Entity-Relationship Model

CS 564- Fall 2016

How To Build a DB Application

- Pick an application
- Figure out what to model (ER model)
  - Output: ER diagram
- Transform the ER diagram to a relational schema
- Refine the relational schema (normalization)
- Now ready to implement the schema and load the data!

RUNNING EXAMPLE

We want to store information about:

- companies and employees
  - Each **company** has a name, an address, …
  - Each company has a list of **employees**
- products manufactured by these companies
  - Each **product** has a name, a description, …

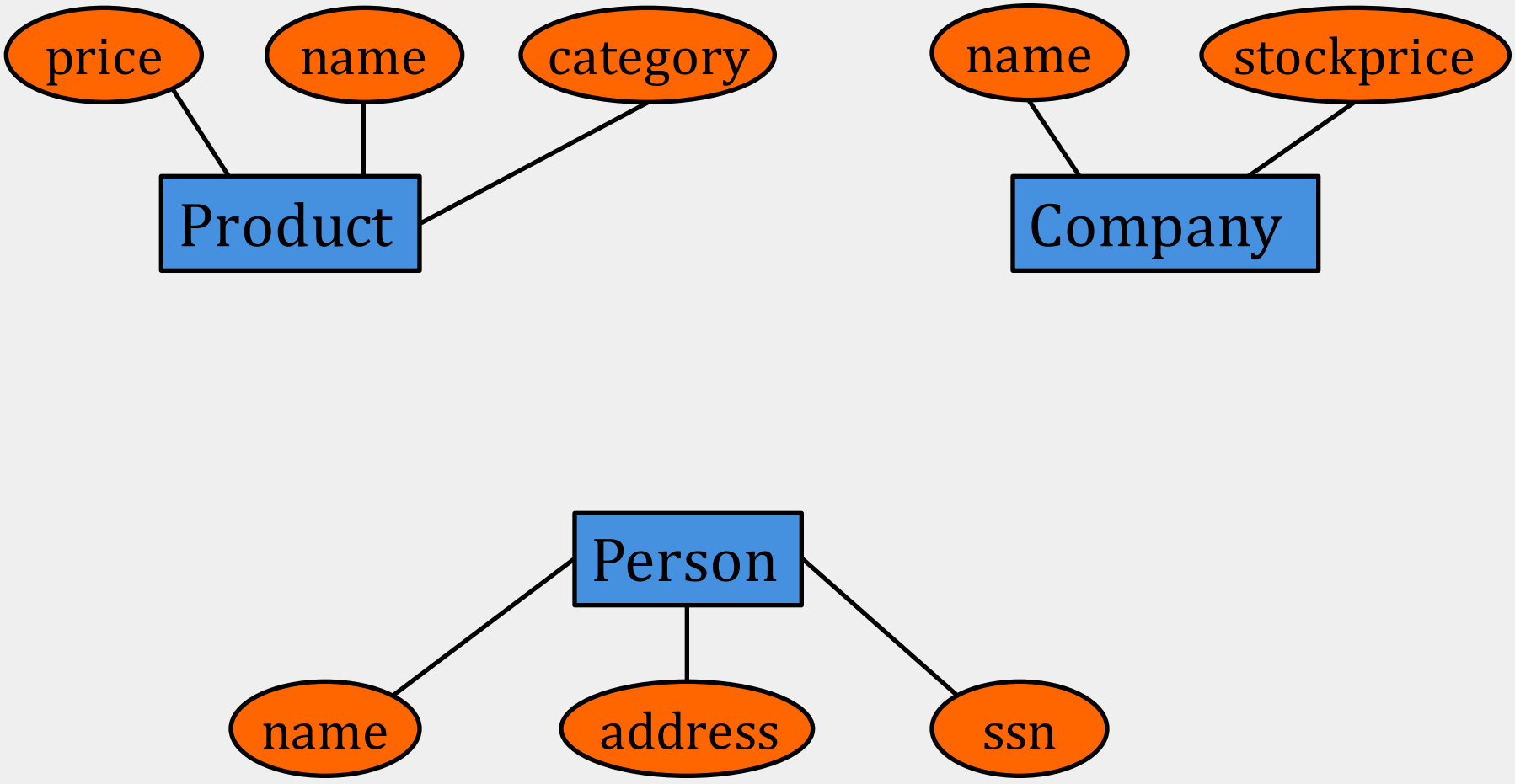
The ER Model

- Gives us a language to specify
  - what information the DB must hold
  - what are the relationships among components of that information
- Proposed by Peter Chen in 1976
- What we will cover:
  1. basic stuff: entities, attributes, relationships
  2. constraints
  3. weak entity sets
  4. design principles

ENTITIES & ATTRIBUTES

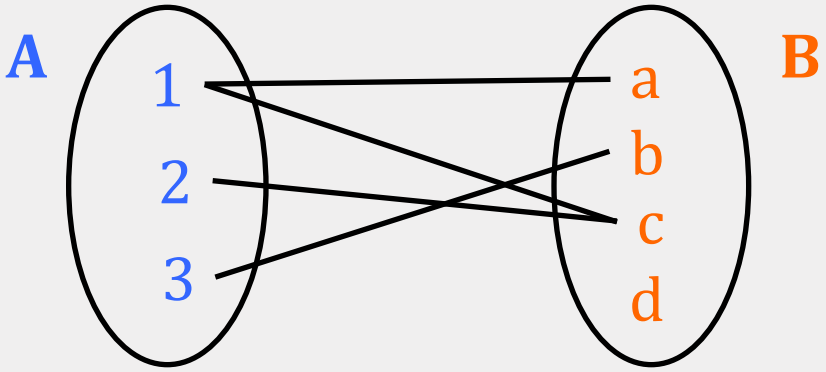
- **Entity**
  - an object distinguishable from other object
  - described using a set of attributes
- **Attribute**
  - each has an atomic domain: string, integer, …
- **Entity set**
  - a collection of similar entities

ENTITIES & ATTRIBUTES

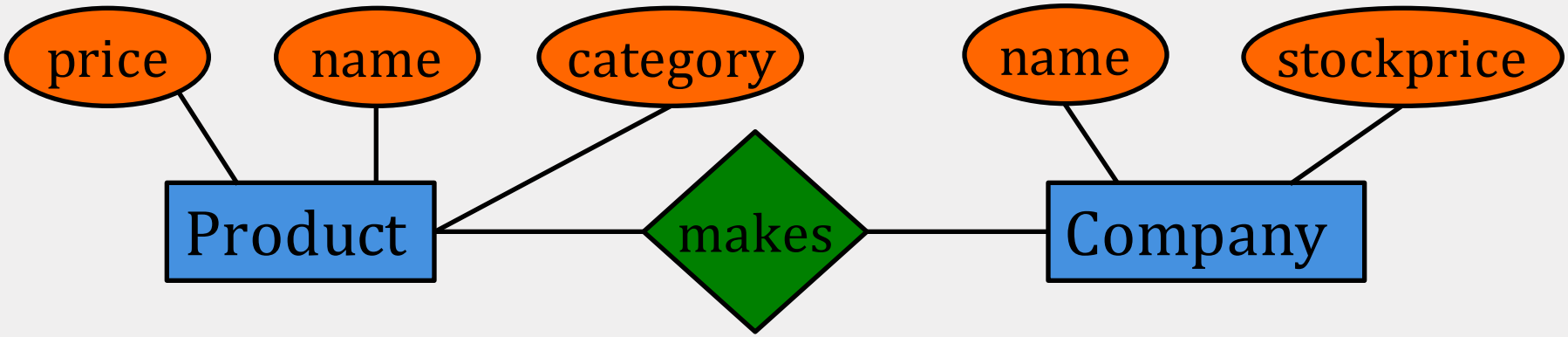


RELATIONS

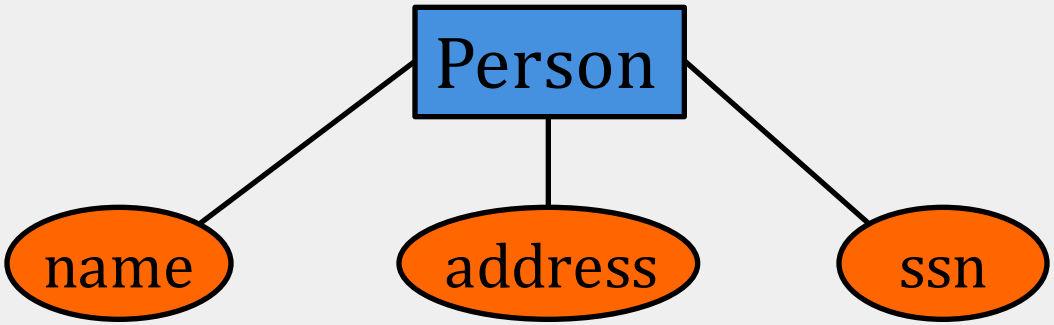
- A mathematical definition:
  - if **A**, **B** are sets, then a **relation R** is a subset of **A** x **B**
- Example
  - **A** = {1, 2, 3}, **B** = {a, b, c, d}
  - **R** = {(1, a), (1, c), (2, c), (3, b)}



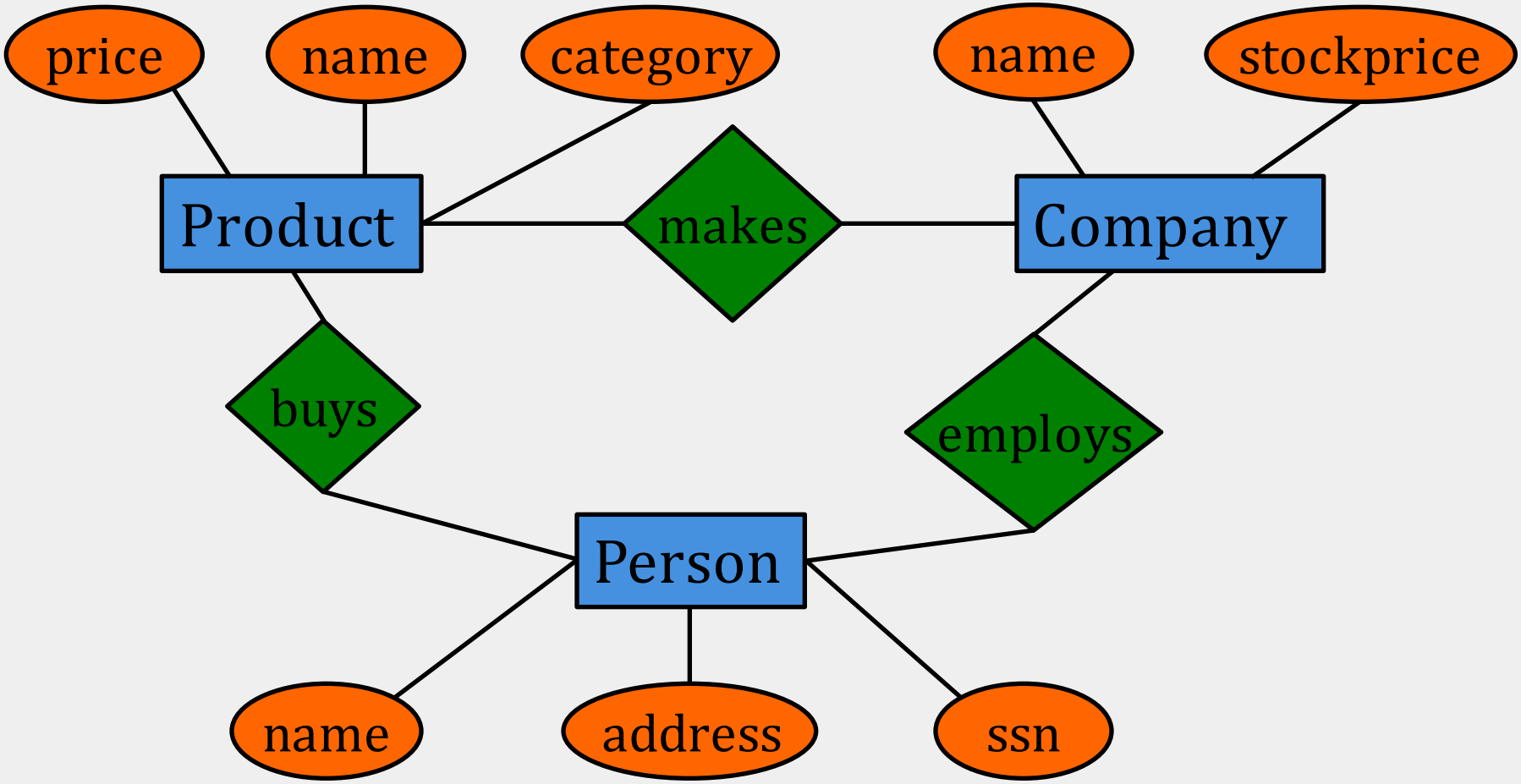
RELATIONSHIPS



makes is a subset of **Product** x **Company**

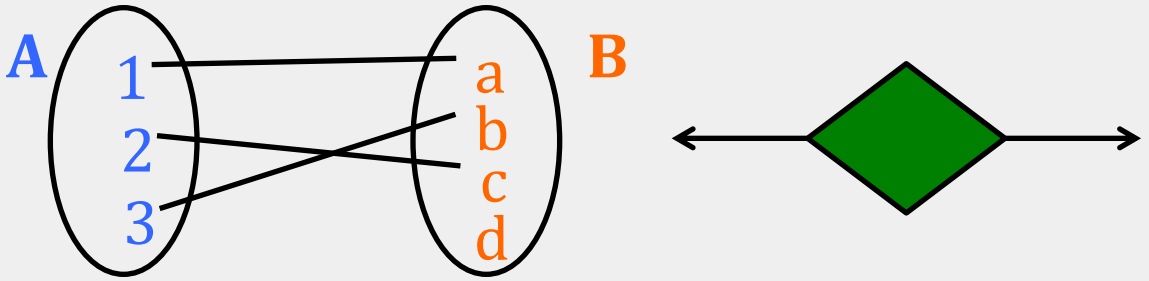


RELATIONSHIPS

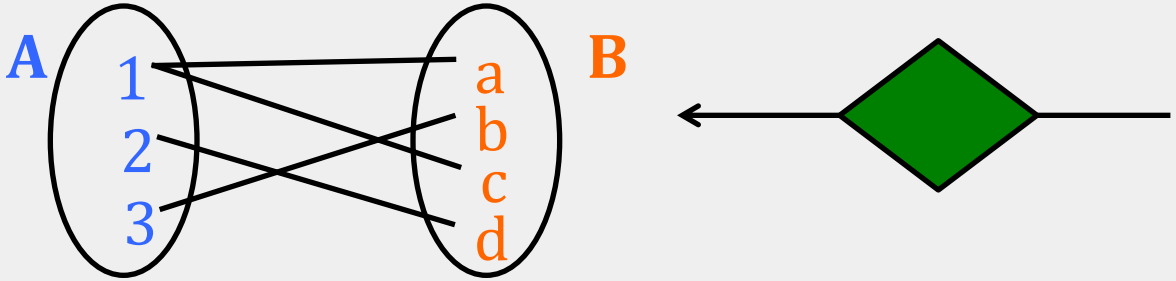


Multiplicity of Relationships

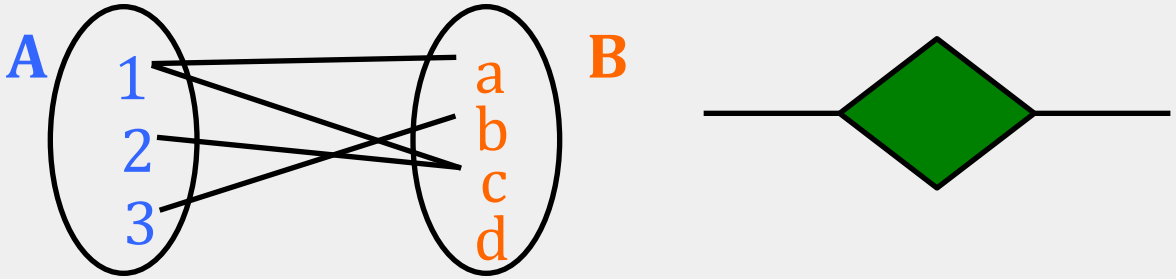
- one-one



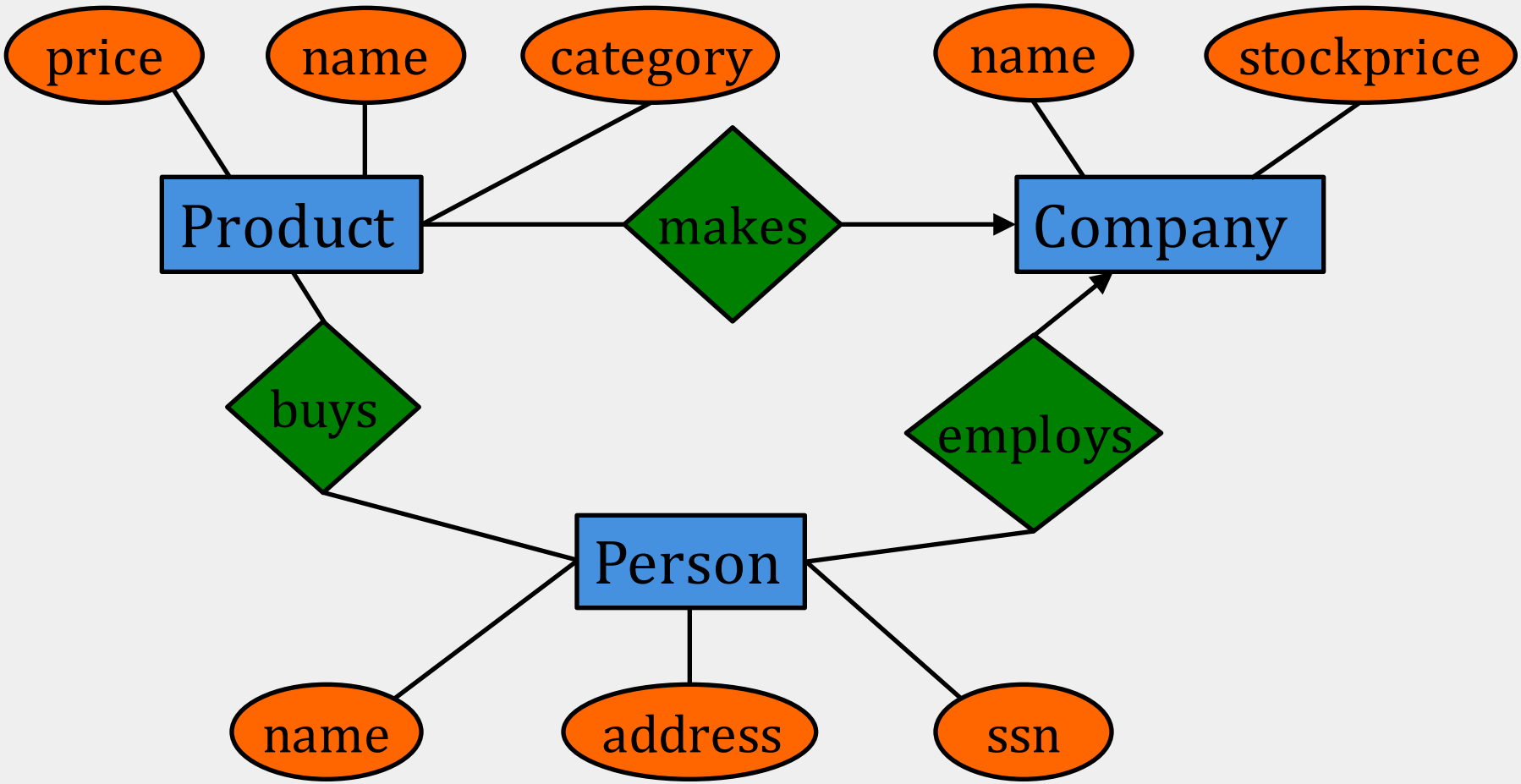
- many-one



- many-many

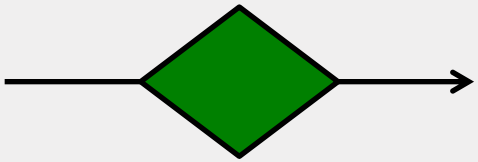


Multiplicity of Relationships

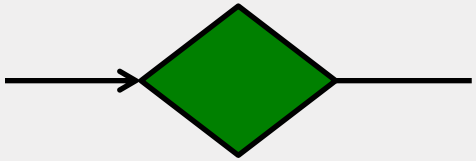


Notation Difference

- We use:



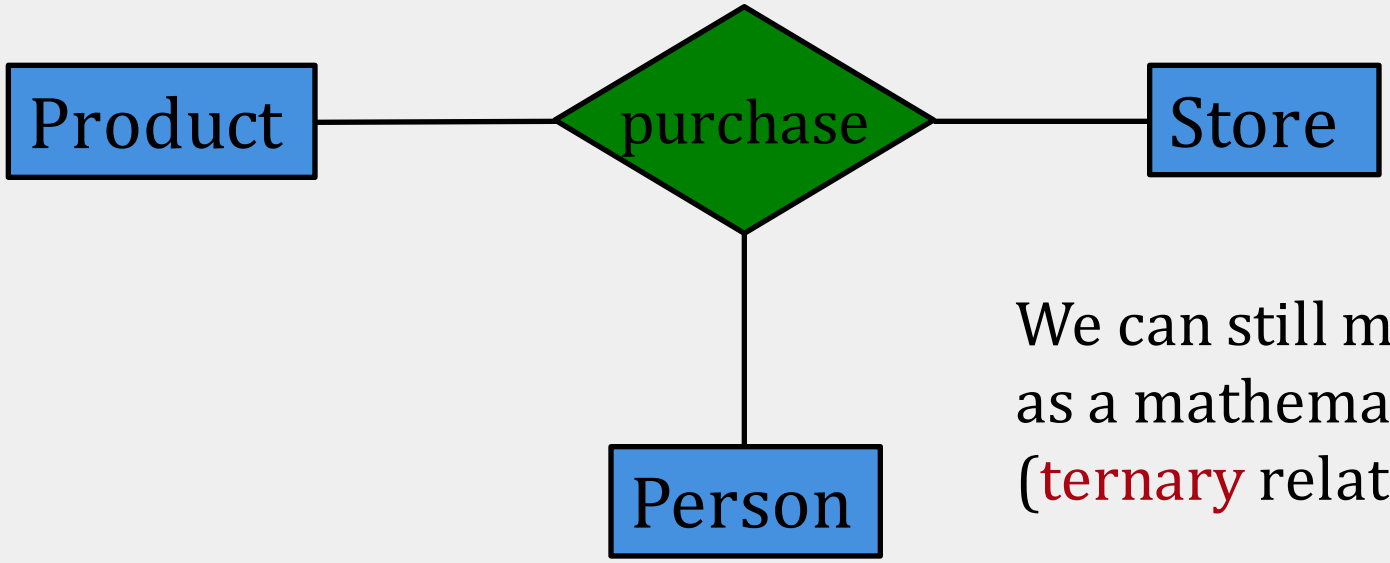
- The cow book uses (page 33):



You should use the notation in the slides!

MULTI-WAY RELATIONSHIPS

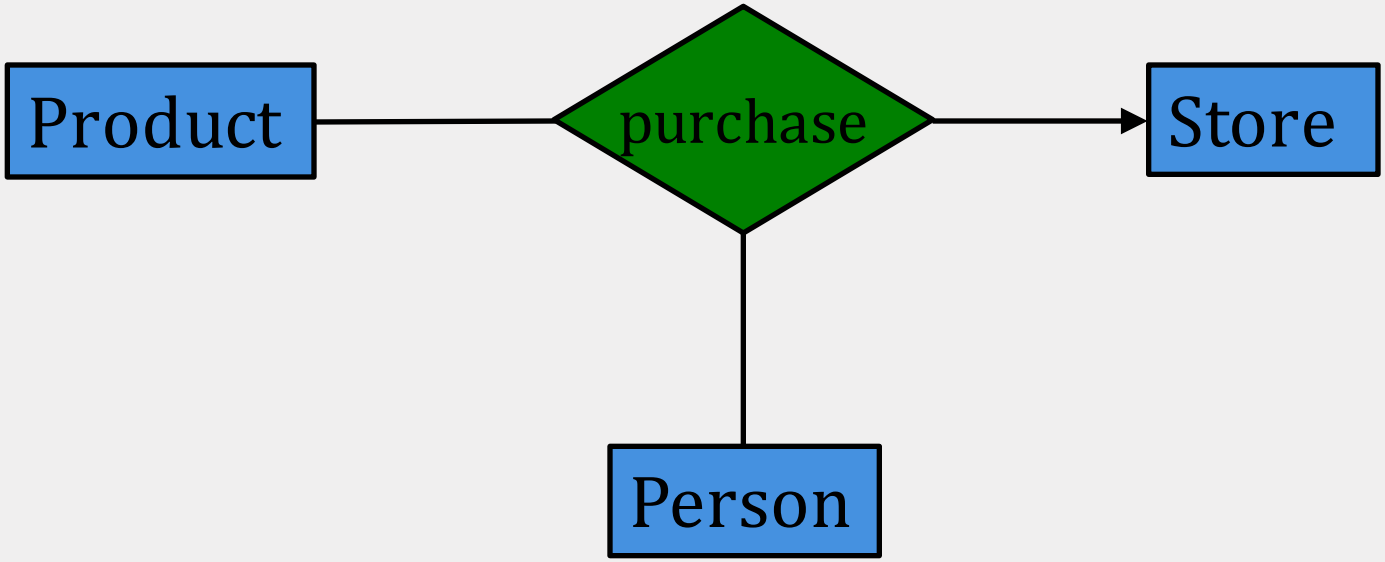
How do we model a **purchase** relation between **buyers**, **products** and **stores**?



We can still model this as a mathematical set (ternary relation)

Arrows in Multi-way Relationships

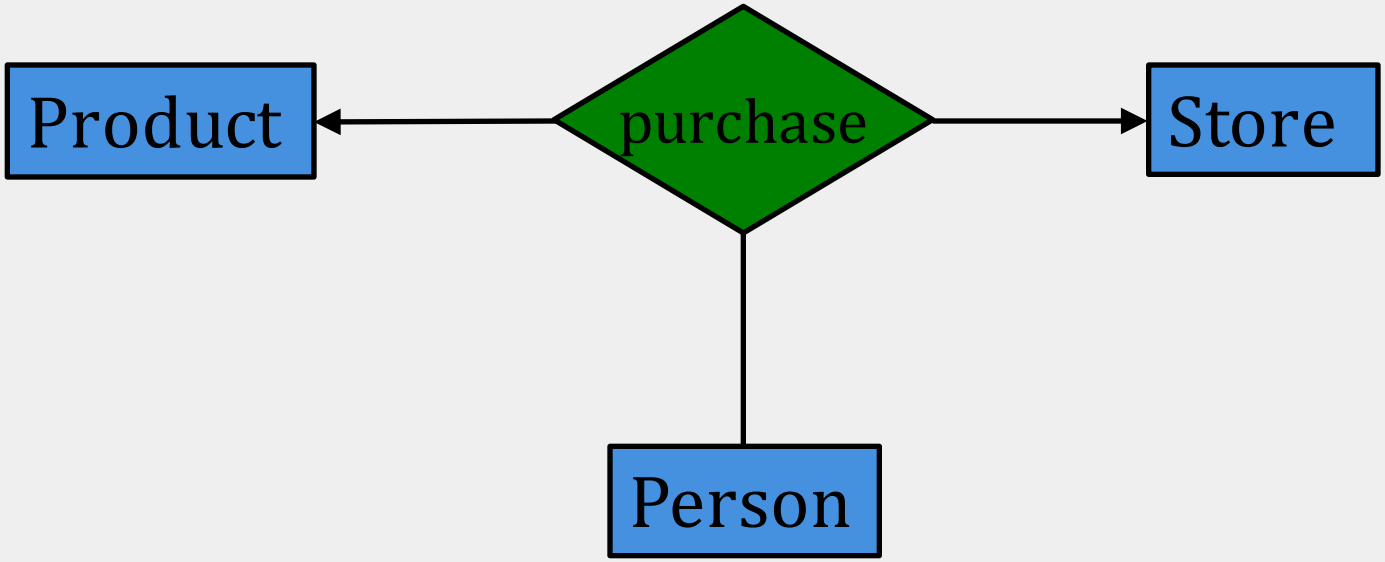
What does the arrow mean here?



A given **person** can **purchase** a given **product** from at most one **store**!

Arrows in Multi-way Relationships

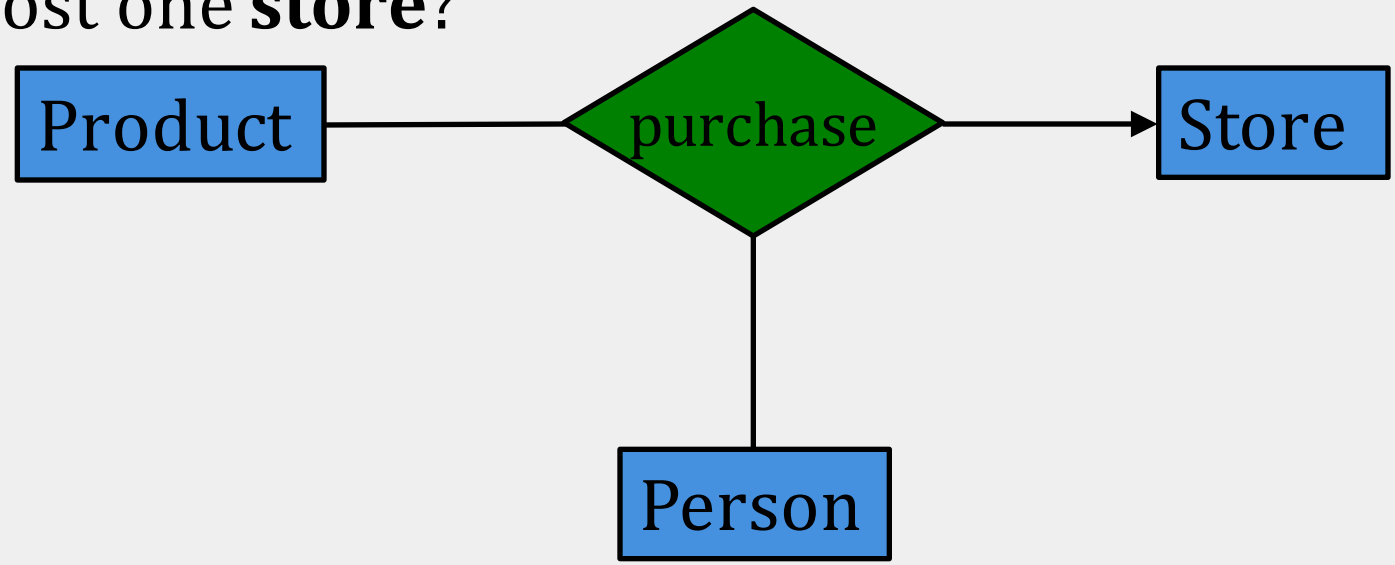
What about here?



A given **person** can **purchase** a given **product** from at most one **store** *AND* a given **store** sells to a given **person** at most one **product**

ARROWS IN MULTI-WAY RELATIONSHIPS

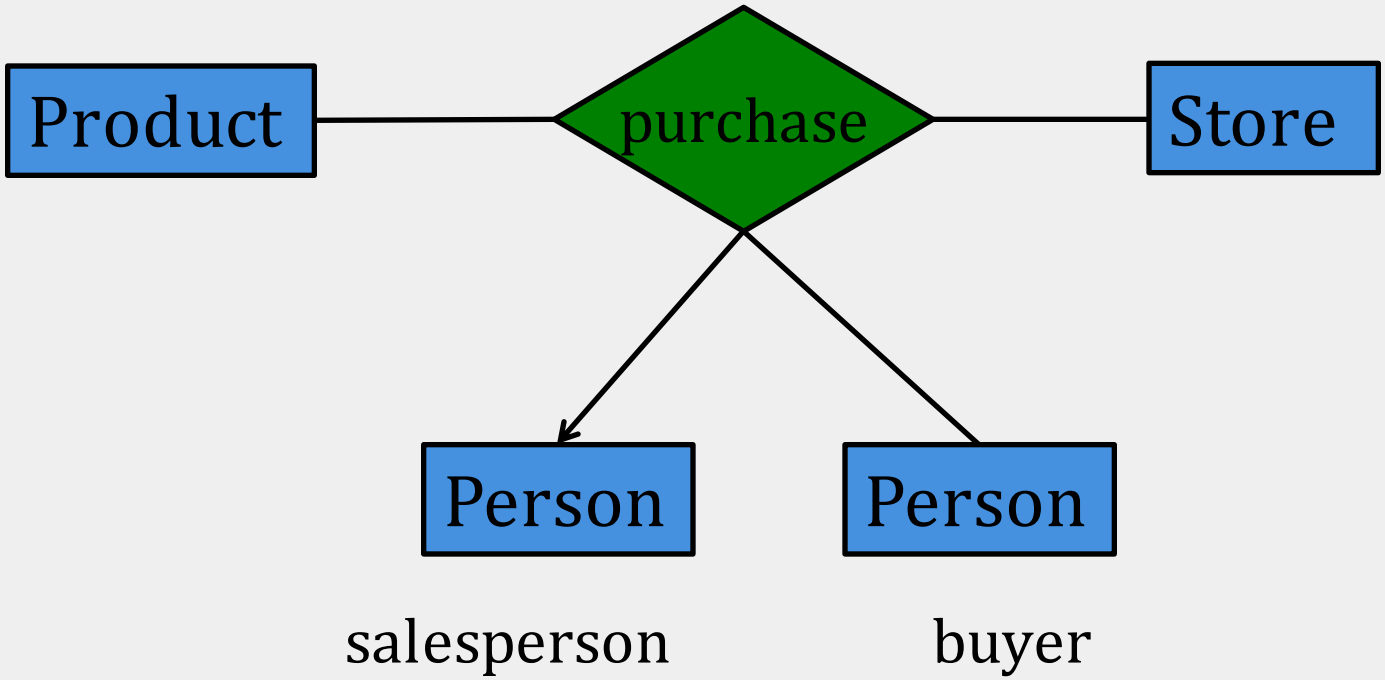
How can we say that a given **person** buys from at most one **store**?



Not possible, we can only approximate!

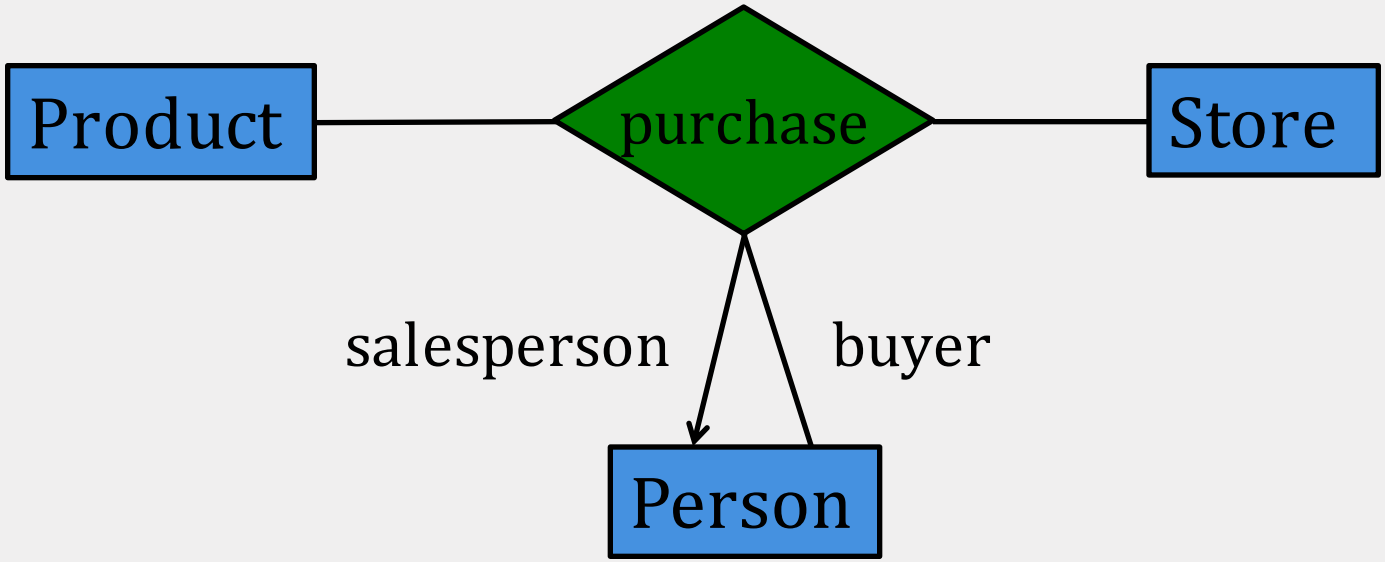
Roles in Relationships

What if we need an entity set twice in a relationship?

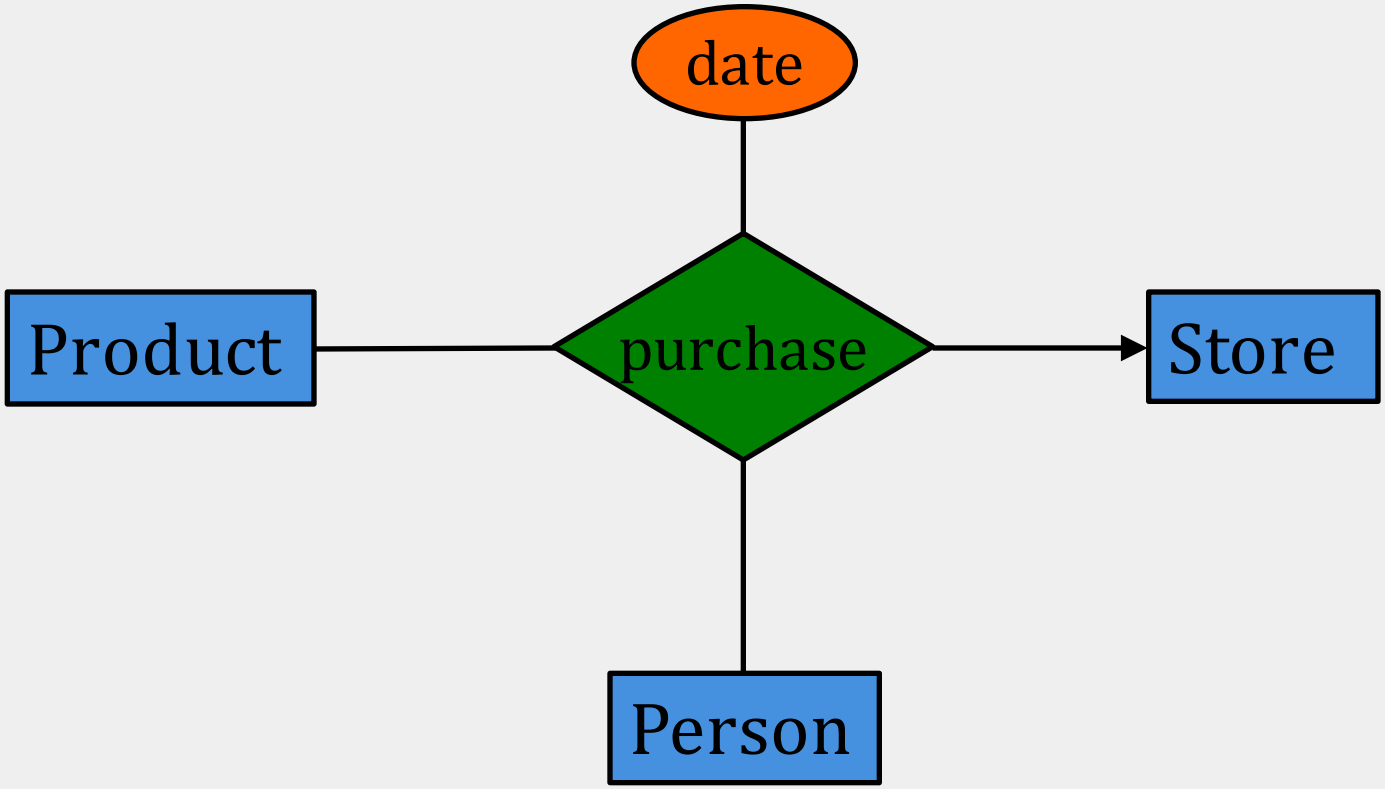


Roles in Relationships

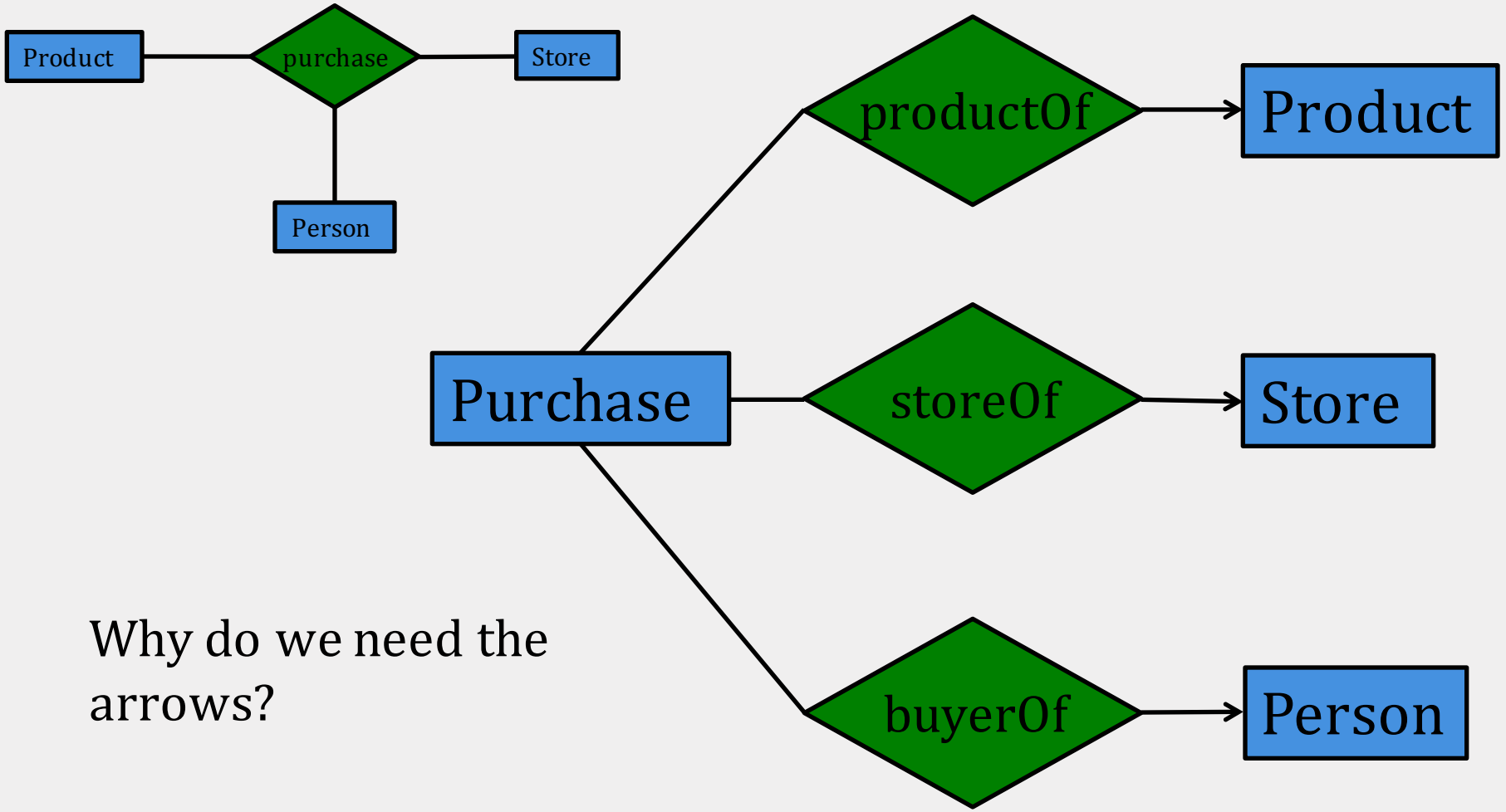
- Label the edges to indicate the roles
- Collapse the two entity sets into one



Attributes in Relationships



Multi-way to Binary Relationships



Why do we need the arrows?

Relationships: Recap

- Modeled as a mathematical set
- Binary and multi-way relationships
- Converting a multi-way one into many binary ones
- Constraints on the degree of the relationship
  - many-one, one-one, many-many
  - limitations of arrows
- Attributes of relationships
  - not necessary, but useful

Additional Features

SUBCLASSES

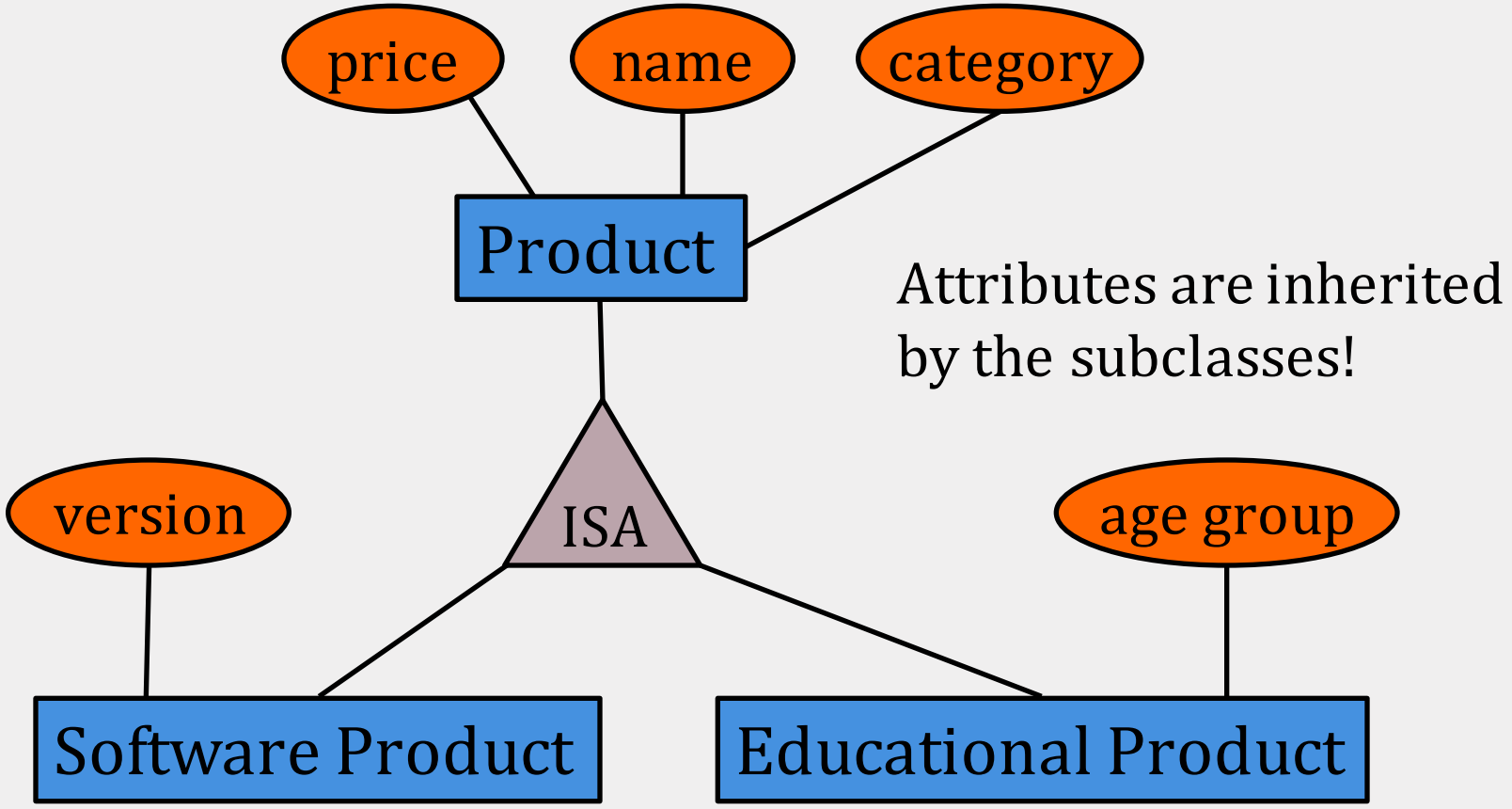
subclass = specialized case

= fewer entities

= more properties

- Example: Products
  - Software products
  - Educational products

SUBCLASSES


price
name
category
Product
Attributes are inherited by the subclasses!
version
ISA
age group
Software Product
Educational Product

CONSTRAINTS

constraint = an assertion about the database that must be true at all times

- part of the database schema
- central in database design

When creating the ER diagram, you need to find as many constraints as possible!

TYPES OF CONSTRAINTS

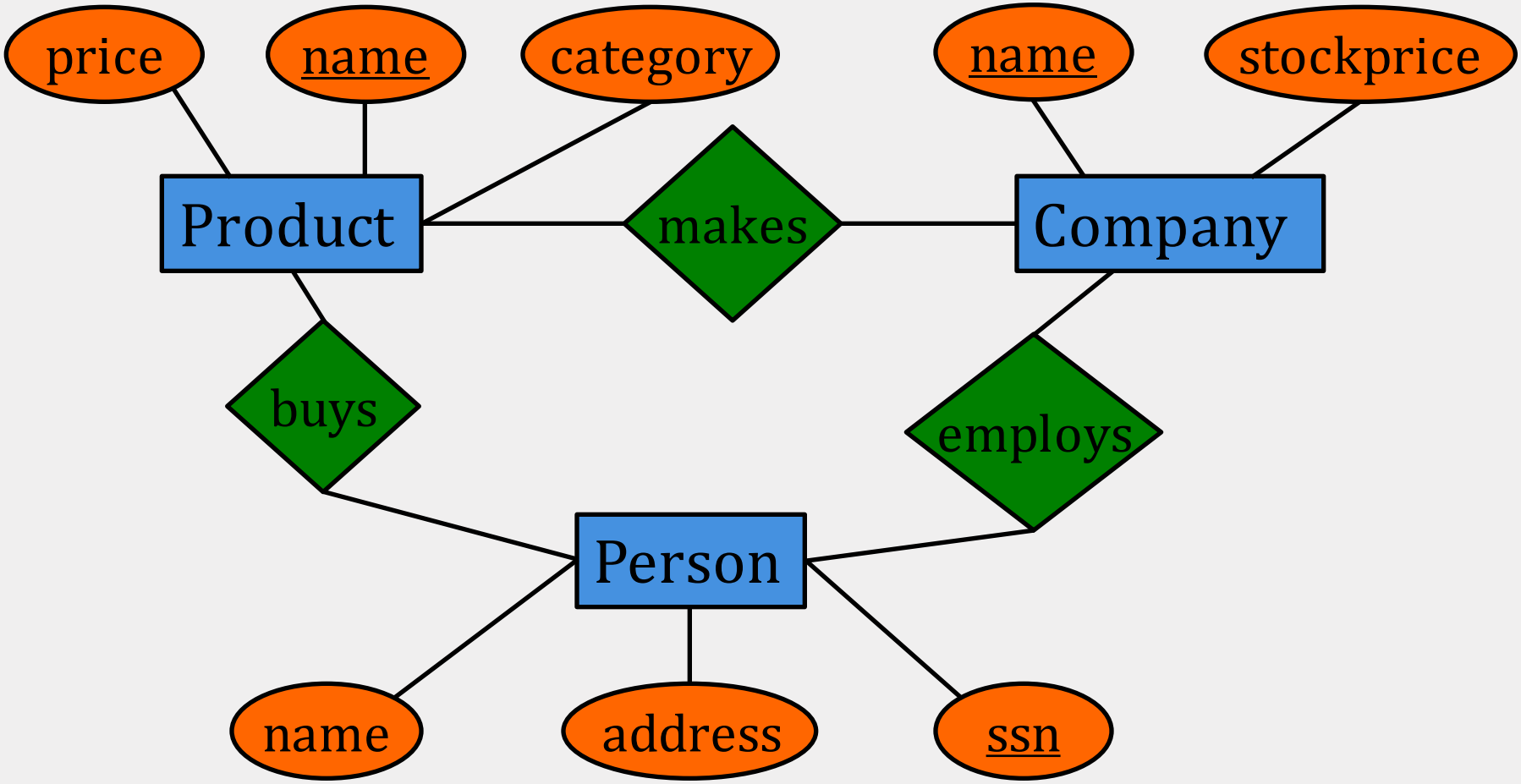
- **keys**: SSN uniquely identifies a person
- **single-value**: a person can have only one father
- **referential integrity**: if you work for a company, it must exist in the database
- **domain**: age is between 0 and 150

- **other**: e.g. at most 80 students enroll in a class

Why do we Need Constraints?

- Give more semantics to the data
  - help us better understand it
- Prevent wrong data entry
- Allow us to refer to entities (e.g. using keys)
- Enable efficient storage and data lookup

KEY CONSTRAINTS



KEY CONSTRAINTS

- *Every entity set must have a key*
- A key can consist of more than one attribute
- There can be more than one key for an entity set
  - one key will be designated as primary key
- No formal way to specify multiple keys in an ER diagram

SINGLE-VALUE CONSTRAINTS

An entity may have **at most one value** for a given attribute or relationship

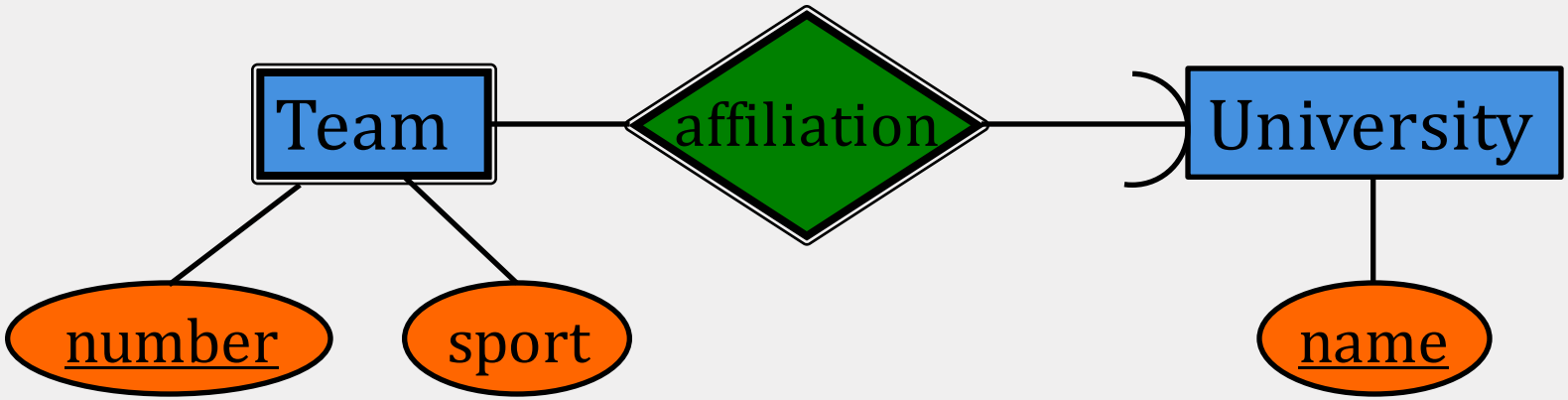
- an attribute of an entity set has a single value
- a many-one relation implies a single value constraint

Referential Integrity Constraint

A relationship has **one value** and the value must exist

Weak Entity Sets

Entity sets are **weak** when their key attributes come from other classes to which they are related

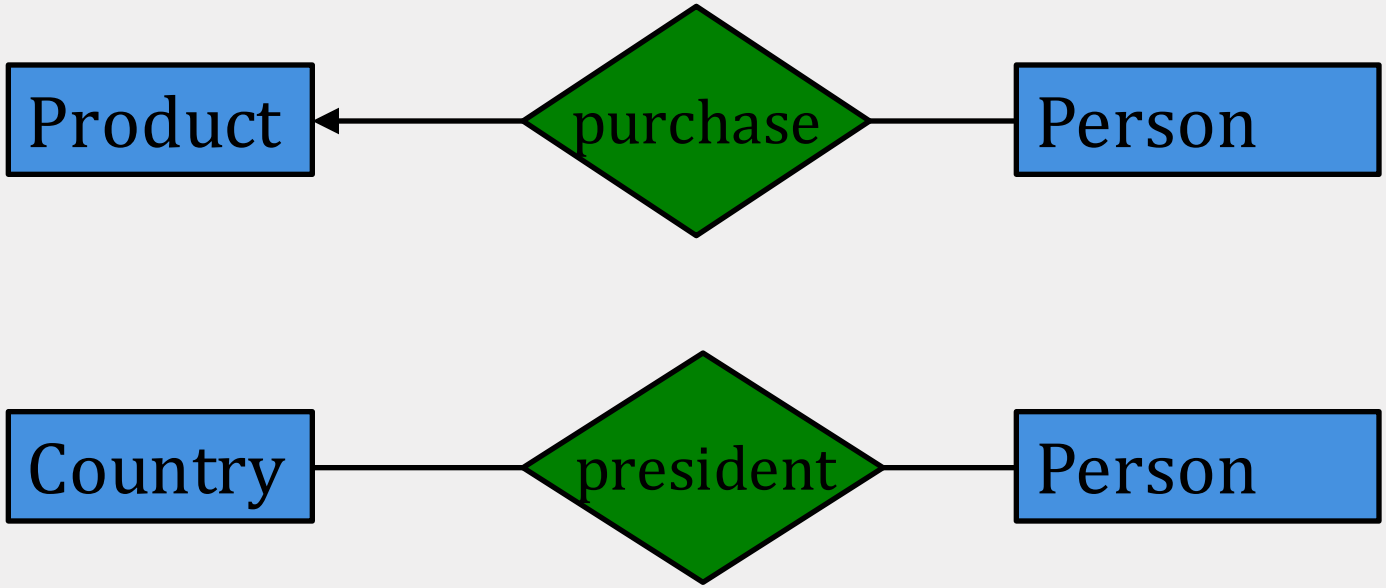


entities of an entity set need "help" to identify them uniquely!

Design Principles

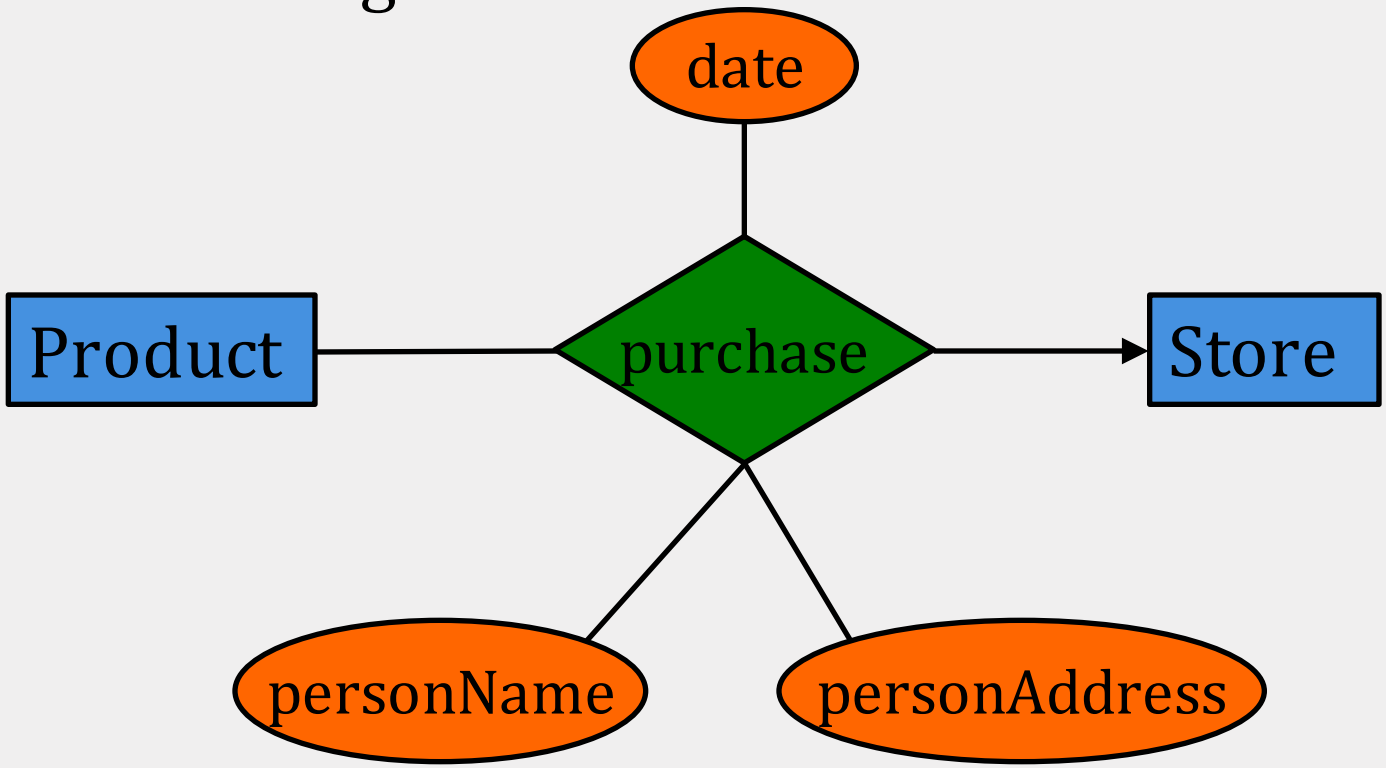
1. Be Faithful to the App!

What is wrong here?



2. Avoid Redundancy!

What is wrong here?

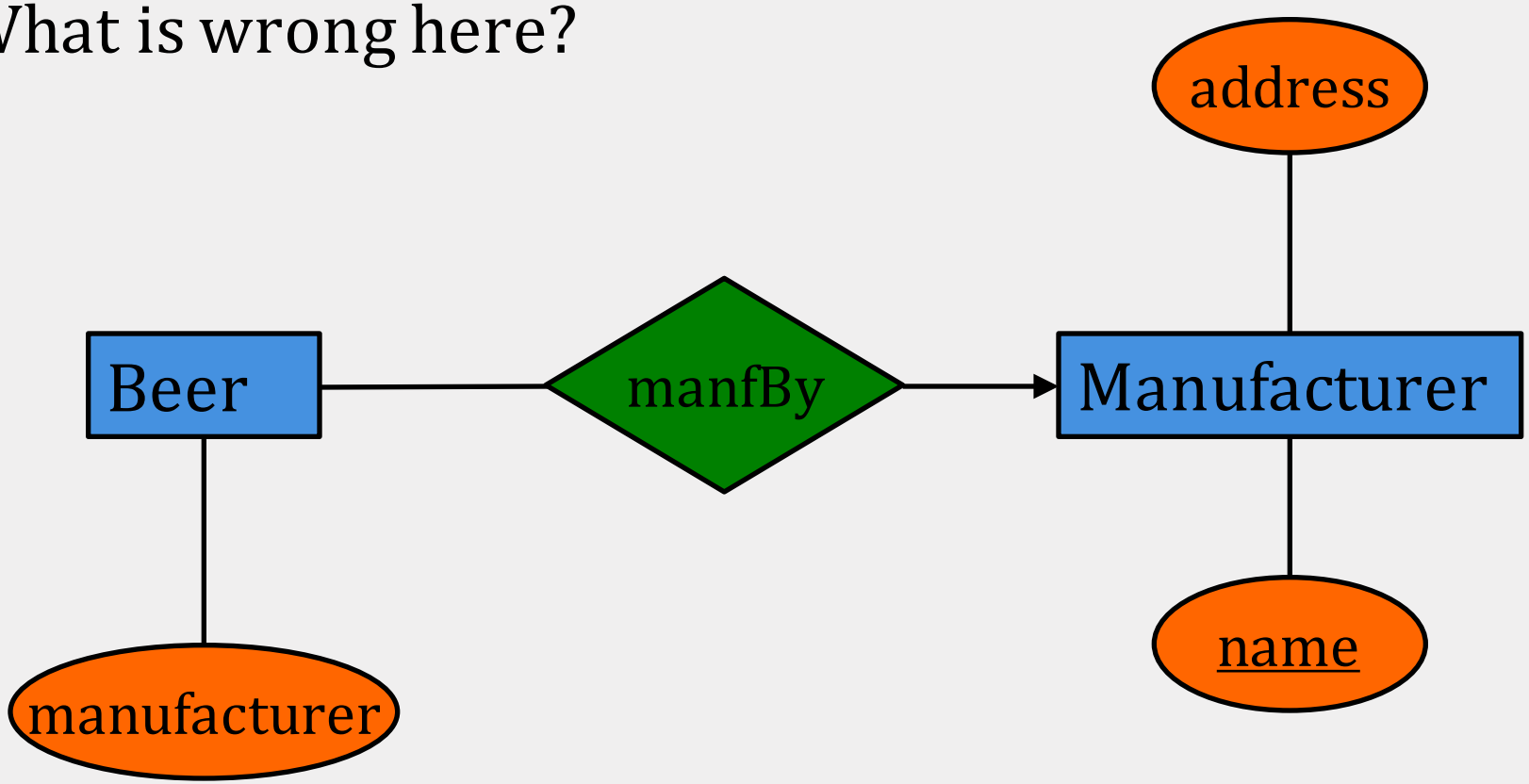


2. Avoid Redundancy!

- Redundancy occurs when we say the same thing in two different ways
- Redundancy wastes space and encourages inconsistency
  - The two instances of the same fact may become inconsistent if we change one and forget to change the other

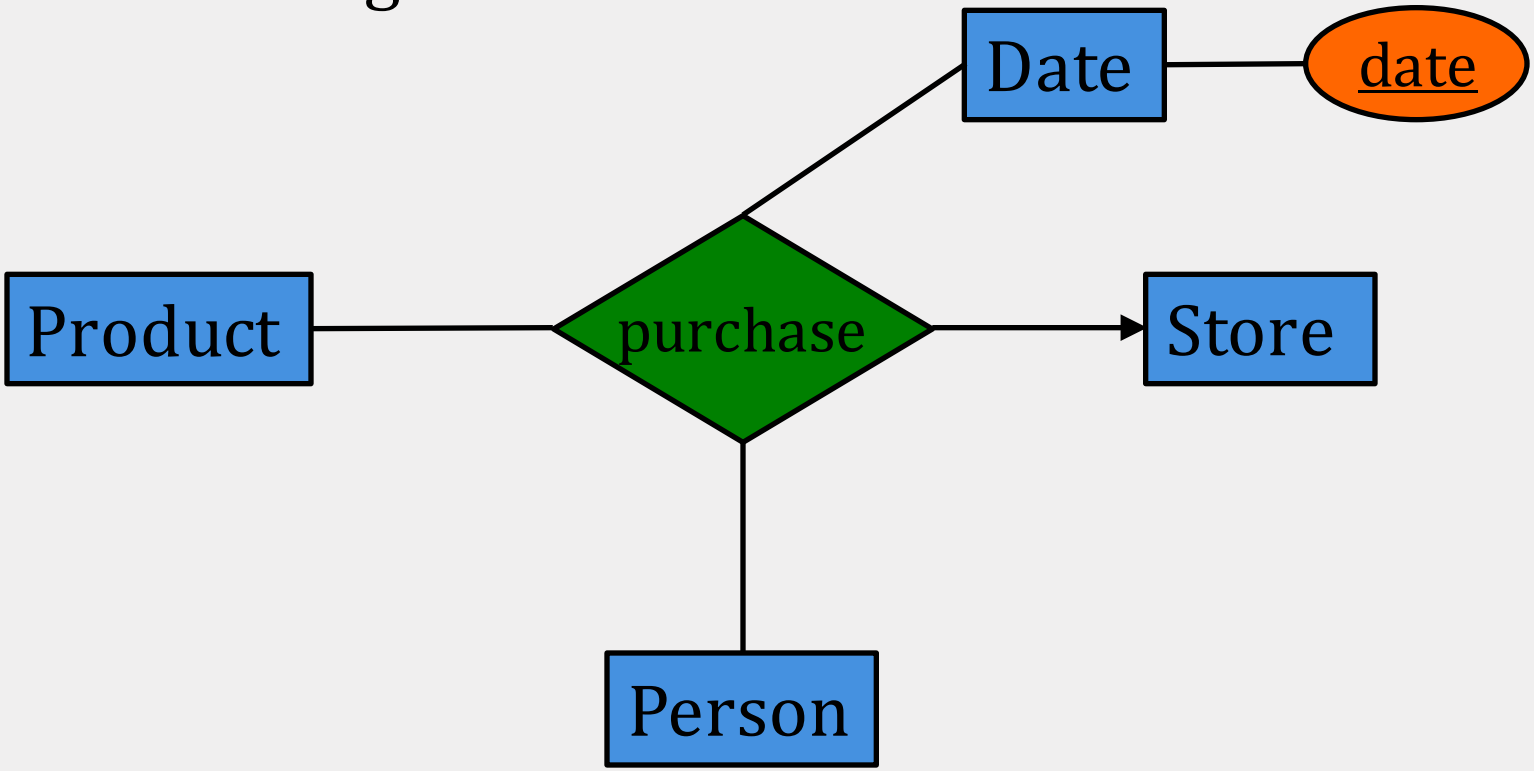
2. Avoid Redundancy!

What is wrong here?



3. KEEP IT SIMPLE!

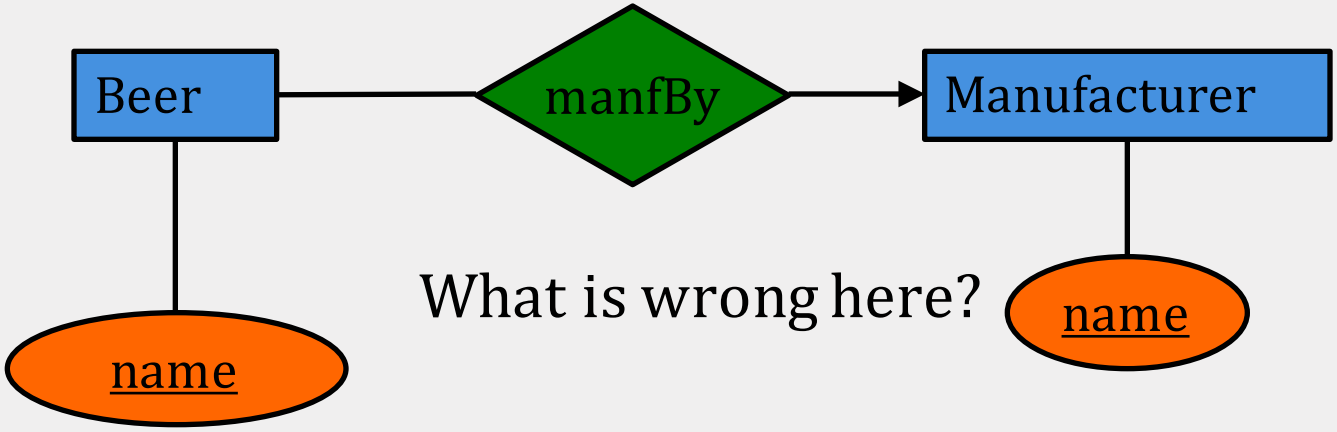
What is wrong here?



4. Attributes Over Entities

An entity set should satisfy at least one of the following conditions

- it is more than the name of something; it has at least one non-key attribute
- it is the "many" in a many-one or many-many relationship



5. Don't Overuse Weak Entity Sets

- Beginning database designers often doubt that anything could be a key by itself
  - They make all entity sets weak, supported by all other entity sets to which they are linked
- In reality, we create unique IDs for entity sets
  - Examples: SSN, ISBN, …

ER Model: Recap

Key concepts:

- entity, attribute, entity set
- relation: binary, multi-way
- relationship roles, attributes on relationships
- subclasses (ISA)
- weak entity sets
- constraints
  - many-one, one-one, many-many
  - keys, single-valued, referential integrity
- design principles