

# CS150: Database & Datamining

## Lecture 5: The E/R Model

Xuming He  
Spring 2019

*Acknowledgement: Slides are adopted from the Berkeley course CS186 by Joey Gonzalez and Joe Hellerstein, Stanford CS145 by Peter Bailis.*

# Today's Lecture

1. E/R Basics: Entities & Relations
  - ACTIVITY: Crayon time!
2. E/R Design considerations
  - ACTIVITY: Crayon time pt. II
3. Advanced E/R Concepts
  - ACTIVITY: E/R Translation

# 1. E/R Basics: Entities & Relations

# What you will learn about in this section

1. High-level motivation for the E/R model
2. Entities
3. Relations
4. ACTIVITY: Crayon Time! Drawing E/R diagrams

# Database Design

- **Database design: Why do we need it?**
  - Agree on structure of the database before deciding on a particular implementation
- **Consider issues such as:**
  - What entities to model
  - How entities are related
  - What constraints exist in the domain
  - How to achieve good designs
- **Several formalisms exist**
  - We discuss one flavor of E/R diagrams

# Database Design Process

1. Requirements Analysis

2. Conceptual Design

3. Logical, Physical, Security, etc.

## 1. Requirements analysis

- What is going to be stored?
- How is it going to be used?
- What are we going to do with the data?
- Who should access the data?

Technical and non-technical people are involved

# Database Design Process

1. Requirements Analysis

2. Conceptual Design

3. Logical, Physical, Security, etc.

## 2. Conceptual Design

- A high-level description of the database
- Sufficiently precise that technical people can understand it
- But, not so precise that non-technical people can't participate

This is where E/R fits in.

# Database Design Process

1. Requirements Analysis

2. Conceptual Design

3. Logical, Physical, Security, etc.

## 3. More:

- Logical Database Design
- Physical Database Design
- Security Design



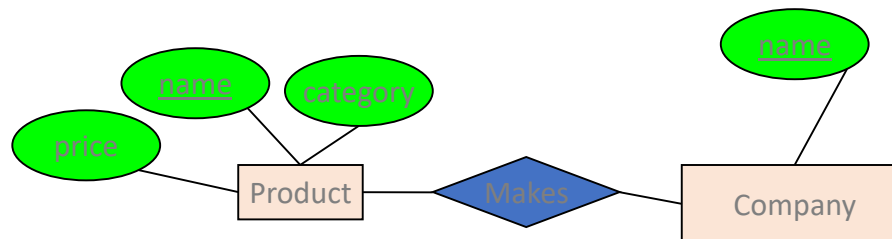
# Database Design Process

1. Requirements Analysis

2. Conceptual Design

3. Logical, Physical, Security, etc.

E/R Model & Diagrams used



This process is iterated **many** times

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

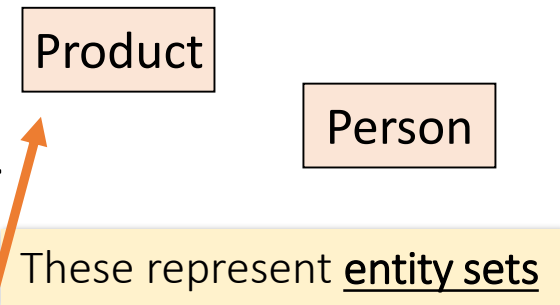
# Interlude: Impact of the ER model

- The E/R 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



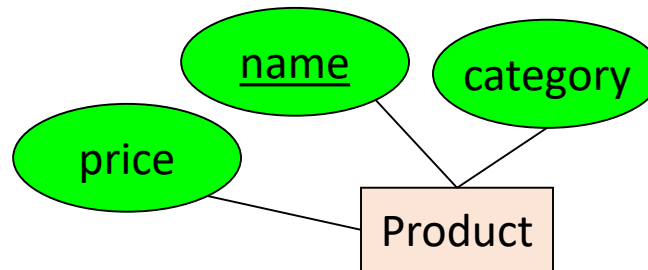
# Entities and Entity Sets

- **Entities & entity sets** are the primitive unit of the E/R 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

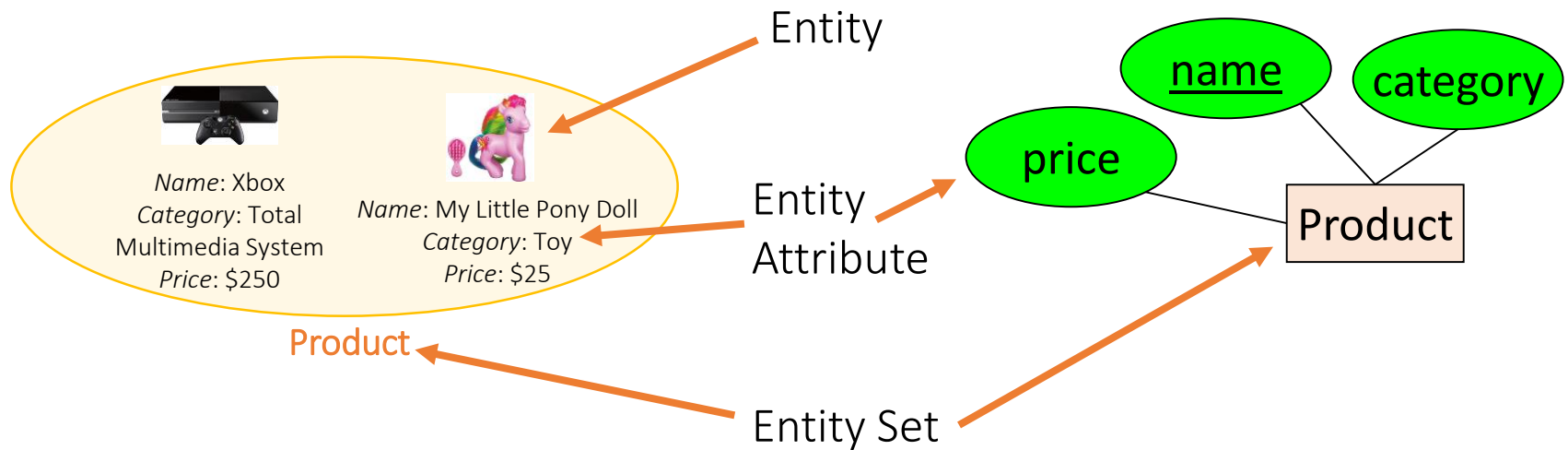


Shapes are important.  
Colors are not.

# Entities vs. Entity Sets

*Example:*

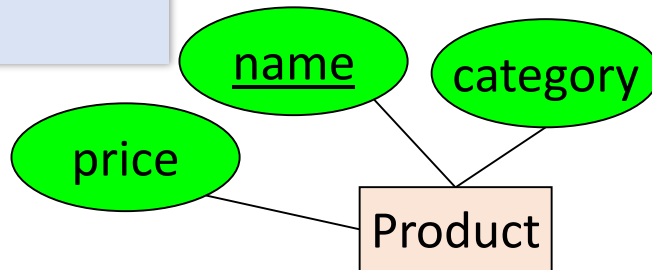
Entities are not explicitly represented in E/R diagrams!



# Keys

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

Denote elements of the primary key by underlining.



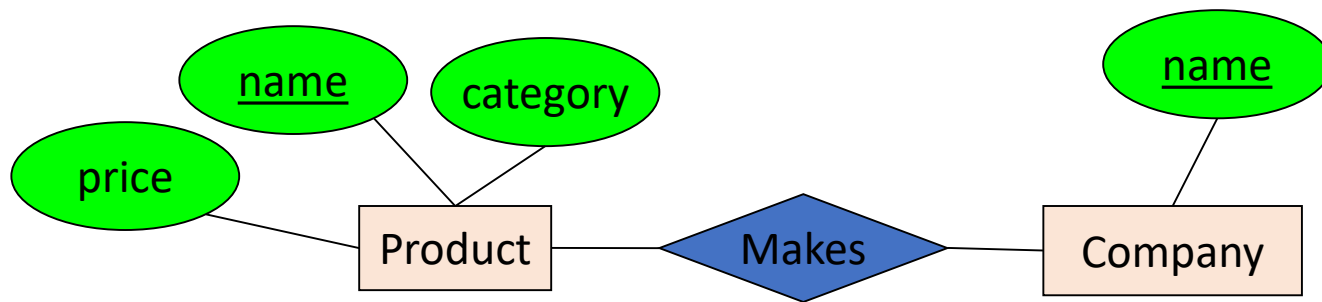
Here, {name, category} is not a key (it is not *minimal*).

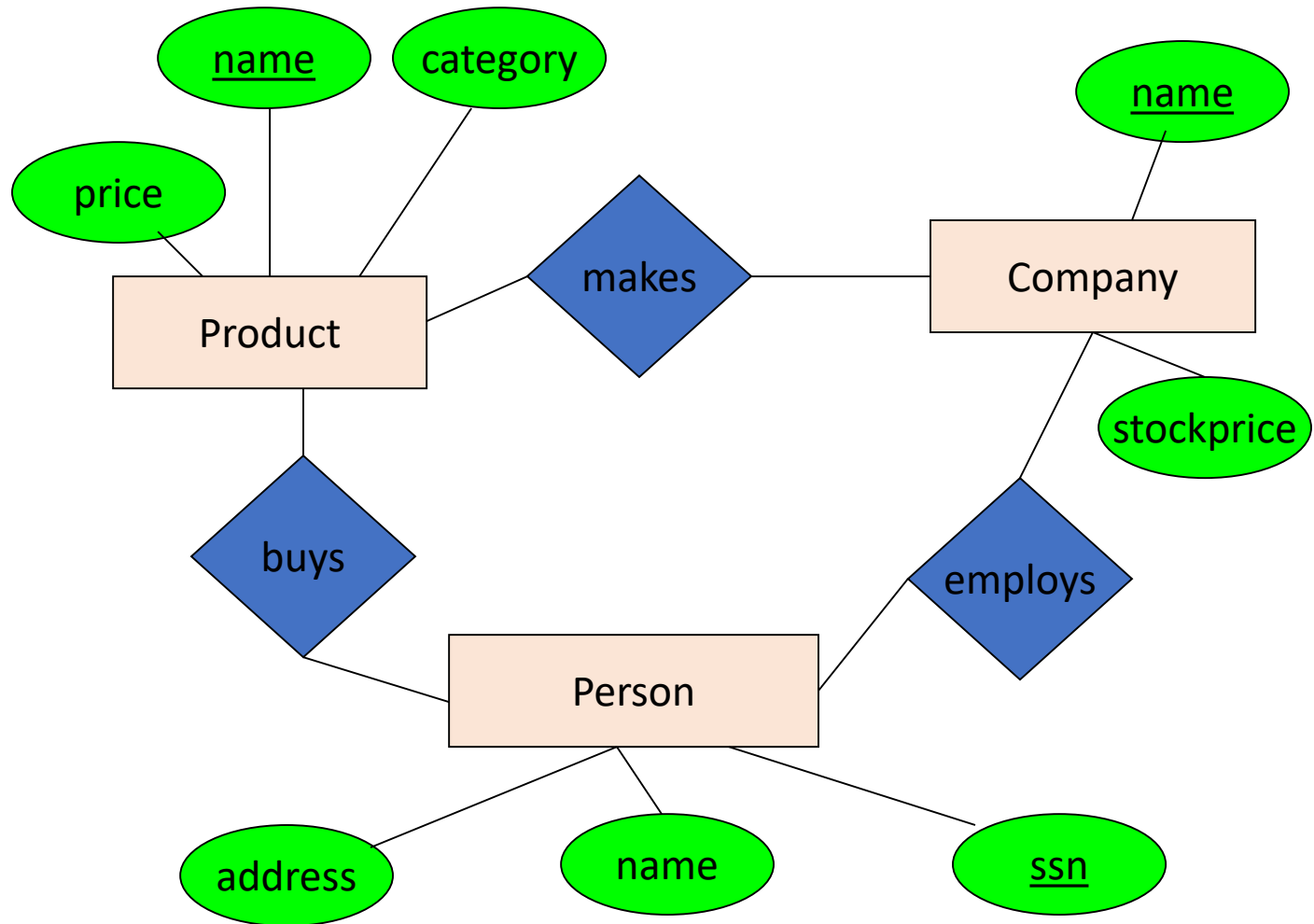
*If it were, what would it mean?*

The E/R model forces us to designate a single primary key, though there may be multiple candidate keys

# The R in E/R: Relationships

- A **relationship** is between two entities



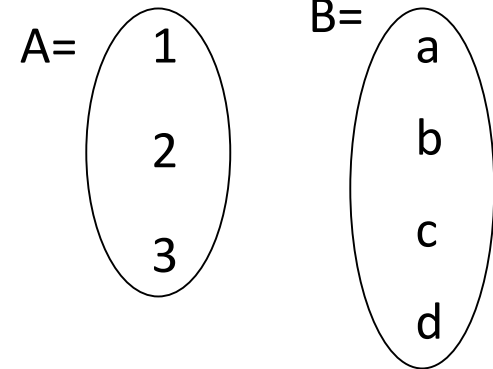




# What is a Relationship?

- ***A mathematical definition:***

- Let A, B be sets
  - $A=\{1,2,3\}, \quad B=\{a,b,c,d\}$



# What is a Relationship?

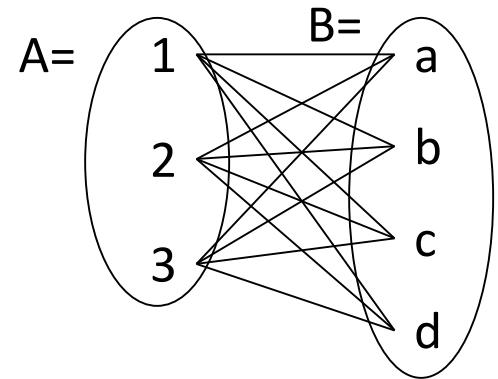
- ***A mathematical definition:***

- Let A, B be sets

- $A=\{1,2,3\}, \quad B=\{a,b,c,d\}$

- $A \times B$  (the ***cross-product***) is the set of all pairs (a,b)

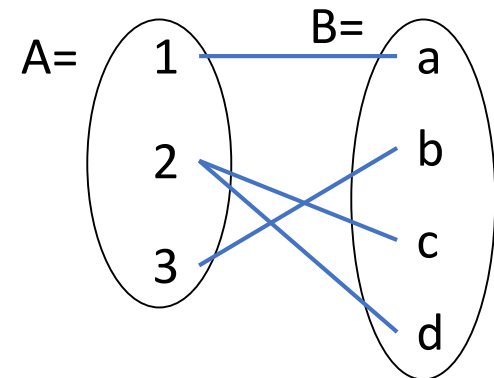
- $A \times B = \{(1,a), (1,b), (1,c), (1,d), (2,a), (2,b), (2,c), (2,d), (3,a), (3,b), (3,c), (3,d)\}$



# What is a Relationship?

- ***A mathematical definition:***

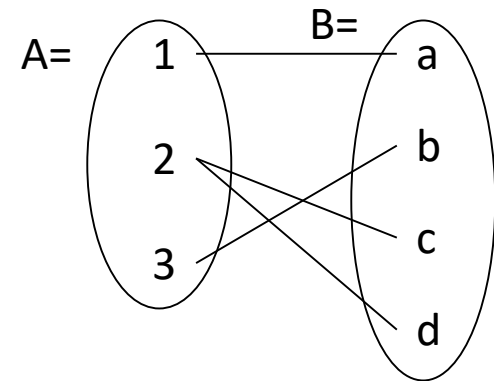
- Let A, B be sets
  - $A = \{1, 2, 3\}$ ,  $B = \{a, b, c, d\}$ ,
- $A \times B$  (the ***cross-product***) is the set of all pairs (a,b)
  - $A \times B = \{(1,a), (1,b), (1,c), (1,d), (2,a), (2,b), (2,c), (2,d), (3,a), (3,b), (3,c), (3,d)\}$
- We define a relationship to be a subset of  $A \times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



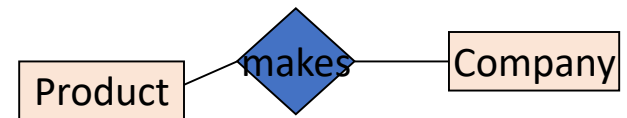
# What is a Relationship?

- ***A mathematical definition:***

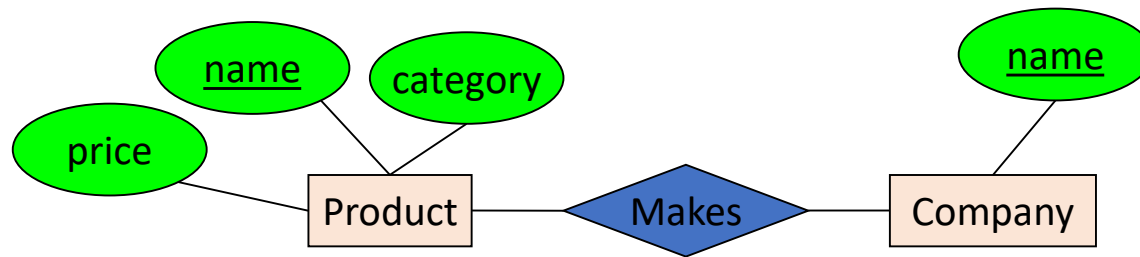
- Let A, B be sets
- $A \times B$  (the ***cross-product***) is the set of all pairs
- A relationship is a subset of  $A \times B$



- **Makes** is relationship- it is a ***subset*** of **Product**  $\times$  **Company**:



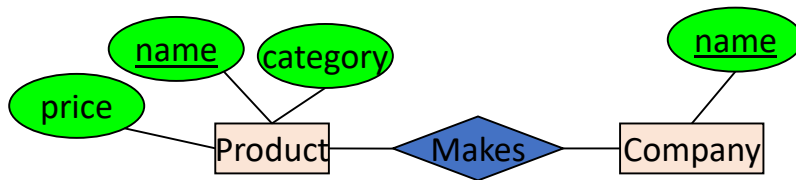
# What is a Relationship?



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		Product		
<u>name</u>		<u>name</u>	category	price
GizmoWorks		Gizmo	Electronics	\$9.99
GadgetCorp		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

<u>name</u>
GizmoWorks
GadgetCorp

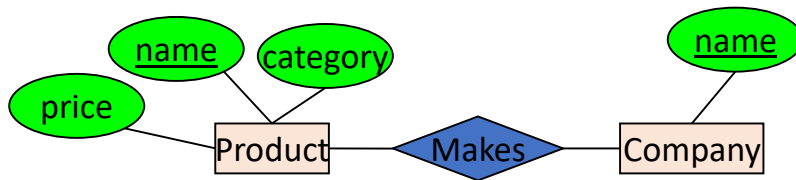
Product

<u>name</u>	category	price
Gizmo	Electronics	\$9.99
GizmoLite	Electronics	\$7.50
Gadget	Toys	\$5.50



Company C × Product P

<u>C.name</u>	<u>P.name</u>	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

<u>name</u>
GizmoWorks
GadgetCorp

Product

<u>name</u>	category	price
Gizmo	Electronics	\$9.99
GizmoLite	Electronics	\$7.50
Gadget	Toys	\$5.50



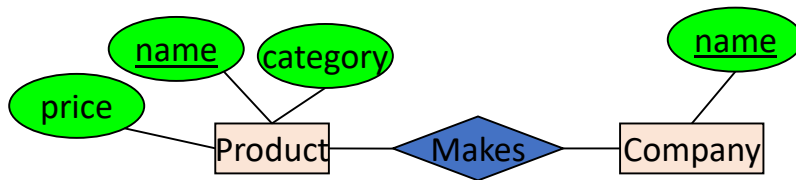
Company C × Product P

<u>C.name</u>	<u>P.name</u>	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

<u>C.name</u>	<u>P.name</u>
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*

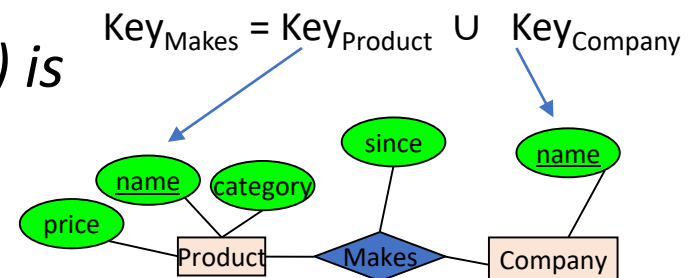


# What is a Relationship?

- There can only be **one relationship for every unique combination of entities**
- This also means that **the relationship is uniquely determined by the keys of its entities**
- *Example: the “key” for Makes (to right) is  $\{Product.name, Company.name\}$*

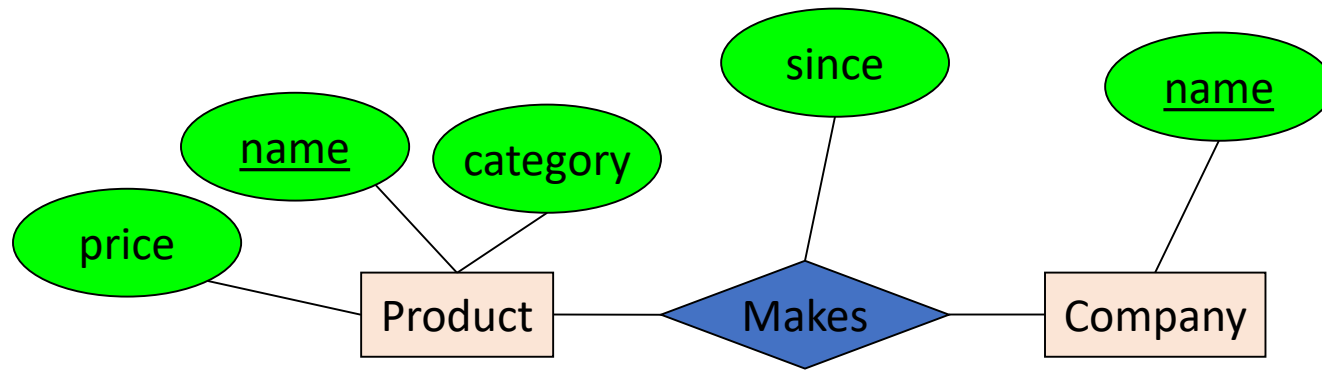
This follows from our mathematical definition of a relationship- it's a SET!

Why does this make sense?



# Relationships and Attributes

- Relationships may have attributes as well.



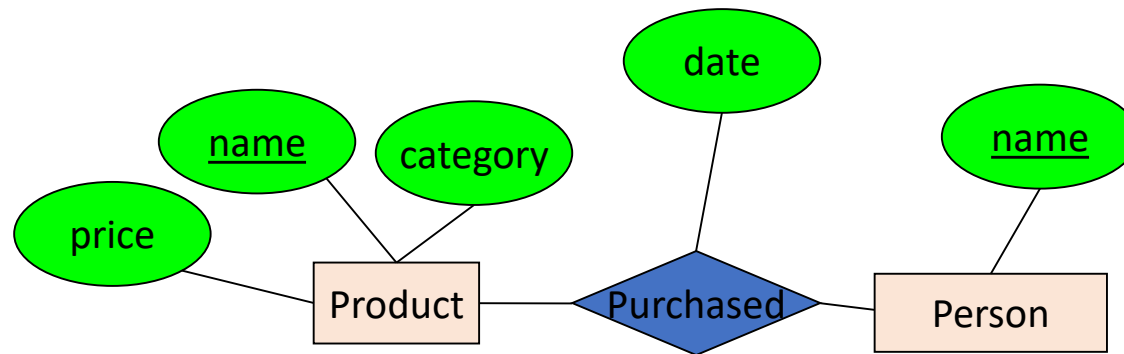
For example: “since” records when company started making a product

Note: “since” is implicitly unique per pair here! Why?

Note #2: Why not “how long”?

# Decision: Relationship vs. Entity?

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

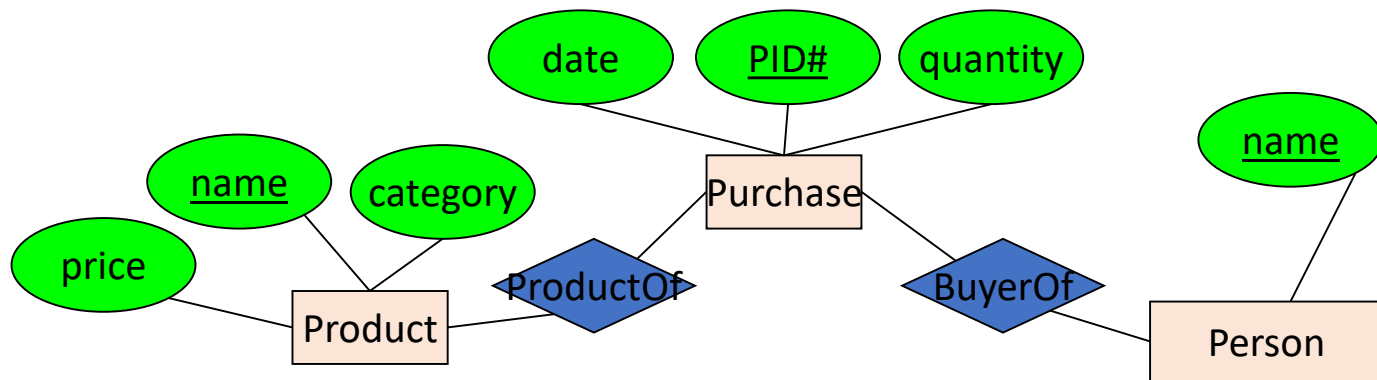


- **A:** A person can only buy a specific product once (on one date)

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!

# ACTIVITY: E/R Diagrams Pt. I

# Draw an E/R diagram for football

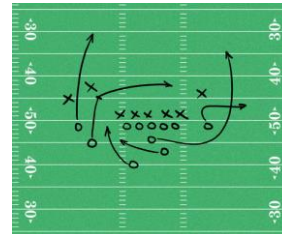
Use the following simplified model of a football season  
(concepts to include are underlined):



Teams play each other in Games. Each pair of teams can play each other multiple times



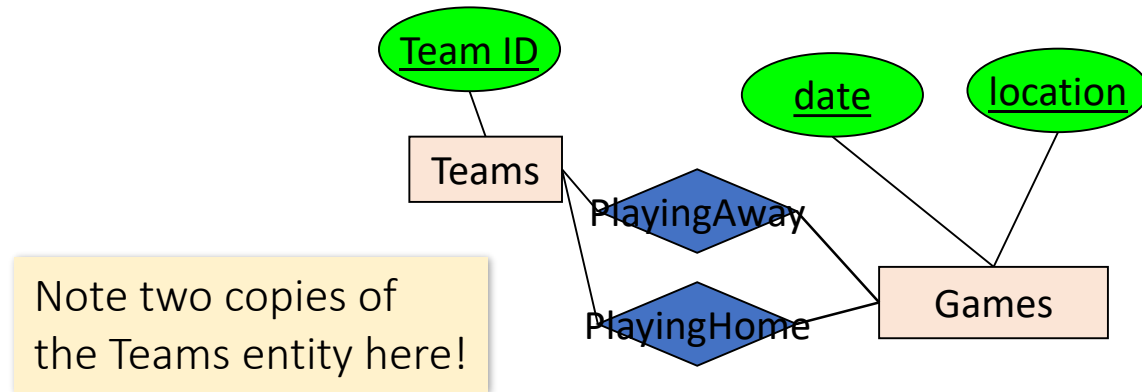
Players belong to Teams (assume no trades / changes).



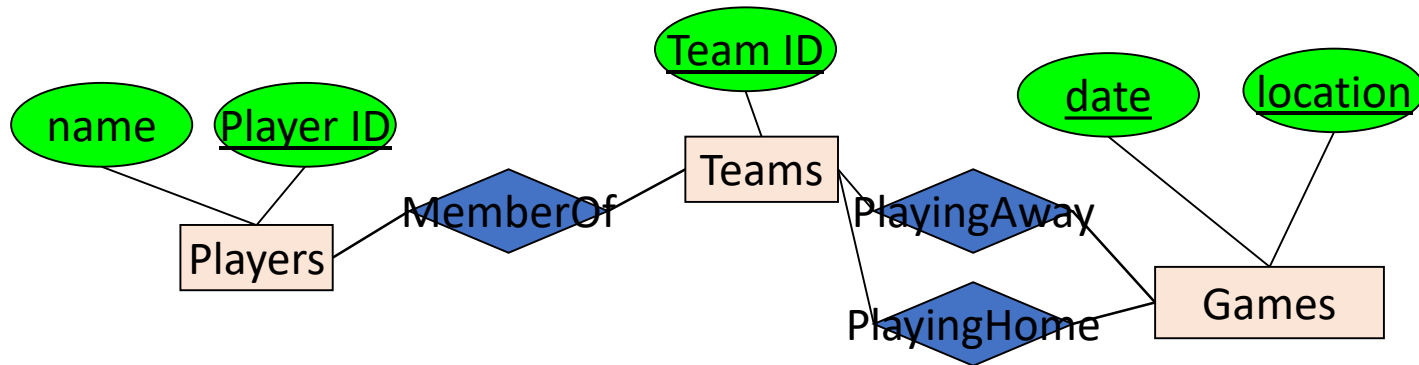
A Game is made up of Plays that result in a yardage gain/loss, and potentially a touchdown



A Play will contain either a Pass from one player to another, or a Run by one player

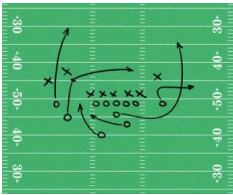
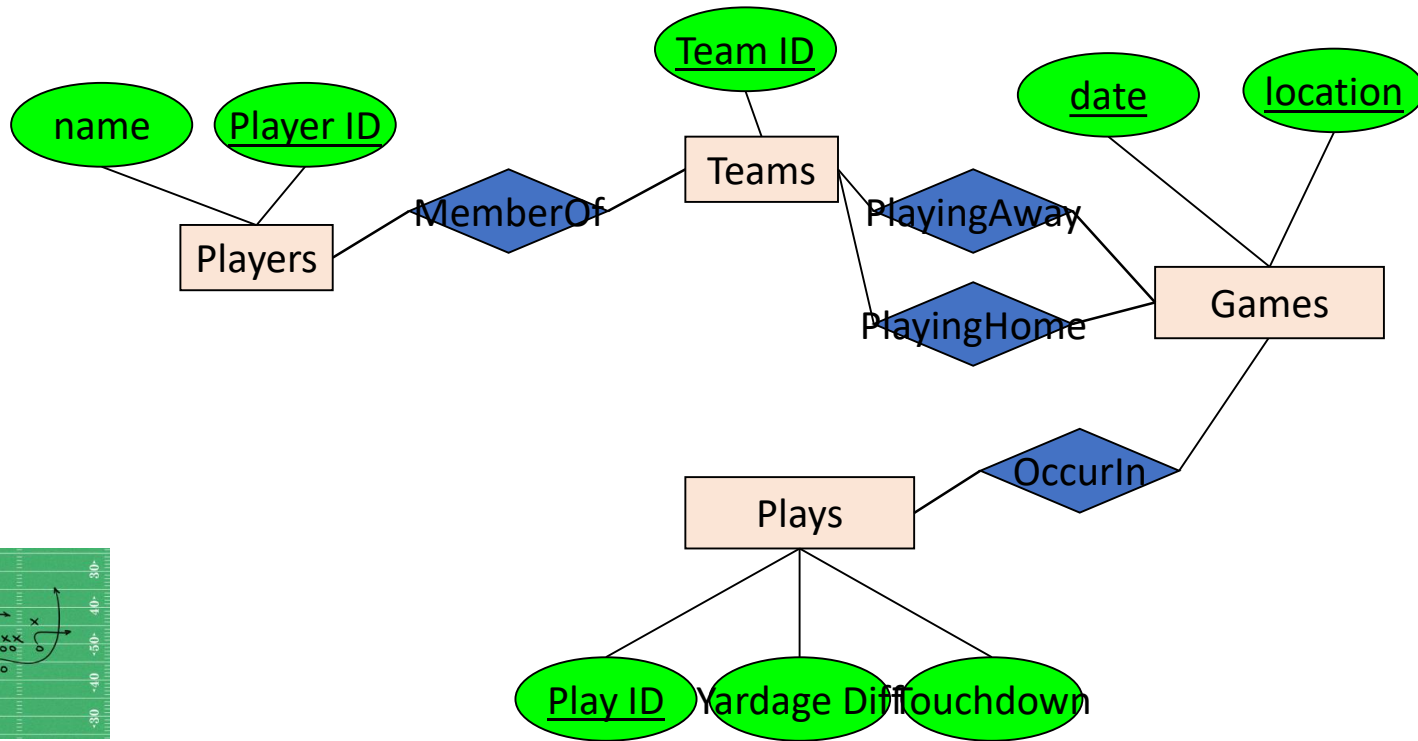


Teams play each other in Games.  
Each pair of teams can play each other multiple times

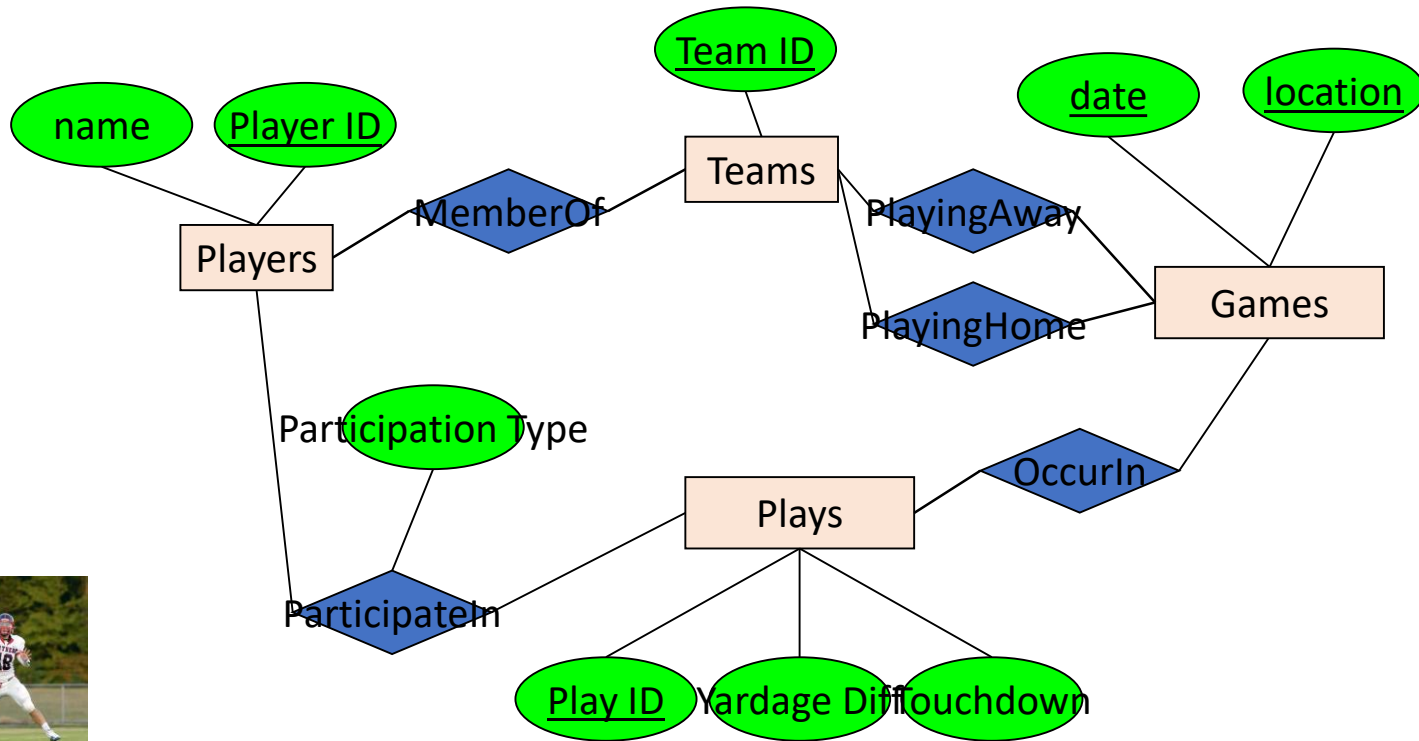


Players belong to  
Teams (assume no  
trades / changes)





A Game is made up of Plays that result in a yardage gain/loss, and potentially a touchdown



A Play will contain either a Pass from one player to another, or a Run by one player

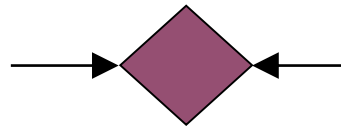
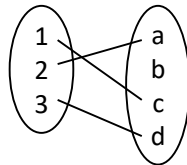
## 2. E/R Design Considerations

# What you will learn about in this section

1. Relationships cont'd: multiplicity, multi-way
2. Design considerations
3. Conversion to SQL
4. ACTIVITY: Crayon Time! Drawing E/R diagrams Pt. II

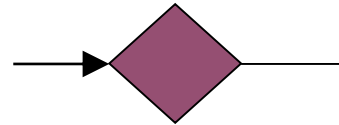
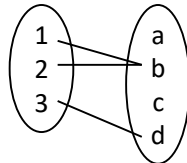
# Multiplicity of E/R Relationships

One-to-one:

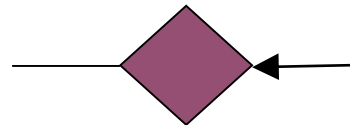
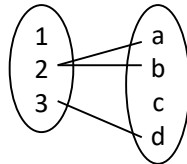


Indicated using  
arrows

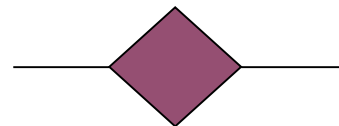
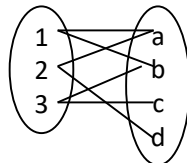
Many-to-one:



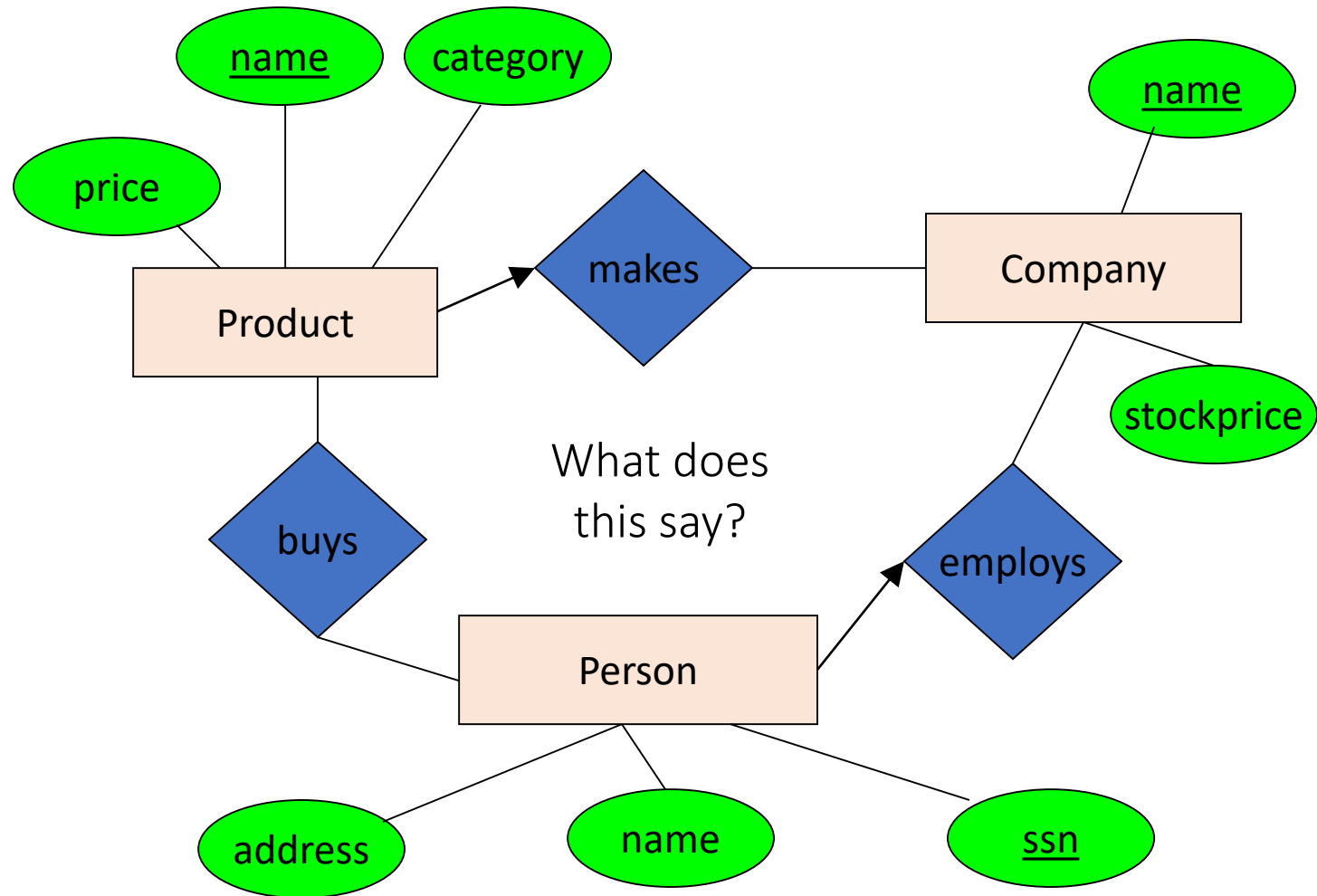
One-to-many:



Many-to-many:

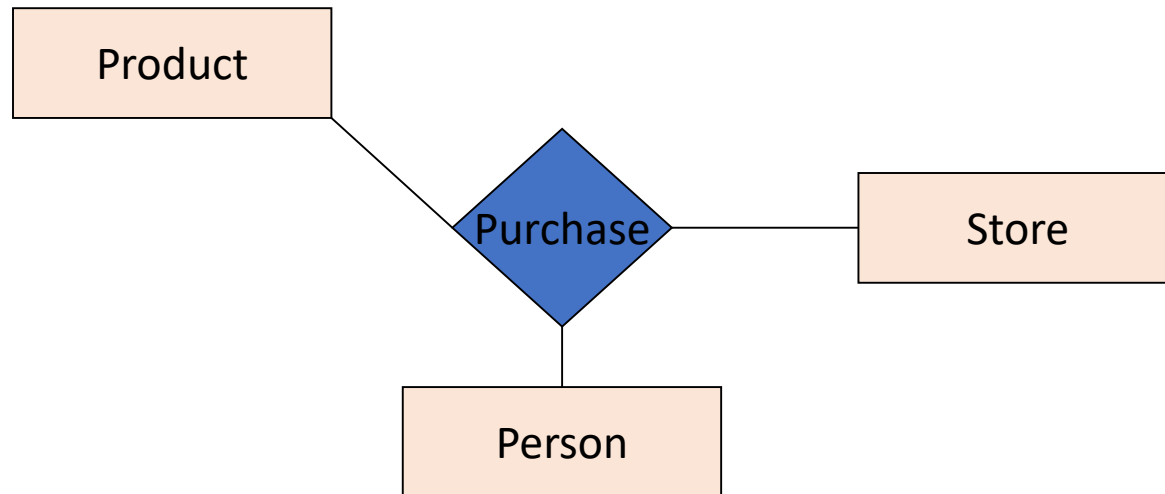


$X \rightarrow Y$  means  
there exists a  
function mapping  
from X to Y (*recall*  
*the definition of a*  
*function*)



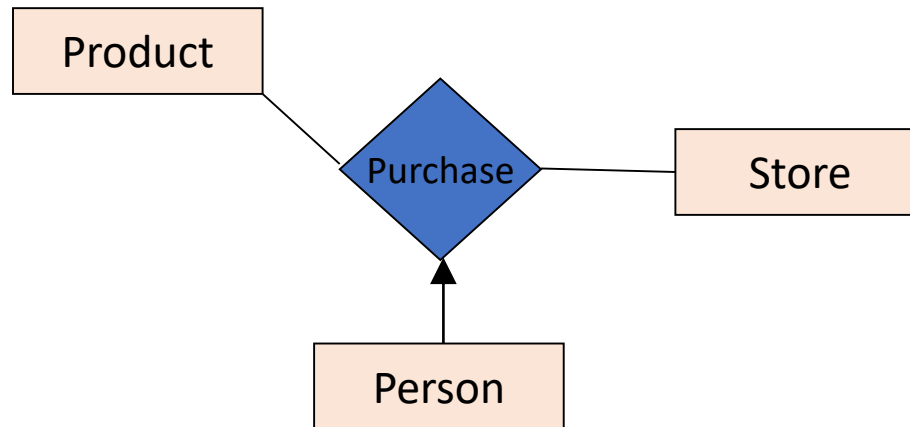
# Multi-way Relationships

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



# Arrows in Multiway Relationships

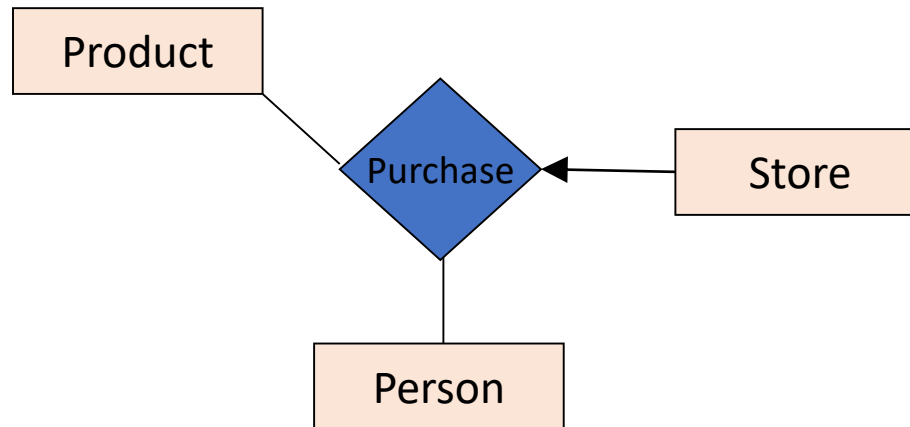
**Q:** What does the arrow mean ?





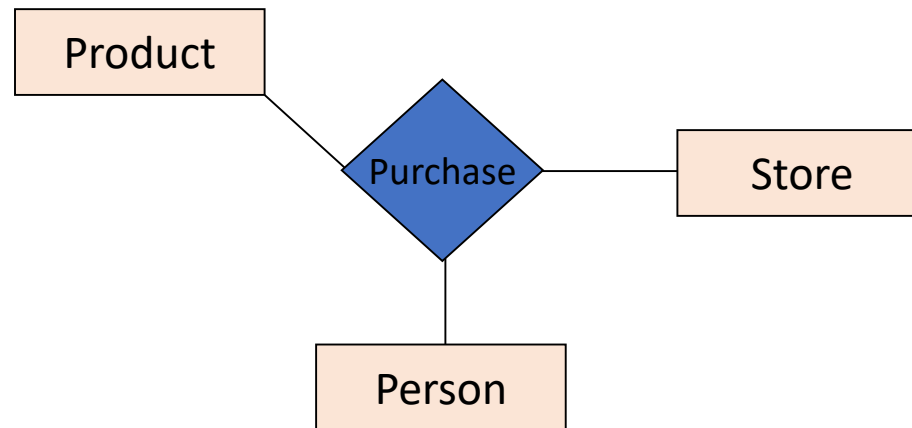
# Arrows in Multiway Relationships

**Q:** What does the arrow mean ?



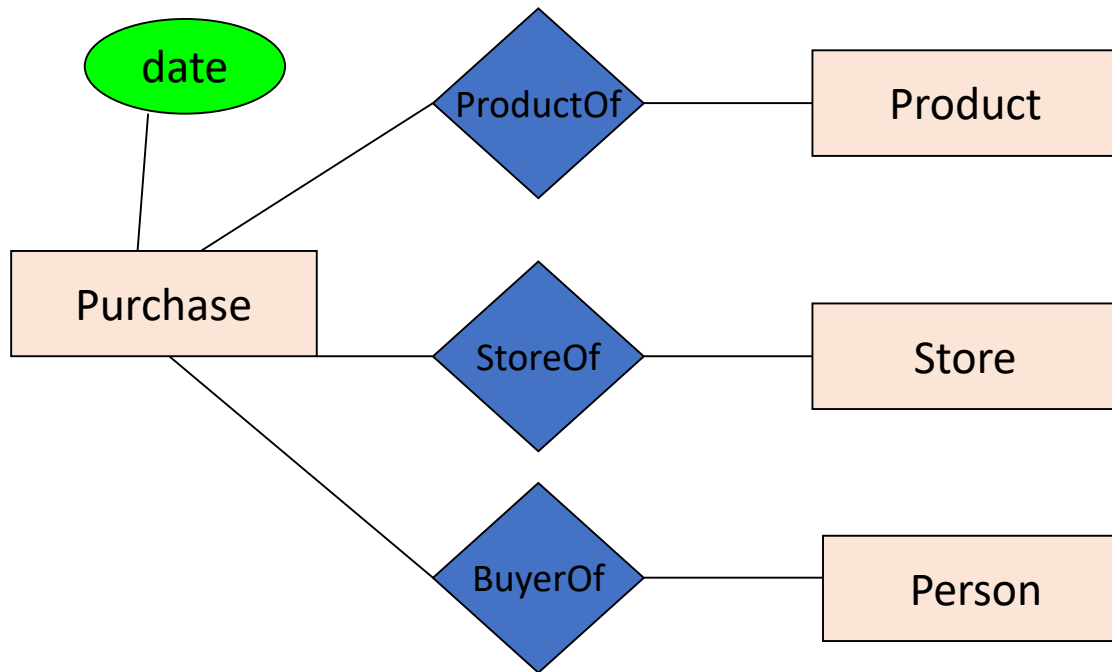
# Arrows in Multiway Relationships

**Q:** How do we say that every person shops in at most one store ?



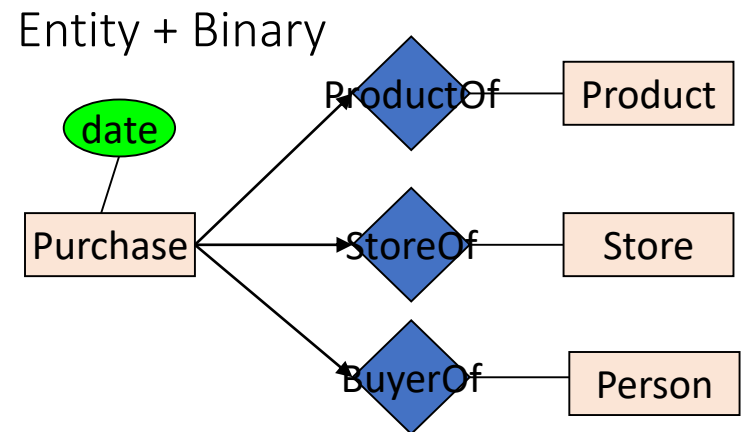
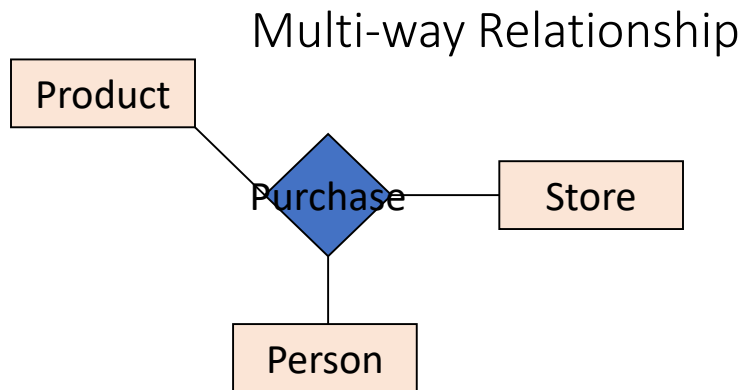
**A:** Cannot. This is the best approximation.  
(Why only approximation ?)

# Converting Multi-way Relationships to Binary



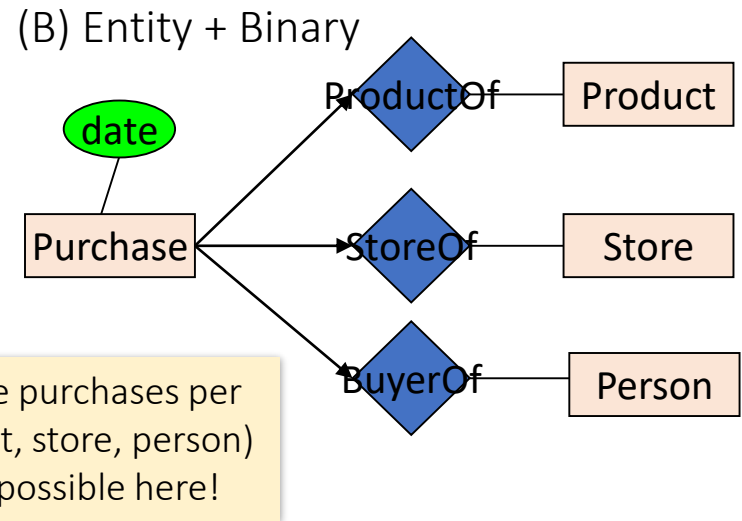
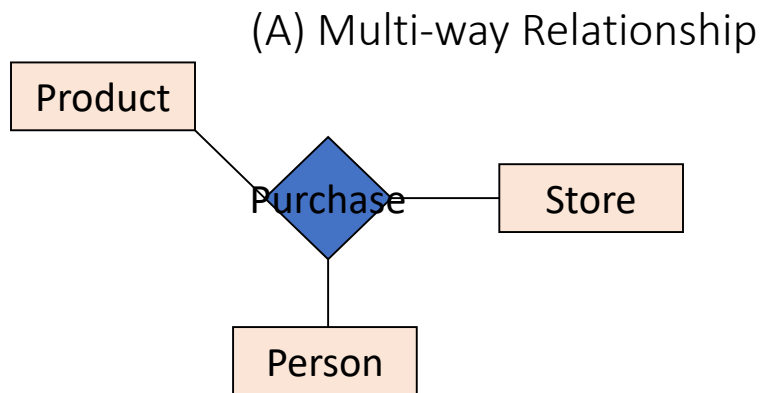
From what we had on previous slide to this - what did we do?

# Decision: Multi-way or New Entity + Binary



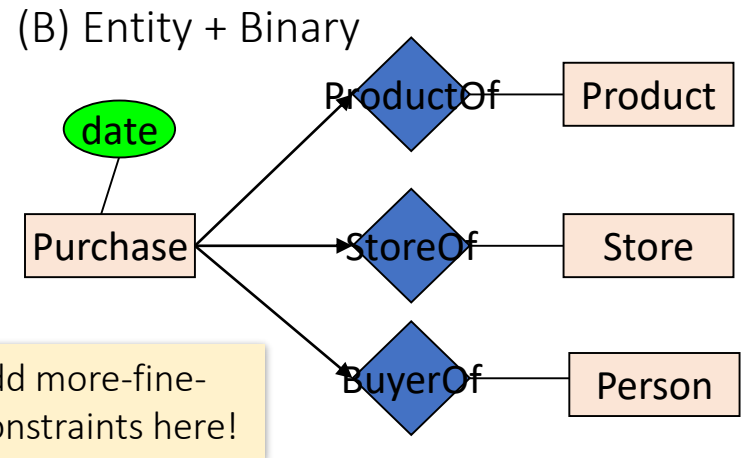
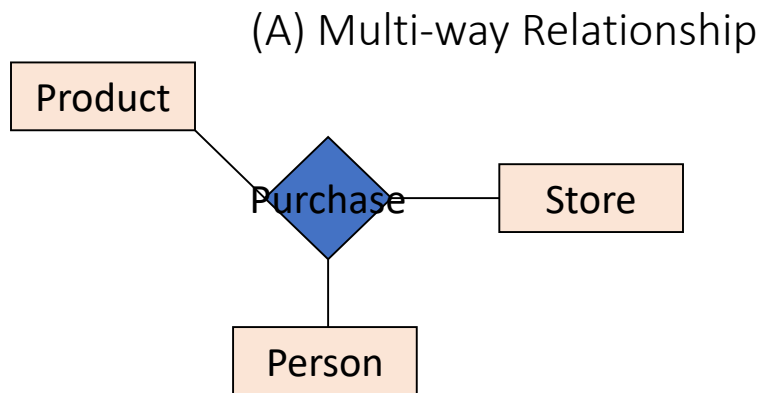
Should we use a single **multi-way relationship** or a *new entity with binary relations*?

# Decision: Multi-way or New Entity + Binary



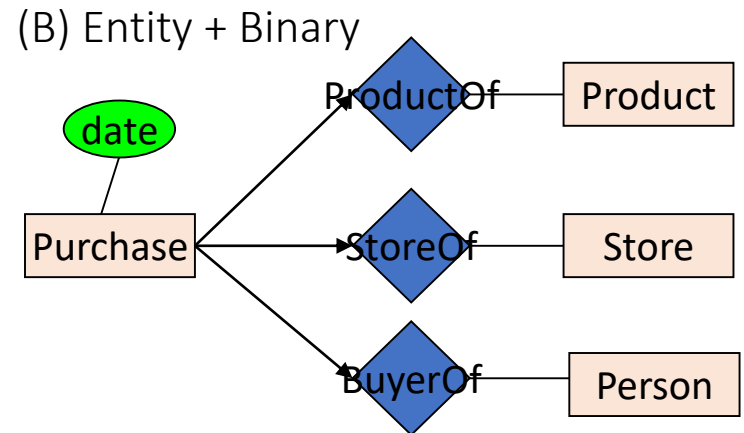
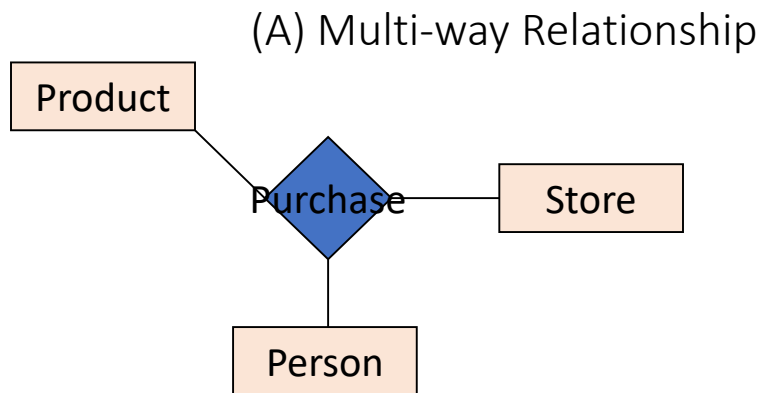
- *Covered earlier:* (B) is useful if we want to have multiple instances of the “relationship” per entity combination

# Decision: Multi-way or New Entity + Binary



- (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”

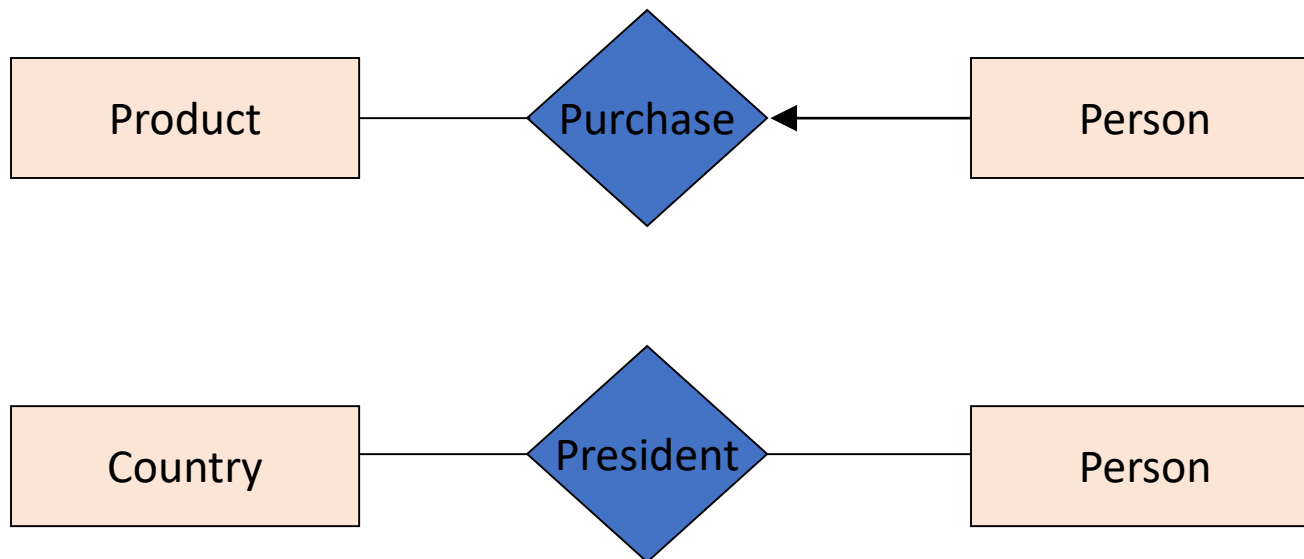
# Decision: Multi-way or New Entity + Binary



- (A) is useful when a relationship really is between multiple entities
  - *Ex: A three-party legal contract*

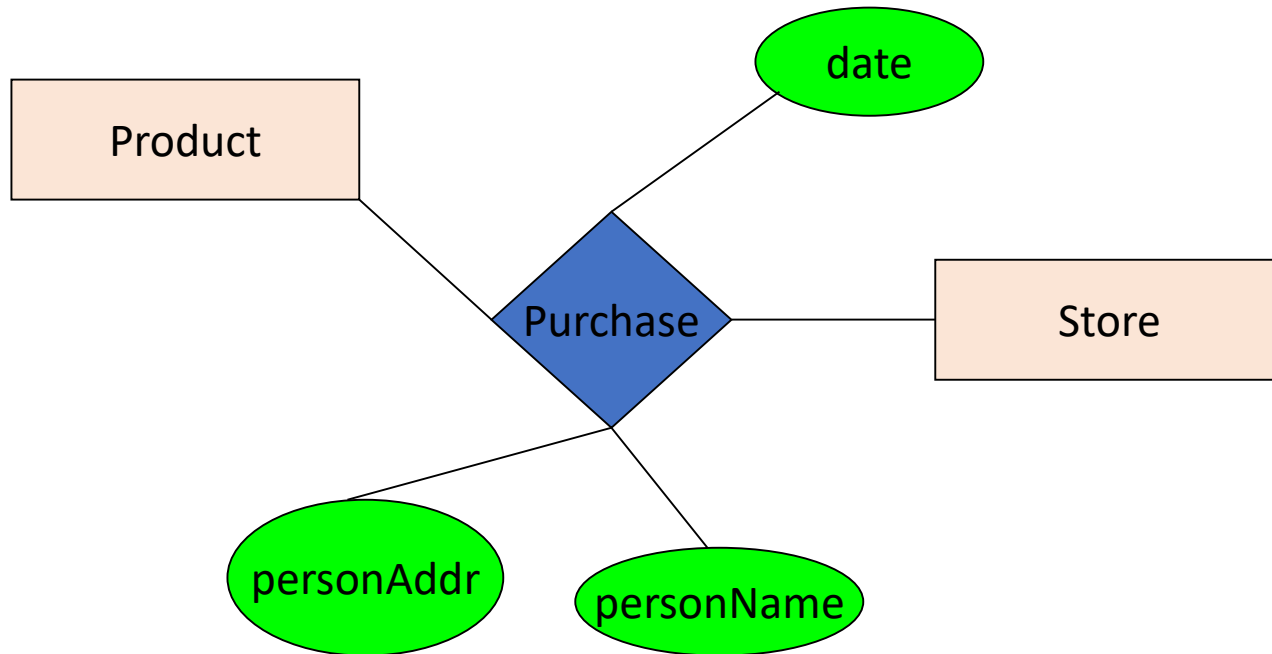
### 3. Design Principles

What's wrong with these examples?

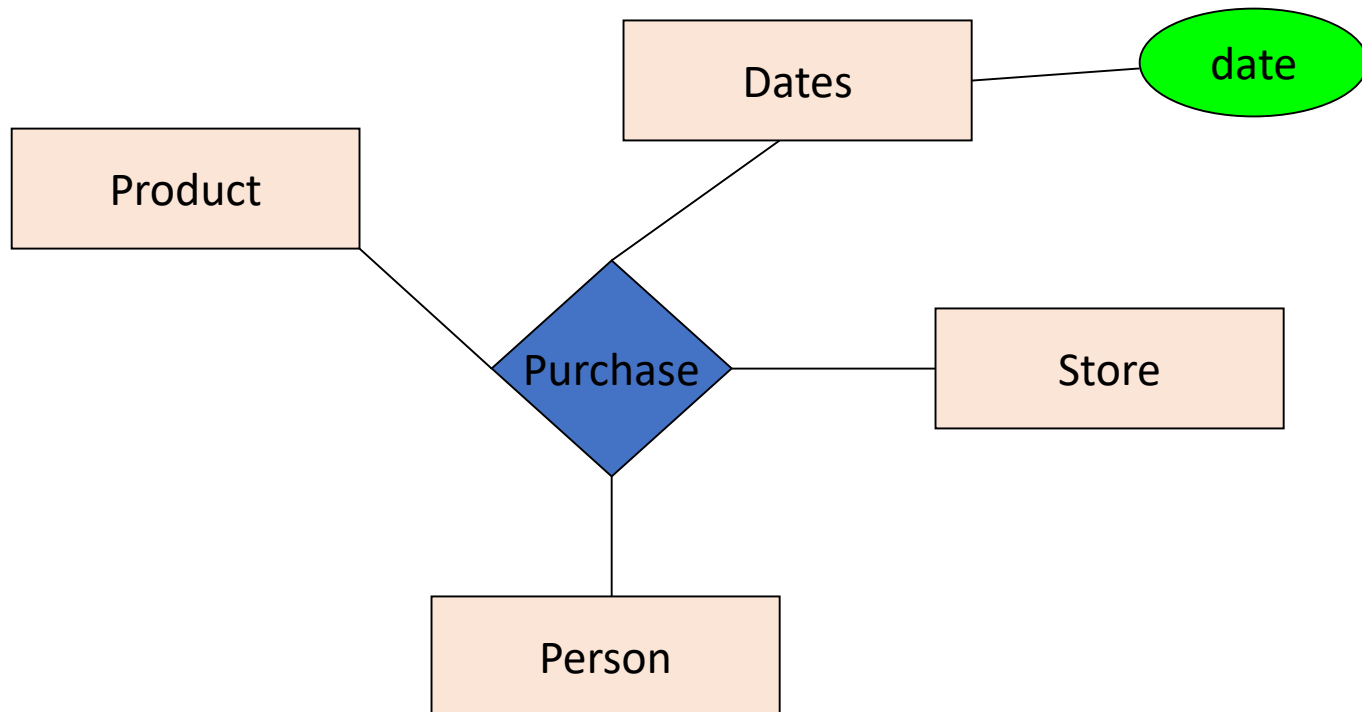




# Design Principles: What's Wrong?

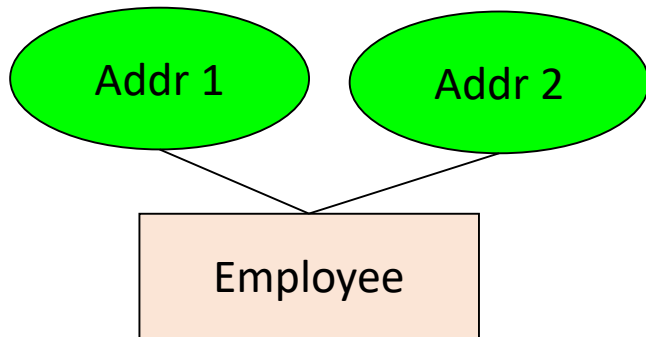


# Design Principles: What's Wrong?

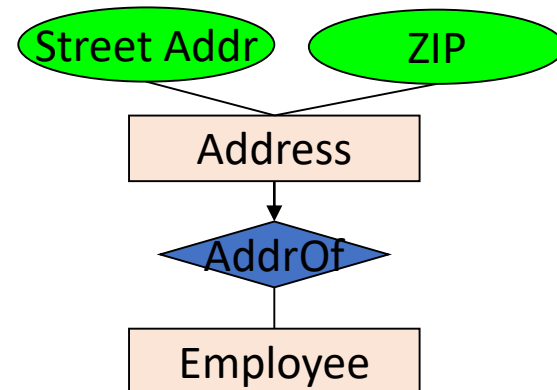


# Examples: Entity vs. Attribute

Should address (A)  
be an attribute?

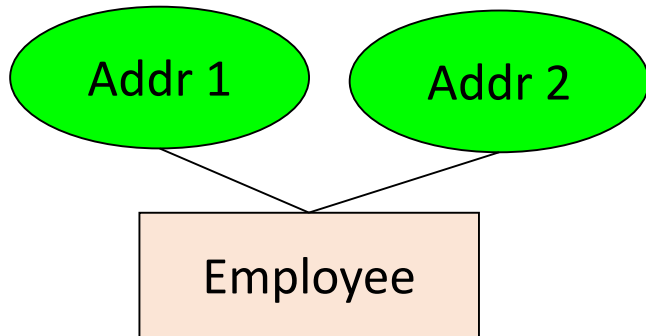


Or (B) be an entity?



# Examples: Entity vs. Attribute

Should address (A)  
be an attribute?

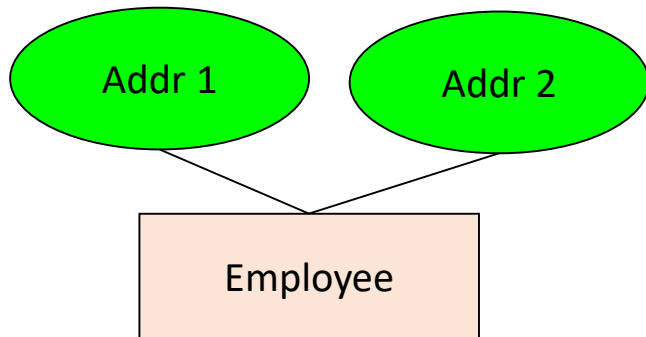


How do we handle employees  
with multiple addresses here?

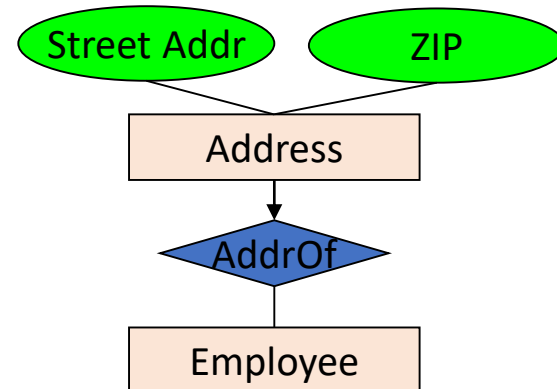
How do we handle addresses  
where internal structure of the  
address (e.g. zip code, state) is  
useful?

# 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

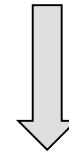
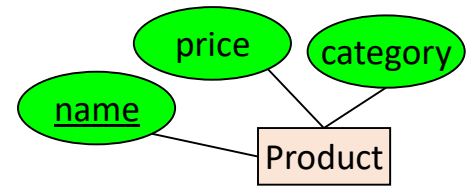
# From E/R Diagrams to Relational Schema

- Key concept:

Both ***Entity sets*** and ***Relationships*** become relations (tables in RDBMS)

# From E/R Diagrams to Relational Schema

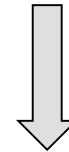
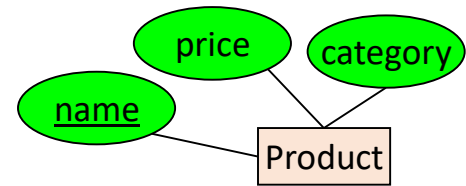
- An entity set becomes a relation (multiset of tuples / table)
  - Each tuple is one entity
  - Each tuple is composed of the entity's attributes, and has the same primary key



Product		
<u>name</u>	price	category
Gizmo1	99.99	Camera
Gizmo2	19.99	Edible

# From E/R Diagrams to Relational Schema

```
CREATE TABLE Product(  
  name CHAR(50) PRIMARY KEY,  
  price DOUBLE,  
  category VARCHAR(30)  
)
```



Product

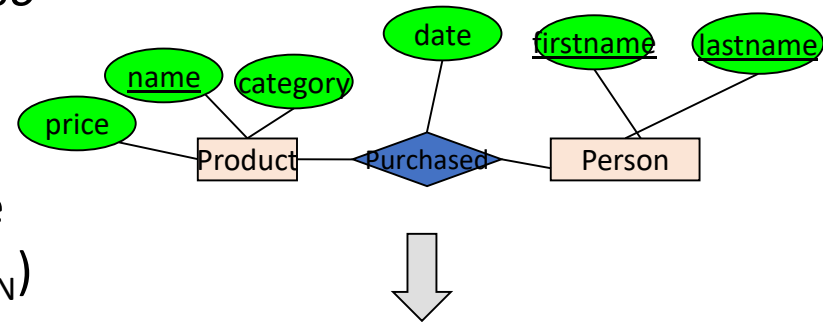
<u>name</u>	price	category
Gizmo1	99.99	Camera
Gizmo2	19.99	Edible



# From E/R Diagrams to Relational Schema

- A relation between entity sets  $A_1, \dots, A_N$  *also* becomes a multiset of tuples / a table

- Each row/tuple is one relation, i.e. one unique combination of entities ( $a_1, \dots, a_N$ )



- Each row/tuple is

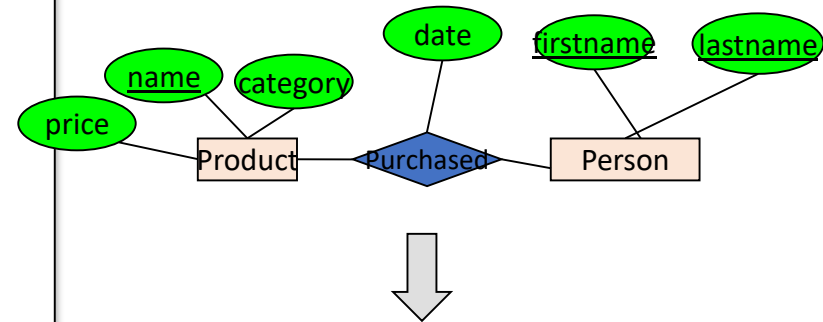
- composed of the **union of the entity sets' keys**
- has the entities' primary keys as foreign keys
- has the union of the entity sets' keys as primary key

Purchased

<u>name</u>	<u>firstname</u>	<u>lastname</u>	<u>date</u>
Gizmo1	Bob	Joe	01/01/15
Gizmo2	Joe	Bob	01/03/15
Gizmo1	JoeBob	Smith	01/05/15

# From E/R Diagrams to Relational Schema

```
CREATE TABLE Purchased(  
  name CHAR(50),  
  firstname CHAR(50),  
  lastname CHAR(50),  
  date DATE,  
  PRIMARY KEY (name, firstname, lastname),  
  FOREIGN KEY (name)  
    REFERENCES Product,  
  FOREIGN KEY (firstname, lastname)  
    REFERENCES Person  
)
```

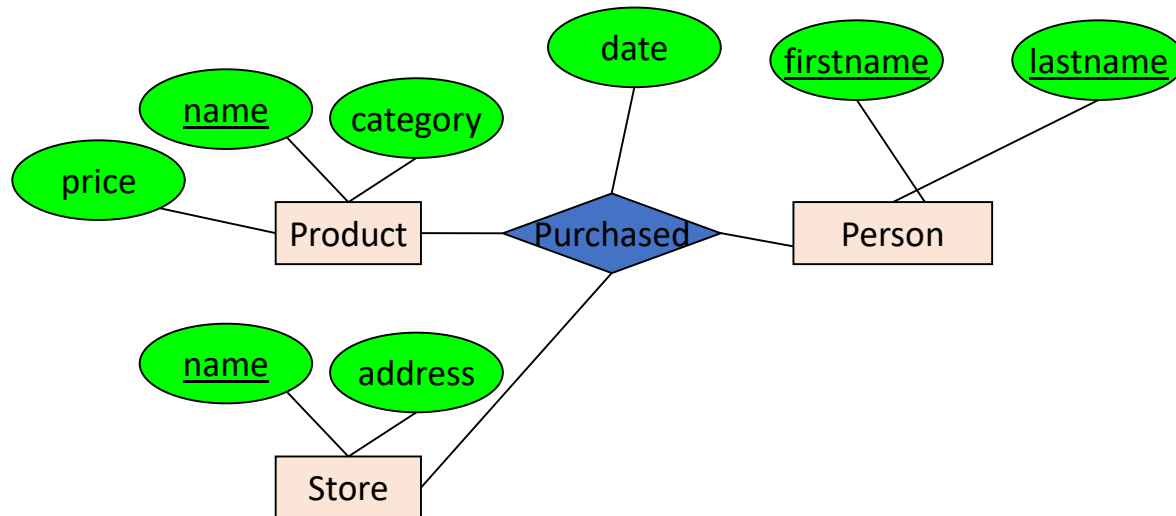


Purchased

<u>name</u>	<u>firstname</u>	<u>lastname</u>	date
Gizmo1	Bob	Joe	01/01/15
Gizmo2	Joe	Bob	01/03/15
Gizmo1	JoeBob	Smith	01/05/15

# From E/R Diagram to Relational Schema

How do we represent this as a relational schema?



# ACTIVITY: E/R Diagrams Pt. II

# Add arrows to your E/R diagram!

Also make sure to add (new concepts underlined):



A player can only belong to one team, a play can only be in one game, a pass/run..?



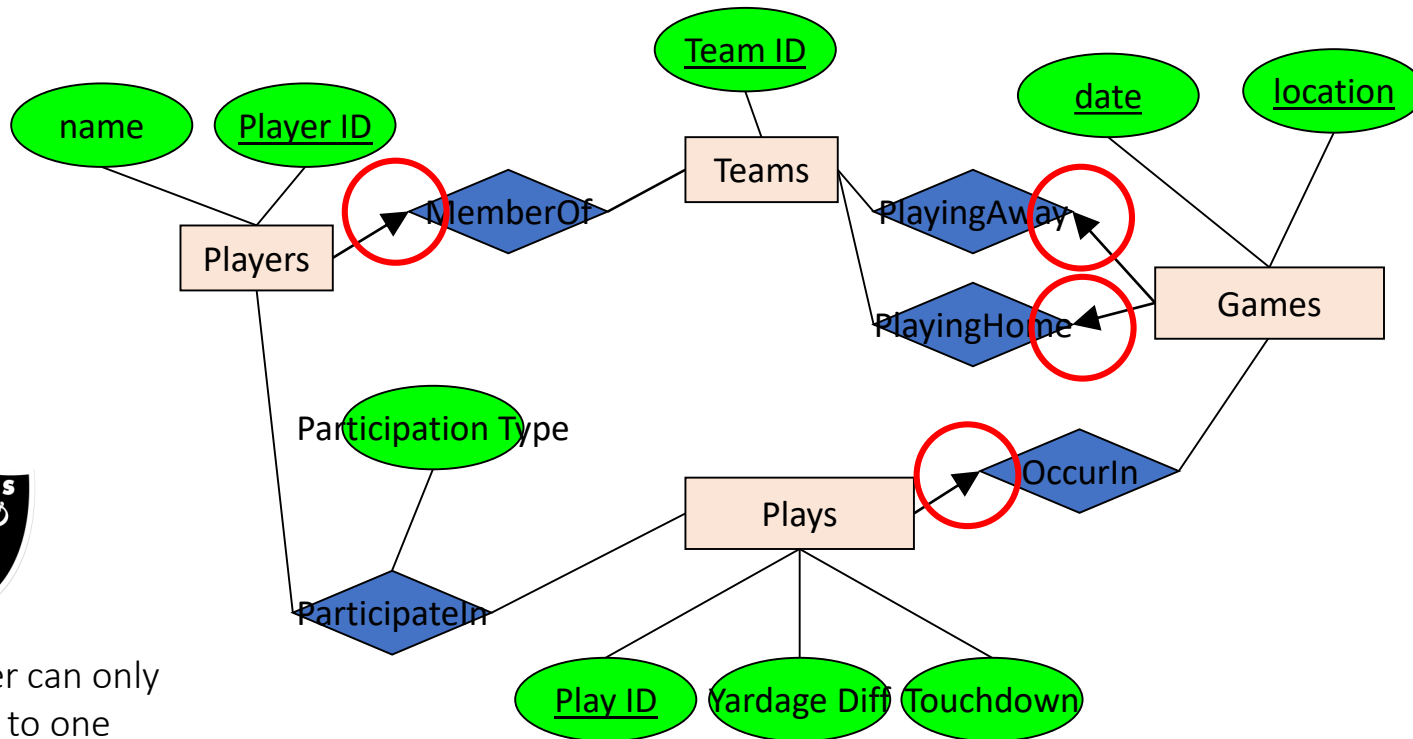
Players can achieve a Personal Record linked to a specific Game and Play



Players have a weight which changes in on vs. off-season



A player can only belong to one team, a play can only be in one game, a pass/run..?



### 3. Advanced E/R Concepts

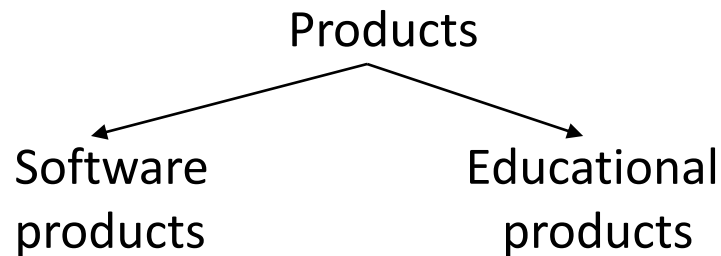
# What you will learn about in this section

1. Subclasses & connection to OO
2. Constraints
3. Weak entity sets
4. ACTIVITY: Crayon Time! Drawing E/R diagrams Pt. III



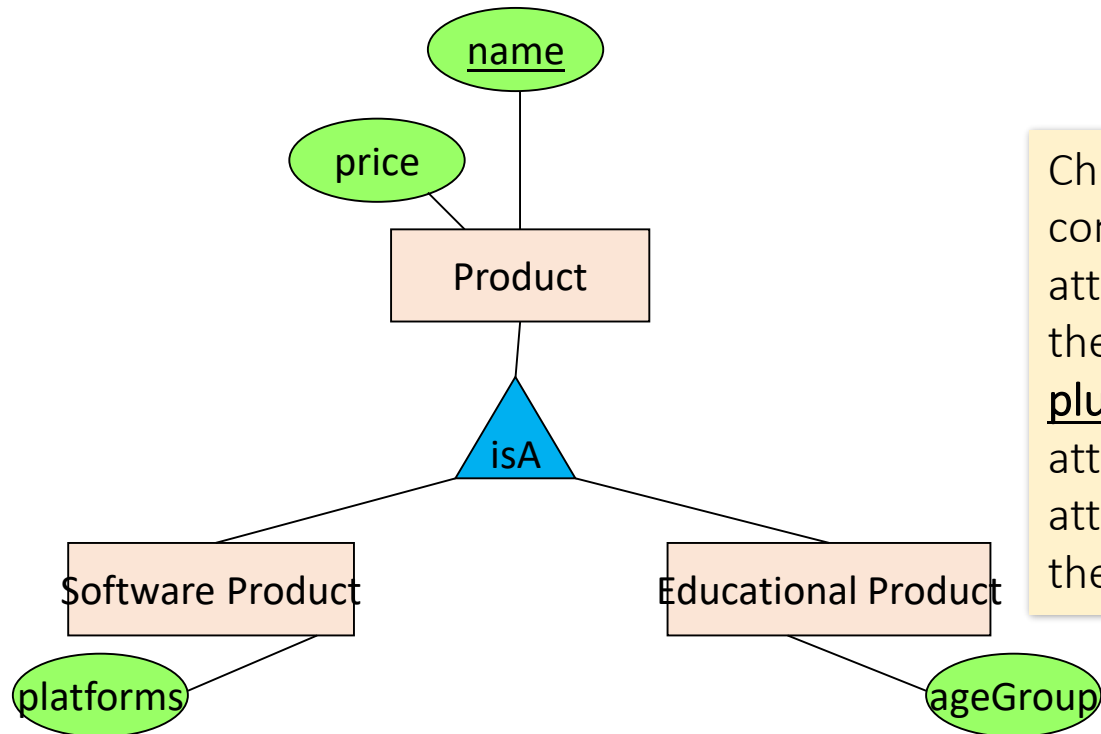
# Modeling Subclasses

- 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 E/R!

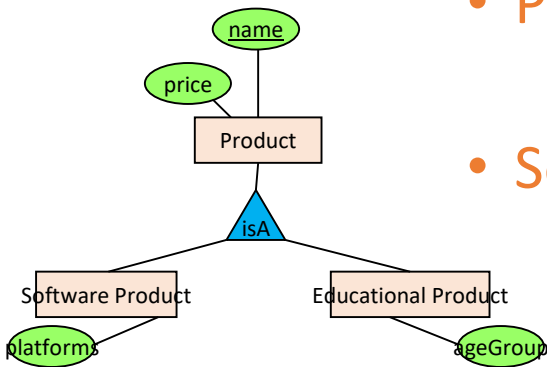
# Modeling Subclasses



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

# Understanding Subclasses

- Think in terms of records; ex:



- Product

- SoftwareProduct

- EducationalProduct

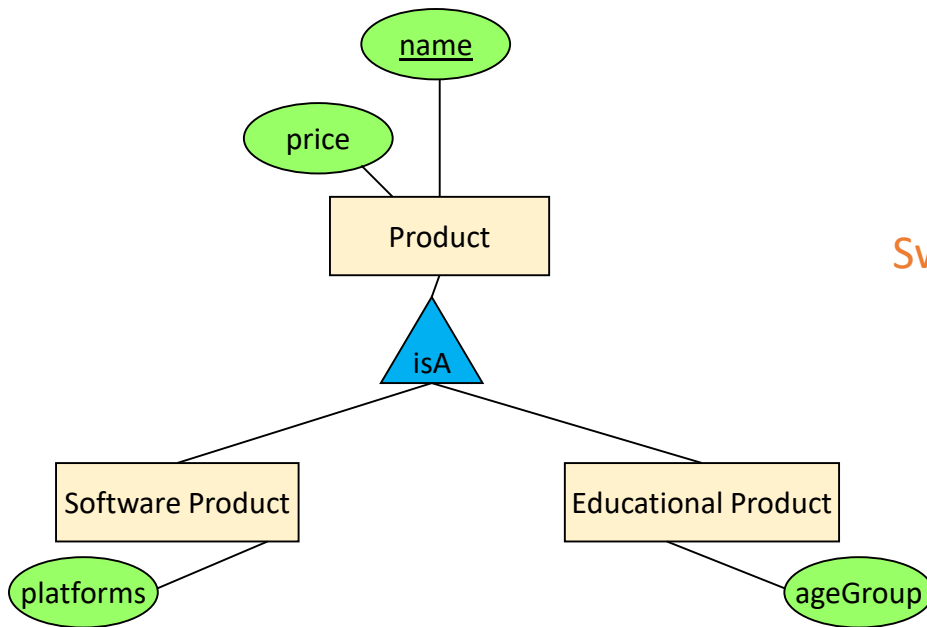
name
price

name
price
platforms

name
price
ageGroup

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

# Think like tables...



Product

<u>name</u>	price	category
Gizmo	99	gadget
Camera	49	photo
Toy	39	gadget

Sw.Product

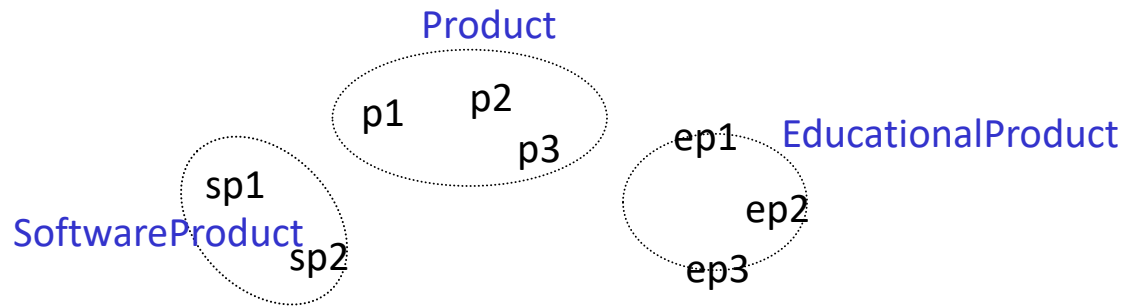
<u>name</u>	platforms
Gizmo	unix

Ed.Product

<u>name</u>	ageGroup
Gizmo	toddler
Toy	retired

# Difference between OO and E/R inheritance

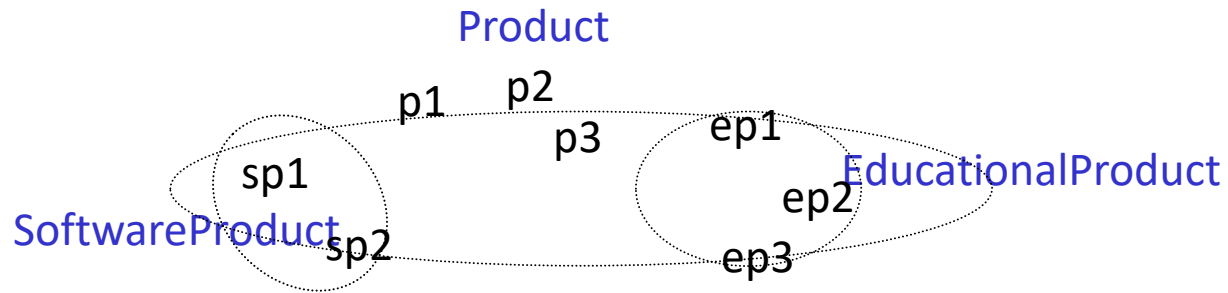
- OO: Classes are disjoint (same for Java, C++)



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

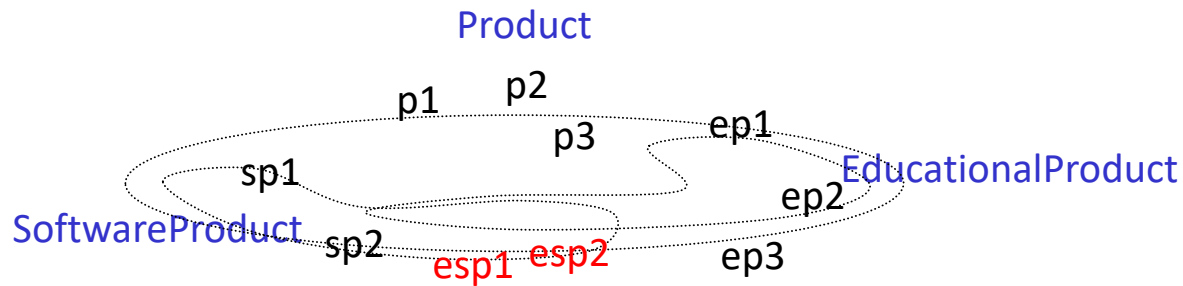
# Difference between OO and E/R inheritance

- E/R: entity sets overlap



# Difference between OO and E/R inheritance

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



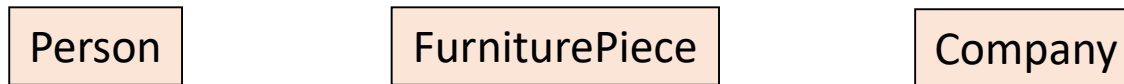
No need for multiple inheritance in E/R

# 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

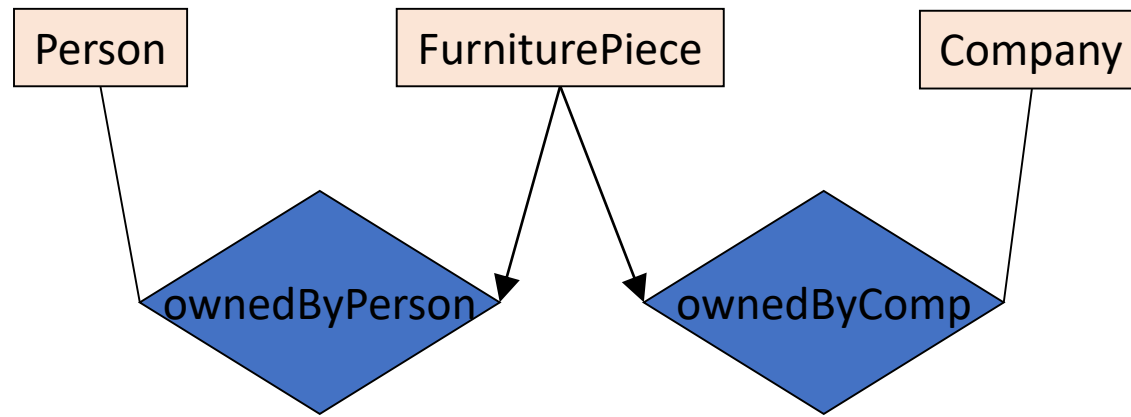


Suppose each piece of furniture is owned either by a person, or by a company. *How do we represent this?*

# Modeling Union Types with Subclasses

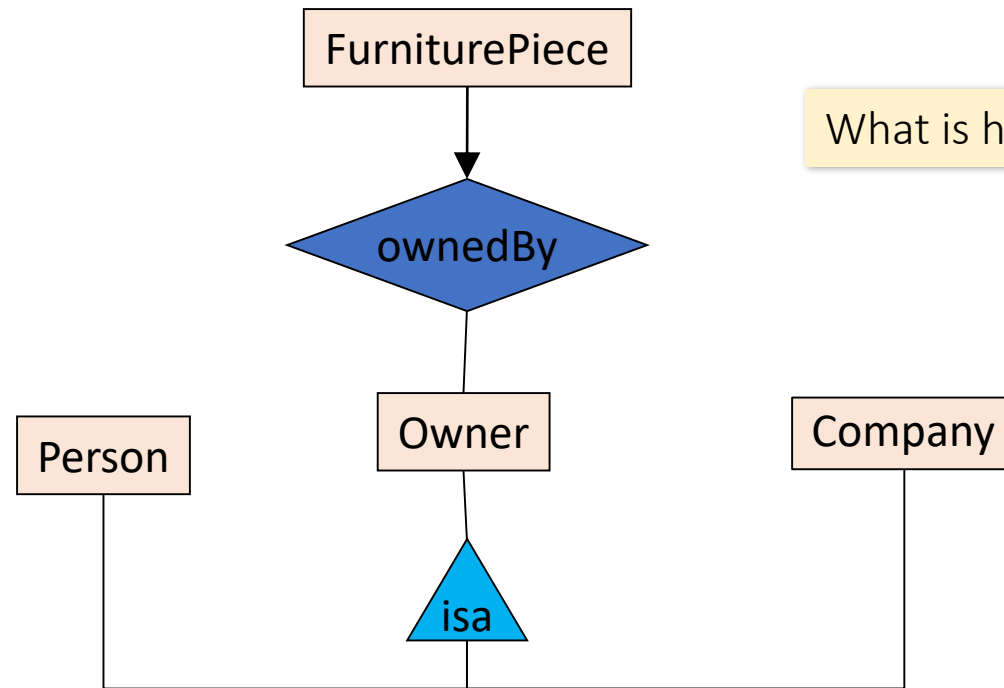
Say: each piece of furniture is owned either by a person, or by a company

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



# Modeling Union Types with Subclasses

Solution 2: better (though more laborious)



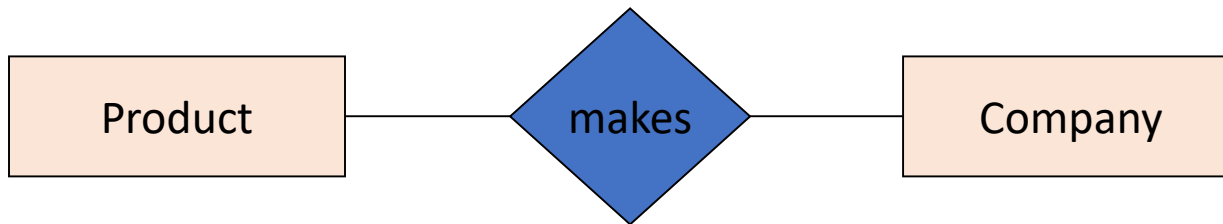
What is happening here?

# Constraints in E/R Diagrams

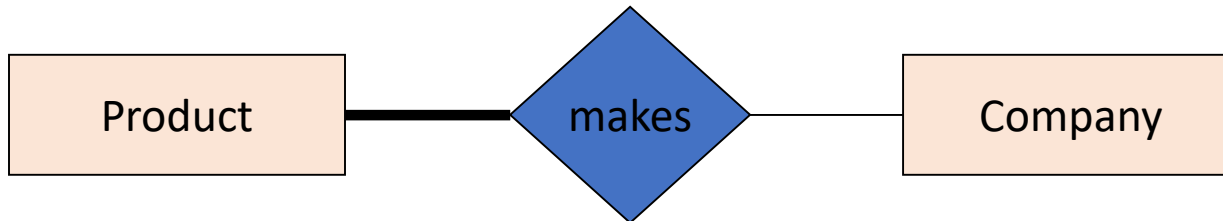
- Finding constraints is part of the E/R modeling process. Commonly used constraints are:
  - Keys: Implicit constraints on uniqueness of entities
    - *Ex: An SSN uniquely identifies a person*
  - Single-value constraints:
    - *Ex: a person can have only one father*
  - Referential integrity constraints: 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!

# Participation Constraints: Partial v. Total



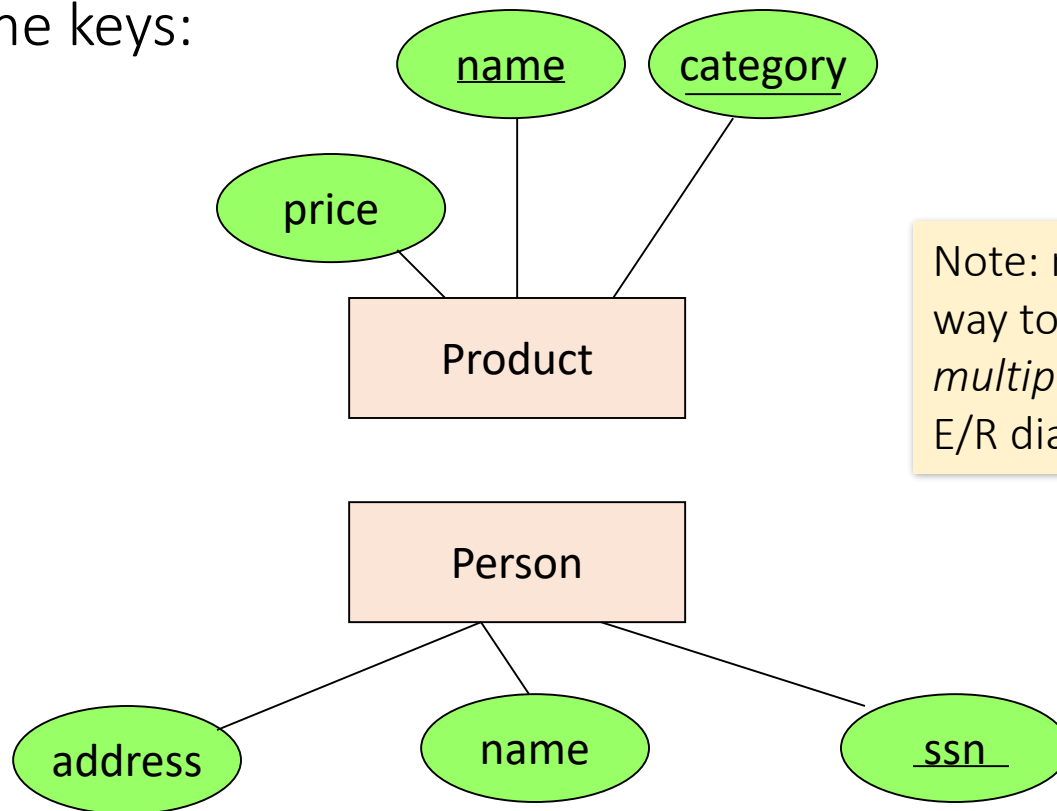
Are there products made by no company?  
Companies that don't make a product?



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

# Keys in E/R Diagrams

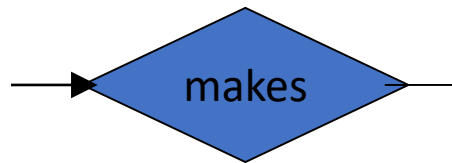
Underline keys:



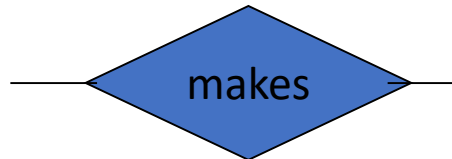
Note: no formal way to specify *multiple* keys in E/R diagrams...

# Single Value Constraints

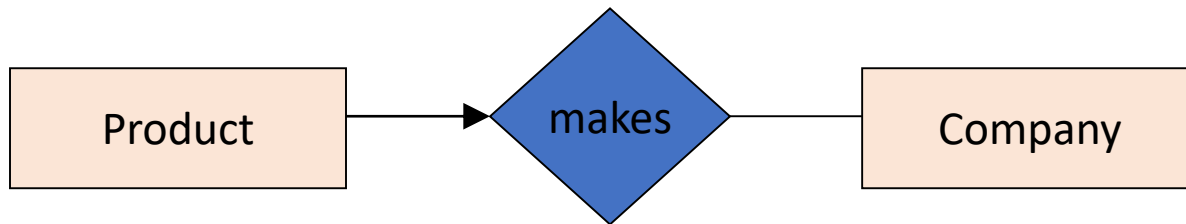
See previous section!



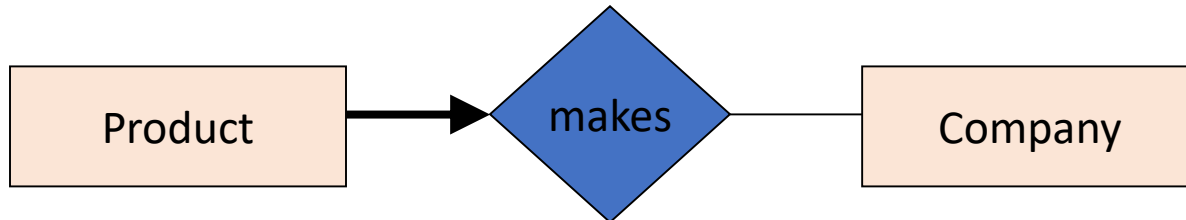
v. s.



# Referential Integrity Constraints



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

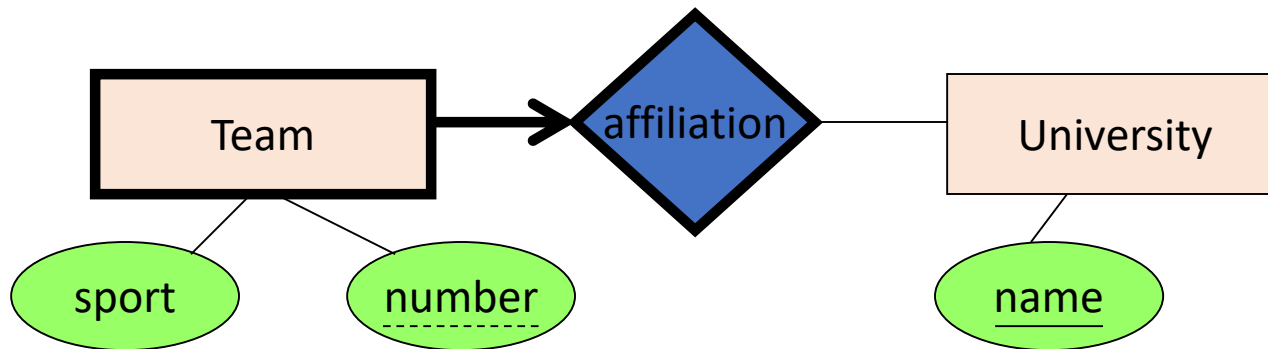


Each product made by exactly one company.



# Weak Entity Sets

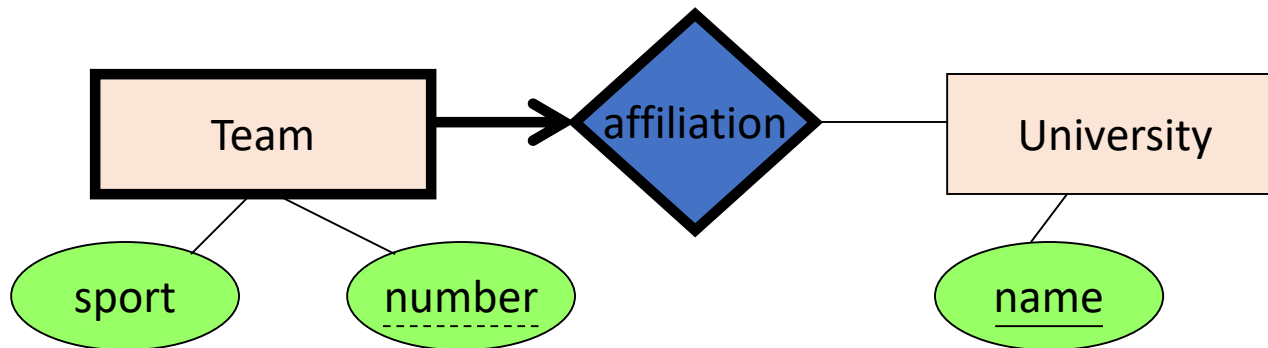
Entity sets are weak when their key comes from other classes to which they are related.



"Football team" v. "*The Stanford* Football team" (E.g., Berkeley has a football team too, sort of)

# Weak Entity Sets

Entity sets are weak when their key comes from other classes to which they are related.



- number is a partial key. (denote with dashed underline).
- University is called the identifying owner.
- Participation in affiliation must be total. Why?

# E/R 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