Introduction to Database Systems CSE 444

Lecture 5: E/R Diagrams

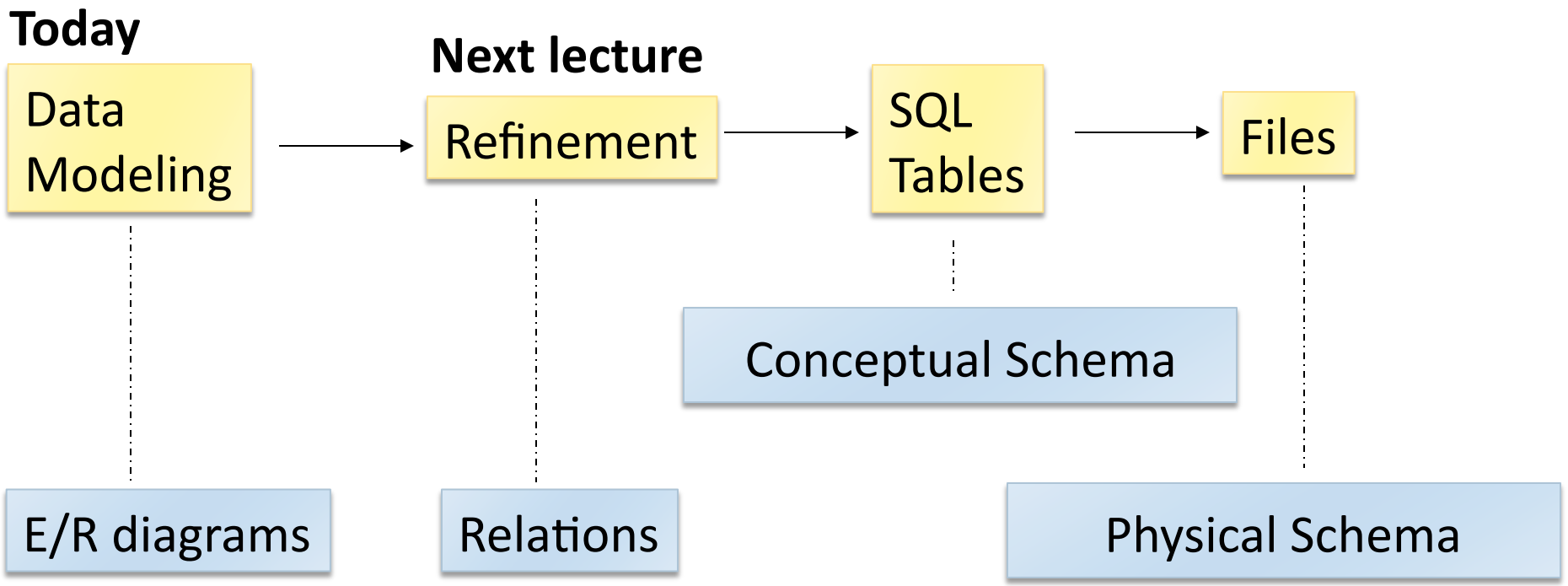
Outline

- Announcements
  - Anyone still not registered??
- 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

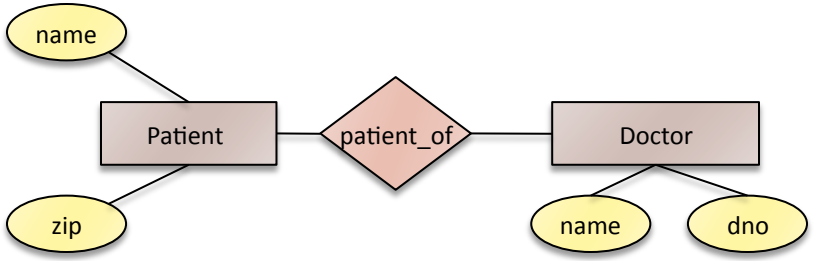
- Why do we need it?
  - Need a way to model real world entities in terms of relations
  - Not easy to go from real-world entities to a database schema
- Consider issues such as:
  - What entities to model
  - How entities are related
  - What constraints exist in the domain
  - How to achieve good designs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

Database Design Process

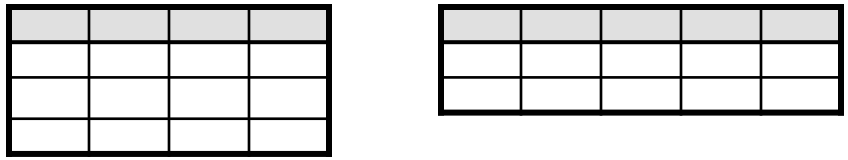


Conceptual Schema Design

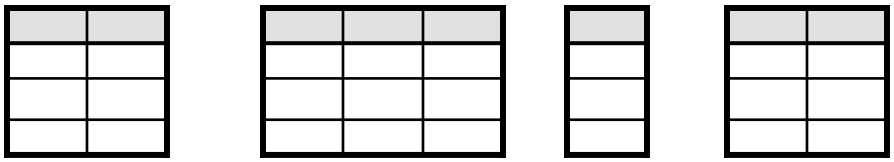
Conceptual Model:



Relational Model:
plus FD's
(FD = Functional Dependency)



Normalization:
Eliminates anomalies



Entity / Relationship Diagrams

Objects → entities
Classes → entity sets

entity set

All entities in the same entity set have the same attributes

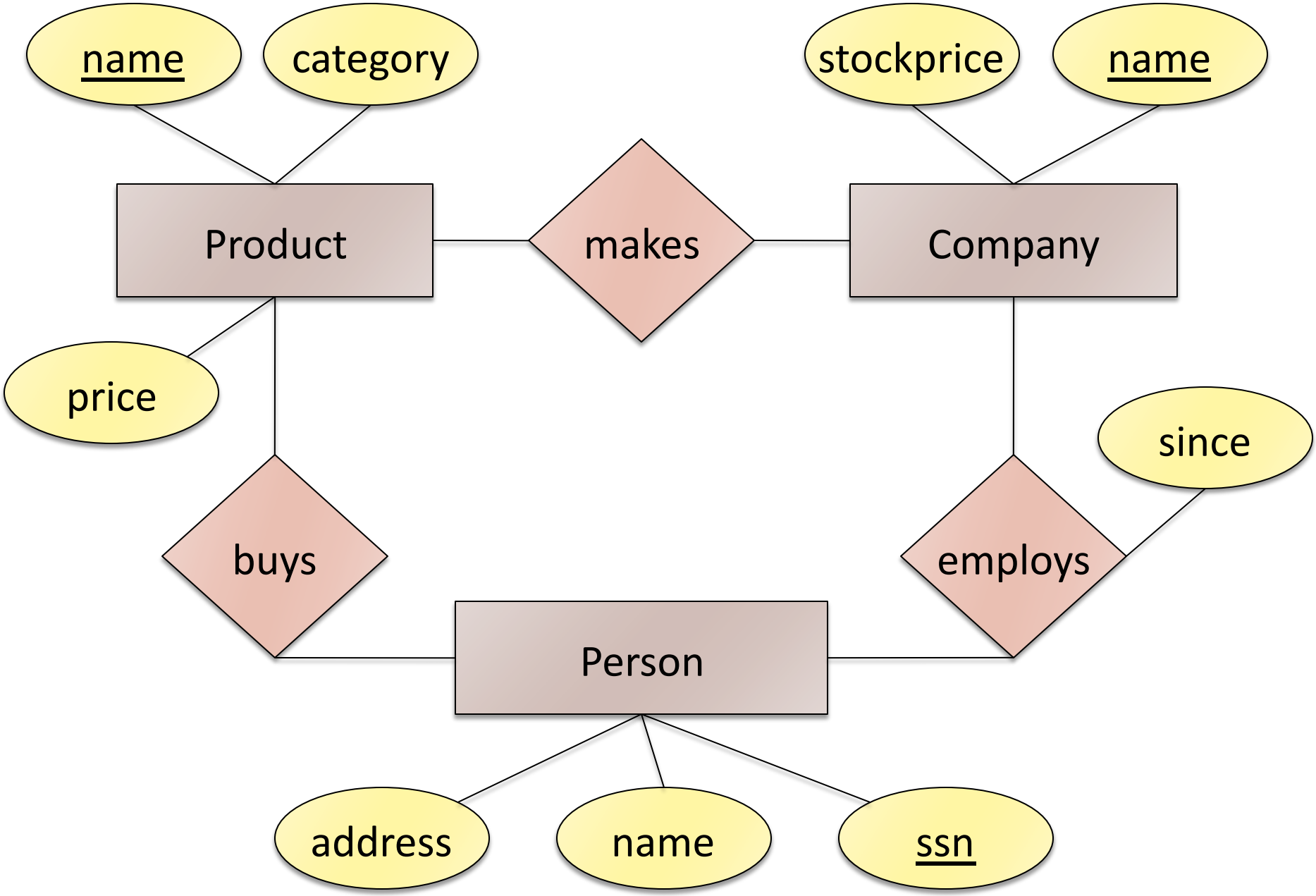
Attributes are like in ODL
(ODL = Object Definition Language)

A relationship may have attributes too!

Relationships: like in ODL except

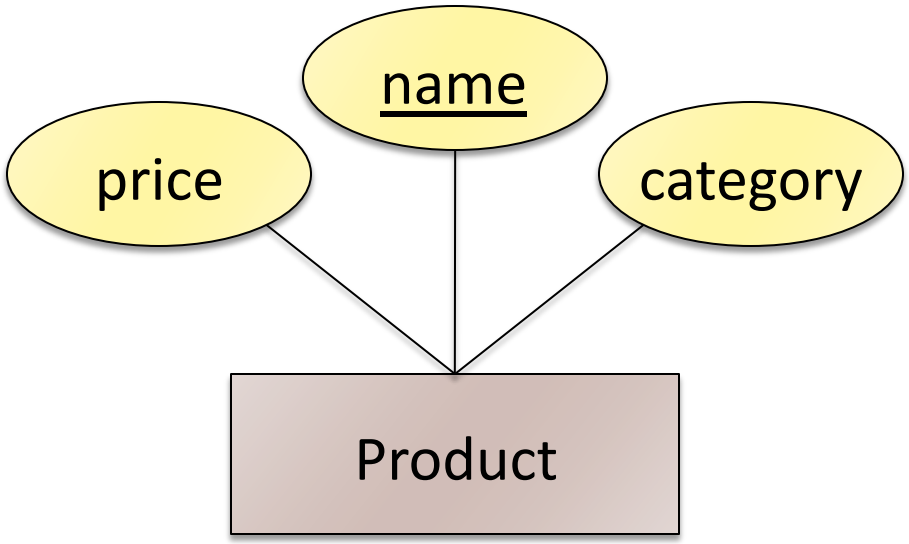
Association between 2 or more entities

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


name
category
stockprice
name
Product
makes
Company
price
since
buys
employs
Person
address
name
ssn

Keys in E/R Diagrams

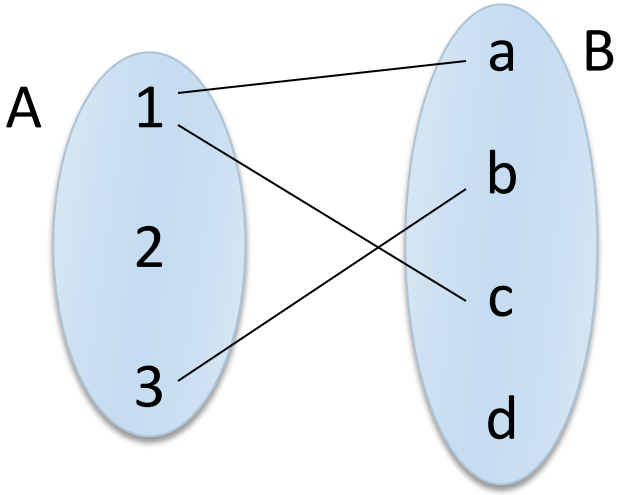
- Every entity set must have a key



What is a Relationship?

- A mathematical definition:
  - if A, B are sets, then a relationship R is a subset of A × B

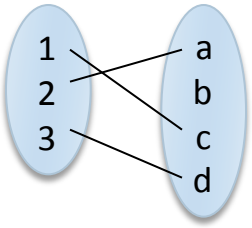
- A={1,2,3}, B={a,b,c,d},
  - A × B = {(1,a),(1,b), . . ., (3,d)}
  - R = {(1,a), (1,c), (3,b)}

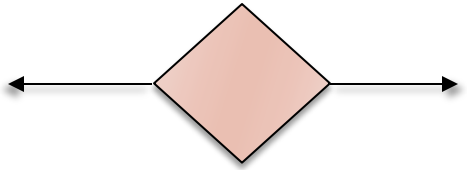


- makes is a subset of Product × Company:

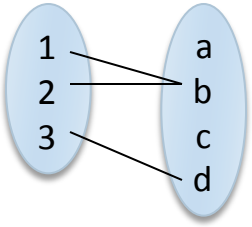
Multiplicity of E/R Relations

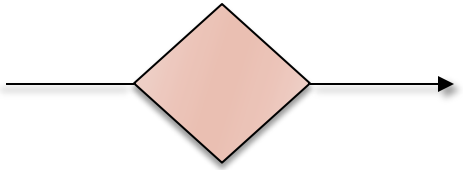
- one-one:



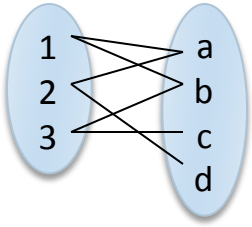


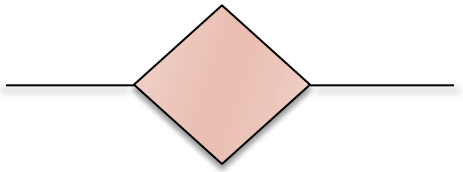
- many-one

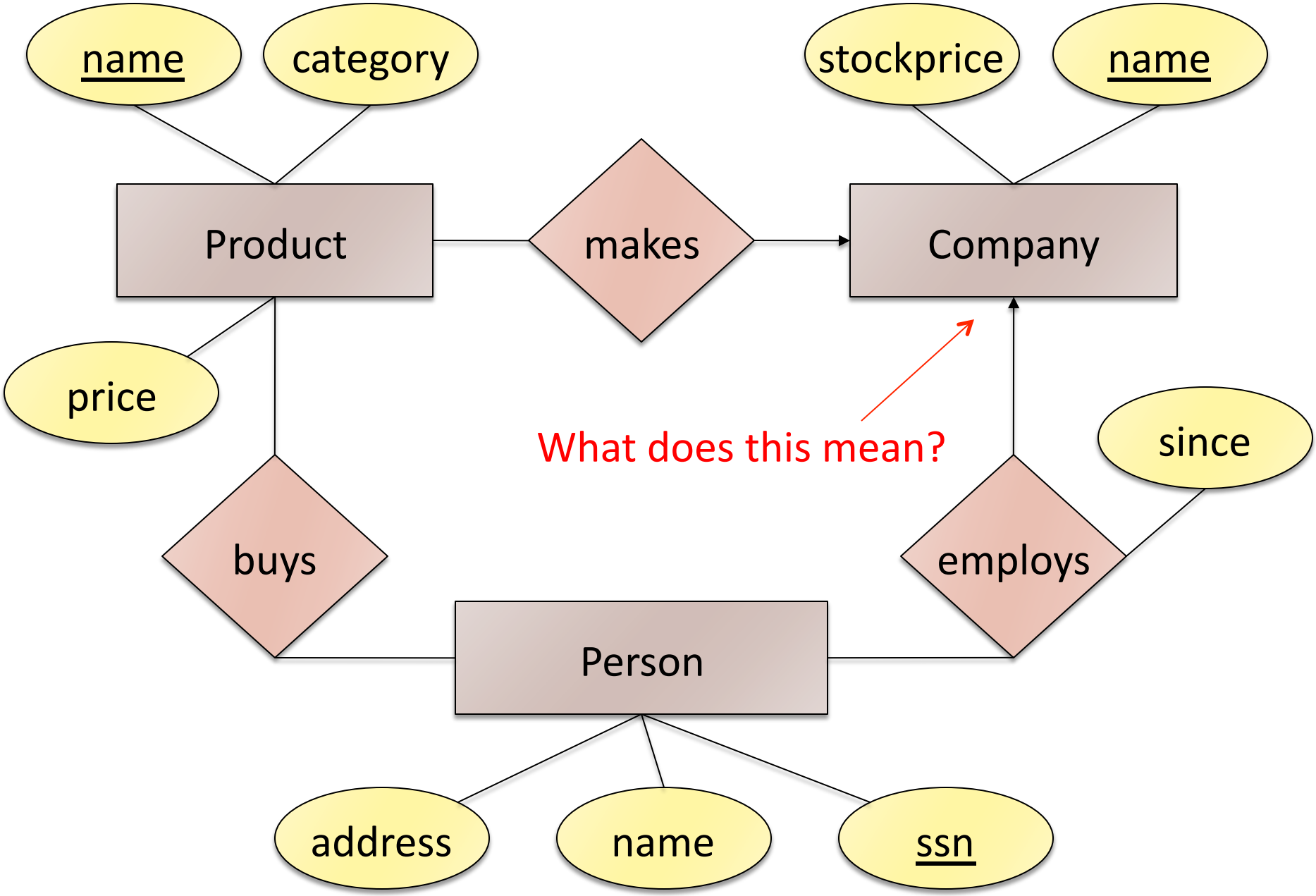




- many-many

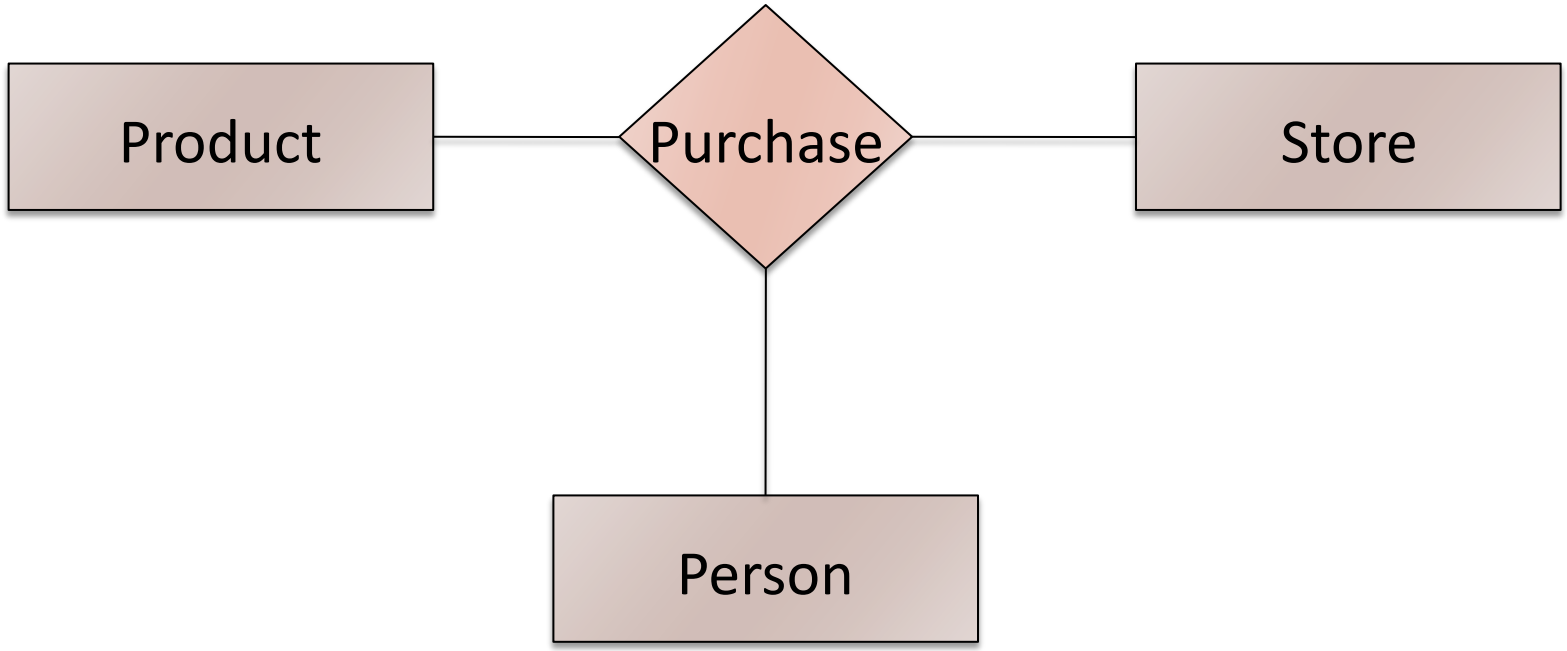





name
category
stockprice
name
Product
makes
Company
price
What does this mean?
since
buys
employs
Person
address
name
ssn

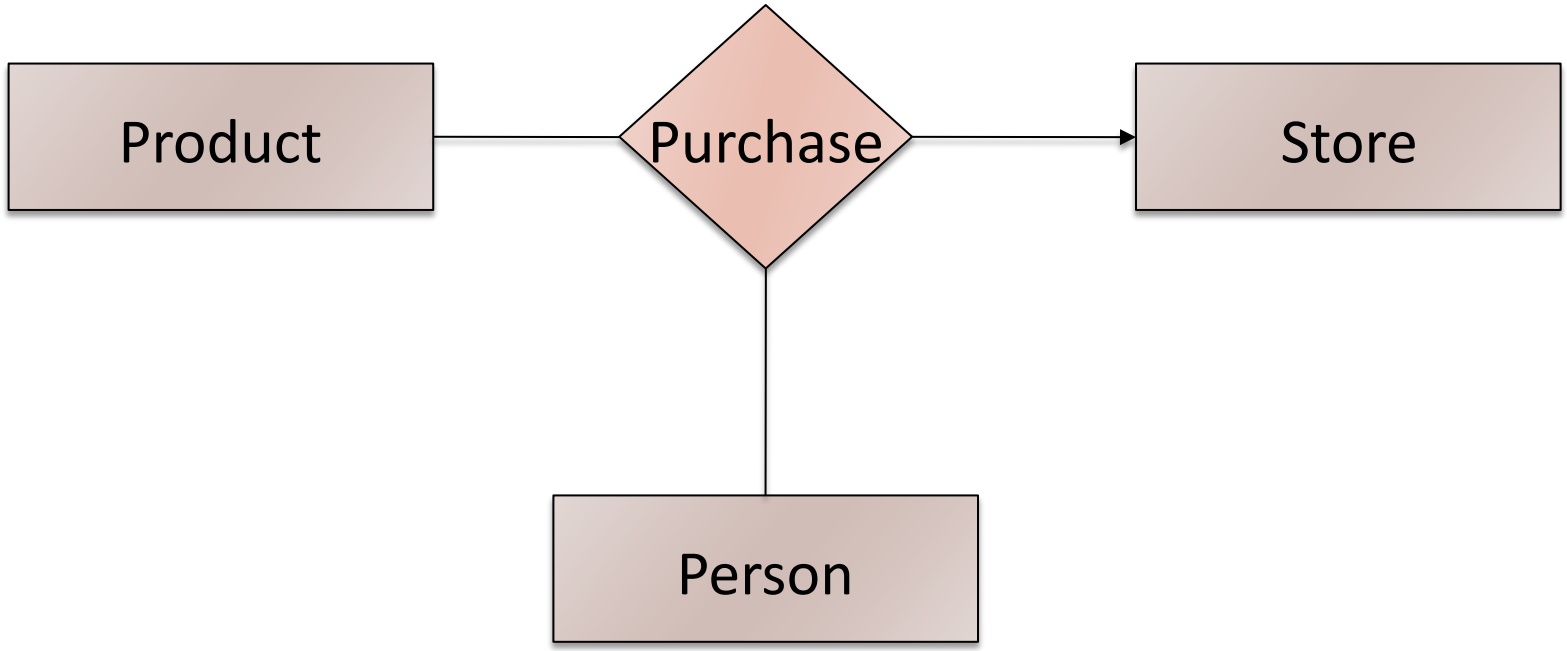
Multi-way Relationships

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



Key Constraints in Multi-way Relationships

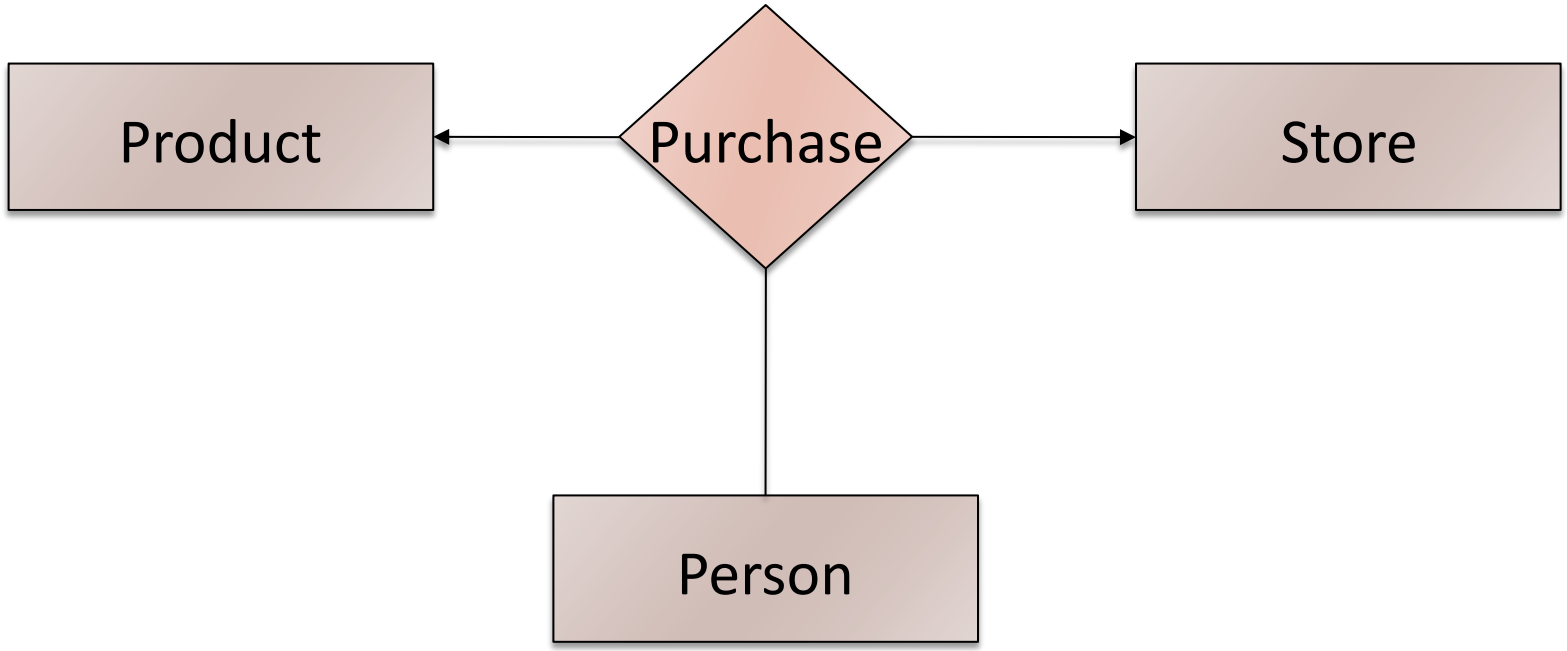
Q: What does the arrow mean?



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

Key Constraints in Multi-way Relationships

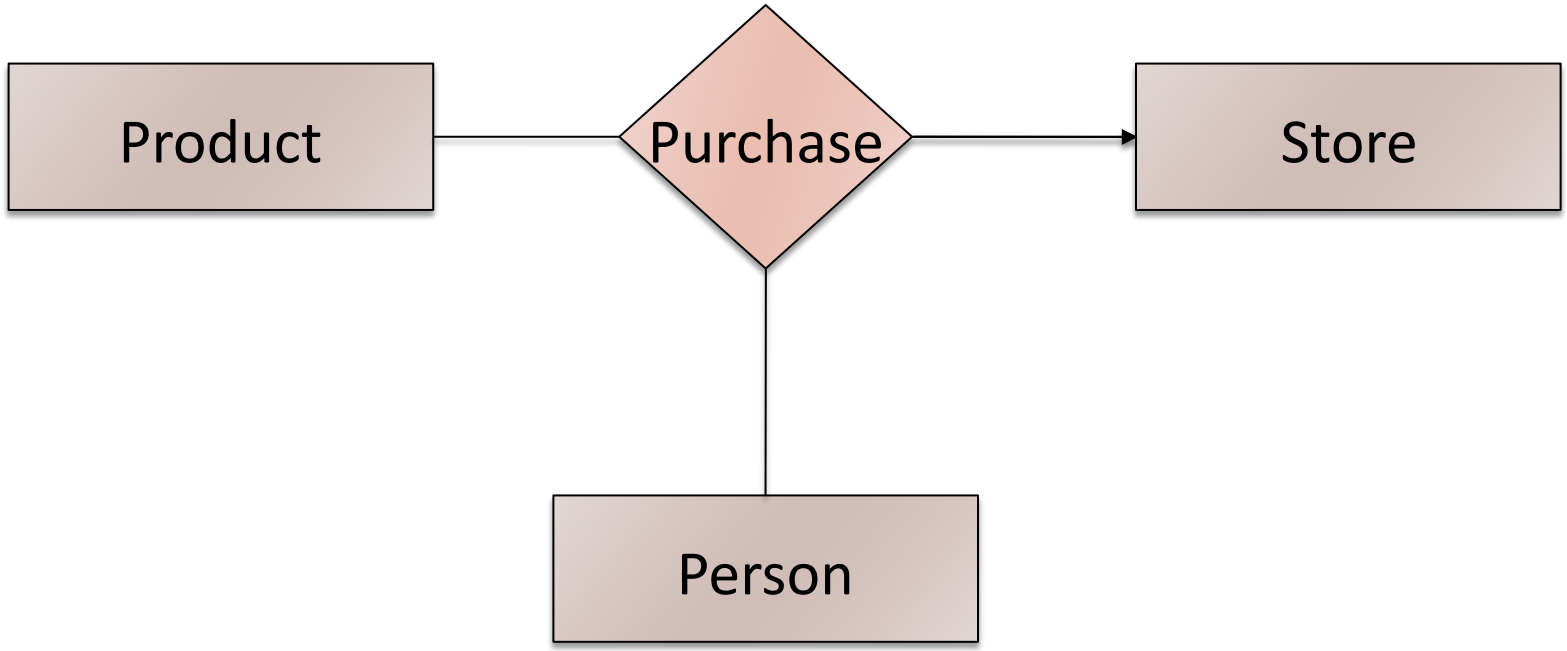
Q: What does the arrow mean?



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

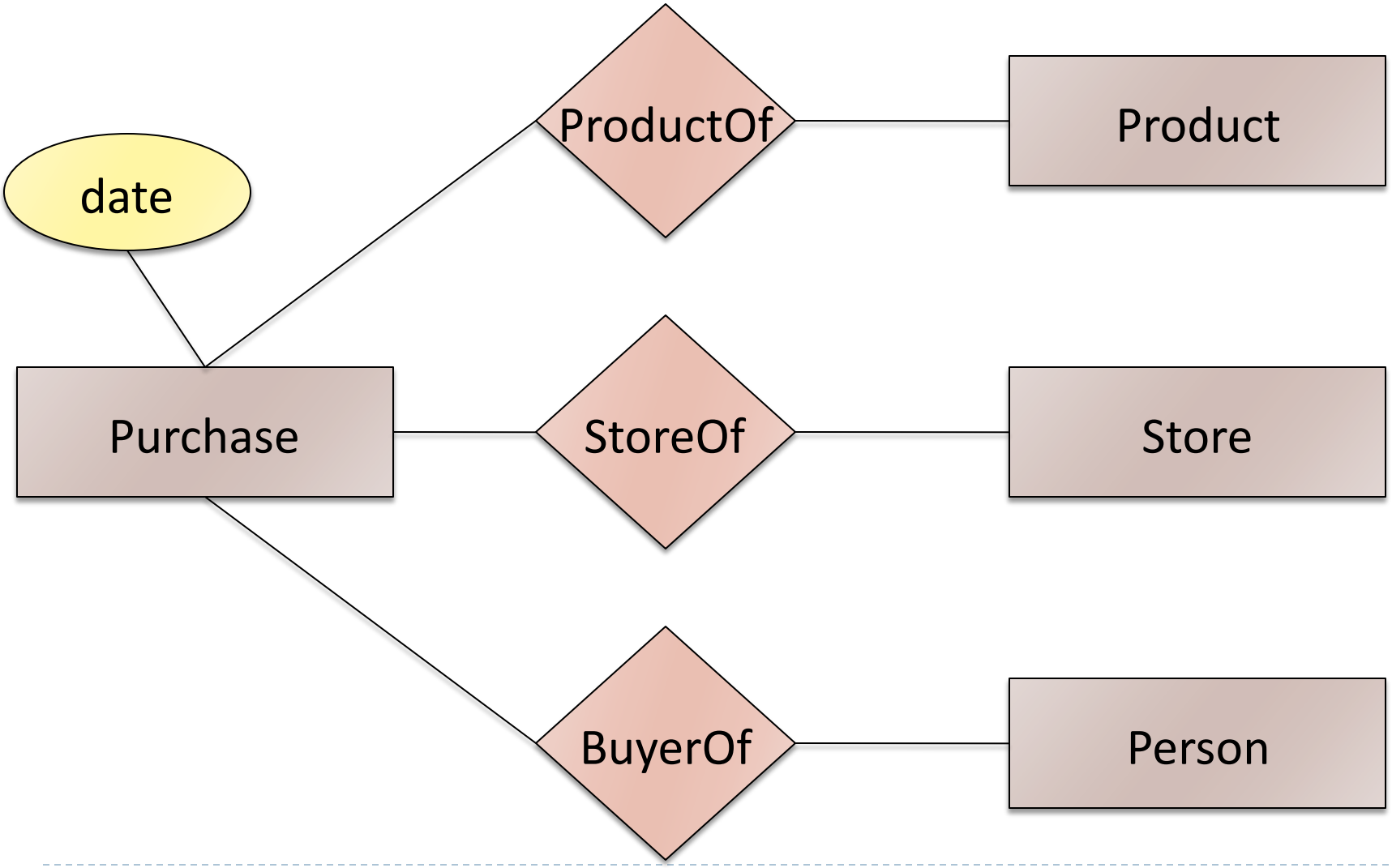
Key Constraints in Multi-way Relationships

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



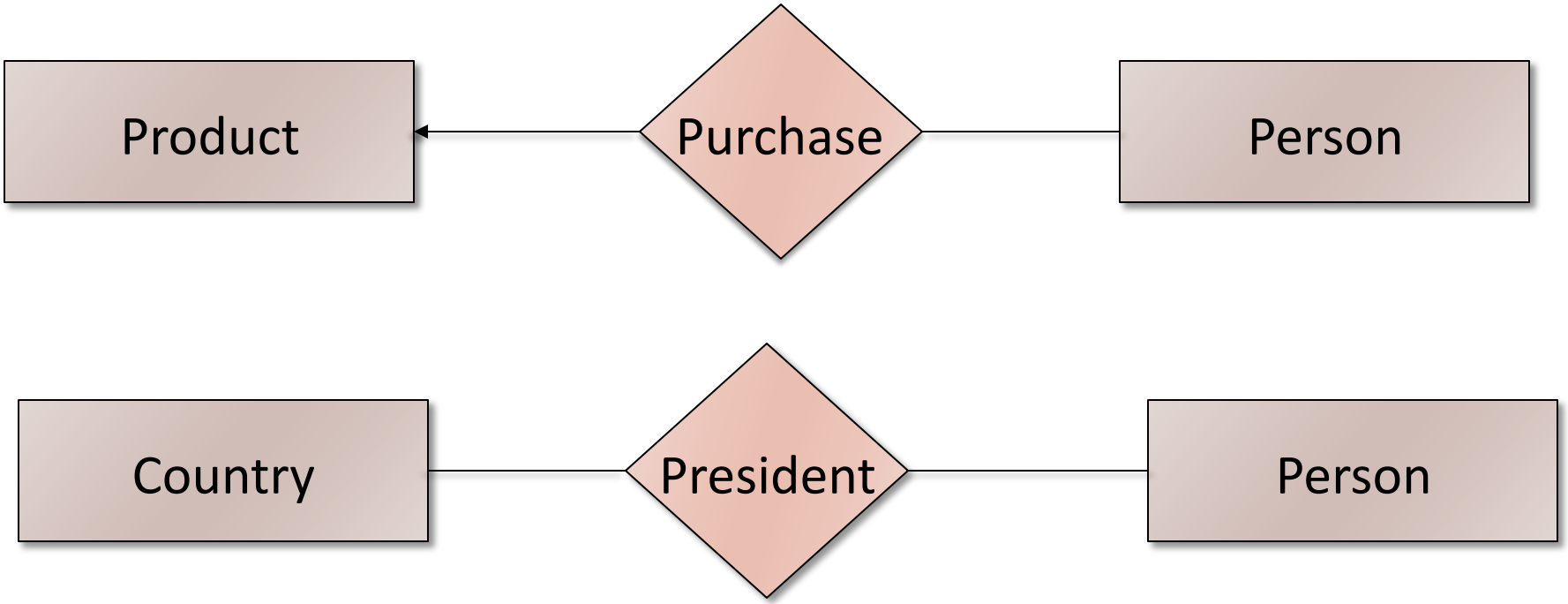
A: Cannot. This is the best approximation.
(Why only approximation ?)

Converting Multi-way Relationships to Binary



Design Principles

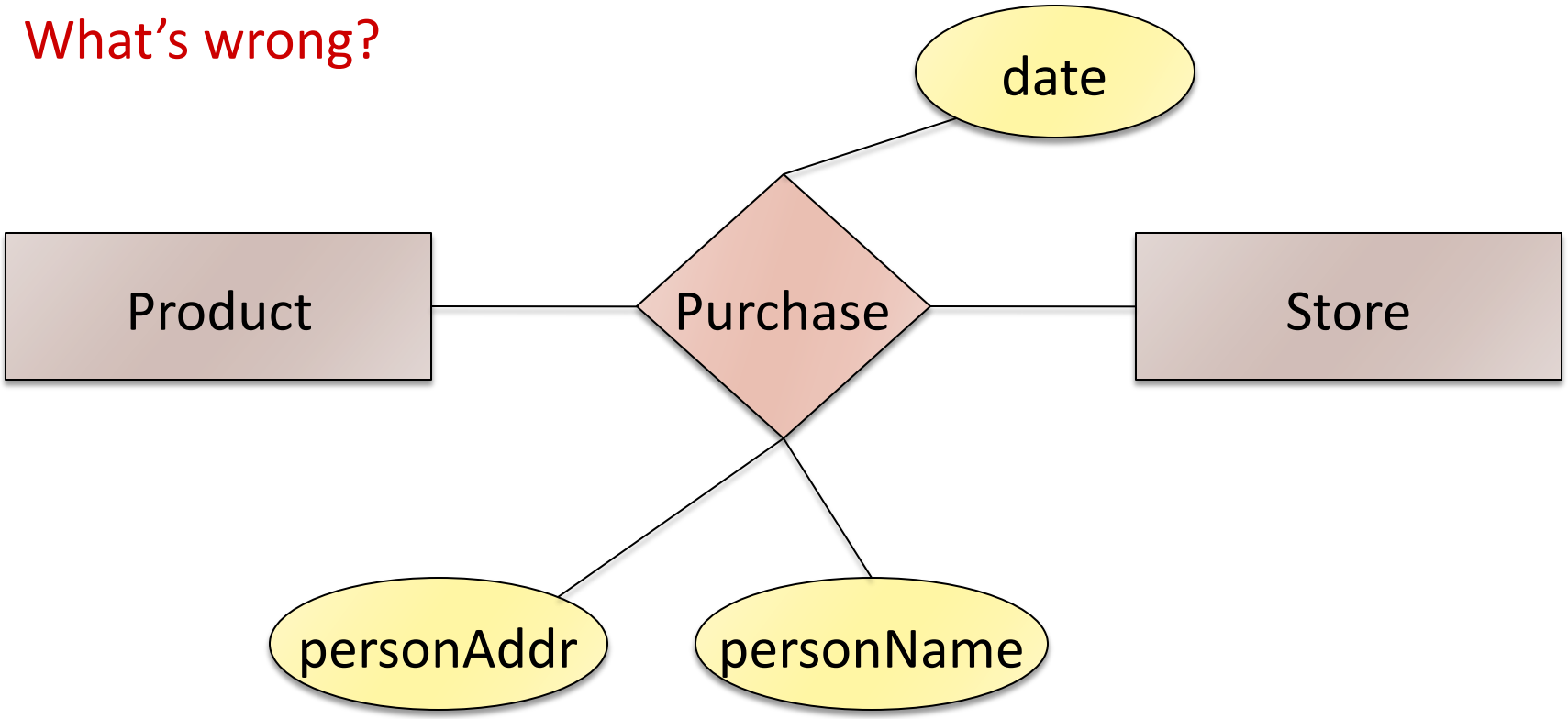
What's wrong?



Moral: be faithful to the specifications of the app!

Design Principles

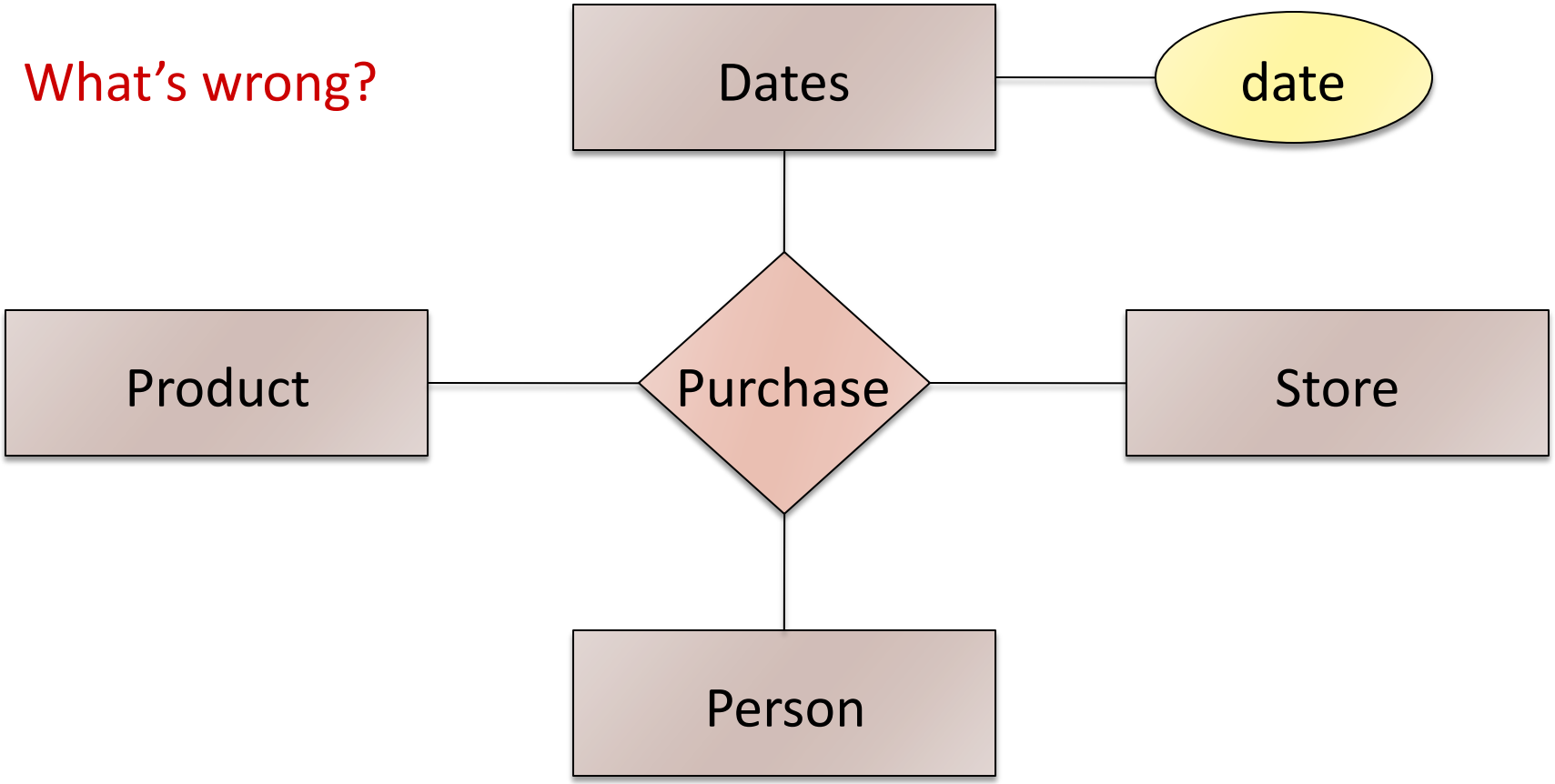
What's wrong?



Moral: pick the right kind of entities!

Design Principles

What's wrong?

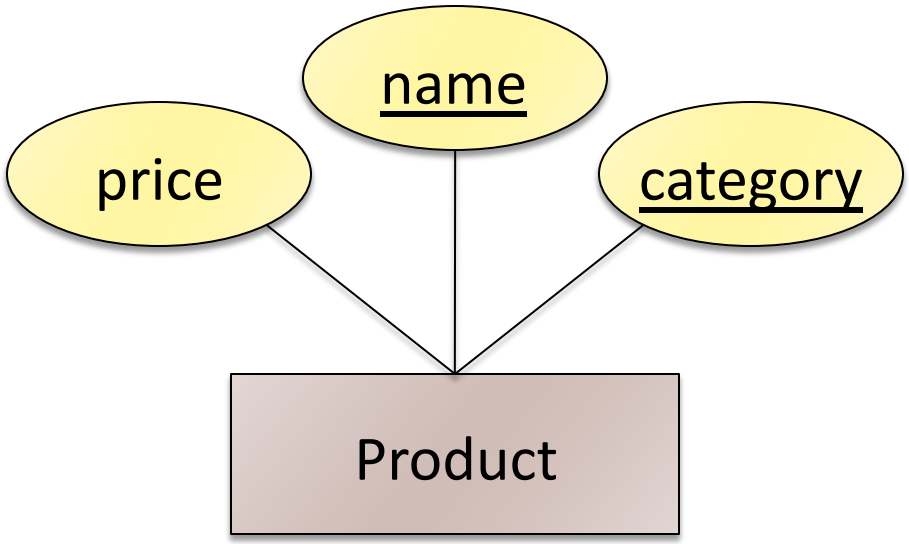


Moral: don't complicate life more than necessary!

From E/R Diagrams to Relational Schema

- Entity set → relation
- Relationship → relation

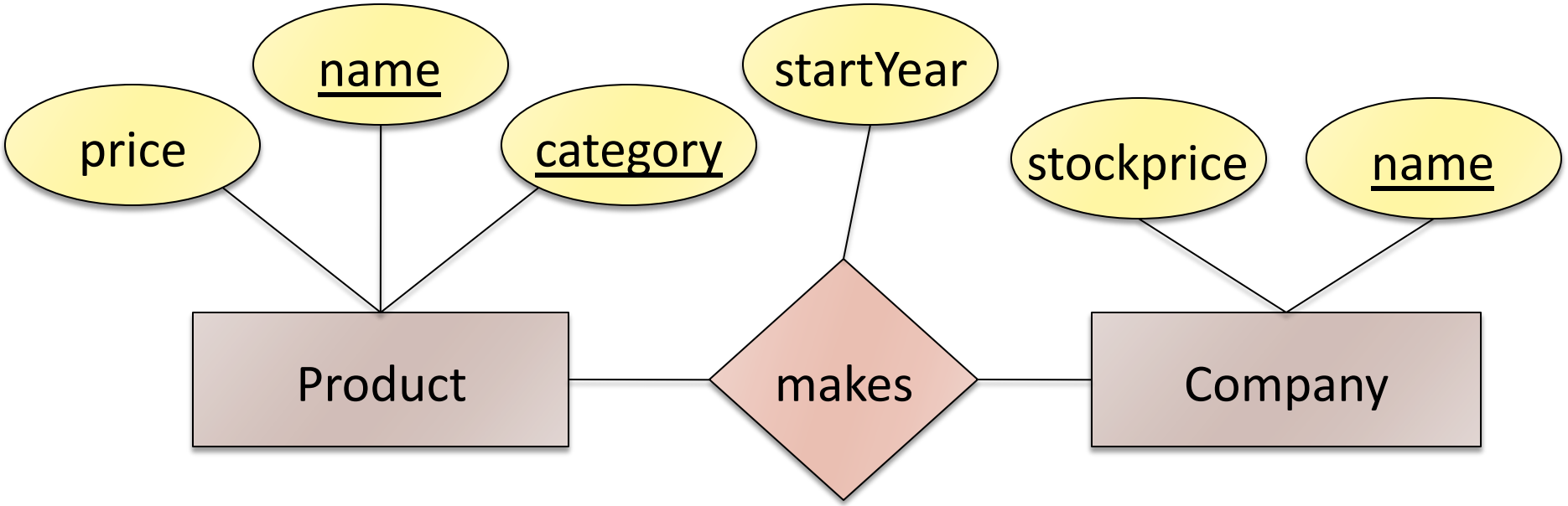
Entity Set to Relation



Product(<u>name, category</u>, price)

Name	Category	Price
Gizmo	Gadgets	$19.99

Relationships to Relations



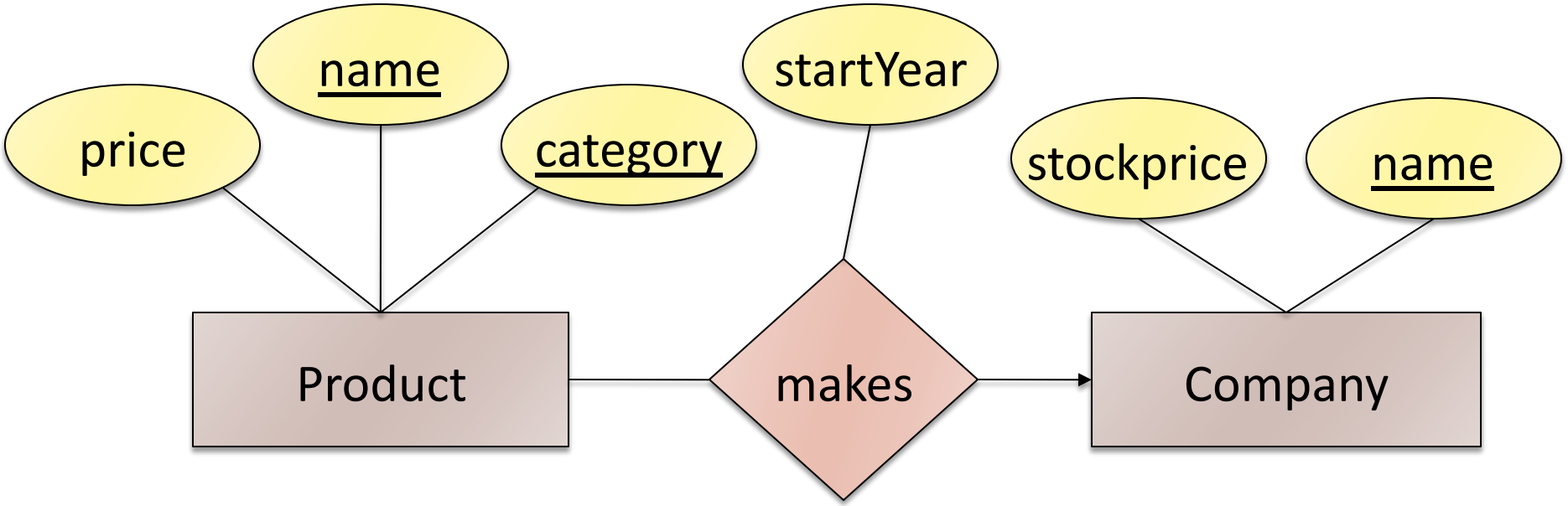
Watch out for attribute name conflicts

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

ProductName	ProductCategory	CompanyName	startYear
Gizmo	Gadgets	GizmoWorks	1963

Foreign keys

Relationships to Relations (with constraints)



Makes(prodName, category, companyName, year)

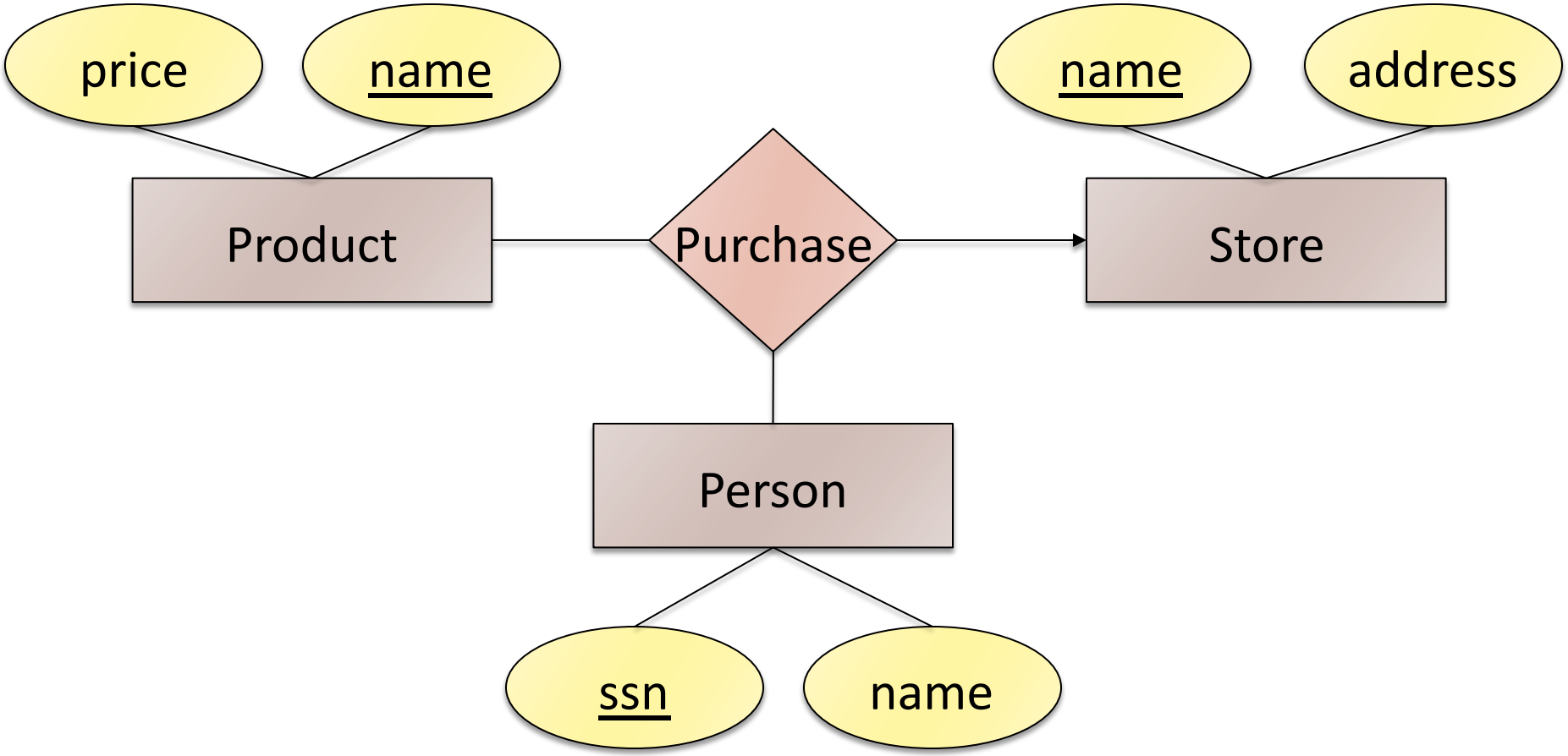
Only keep **Product** keys as primary key

Better solution: get rid of **Makes**, modify **Product**:

prodName	Category	Price	startYear	CompanyName
Gizmo	Gadgets	$19.99	1963	GizmoWorks

x

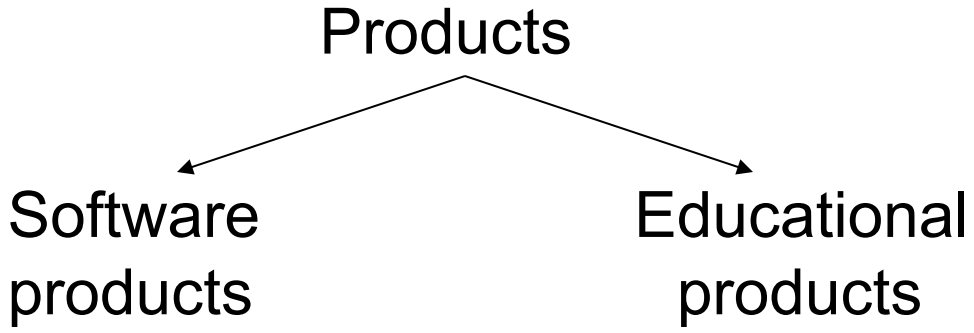
Multi-way Relationships to Relations



Purchase(prodName, storeName, ssn)

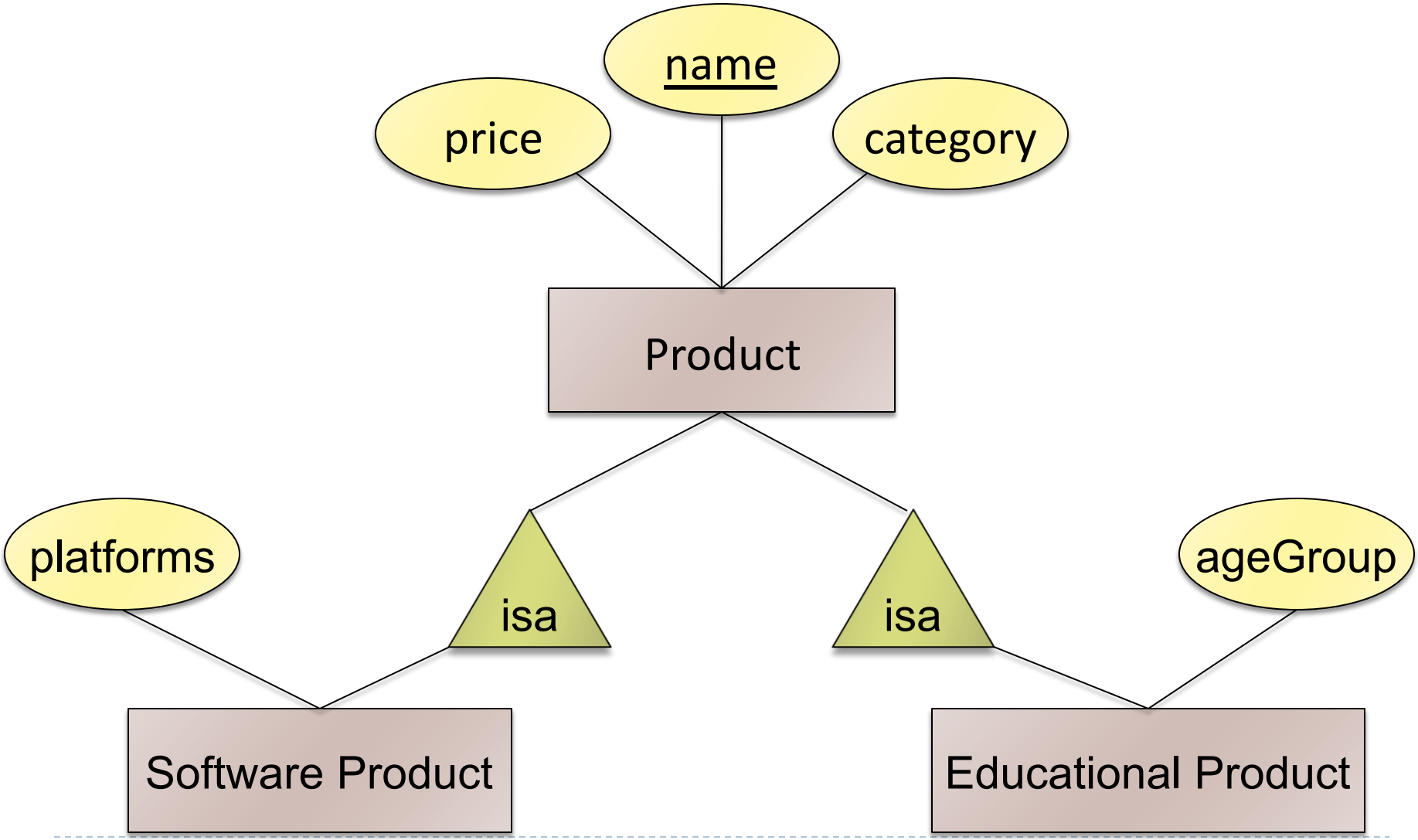
Modeling Subclasses

- Some objects in a class may be special
  - Define a new class
  - Better: define a *subclass*

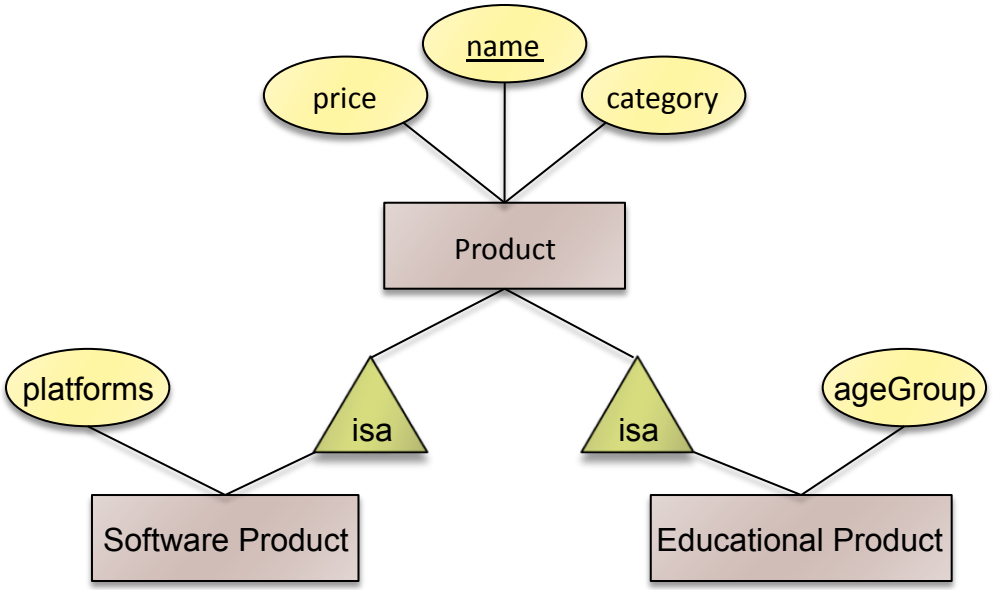


So --- we define subclasses in E/R

Subclasses


name
price
category
Product
platforms
isa
isa
ageGroup
Software Product
Educational Product

Subclasses to Relations



Product

Name	Price	Category
Gizmo	99	gadget
Camera	49	photo
Toy	39	gadget

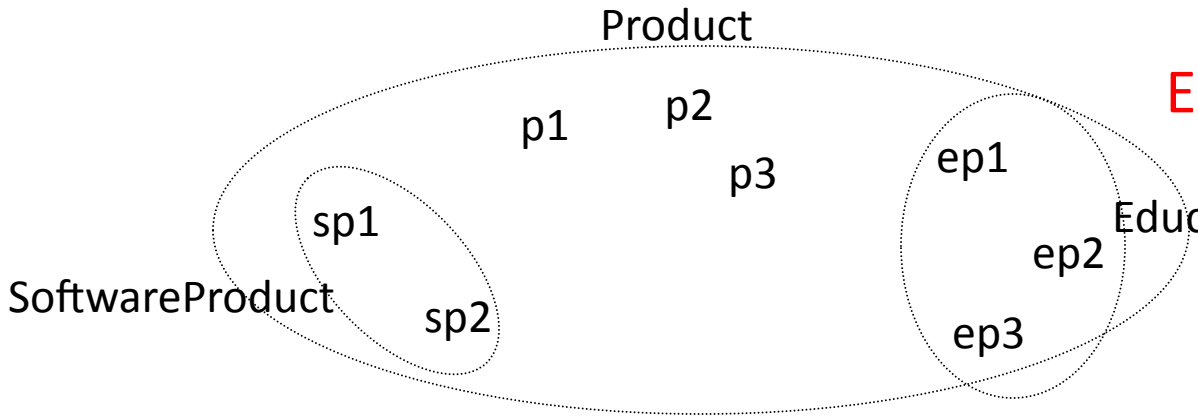
Software Product

Name	platforms
Gizmo	unix

Educational Product

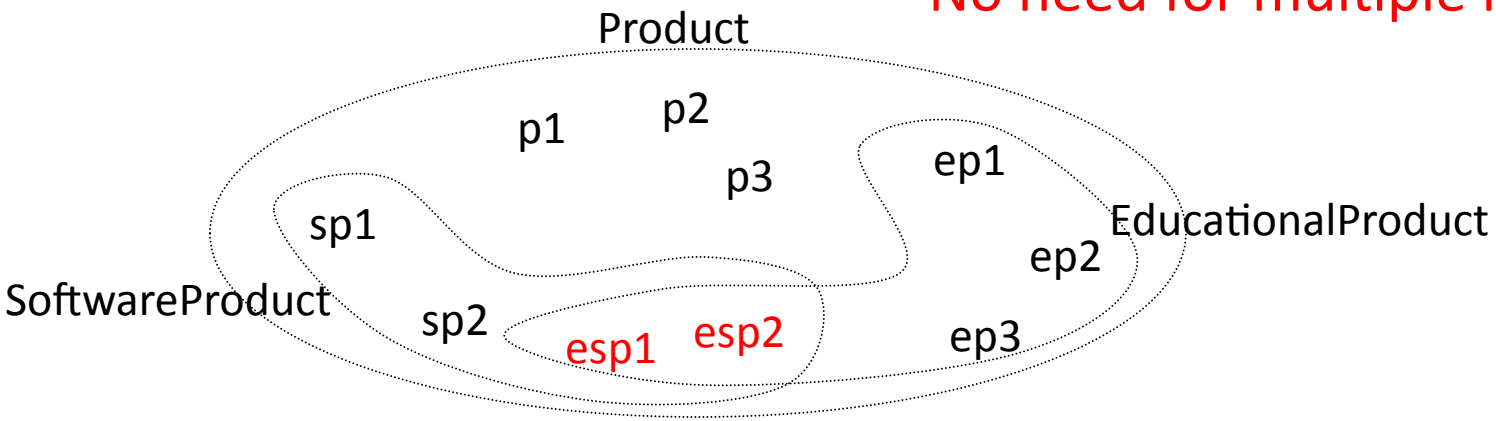
Name	ageGroup
Gizmo	todler
Toy	retired

E/R Inheritance

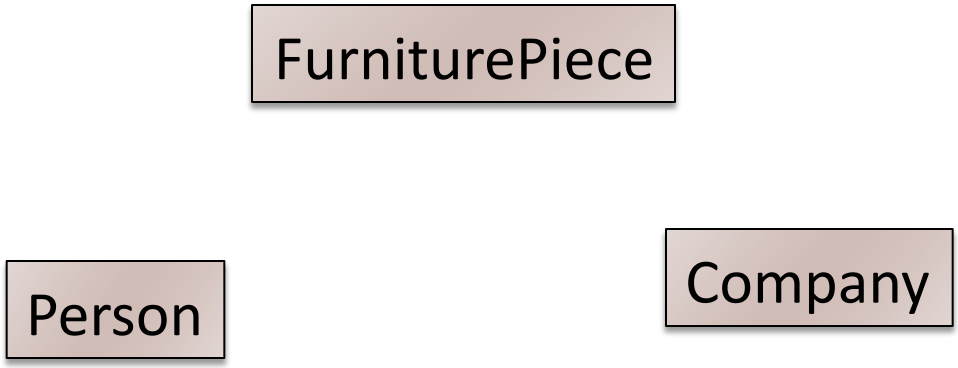


Entity sets overlap

No need for multiple inheritance



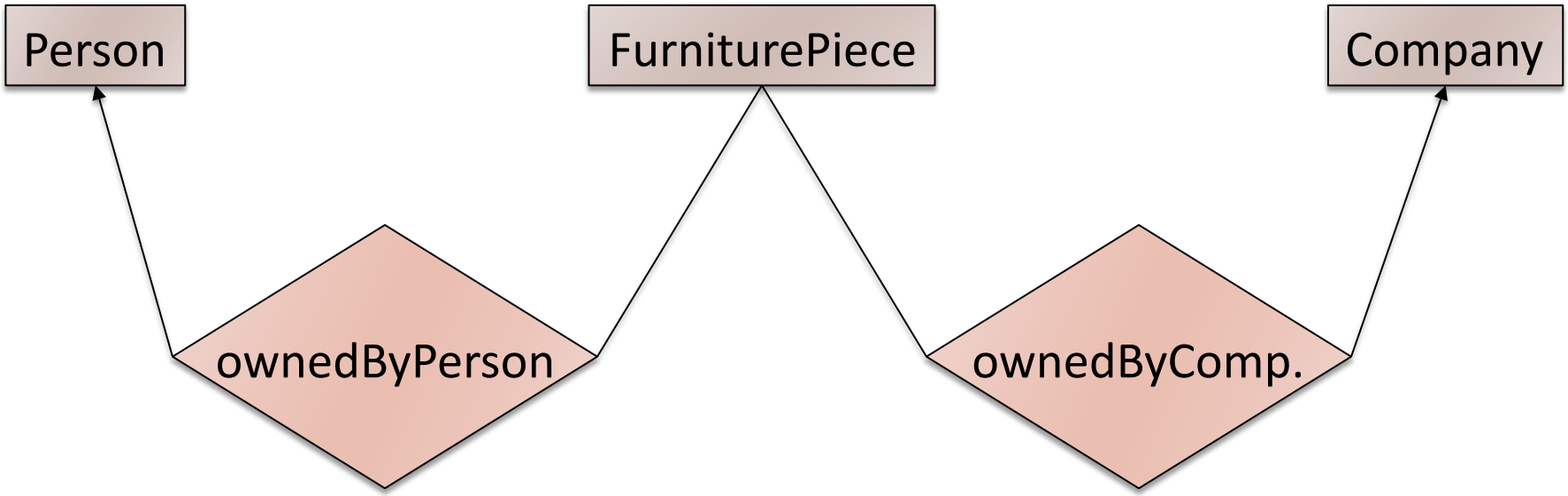
Modeling Union Types With Subclasses



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

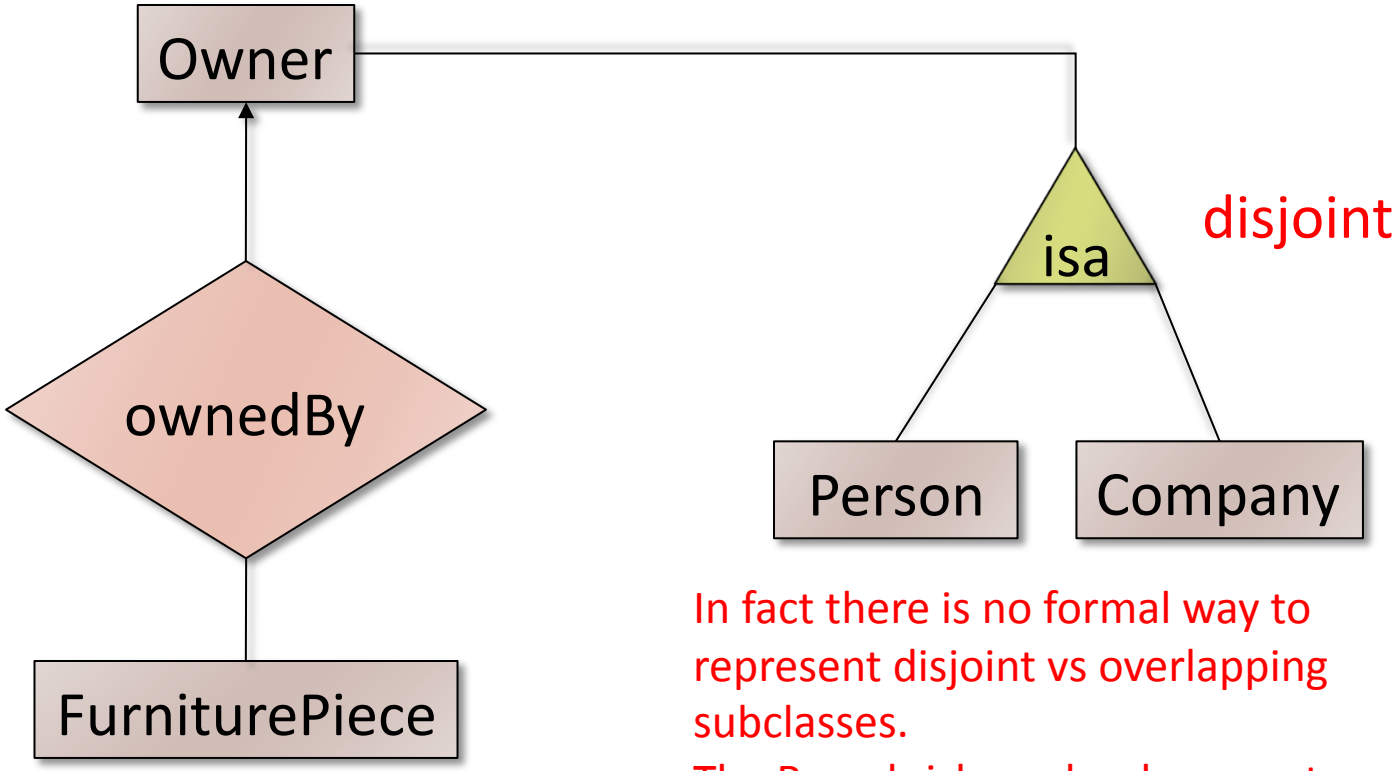
Modeling Union Types with Subclasses

- Solution 1: acceptable, but imperfect (why?)



Modeling Union Types with Subclasses

- Solution 2: better, more laborious



In fact there is no formal way to represent disjoint vs overlapping subclasses.
The Ramakrishnan book suggests writing it down.

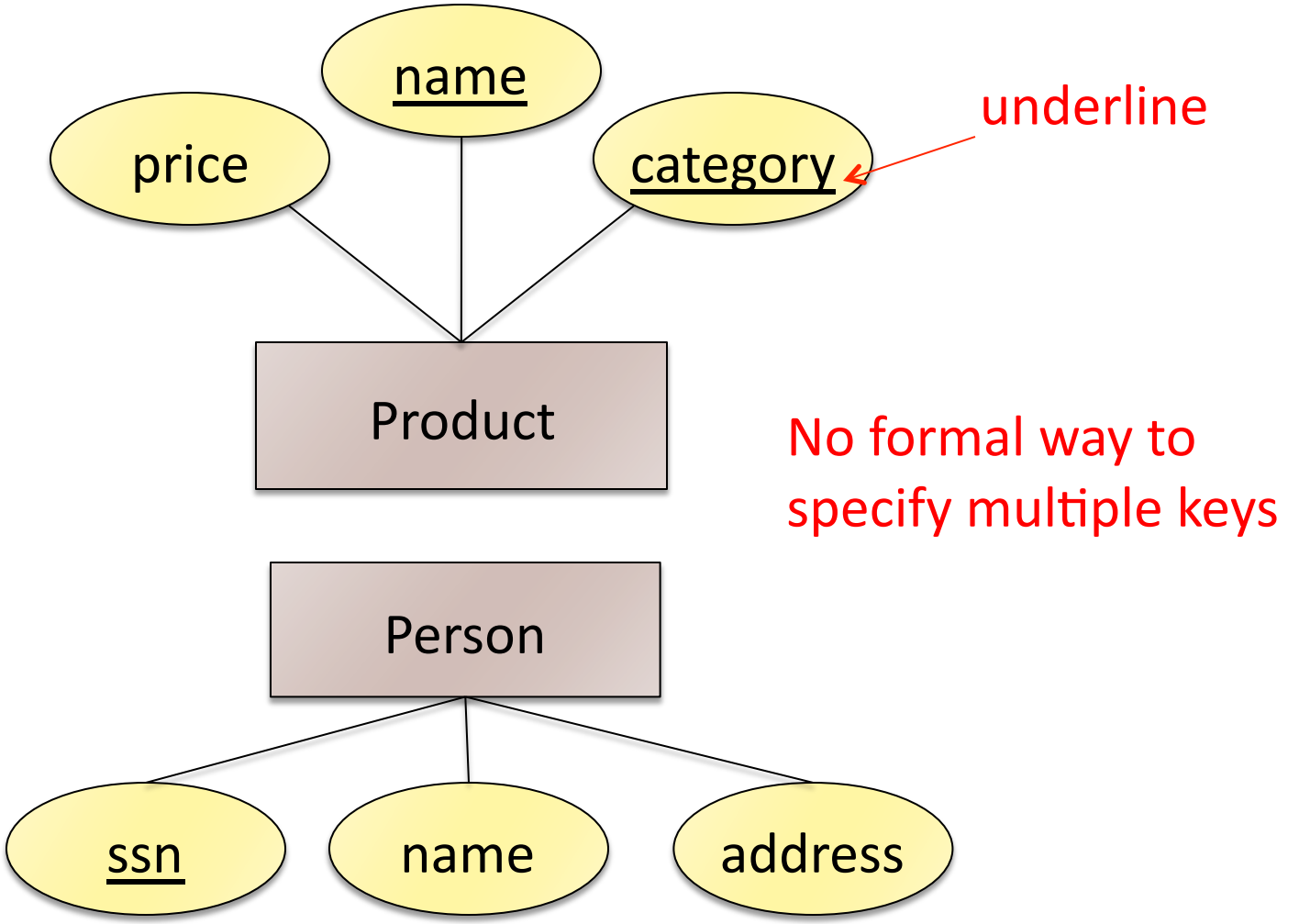
x

Constraints in E/R Diagrams

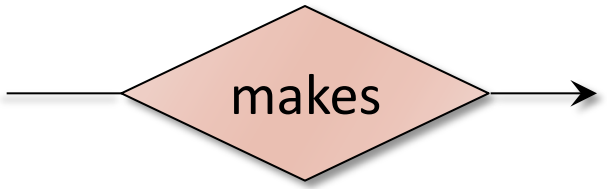
- Finding constraints is part of the modeling process.

- 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.

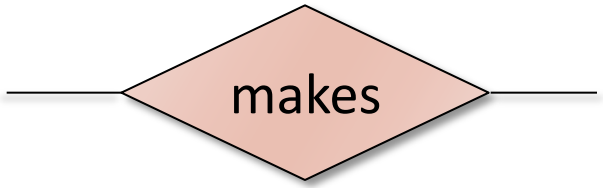
Keys in E/R Diagrams



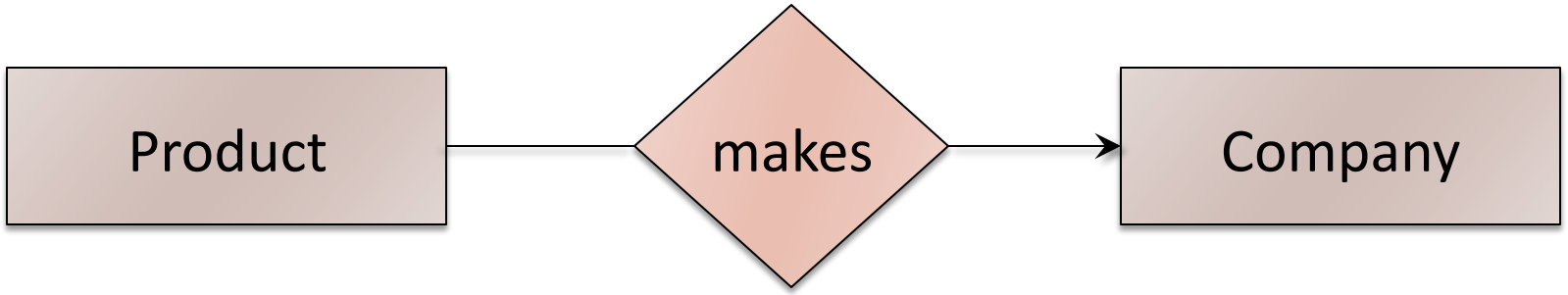
Single Value Constraints



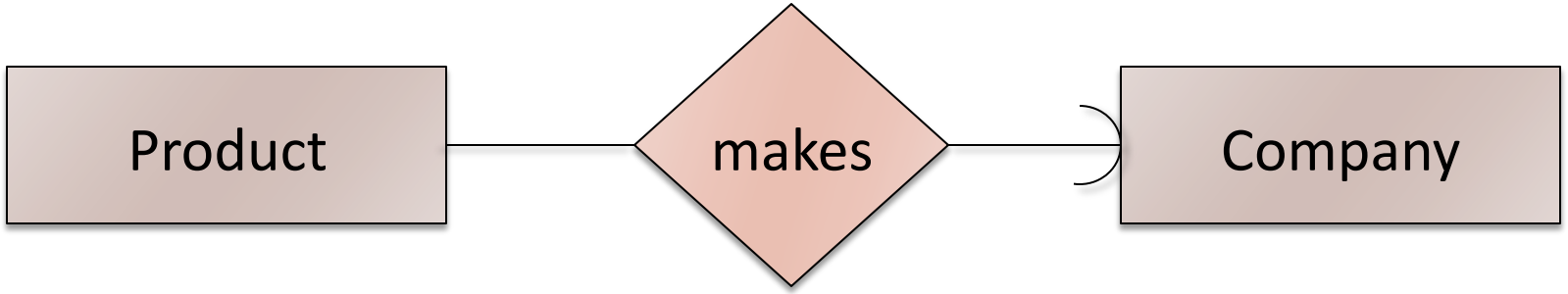
v. s.



Referential Integrity Constraints

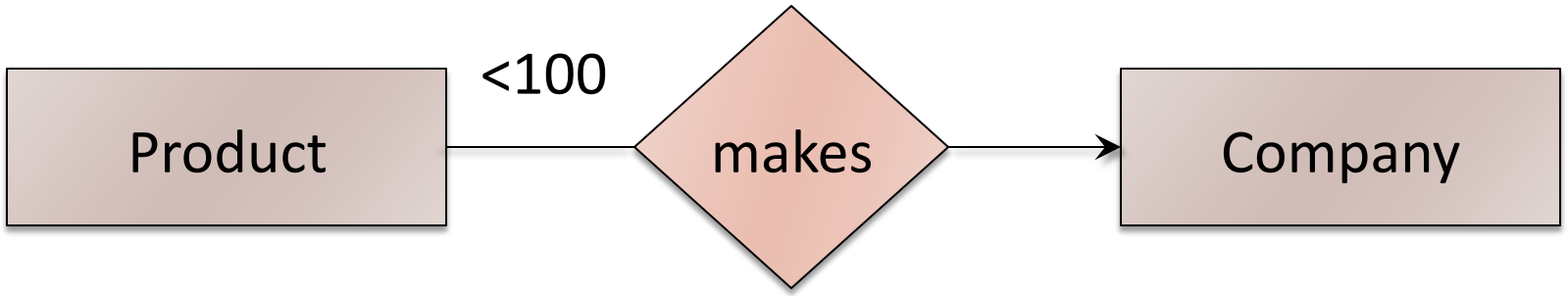


Each product made by at most one company.
Some products made by no company



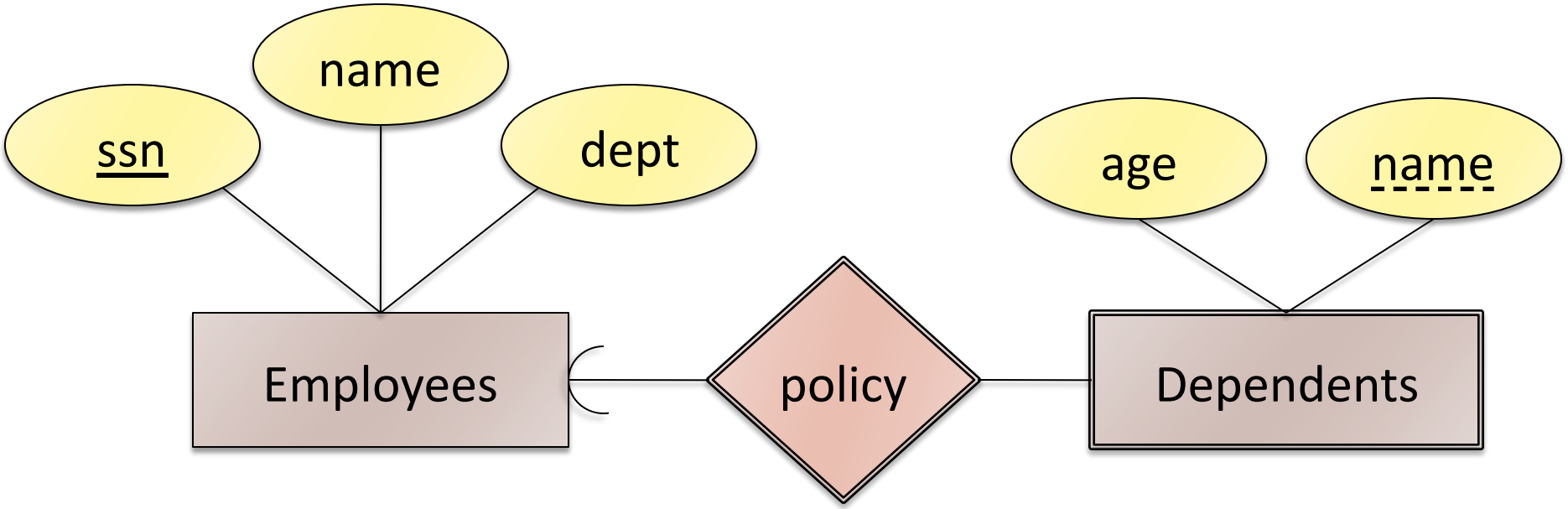
Each product made by *exactly* one company.

Other Constraints



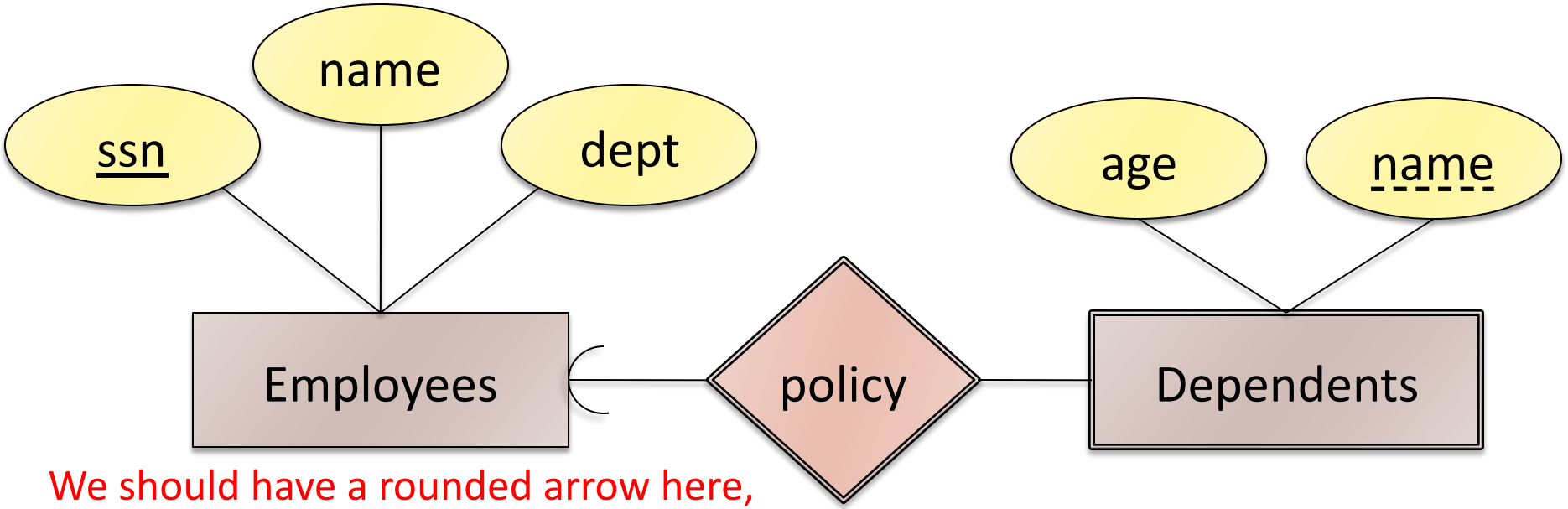
What does this mean ?

Weak Entity Sets



Entity sets are weak when their key comes from classes to which they're related

Handling Weak Entity Sets



We should have a rounded arrow here, otherwise Dependents key would have NULL value!

Employee(ssn, name, dept)
Dependents(ssn,name,age)

No need to represent policy separately