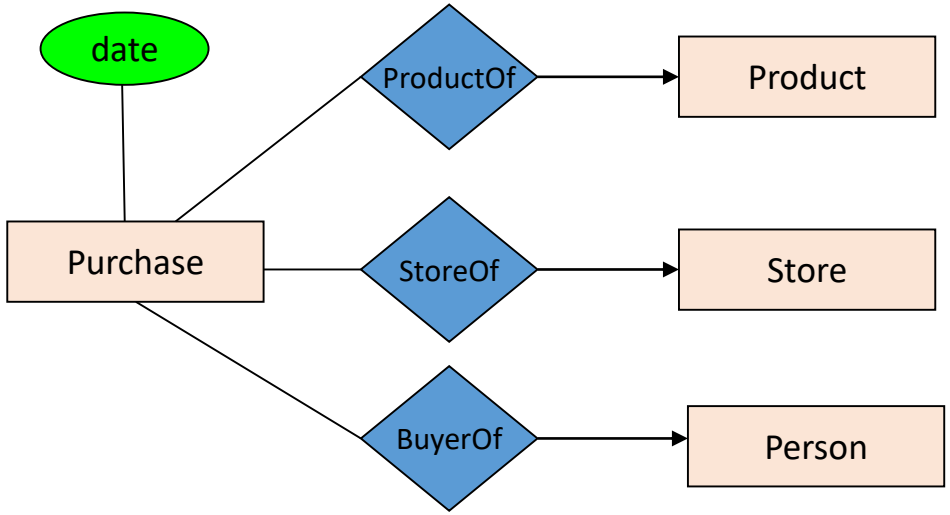


Decision: Multi-way or New Entity + Binary?

(A) Multi-way Relationship



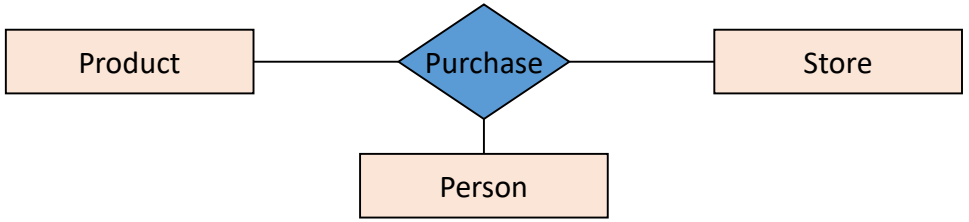
(B) Entity + Binary



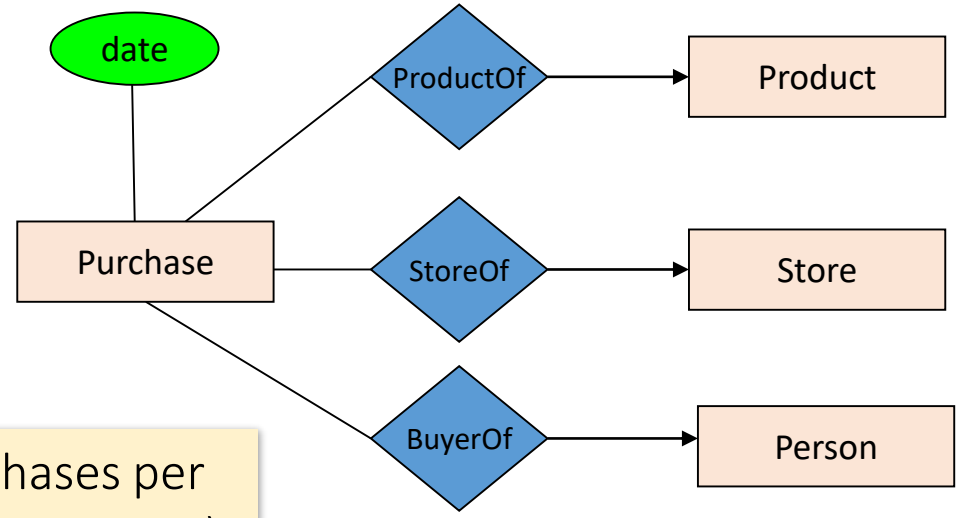
Should we use a single **multi-way relationship** or a ***new entity with binary relations?***

Decision: Multi-way or New Entity + Binary?

(A) Multi-way Relationship



(B) Entity + Binary

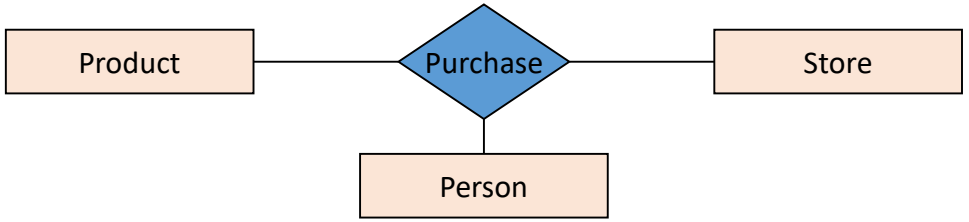


Multiple purchases per (product, store, person) combo possible here!

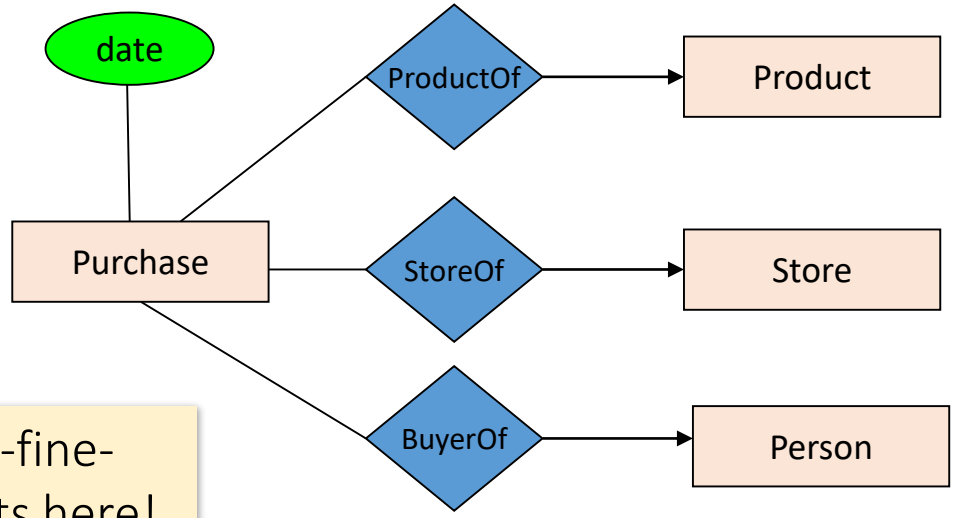
(B) is useful if we want to have multiple instances of the “relationship” per entity combination

Decision: Multi-way or New Entity + Binary?

(A) Multi-way Relationship



(B) Entity + Binary



We can add more-fine-grained constraints here!

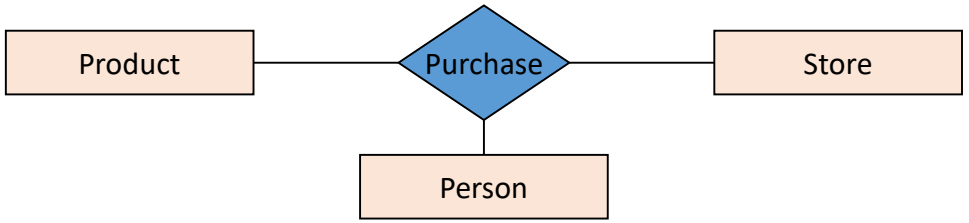
(B) is also useful when we want to add details (constraints or attributes) to the relationship

"A person who shops in only one store"

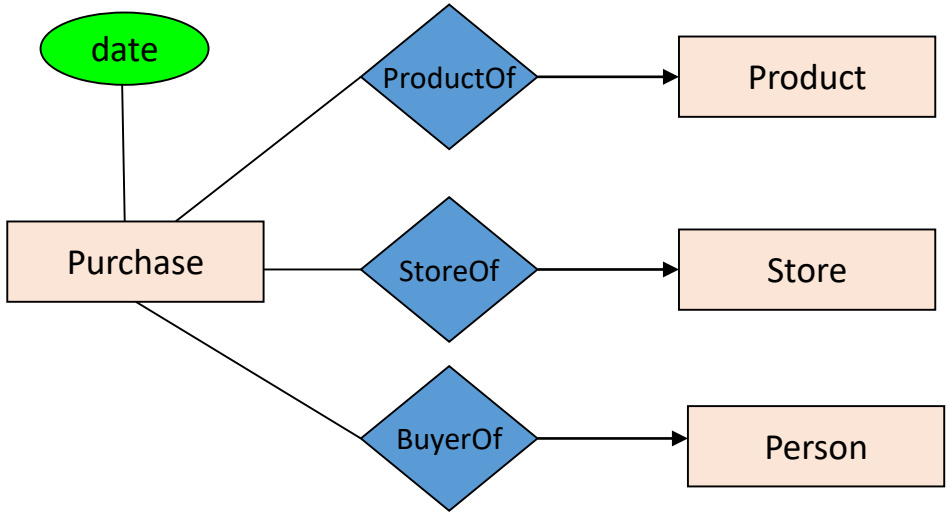
"How long a person has been shopping at a store"

Decision: Multi-way or New Entity + Binary?

(A) Multi-way Relationship



(B) Entity + Binary

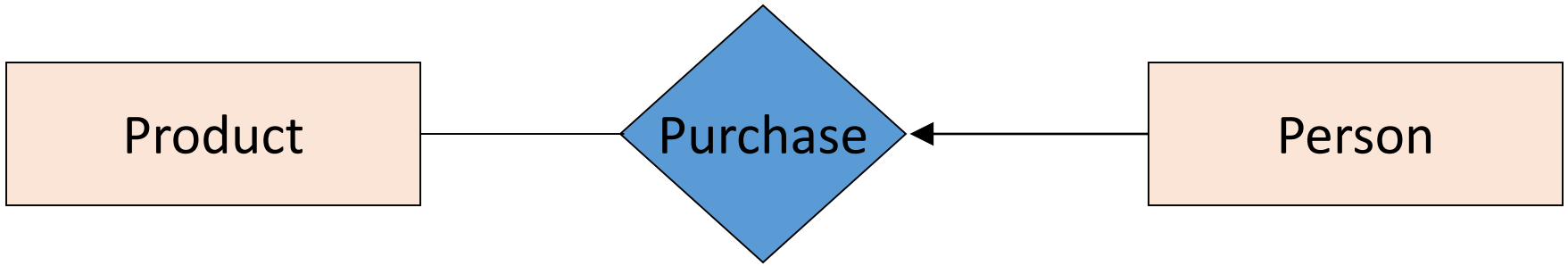


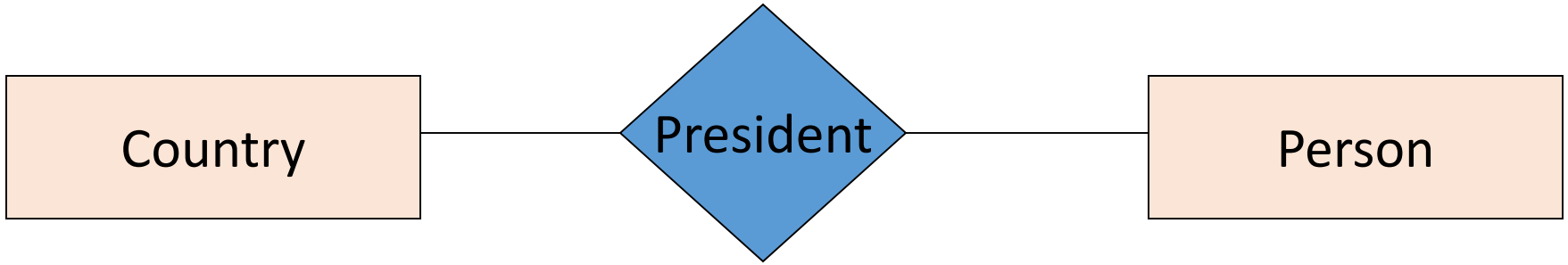
(A) is useful when a relationship really is between multiple entities

- *Ex: A three-party legal contract*

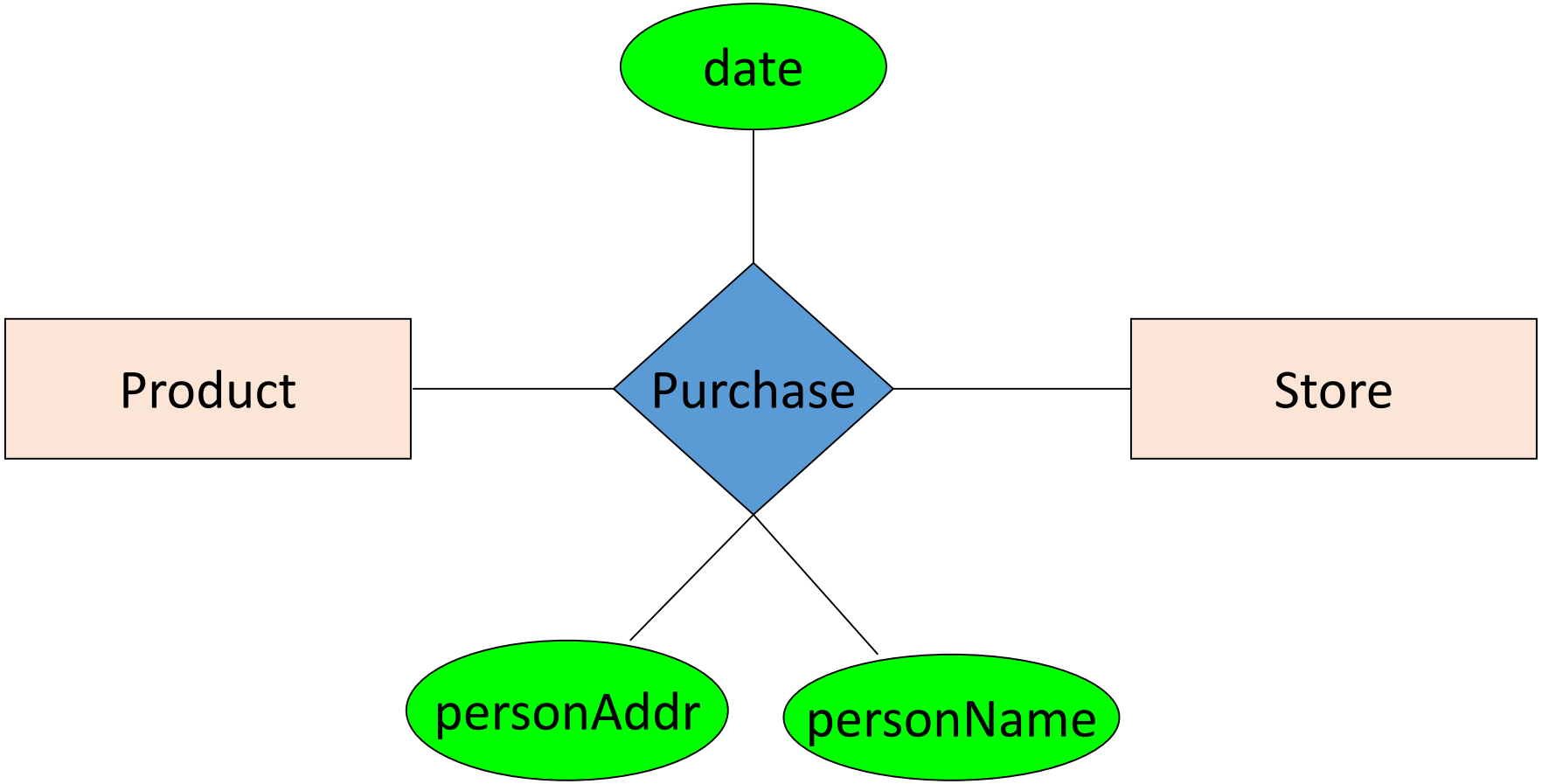
3. Design Principles

What's wrong with these examples?



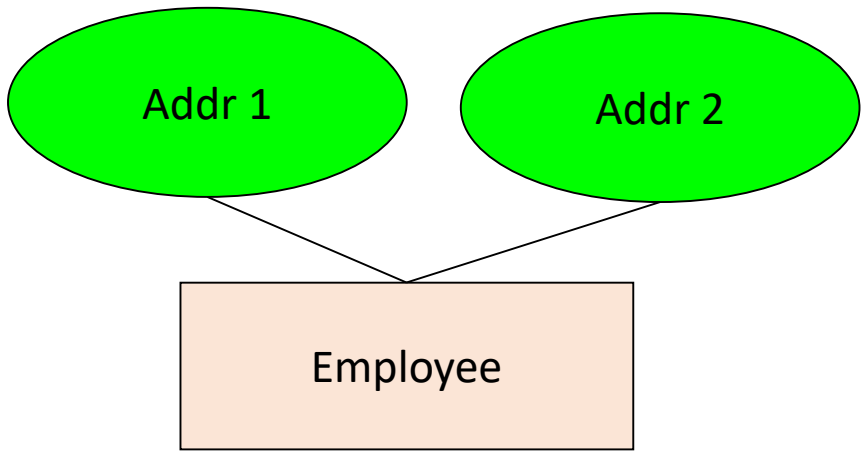


Design Principles: What's Wrong?

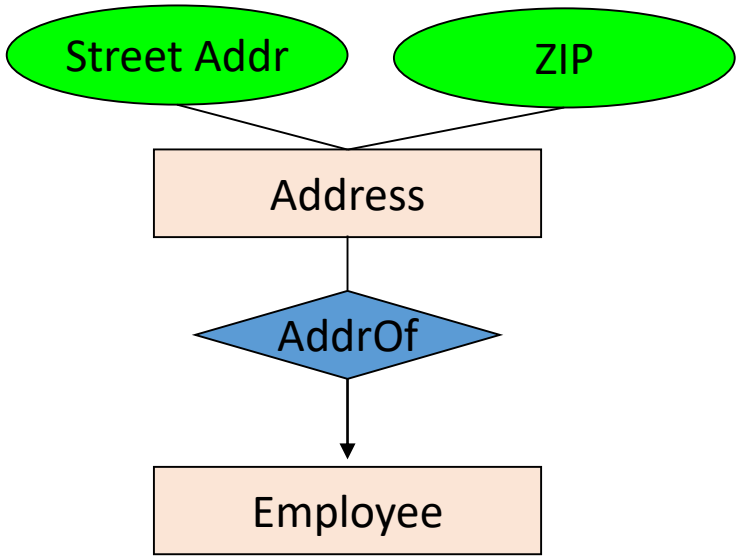


Examples: Entity vs. Attribute

Should address (A) be an attribute?

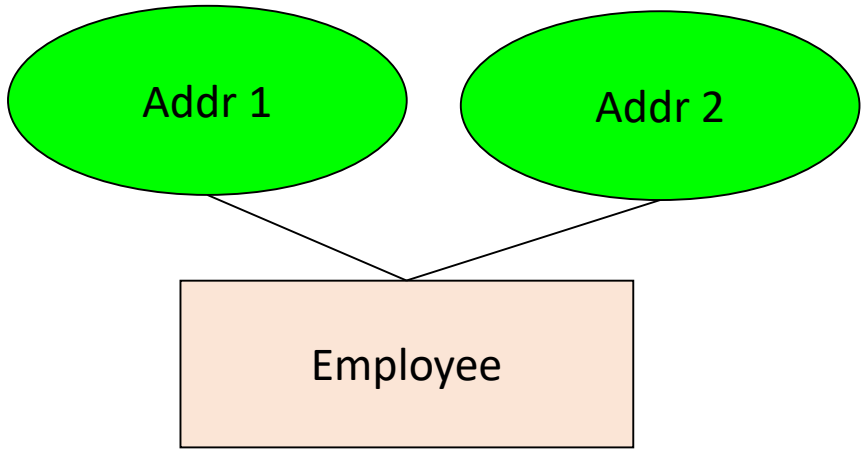


Or (B) be an entity?



Examples: Entity vs. Attribute

Should address (A) be an attribute?

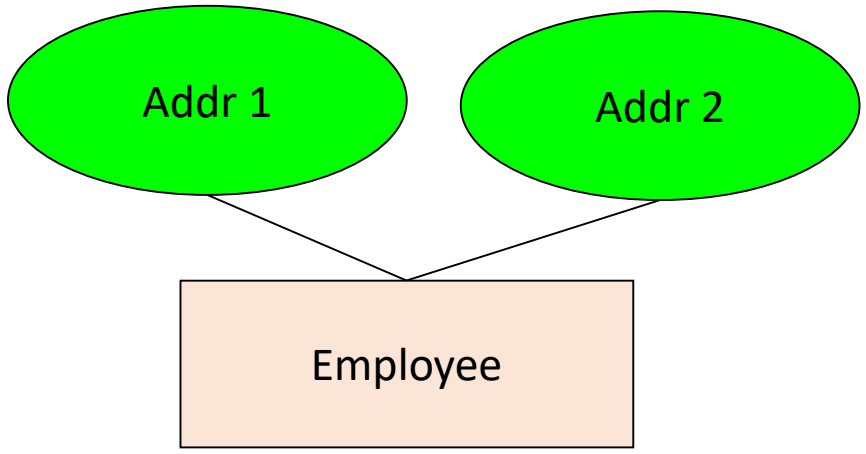


How do we handle employees with multiple addresses here?

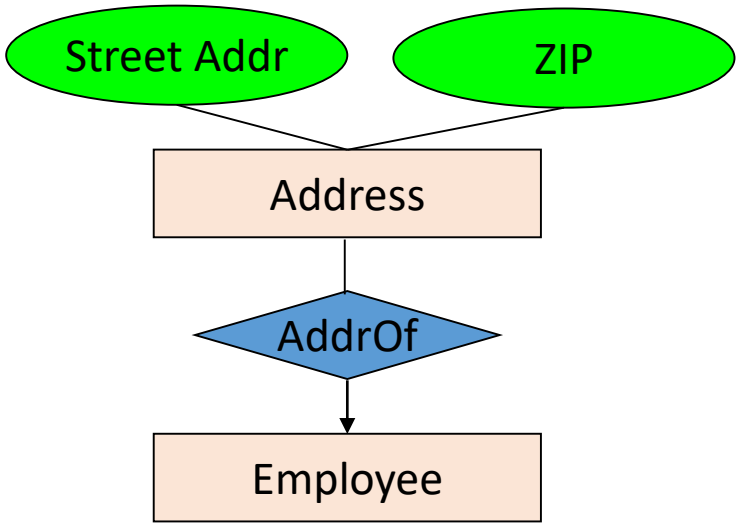
How do we handle addresses where internal structure of the address (e.g. zip code, state) is useful?

Examples: Entity vs. Attribute

Should address (A)
be an attribute?



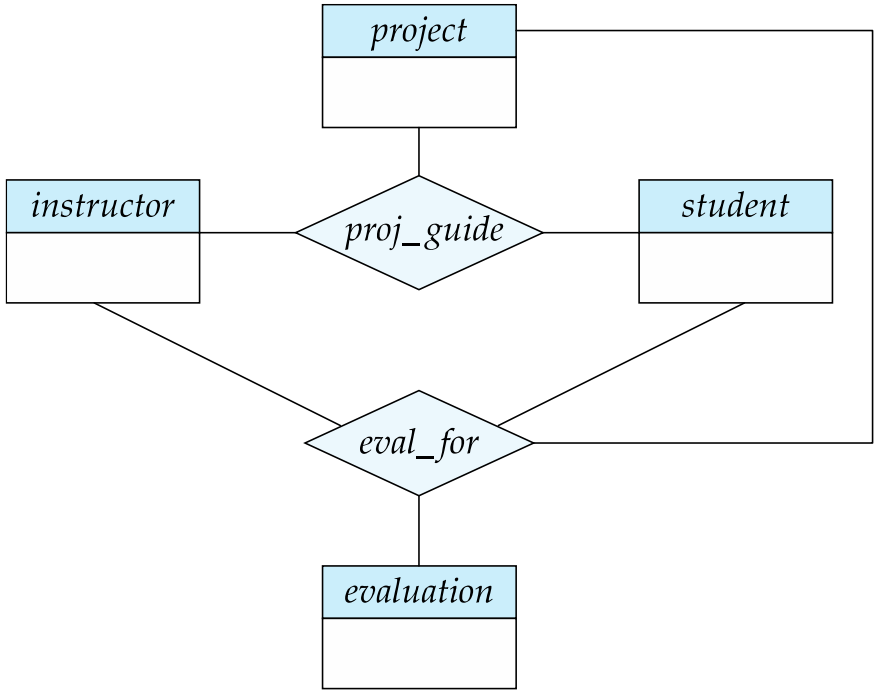
Or (B) be an entity?



In general, when we want to record several values,
we choose new entity

Aggregation

- Suppose we want to record evaluations of a student by a guide on a project

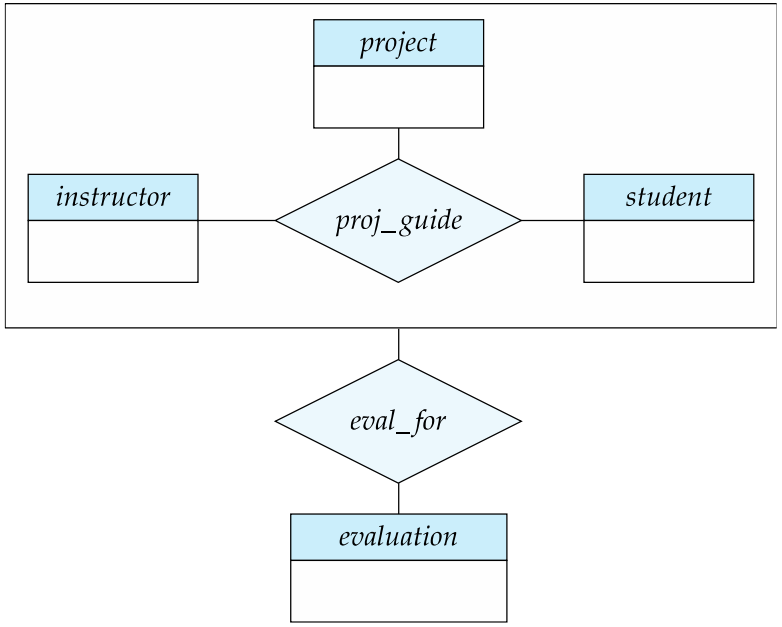


Aggregation (Cont.)

- Relationship sets *eval_for* and *proj_guide* represent overlapping information
  - Every *eval_for* relationship corresponds to a *proj_guide* relationship
  - However, some *proj_guide* relationships may not correspond to any *eval_for* relationships
    - So we can't discard the *proj_guide* relationship
- Eliminate this redundancy via *aggregation*
  - Treat relationship as an abstract entity
  - Allows relationships between relationships
  - Abstraction of relationship into new entity

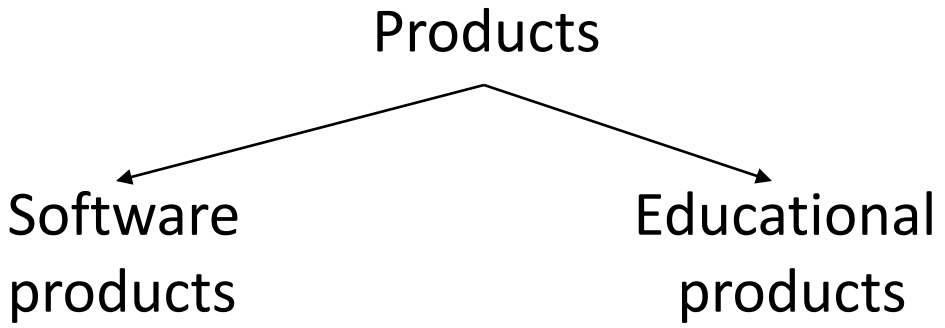
Aggregation (Cont.)

- Eliminate this redundancy via *aggregation* without introducing redundancy, the following diagram represents:
  - A student is guided by a particular instructor on a particular project
  - A student, instructor, project combination may have an associated evaluation



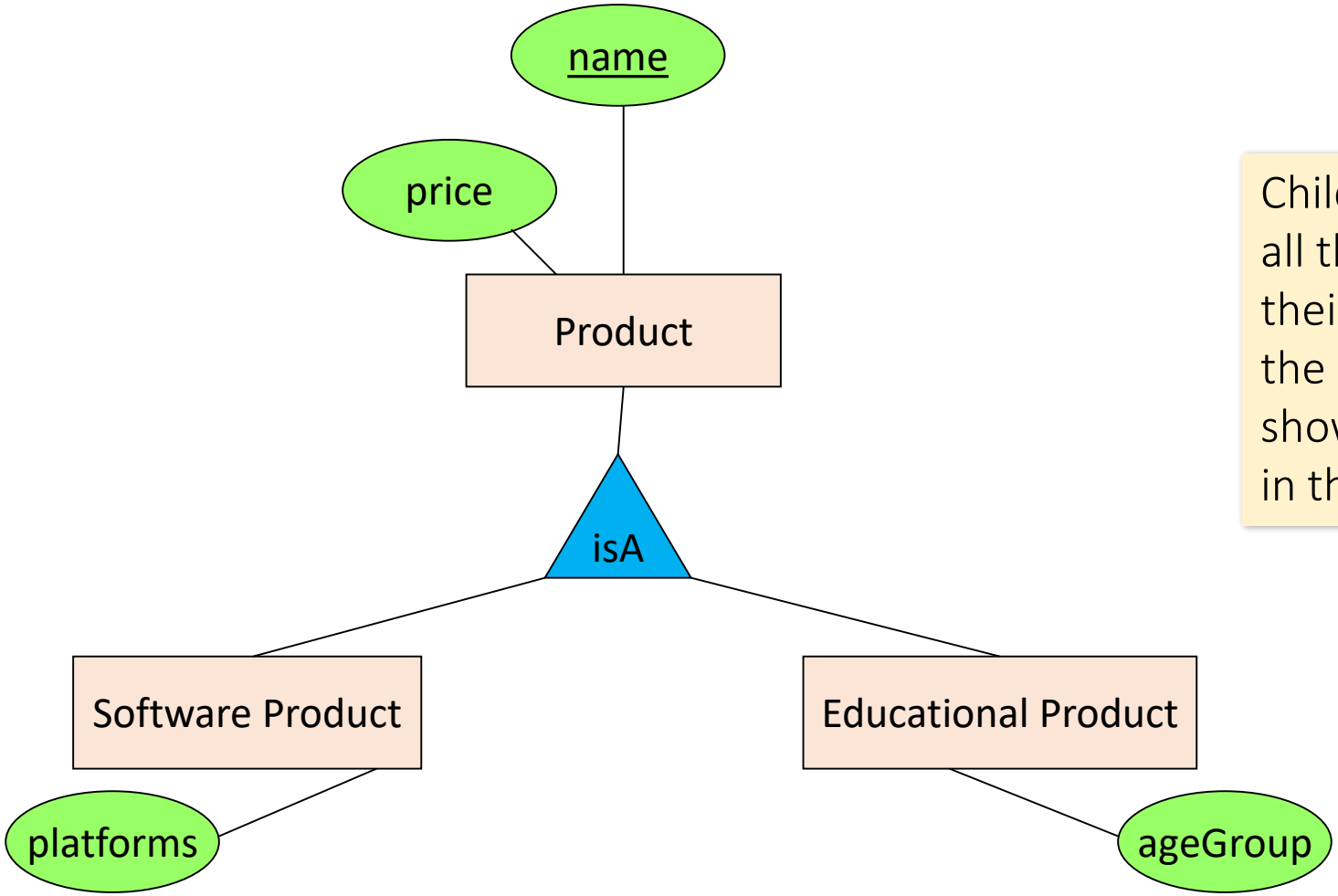
Modeling Subclasses

- Some objects in a class may be special, i.e. worthy of their own class
  - Define a new class?
    - *But what if we want to maintain connection to current class?*
  - Better: define a *subclass*
    - *Ex:*



We can define **subclasses** in ER!

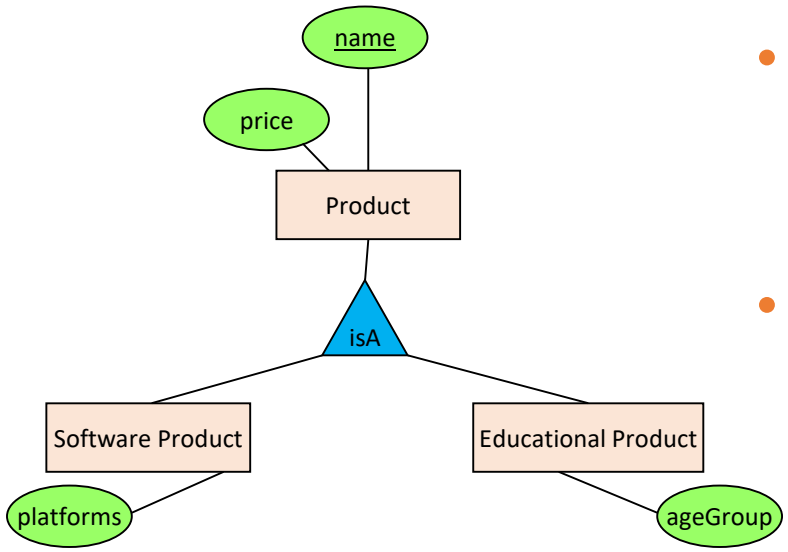
Modeling Subclasses



Child subclasses contain all the attributes of *all* of their parent classes **plus** the new attributes shown attached to them in the ER diagram

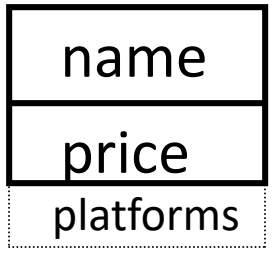
Understanding Subclasses

- Think in terms of records; ex:

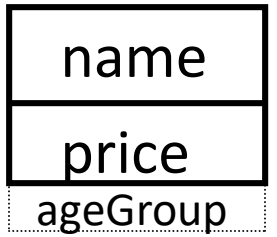


- Product

- SoftwareProduct

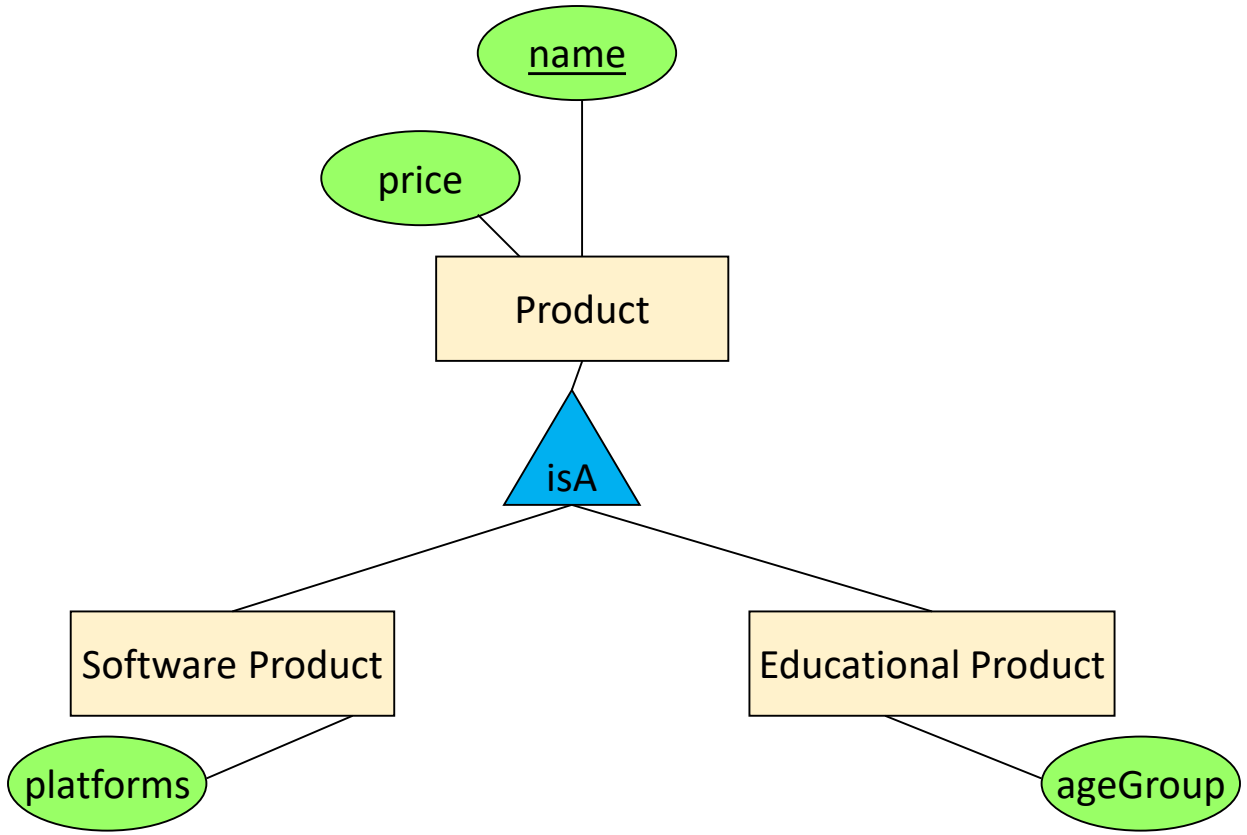


- EducationalProduct



Child subclasses contain all the attributes of *all* of their parent classes **plus** the new attributes shown attached to them in the ER diagram

Think like tables…



Product

name	price	category
Gizmo	99	gadget
Camera	49	photo
Toy	39	gadget

Sw.Product

name	platforms
Gizmo	unix

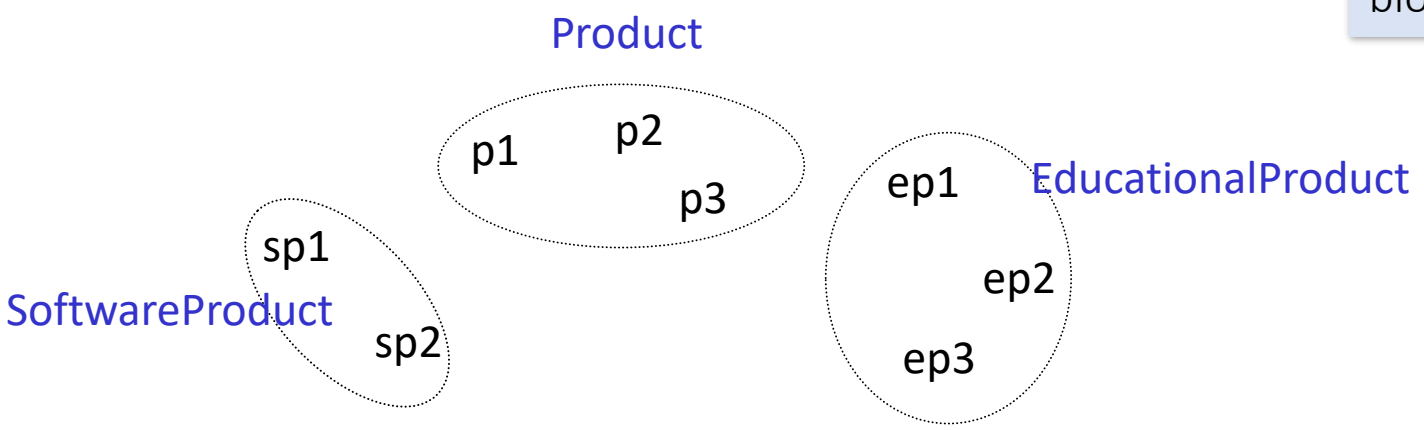
Ed.Product

name	ageGroup
Gizmo	todler
Toy	retired

Difference between OO and ER inheritance

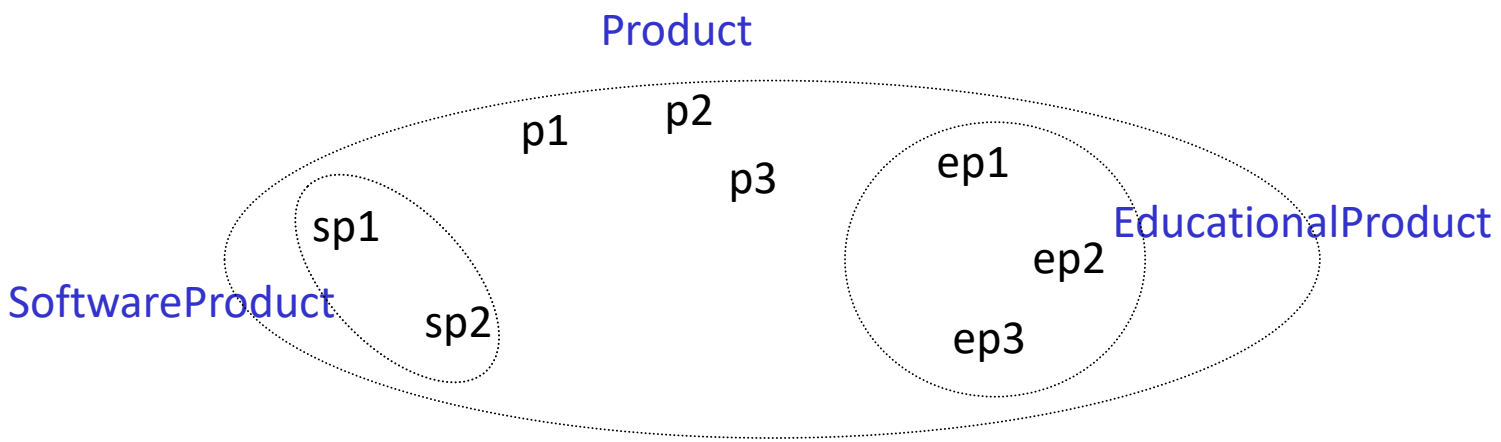
- OO: Classes are disjoint (same for Java, C++)

OO = Object Oriented. E.g. classes as fundamental building block, etc…



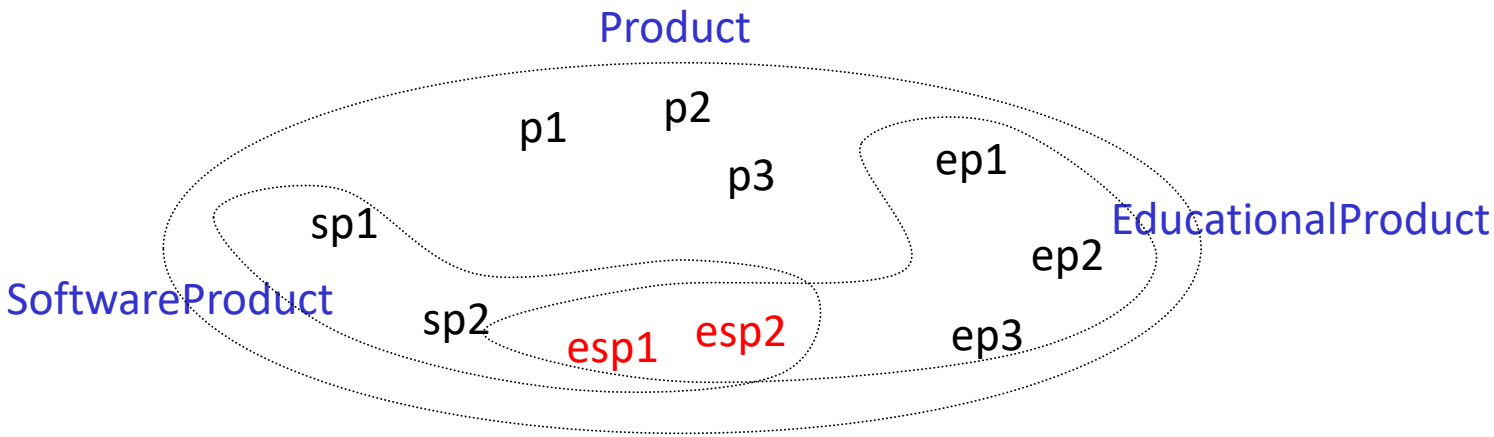
Difference between OO and ER inheritance

- ER: entity sets overlap



Difference between OO and ER inheritance

We have three entity sets, but four different kinds of objects



No need for multiple inheritance in ER

IsA Review

- If we declare ***A IsA B*** then every **A** is a **B**
- We use IsA to
  - Add descriptive attributes to a subclass
  - To identify entities that participate in a relationship
- **No need for multiple inheritance**

Modeling UnionTypes With Subclasses

Person

FurniturePiece

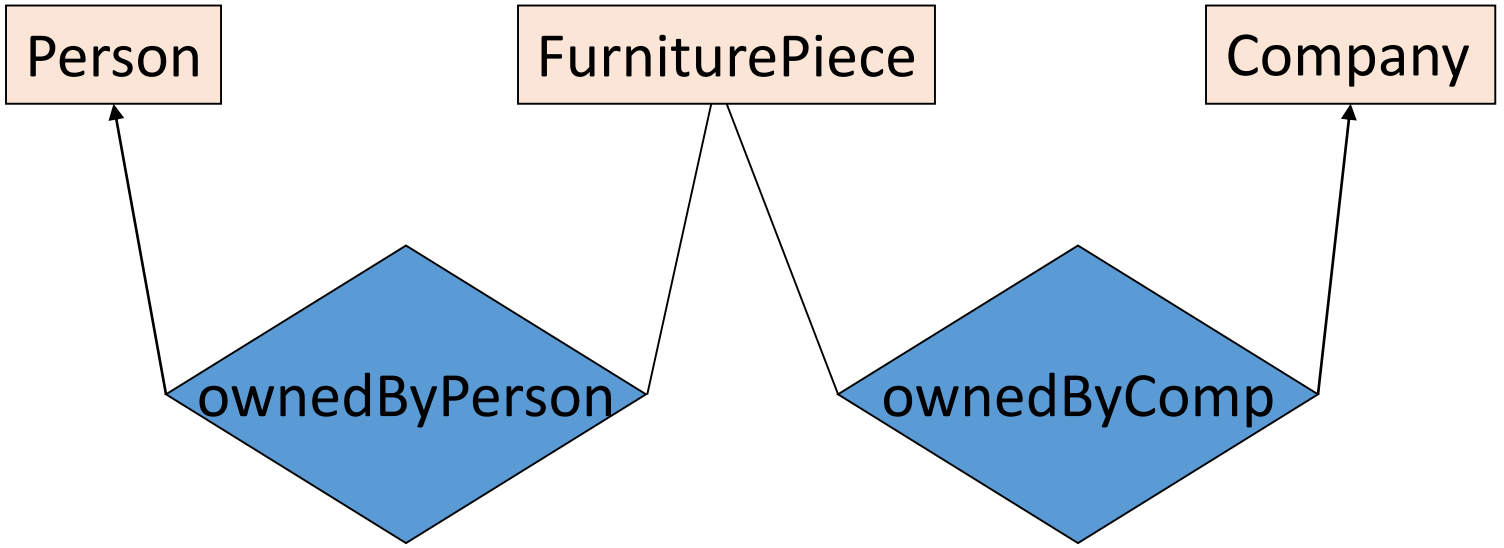
Company

Suppose each piece of furniture is owned either by a person, or by a company. *How do we represent this?*

Modeling Union Types with Subclasses

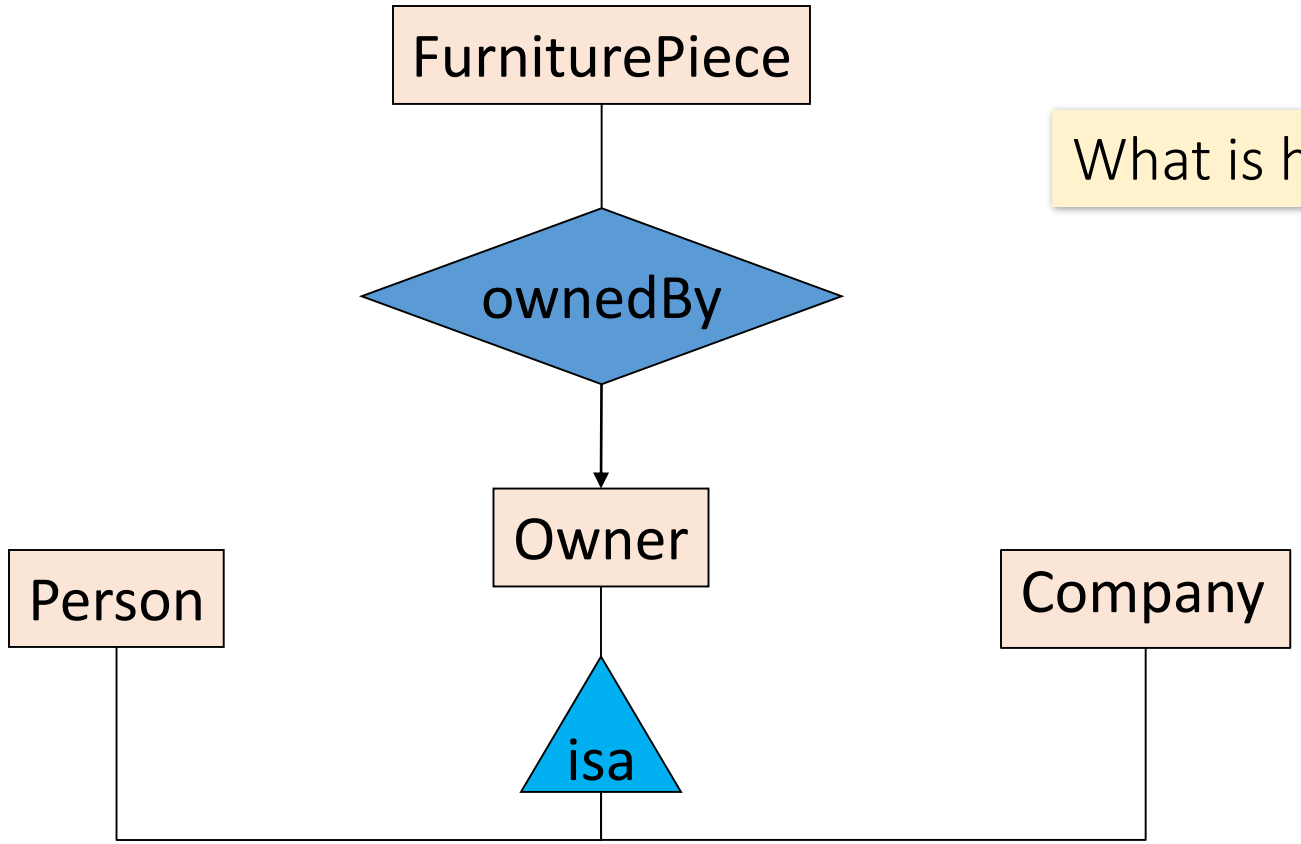
Say: each piece of furniture is owned either by a person, or by a company

Solution 1. Acceptable, but imperfect (What's wrong ?)



Modeling Union Types with Subclasses

Solution 2: better (though more laborious)



What is happening here?

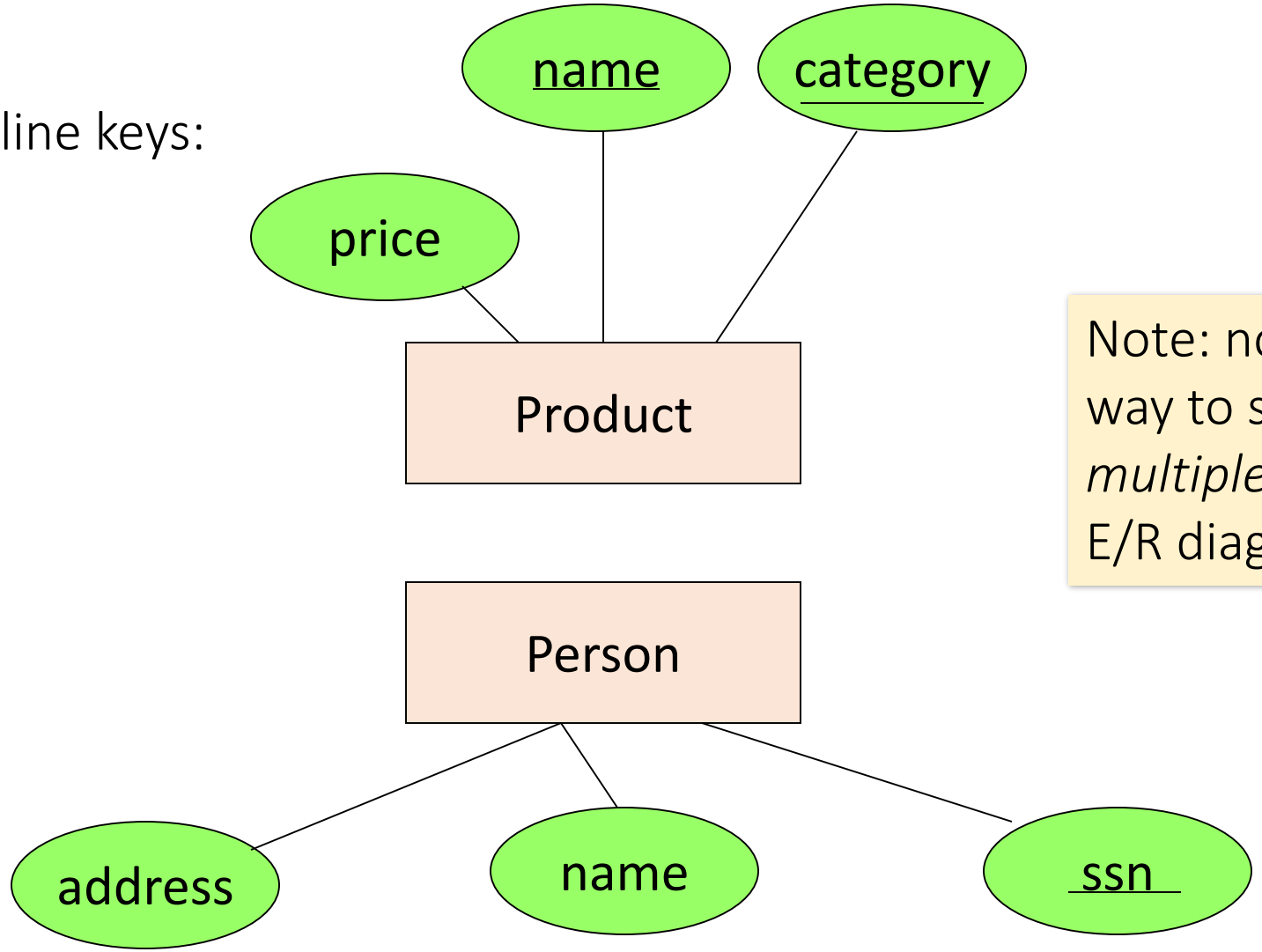
Constraints in ER Diagrams

- Finding constraints is part of the E/R modeling process. Commonly used constraints are:
  - Keys: Implicit constraints on uniqueness of entities
    - *Ex: An SSN uniquely identifies a person*
  - Single-value constraints:
    - *Ex: a person can have only one father*
  - Referential integrity constraints: Referenced entities must exist
    - *Ex: if you work for a company, it must exist in the database*
  - Other constraints:
    - *Ex: peoples' ages are between 0 and 150*

Recall FOREIGN KEYs!

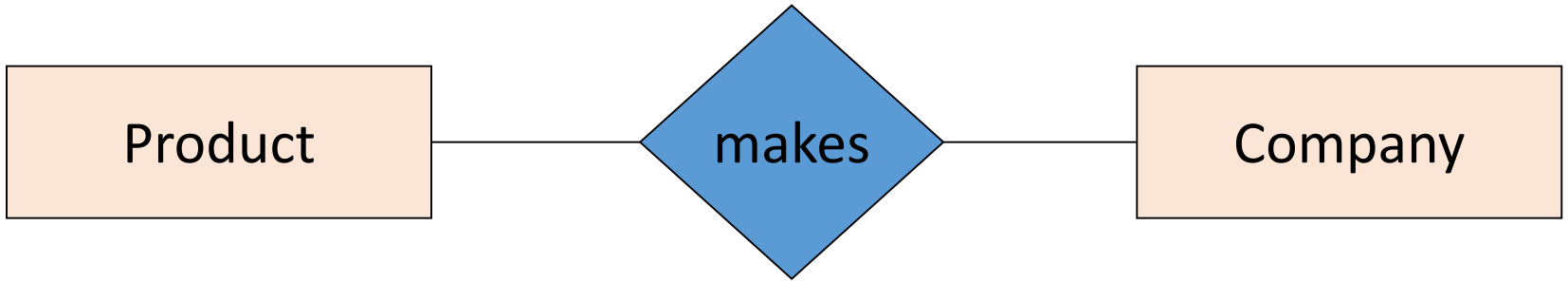
Keys in ER Diagrams

Underline keys:

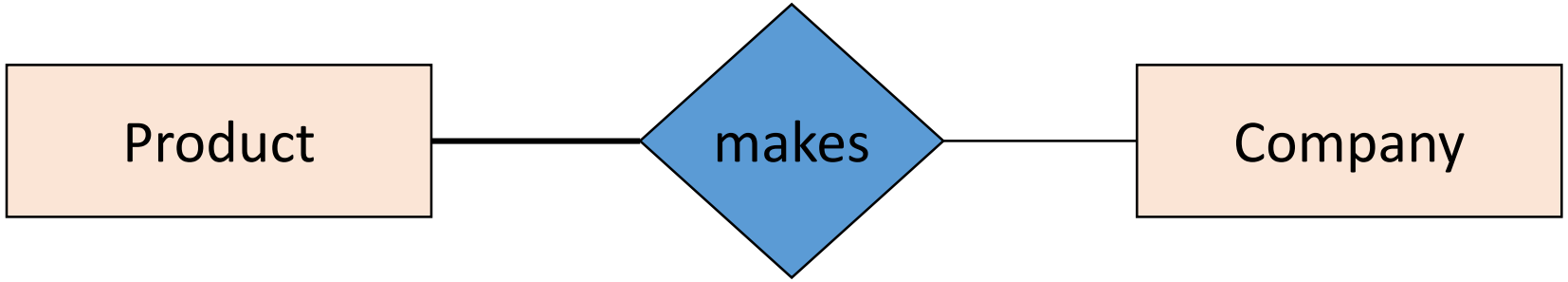


Note: no formal way to specify *multiple* keys in E/R diagrams…

Participation Constraints: Partial v. Total

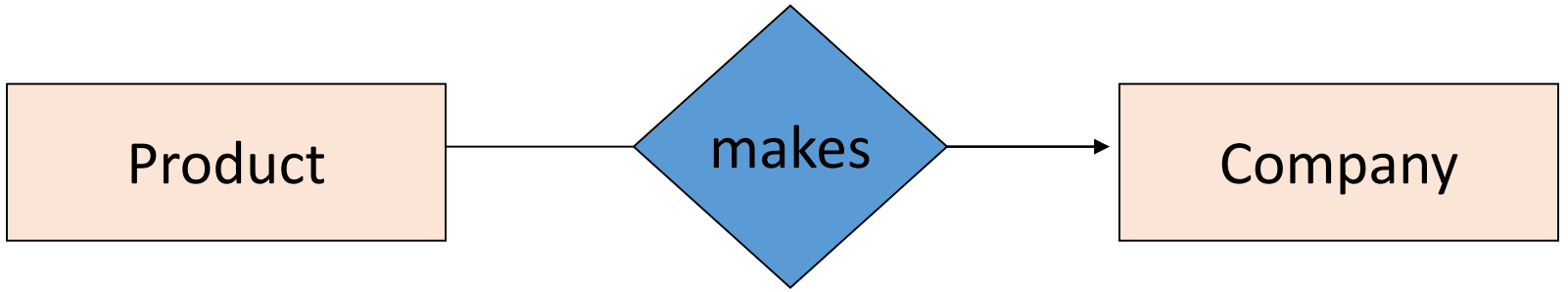


Are there products made by no company?
Companies that don't make a product?

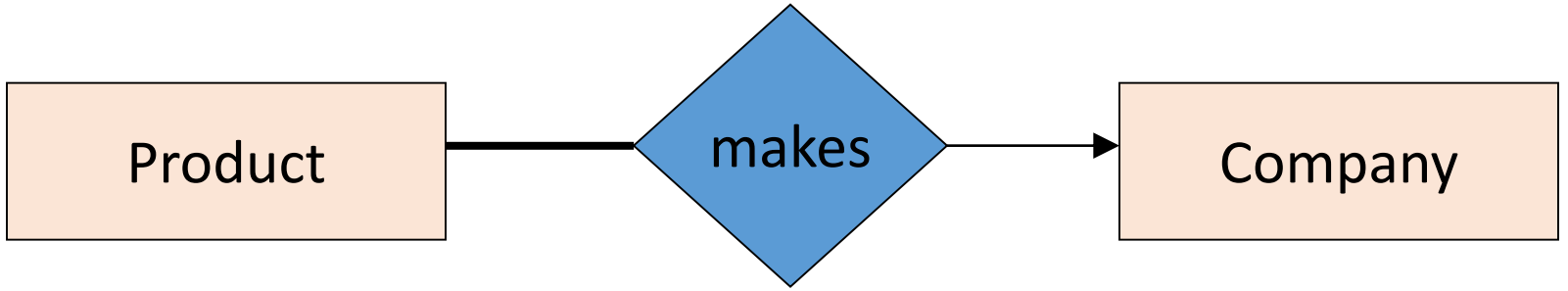


Bold line indicates *total participation* (i.e. here: all products are made by a company)

Referential Integrity Constraints



Each product made by at most one company.
Some products made by no company?



Each product made by *exactly* one company.

Weak Entity Sets

- Consider a section entity, which is uniquely identified by a *course_id, semester, year*, and *sec_id*.
- Clearly, section entities are related to course entities. Suppose we create a relationship set *sec_course* between entity sets section and course.
- Note that the information in *sec_course* is redundant, since section already has an attribute *course_id*, which identifies the course with which the section is related.
- One option to deal with this redundancy is to get rid of the relationship *sec_course*; however, by doing so the relationship between section and course becomes implicit in an attribute, which is not desirable.

Weak Entity Sets (Cont.)

- An alternative way to deal with this redundancy is to not store the attribute *course_id* in the *section* entity and to only store the remaining attributes *section_id*, *year*, and *semester.*
  - However, the entity set *section* then does not have enough attributes to identify a particular *section* entity uniquely
- To deal with this problem, we treat the relationship *sec_course* as a special relationship that provides extra information, in this case, the *course_id*, required to identify *section* entities uniquely.
- A **weak entity set** is one whose existence is dependent on another entity, called its **identifying entity**
- Instead of associating a primary key with a weak entity, we use the identifying entity, along with extra attributes called **discriminator** to uniquely identify a weak entity.

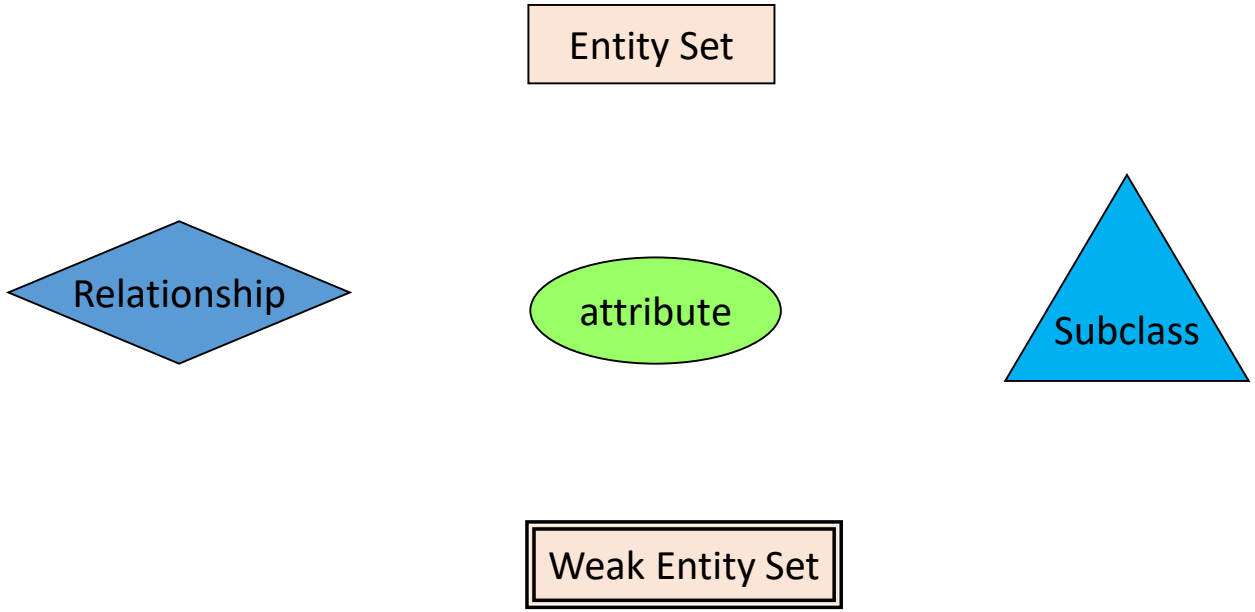
Weak Entity Sets (Cont.)

- An entity set that is not a weak entity set is termed a **strong entity set**.
- Every weak entity must be associated with an identifying entity; that is, the weak entity set is said to be **existence dependent** on the identifying entity set.
- The identifying entity set is said to **own** the weak entity set that it identifies.
- The relationship associating the weak entity set with the identifying entity set is called the **identifying relationship**.
- Note that the relational schema we eventually create from the entity set *section* does have the attribute *course_id*, for reasons that will become clear later, even though we have dropped the attribute *course_id* from the entity set *section.*

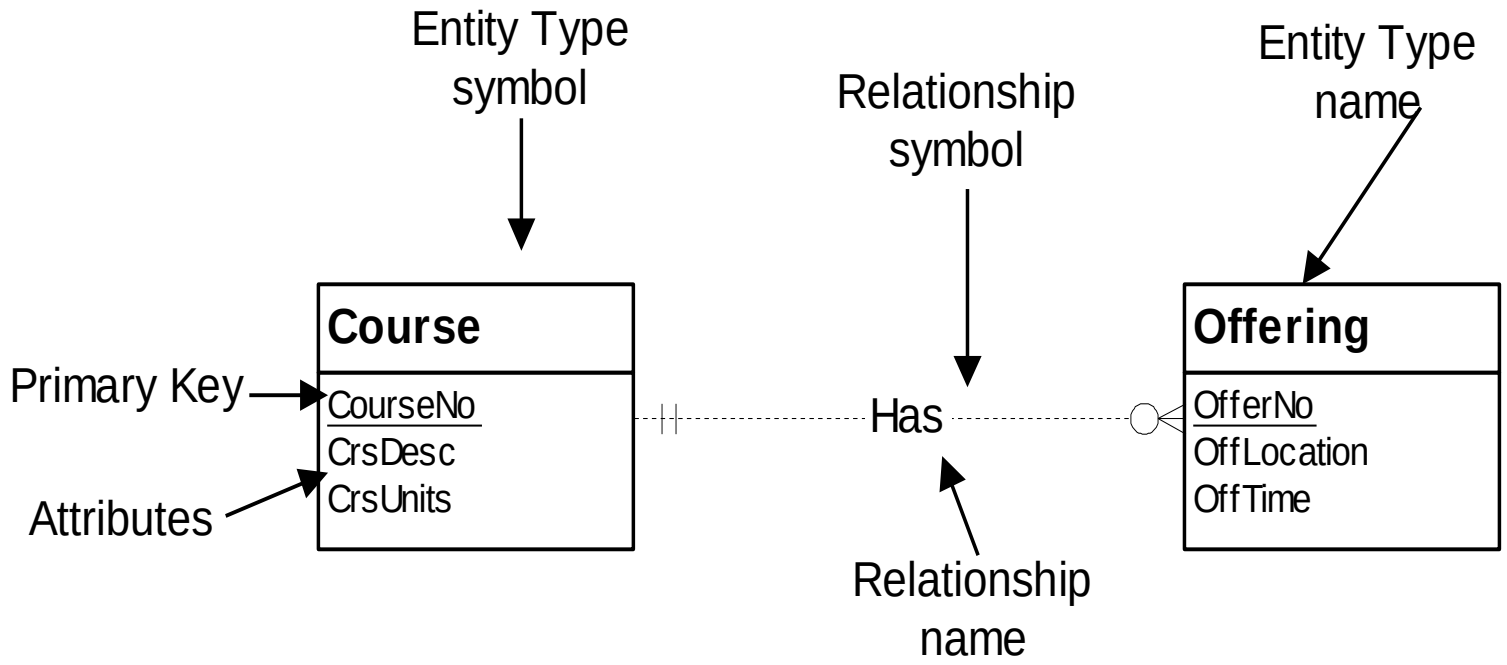
Expressing Weak Entity Sets

- In E-R diagrams, a weak entity set is depicted via a double rectangle.
- We underline the discriminator of a weak entity set with a dashed line.
- The relationship set connecting the weak entity set to the identifying strong entity set is depicted by a double diamond.
- Primary key for *section* – (*course_id, sec_id, semester, year*)

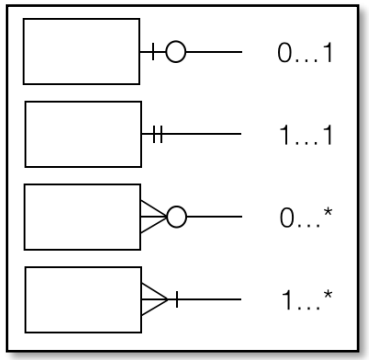
Summary of Used Symbols

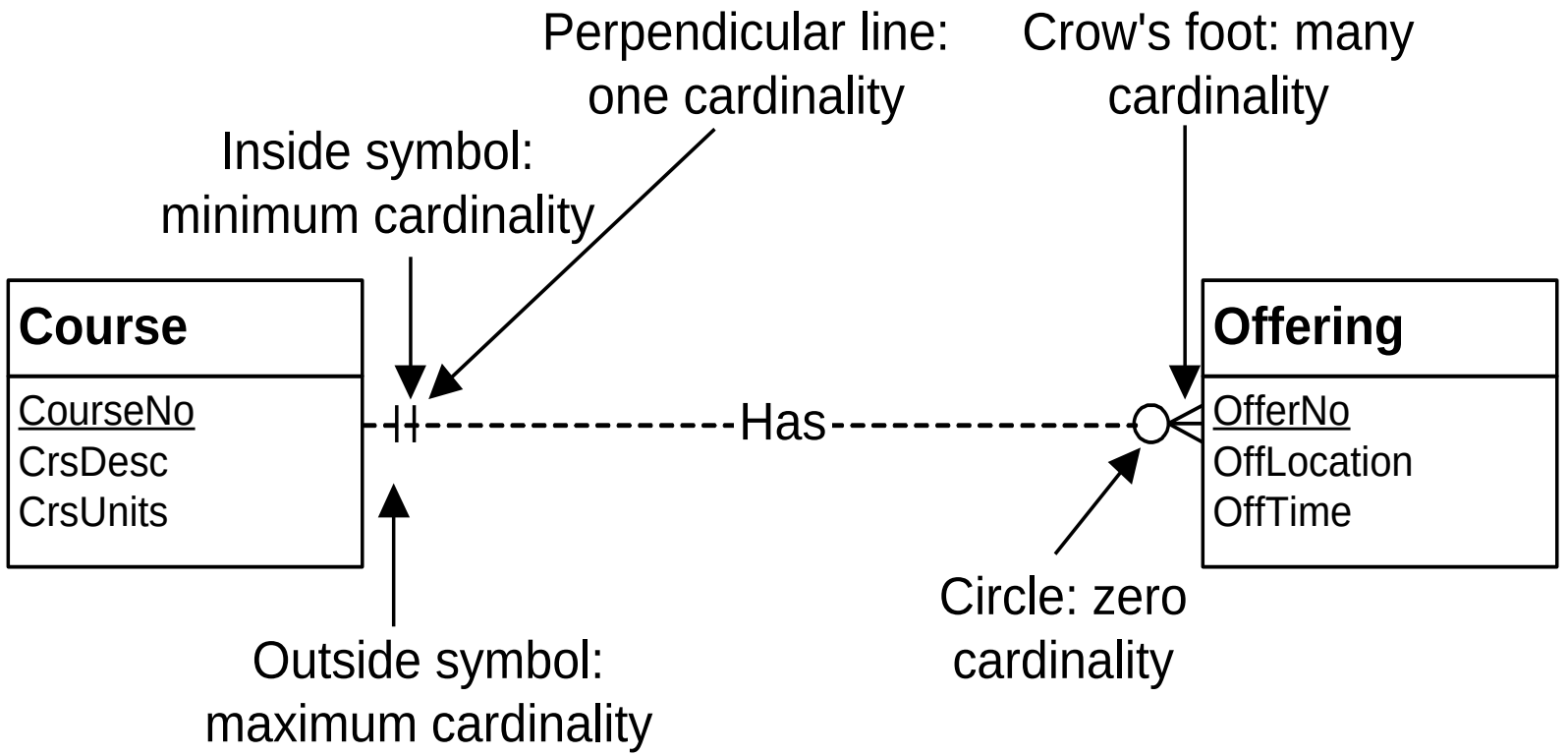


Alternative Representations: Basic Symbols

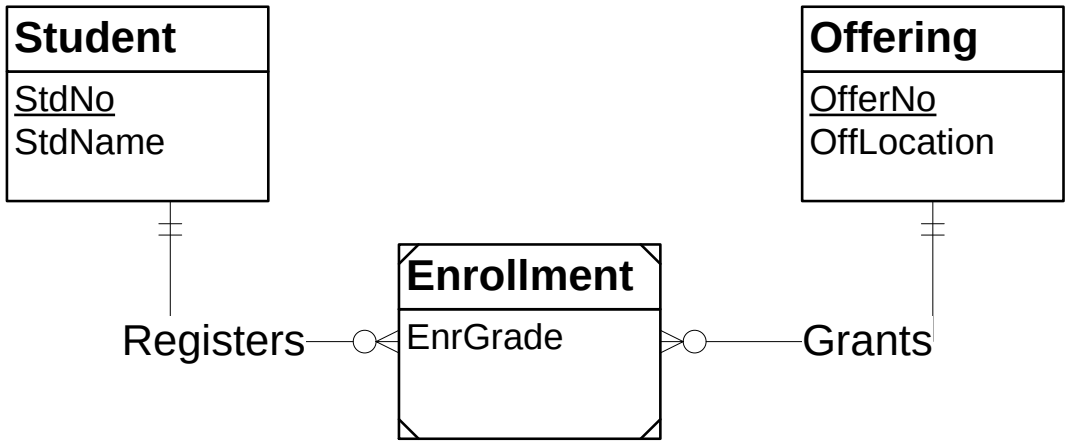


Alternative Representations: Cardinality

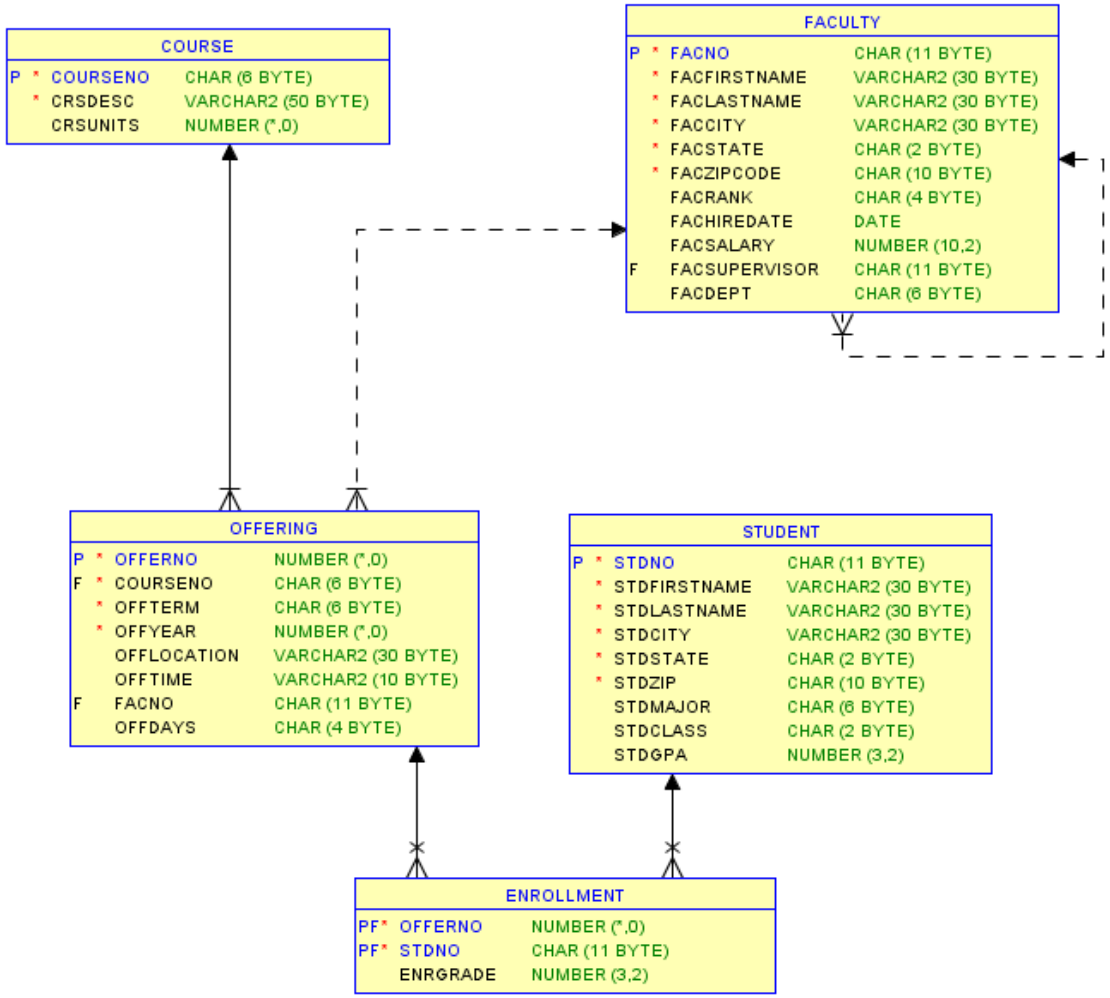




Alternative Representations: Example



Alternative Representations: Tool X



ER Summary

- E/R diagrams are a visual syntax that allows technical and non-technical people to talk
  - For conceptual design

- Basic constructs: **entity**, **relationship**, and **attributes**

- A good design is faithful to the constraints of the application, but not overzealous

ER Design Decisions

- The use of an attribute or entity set to represent an object.
- Whether a real-world concept is best expressed by an entity set or a relationship set.
- The use of a ternary relationship versus a pair of binary relationships.
- The use of a strong or weak entity set.
- The use of specialization/generalization – contributes to modularity in the design.
- The use of aggregation – can treat the aggregate entity set as a single unit without concern for the details of its internal structure.

Acknowledgements

The course material used for this lecture is mostly taken and/or adopted from

- The course materials of the *CS145 Introduction to Databases* lecture given by *Christopher Ré* at *Stanford University* (http://web.stanford.edu/class/cs145/).
- From the slides of the textbook Database System Concepts, Seventh Edition by Avi Silberschatz, Henry F. Korth, S. Sudarshan.