

# Introduction to Database Systems

## CSE 444

### Lecture 5: E/R Diagrams

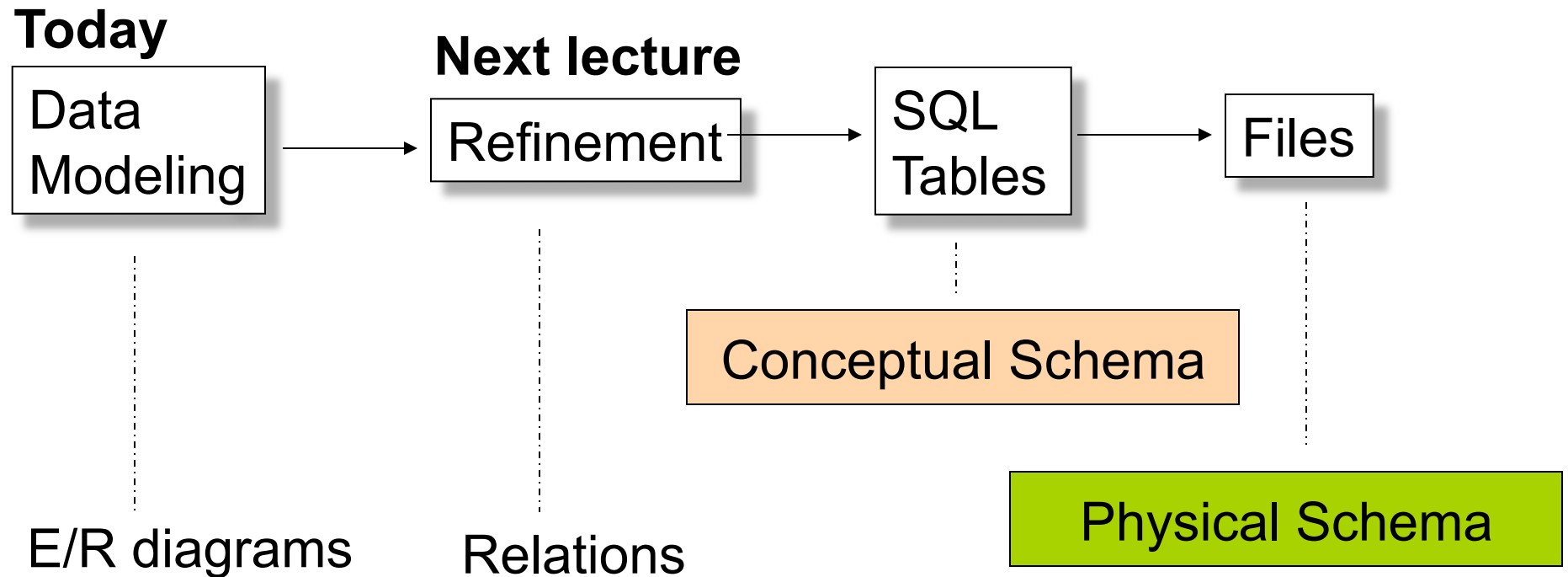
# Outline

- E/R diagrams
  - Sec. 4.1- 4.4 [Old edition: Chapter 2]
- From E/R diagrams to relations
  - Sec. 4.5 and 4.6 [Old edition: Sec. 3.2 and 3.3]

# Database Design

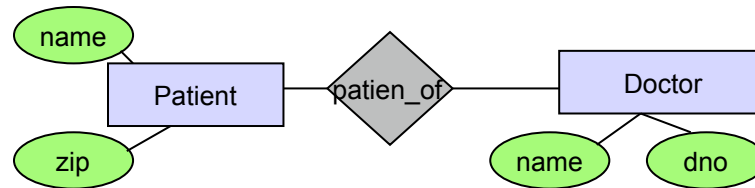
- Why do we need it?
  - Need a way to model real world entities in terms of relations
  - Not easy to go from real-world entities to a database schema
- Consider issues such as:
  - What entities to model
  - How entities are related
  - What constraints exist in the domain
  - How to achieve *good* designs
- Several formalisms exists
  - We discuss E/R diagrams

# Database Design Process

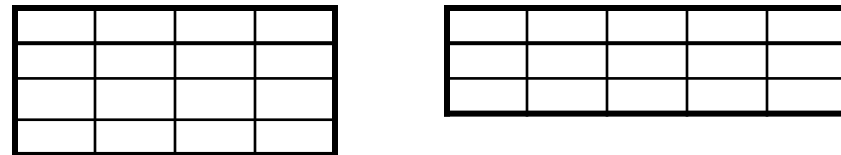


# Conceptual Schema Design

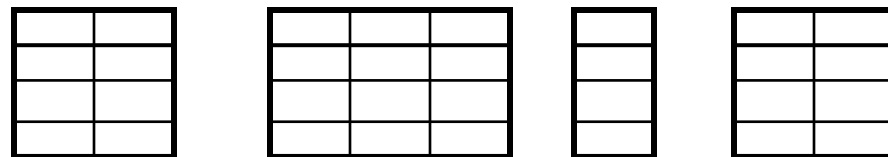
Conceptual Model:



Relational Model:  
plus FD's  
(FD = functional dependency)



Normalization:  
Eliminates anomalies



# Entity / Relationship Diagrams

Objects       $\longrightarrow$       entities  
Classes      $\longrightarrow$       entity sets

Product

This is an  
entity set

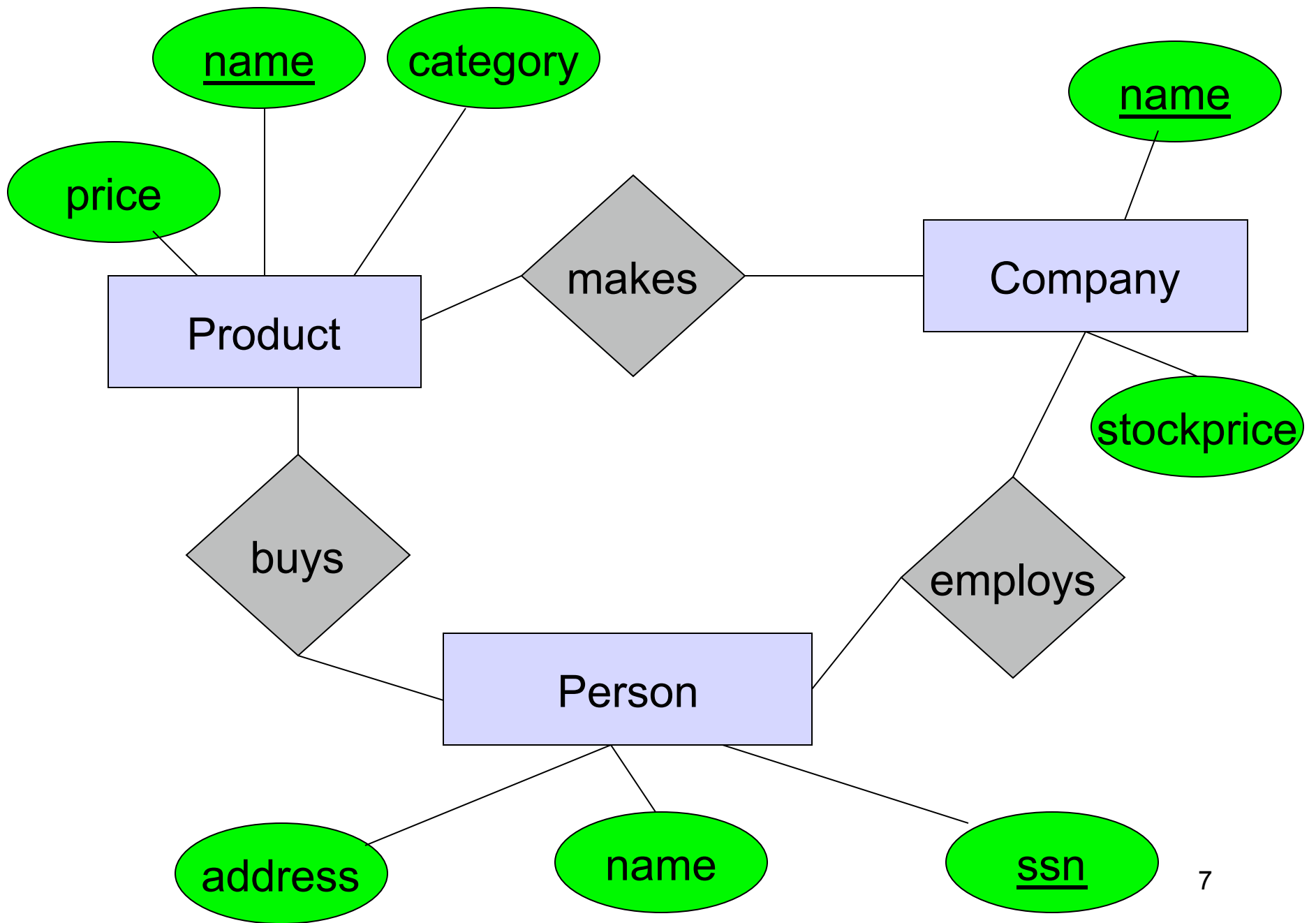
Attributes are like in ODL  
(ODL = Object Definition Language)

address

Relationships: like in ODL except

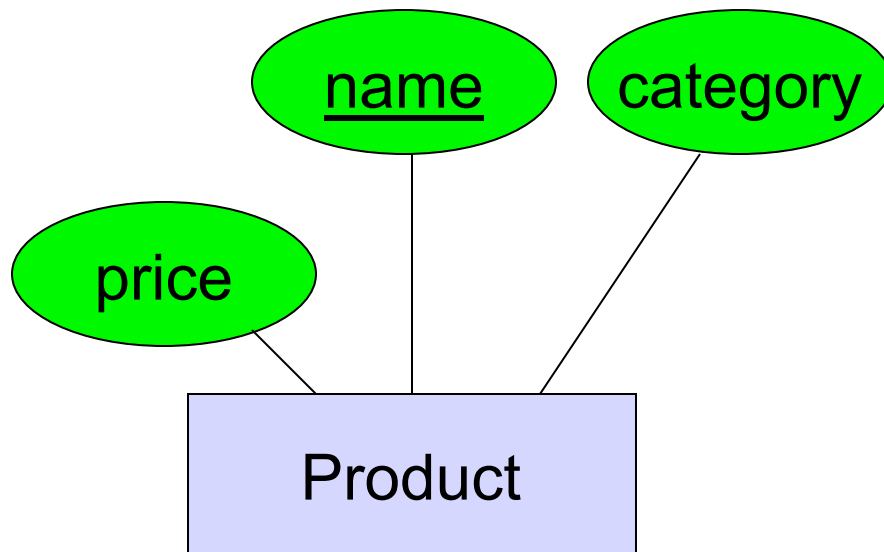
buys

- first class citizens (not associated with classes)
- not necessarily binary



# Keys in E/R Diagrams

- Every entity set must have a key

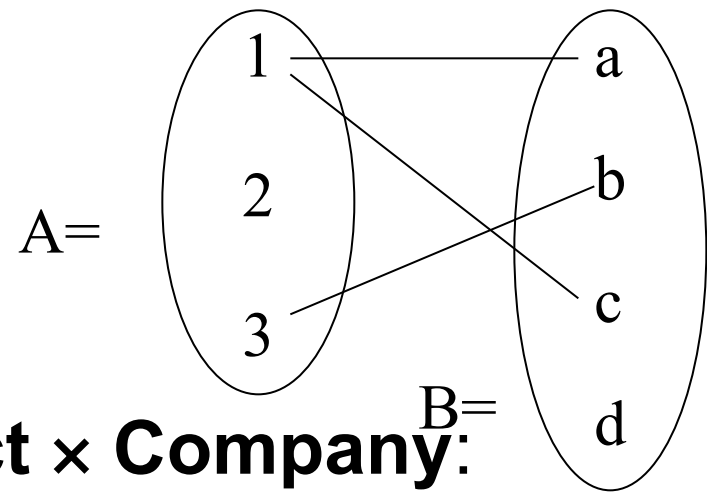




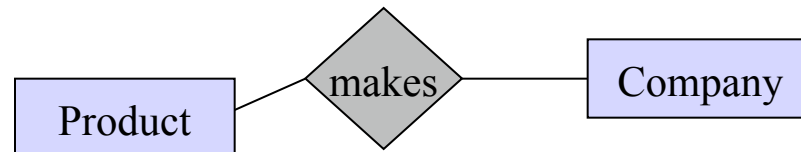
# What is a Relation ?

- A mathematical definition:
  - if  $A, B$  are sets, then a relation  $R$  is a subset of  $A \times B$

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

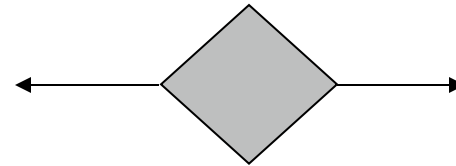
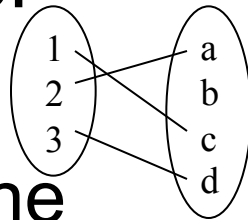


- **makes** is a subset of **Product**  $\times$  **Company**:

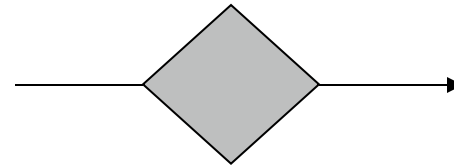
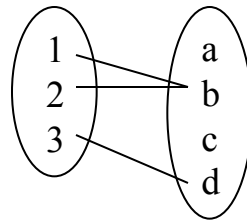


# Multiplicity of E/R Relations

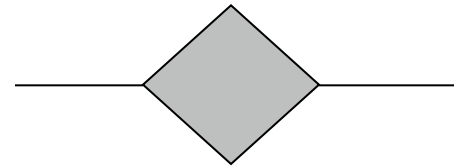
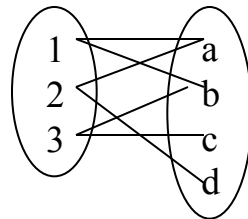
- one-one:

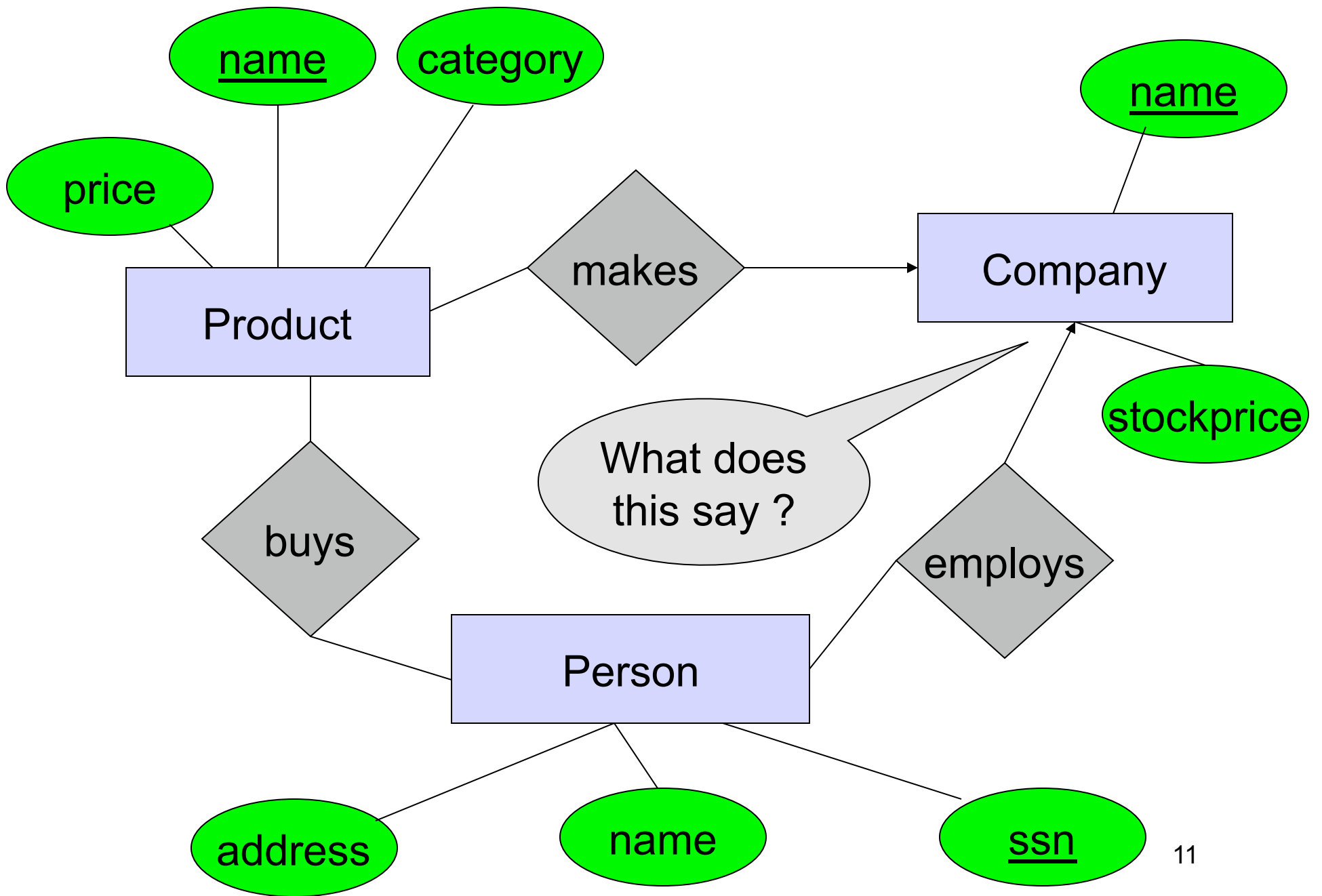


- many-one



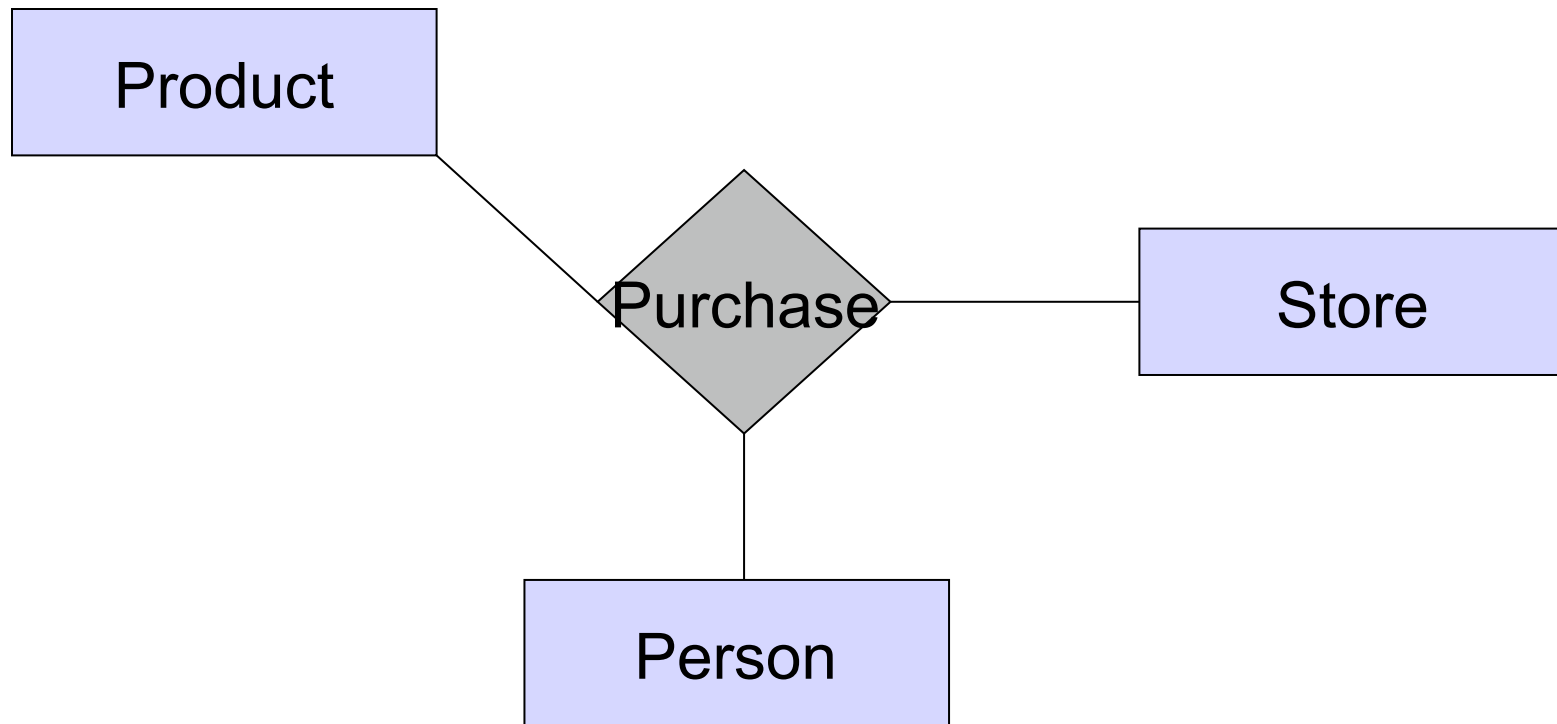
- many-many





# Multi-way Relationships

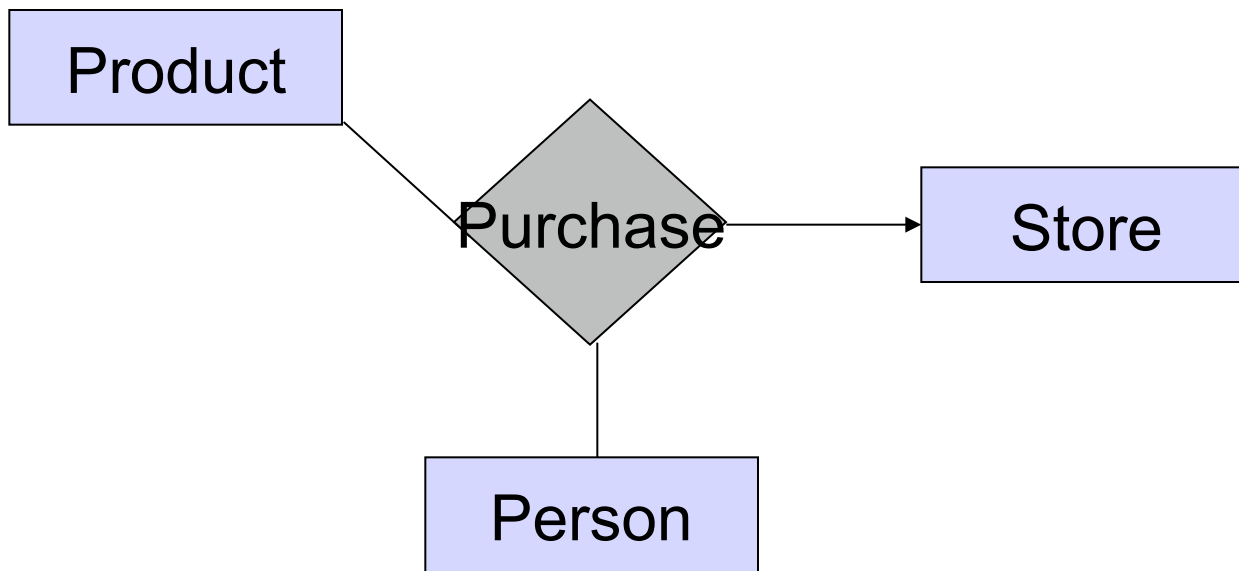
How do we model a purchase relationship between buyers, products and stores?



Can still model as a mathematical set (how ?)

# Arrows in Multiway Relationships

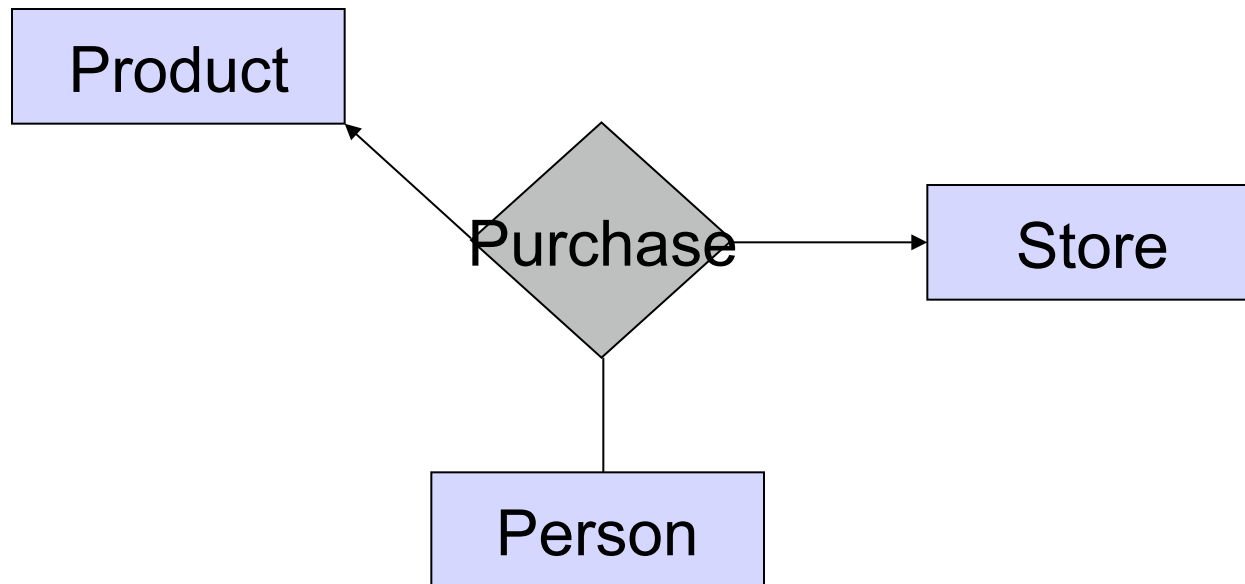
**Q:** What does the arrow mean ?



**A:** A given person buys a given product from at most one store

# Arrows in Multiway Relationships

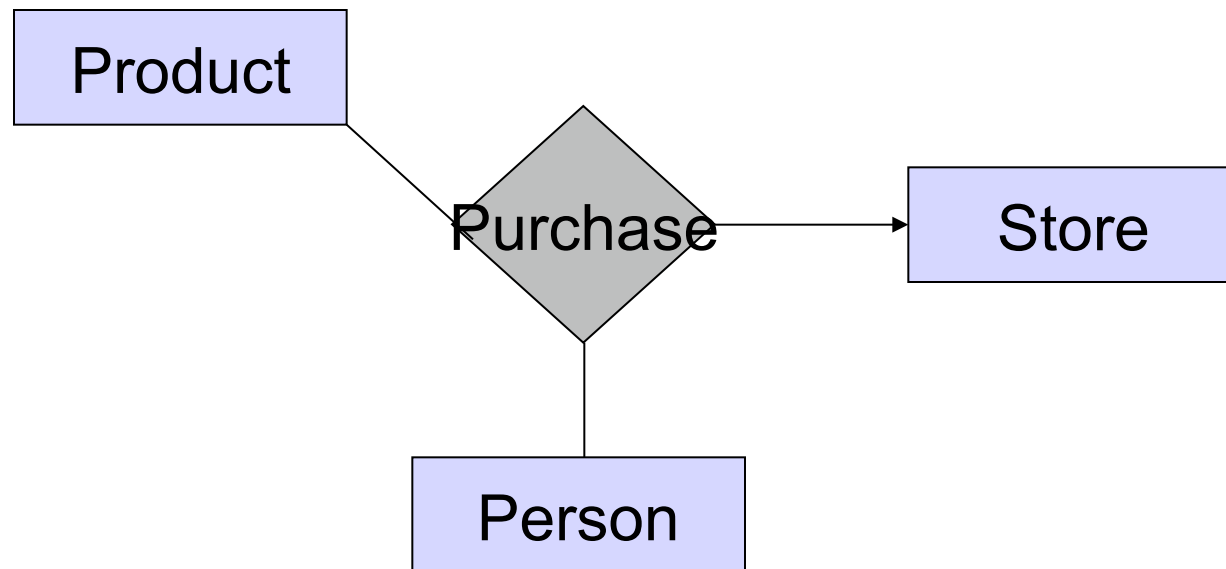
**Q:** What does the arrow mean ?



**A:** A given person buys a given product from at most one store  
AND every store sells to every person at most one product

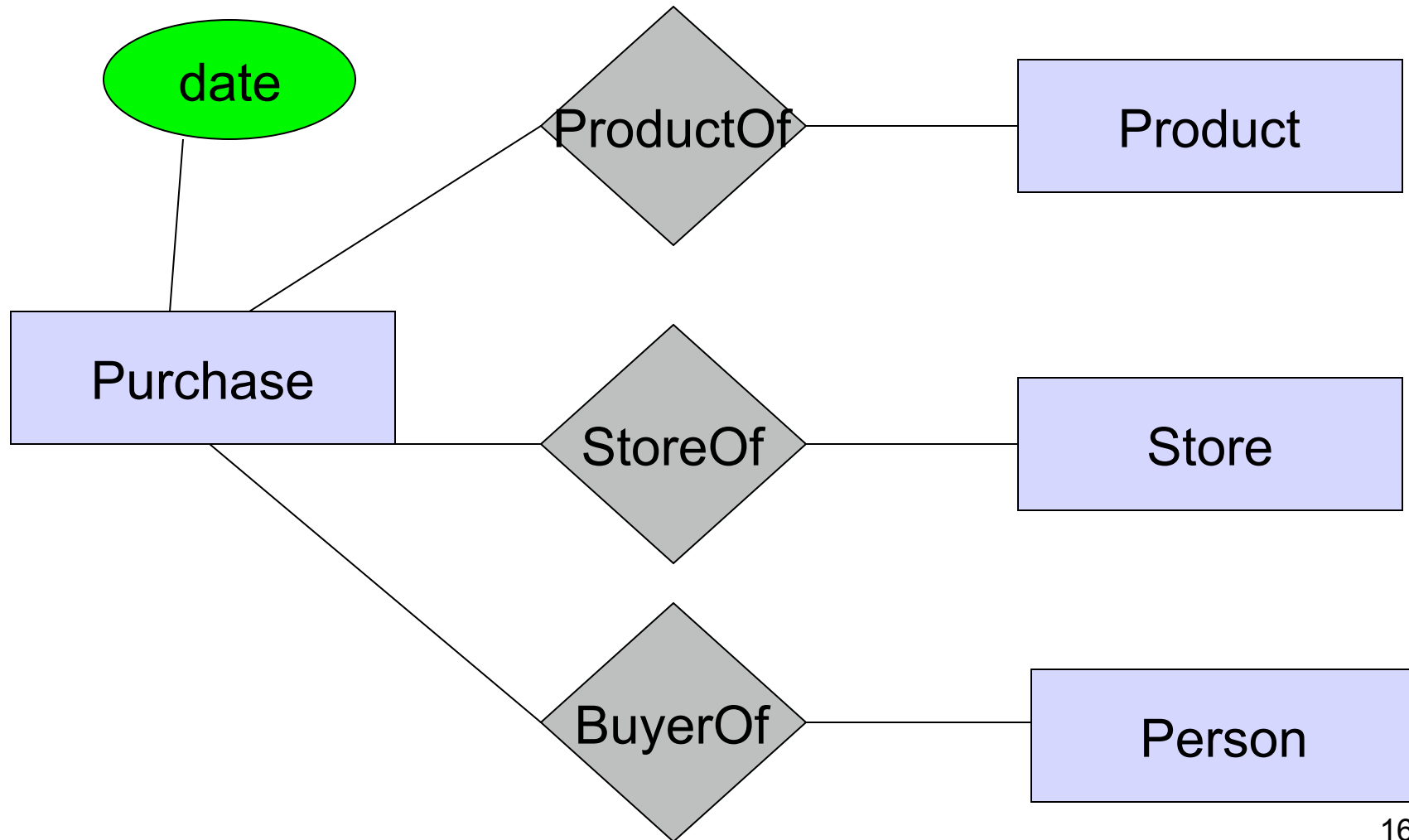
# Arrows in Multiway Relationships

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



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

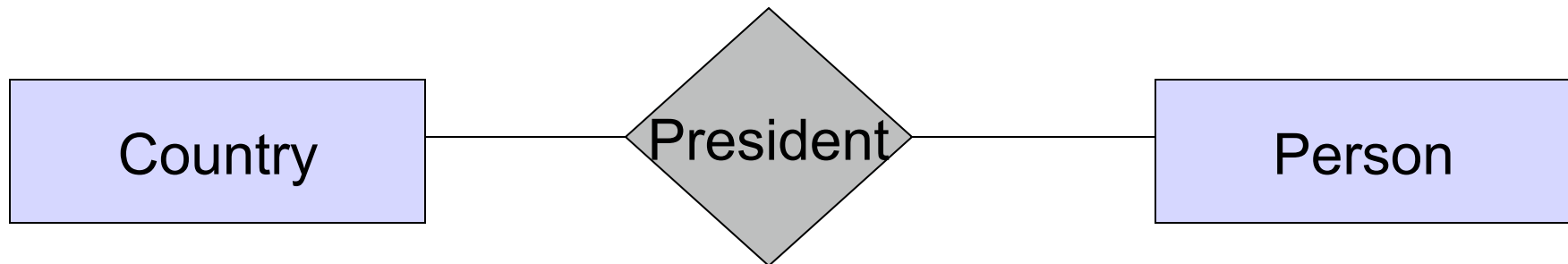
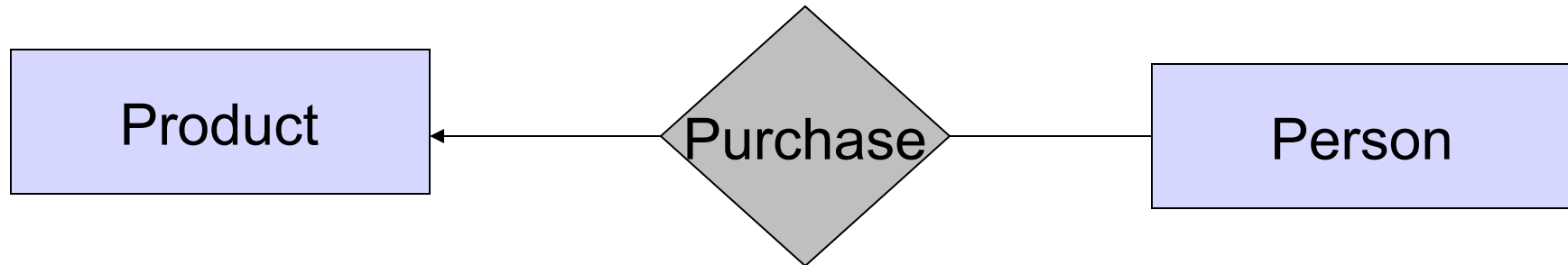
# Converting Multi-way Relationships to Binary





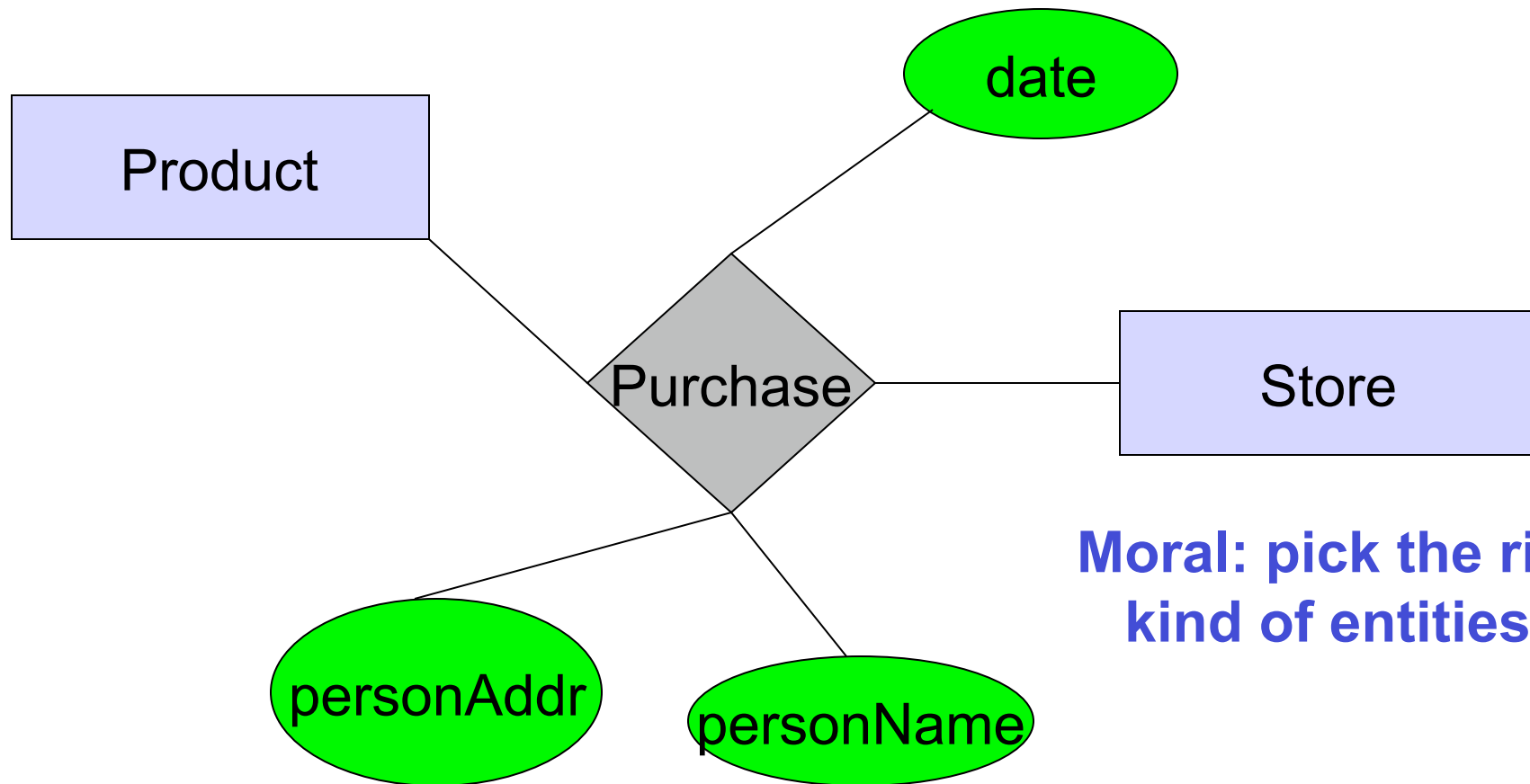
# 3. Design Principles

What's wrong?



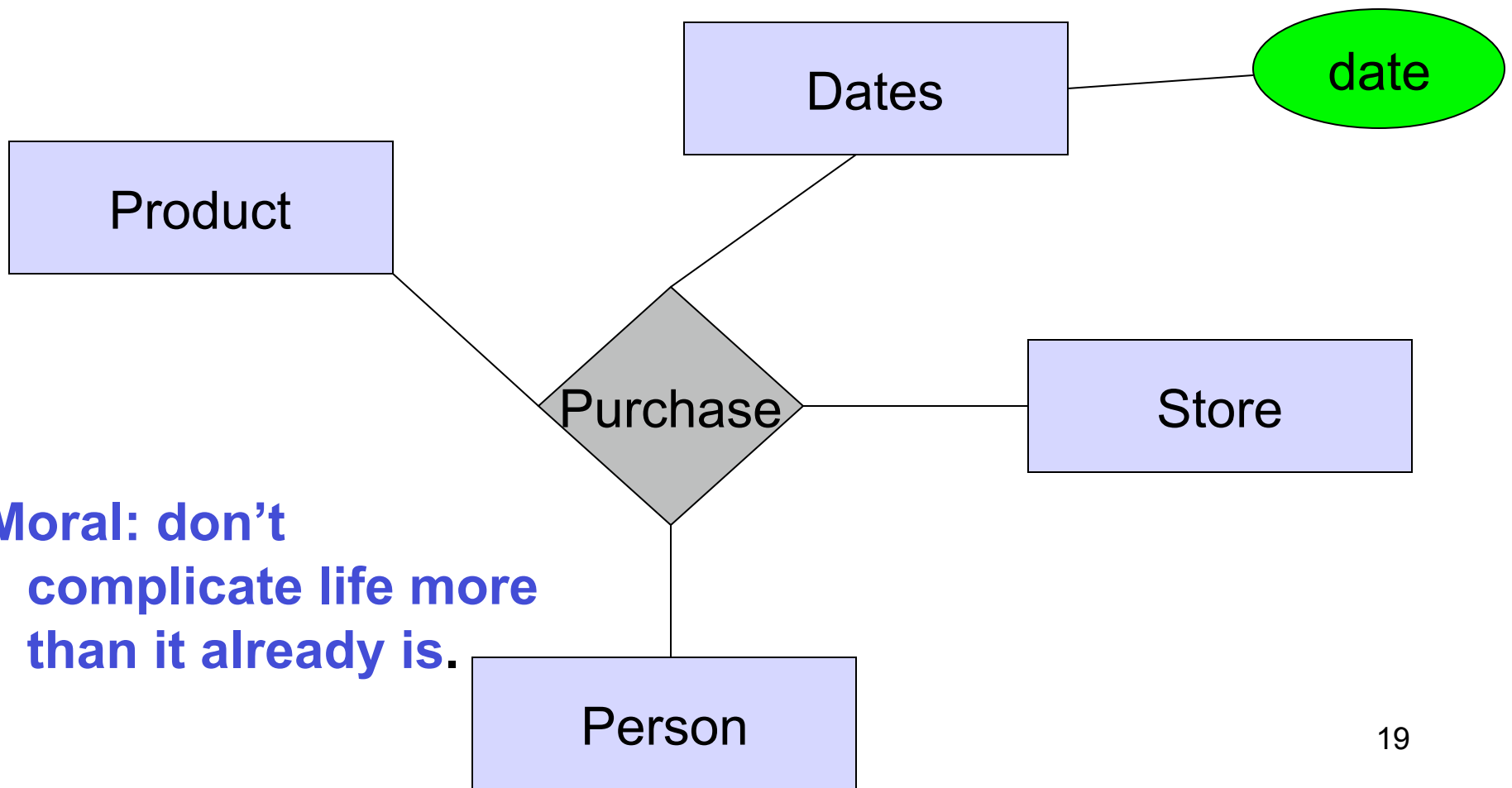
**Moral: be faithful to the specifications of the app!**

# Design Principles: What's Wrong?



**Moral: pick the right  
kind of entities.**

# Design Principles: What's Wrong?

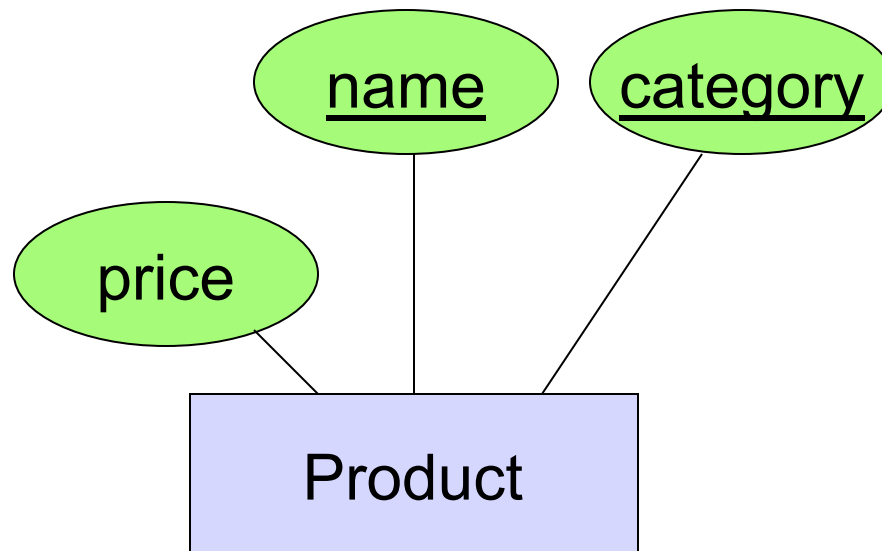


**Moral: don't  
complicate life more  
than it already is.**

# From E/R Diagrams to Relational Schema

- Entity set  $\rightarrow$  relation
- Relationship  $\rightarrow$  relation

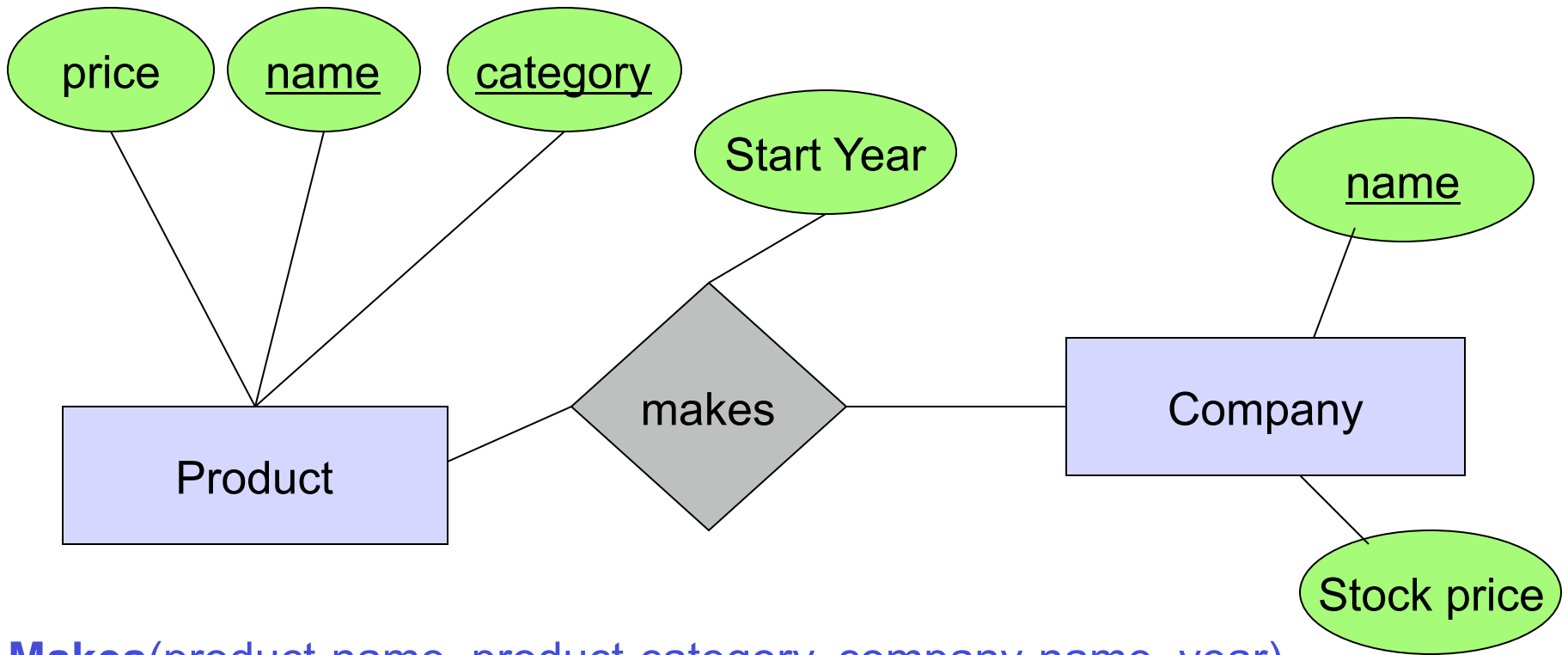
# Entity Set to Relation



**Product**(name, category, price)

name	category	price
gizmo	gadgets	\$19.99

# Relationships to Relations

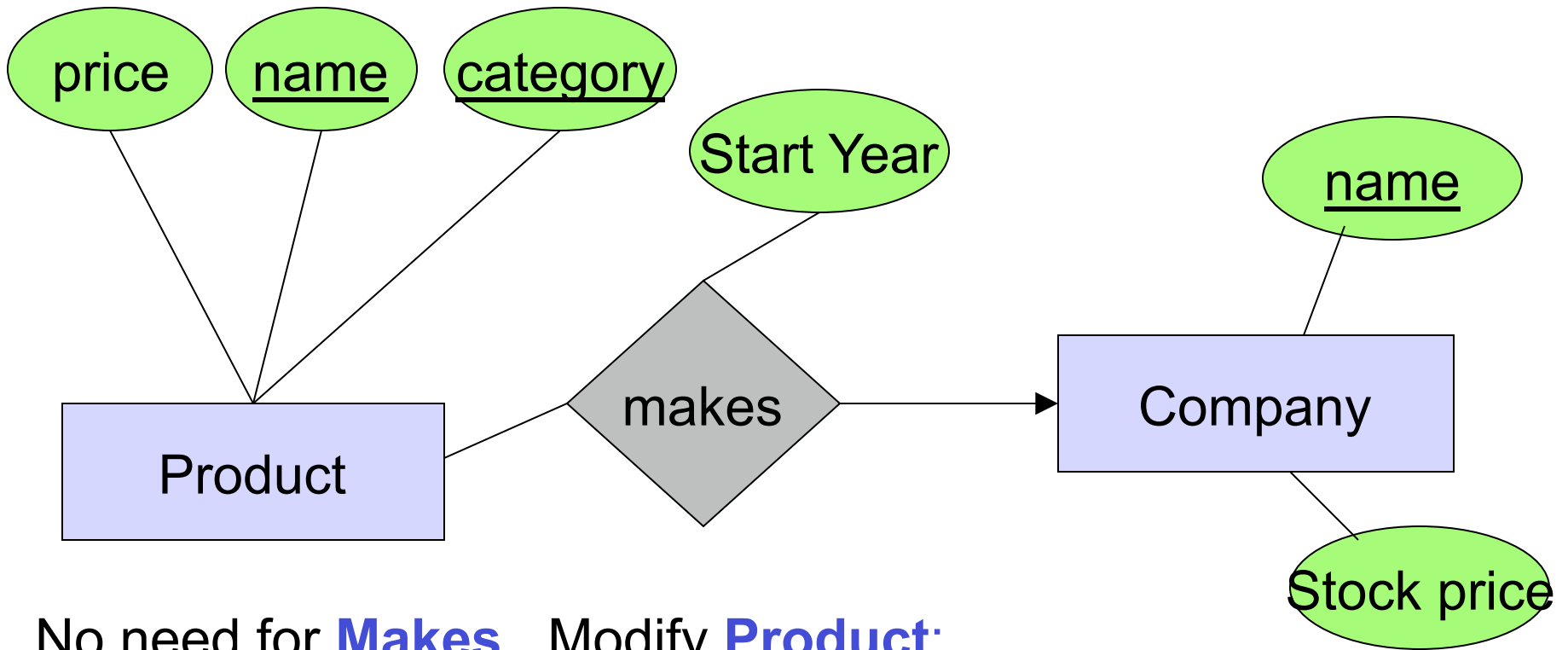


**Makes**(product-name, product-category, company-name, year)

Product-name	Product-Category	Company-name	Starting-year
gizmo	gadgets	gizmoWorks	1963

(watch out for attribute name conflicts)

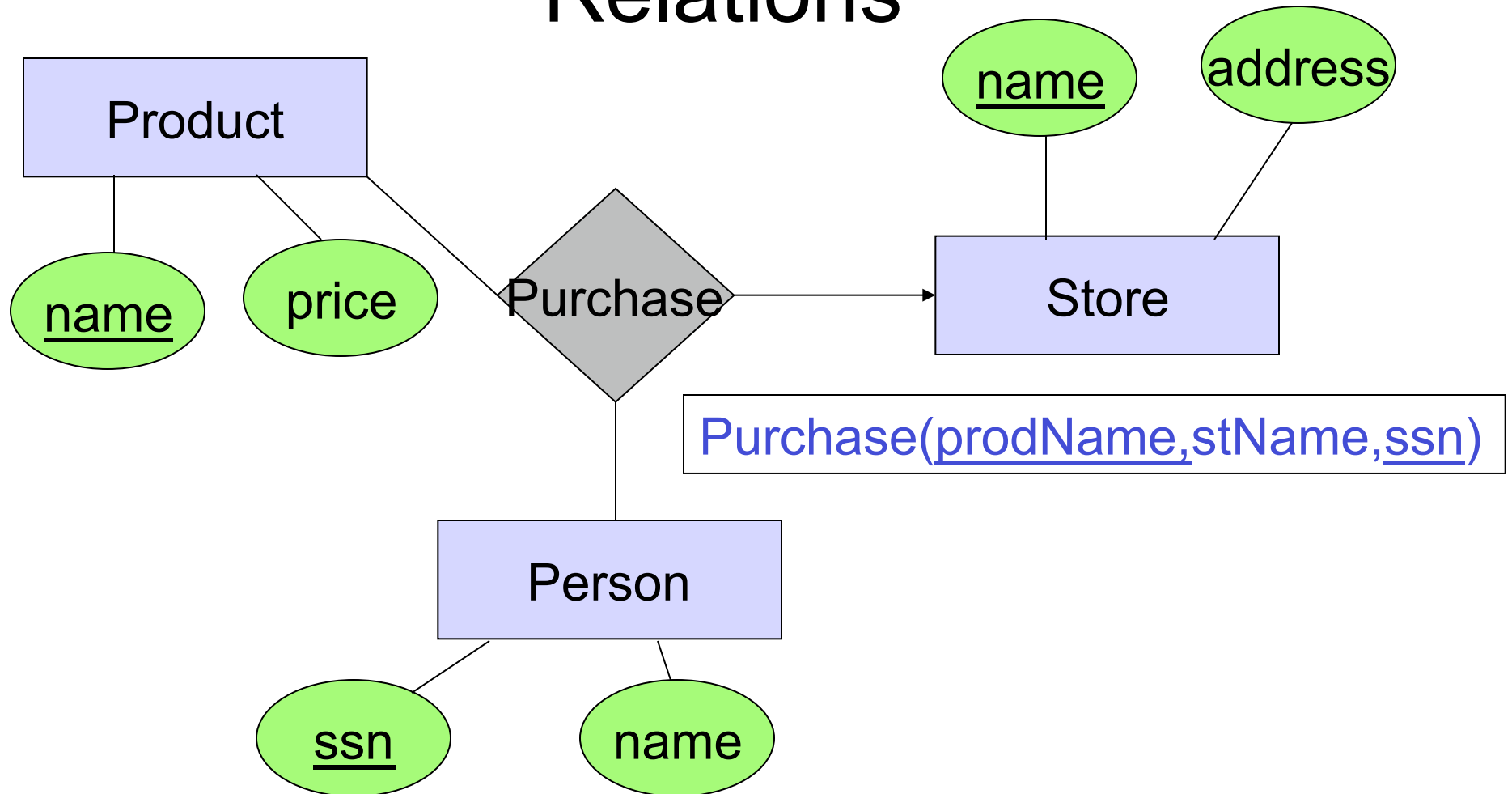
# Relationships to Relations



No need for **Makes**. Modify **Product**:

<u>name</u>	<u>category</u>	<u>price</u>	<u>StartYear</u>	<u>companyName</u>
gizmo	gadgets	19.99	1963	gizmoWorks

# Multi-way Relationships to Relations

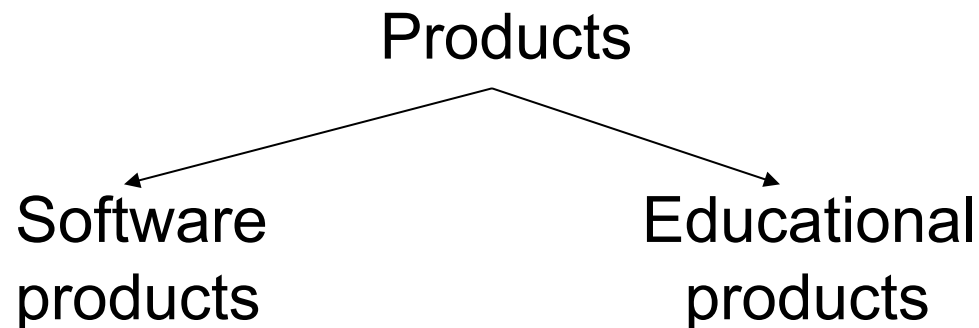




# Modeling Subclasses

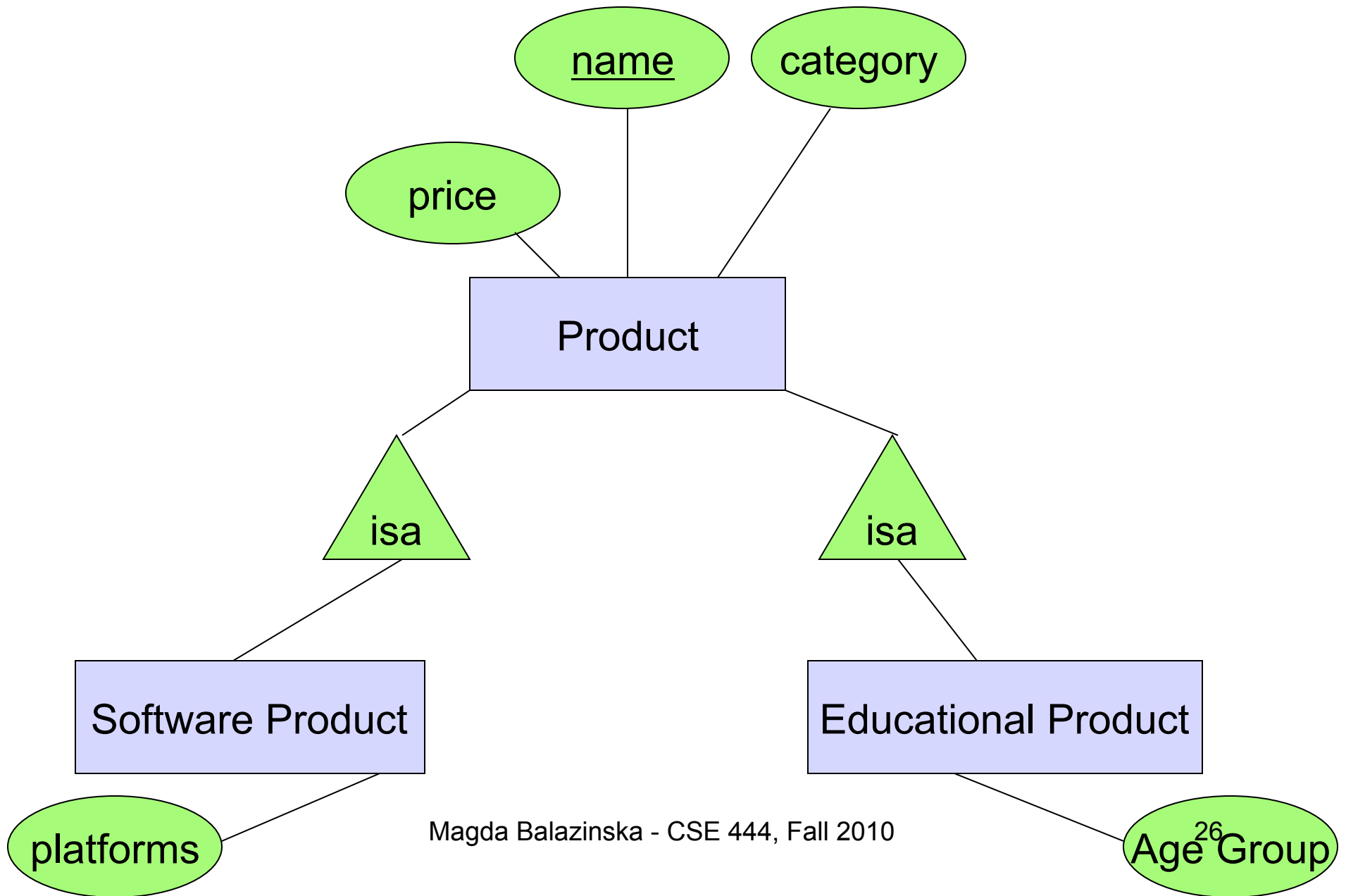
Some objects in a class may be special

- define a new class
- better: define a *subclass*



So --- we define subclasses in E/R

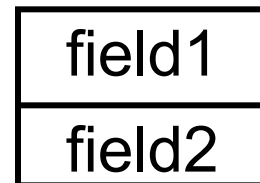
# Subclasses



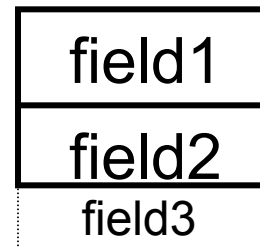
# Understanding Subclasses

- Think in terms of records:

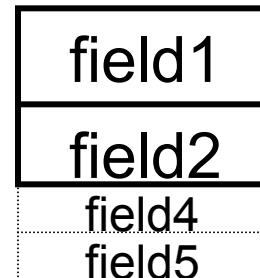
- Product



- SoftwareProduct



- EducationalProduct



# Subclasses to Relations

Product

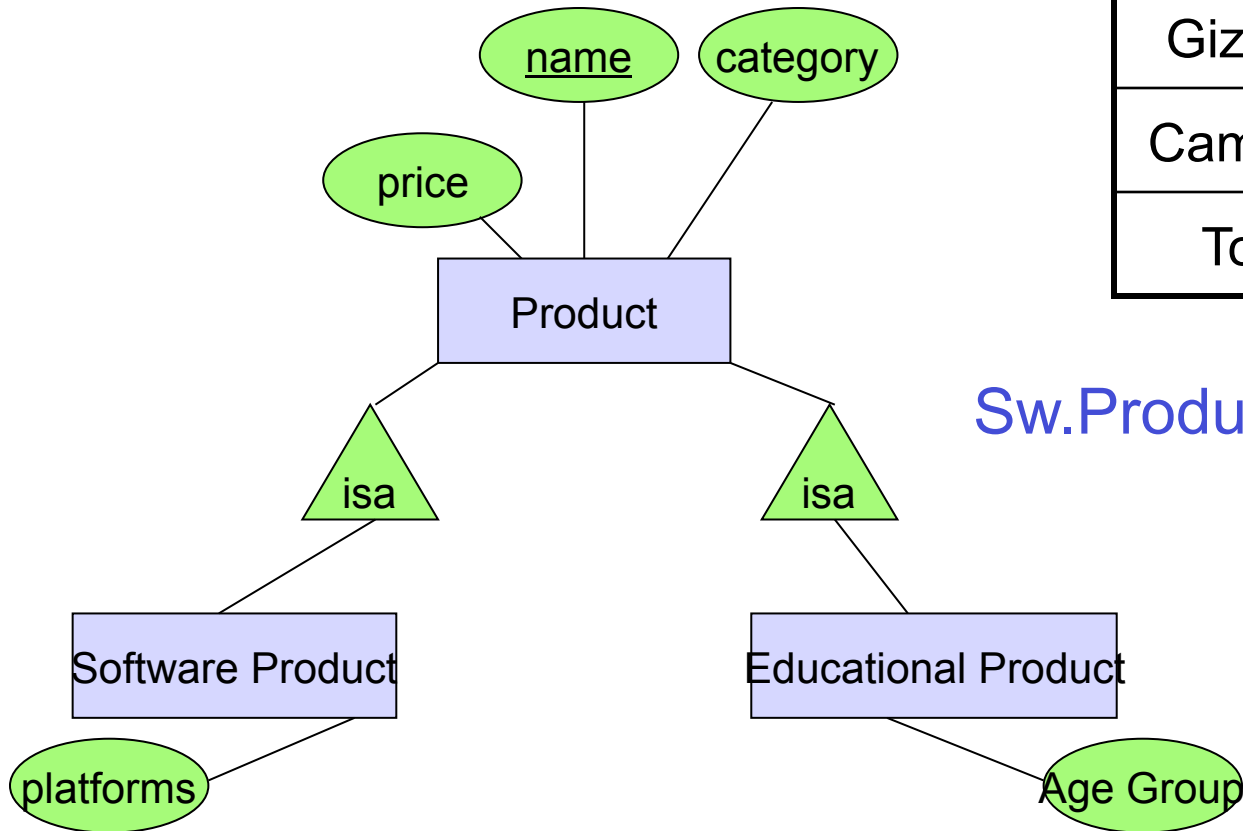
<u>Name</u>	Price	Category
Gizmo	99	gadget
Camera	49	photo
Toy	39	gadget

Sw.Product

<u>Name</u>	platforms
Gizmo	unix

Ed.Product

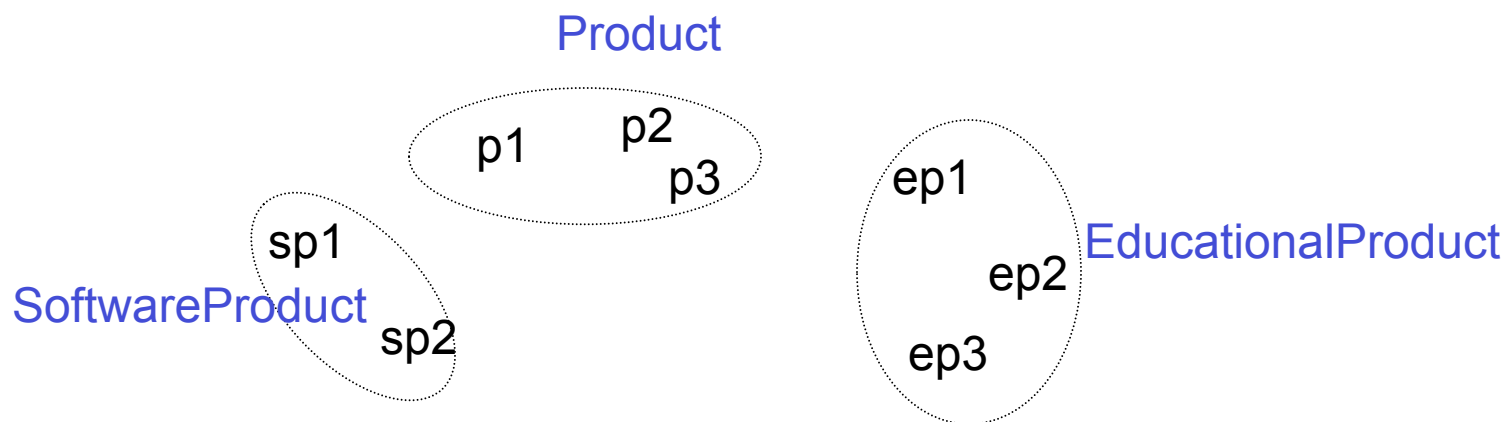
<u>Name</u>	Age Group
Gizmo	todler
Toy	retired



Other ways to convert are possible  
See book sec 4.6 [Old ed: 3.3]

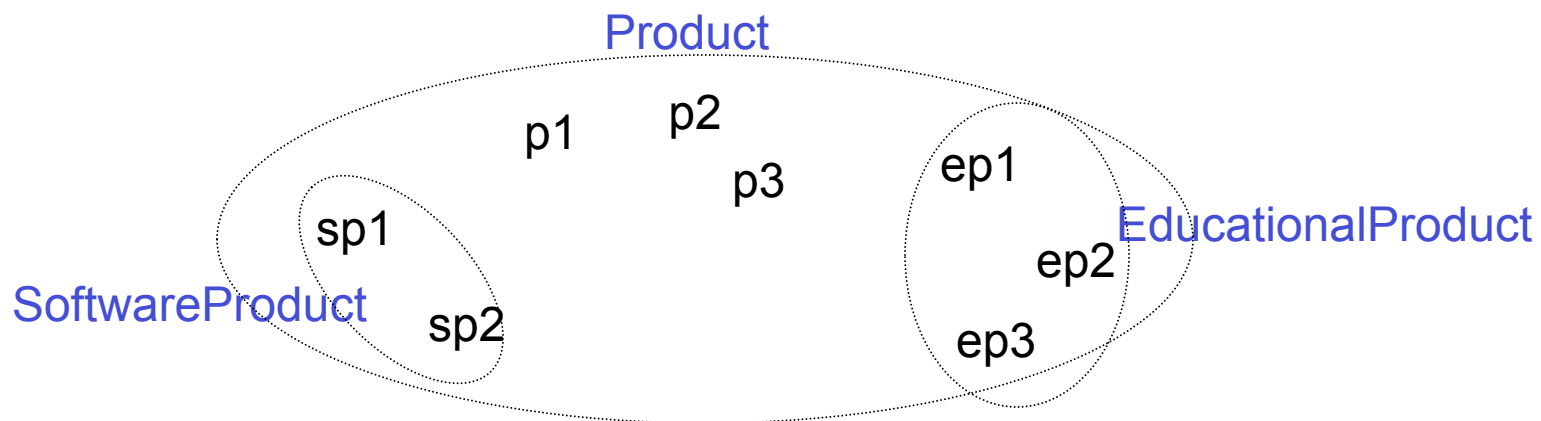
# Difference between OO and E/R inheritance

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



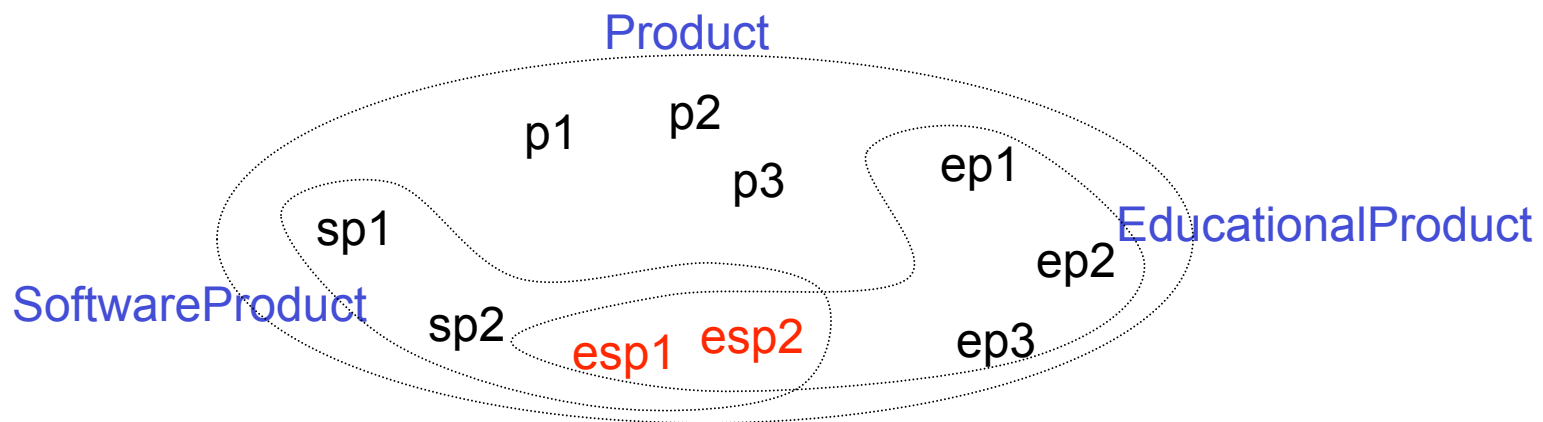
# Difference between OO and E/R inheritance

- E/R: entity sets overlap



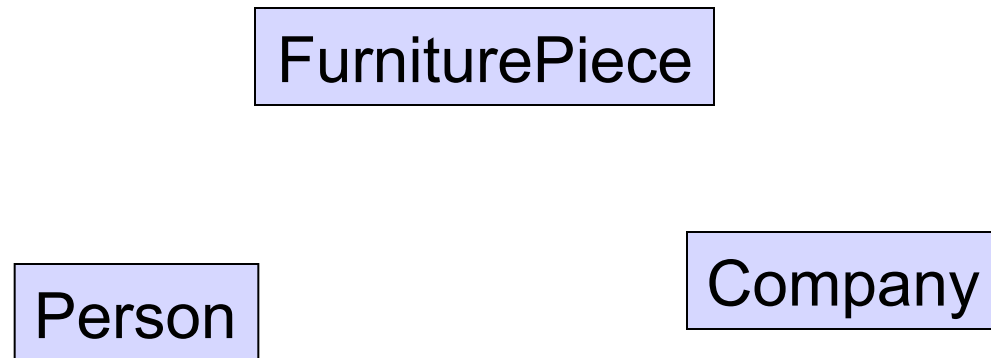
# Difference between OO and E/R inheritance

No need for multiple inheritance in E/R



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

# Modeling UnionTypes With Subclasses



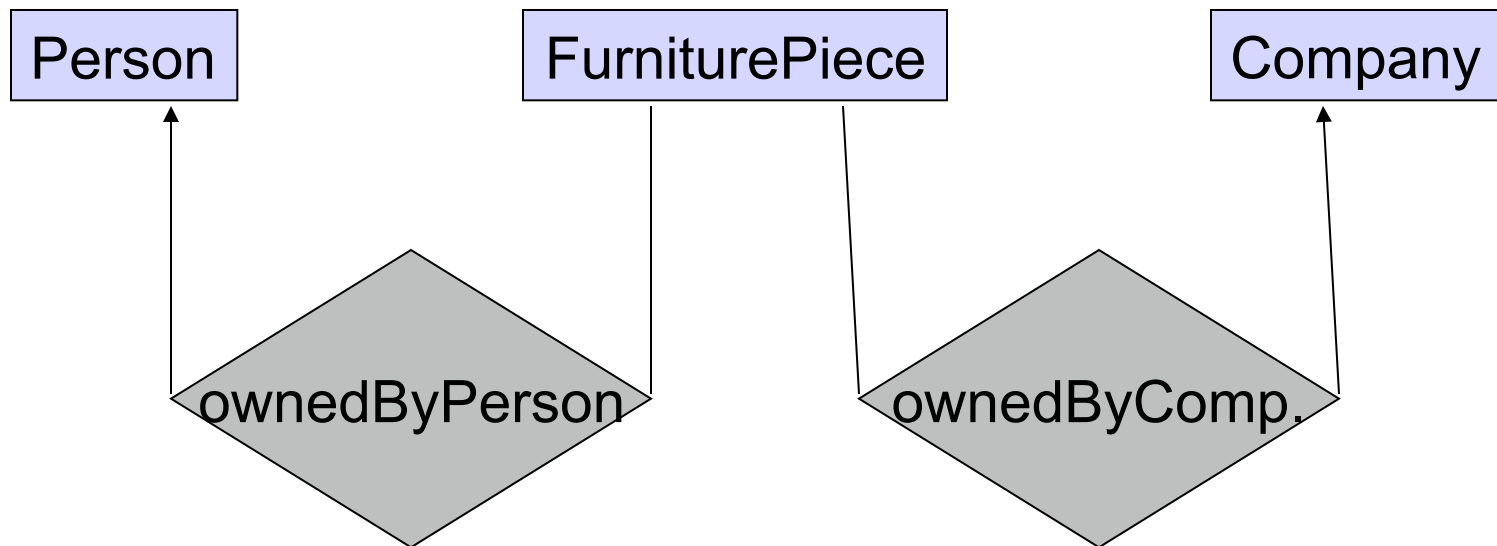
Say: each piece of furniture is owned either by a person, or by a company



# Modeling Union Types with Subclasses

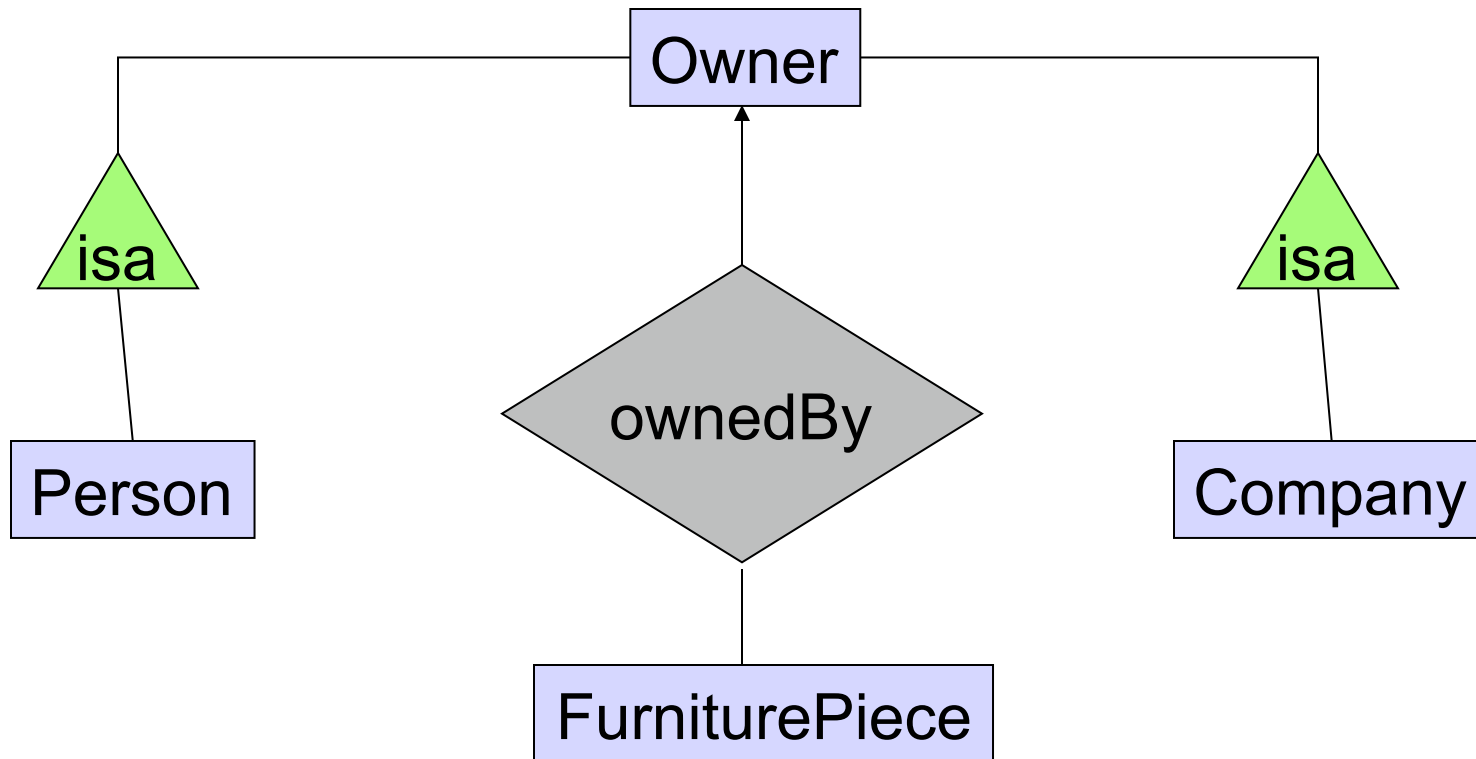
Say: each piece of furniture is owned either by a person, or by a company

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



# Modeling Union Types with Subclasses

Solution 2: better, more laborious



# Constraints in E/R Diagrams

Finding constraints is part of the modeling process.  
Commonly used constraints:

**Keys:** social security number uniquely identifies a person.

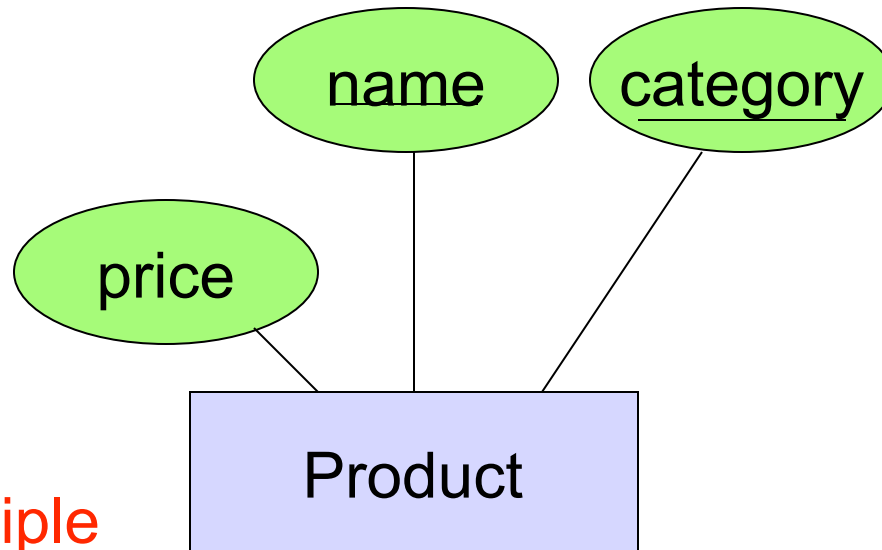
**Single-value constraints:** a person can have only one father.

**Referential integrity constraints:** if you work for a company, it must exist in the database.

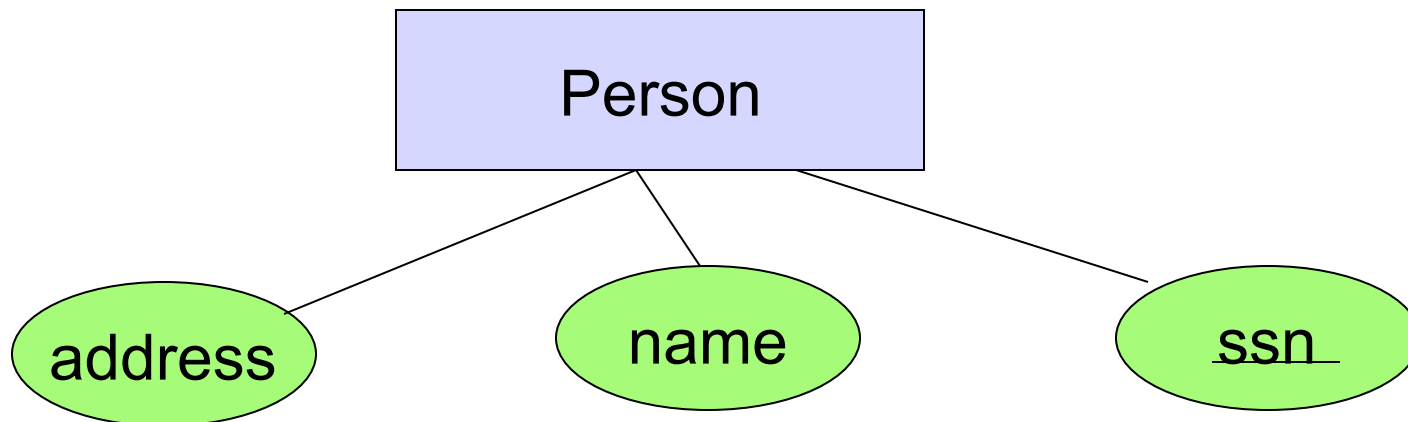
**Other constraints:** peoples' ages are between 0 and 150.

# Keys in E/R Diagrams

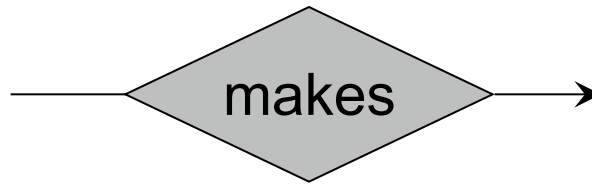
Underline:



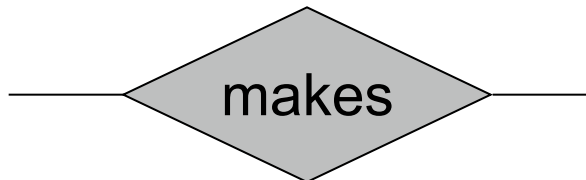
No formal way  
to specify multiple  
keys in E/R diagrams



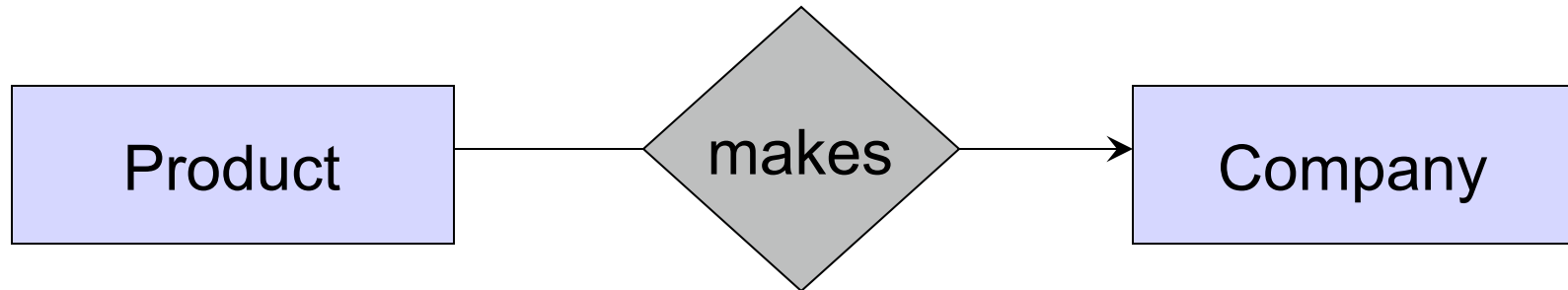
# Single Value Constraints



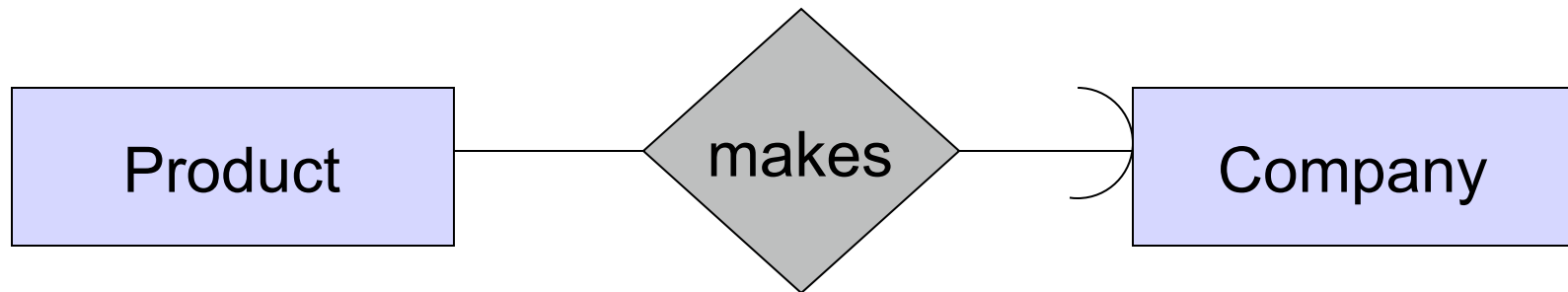
v. s.



# Referential Integrity Constraints

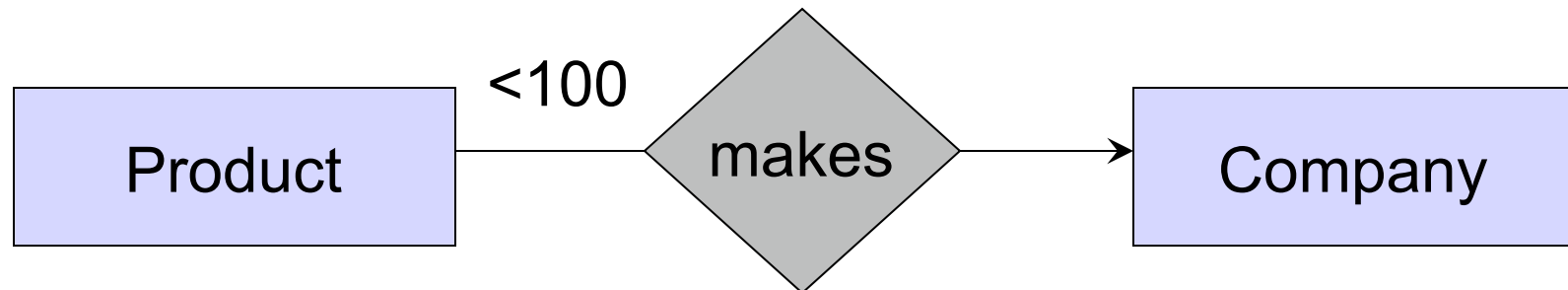


Each product made by at most one company.  
Some products made by no company



Each product made by exactly one company.

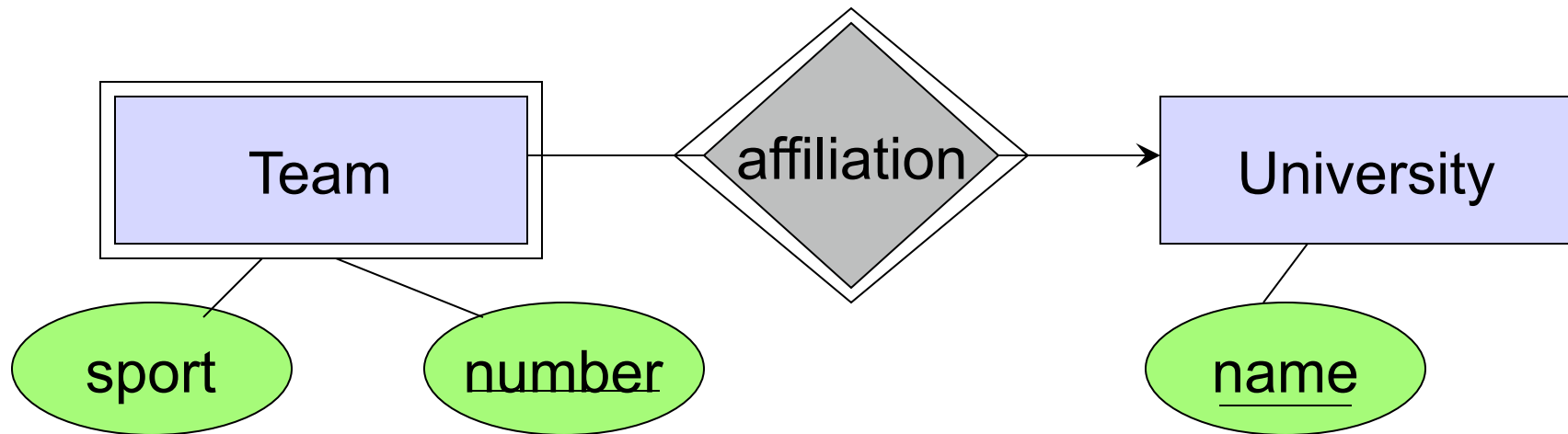
# Other Constraints



What does this mean ?

# Weak Entity Sets

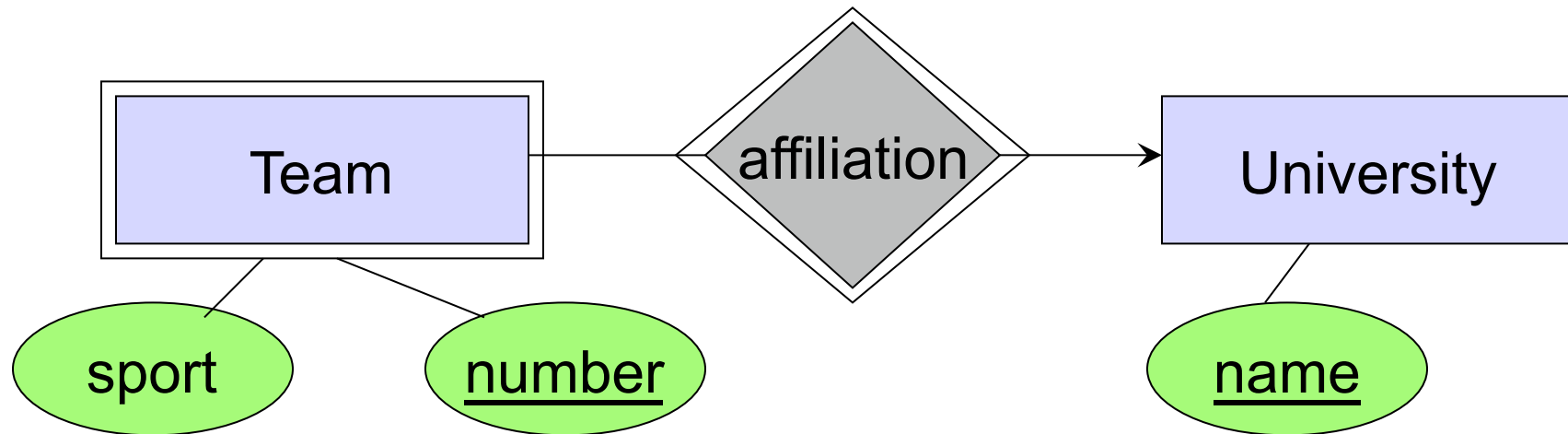
Entity sets are weak when their key comes from other classes to which they are related.



Notice: we encountered this when converting multiway relationships to binary relationships



# Handling Weak Entity Sets



Convert to a relational schema

```
University(name)  
Team(number, universityName, sport)  
No need to represent affiliation separately
```