CS411
Database Systems

02: ER Model

Why do we learn this?

Steps in Building a DB Application

- Suppose you are working on CS411 project
- Step 0: pick an application domain
  - we will talk about this later
- Step 1: conceptual design
  - discuss with your team mates what to model in the application domain
  - need a modeling language to express what you want
  - ER model is the most popular such language
  - output: “an ER diagram” of the application domain

Steps in Building a DB Application

- Step 2: pick a type of DBMS
  - relational DBMS is most popular and is our focus
- Step 3: translate “ER design” to a “relational schema”
  - use a set of rules to translate from ER to rel. schema
  - use a set of schema refinement rules to transform the above rel. schema into a good rel. schema
- At this point
  - you have a good relational schema on paper

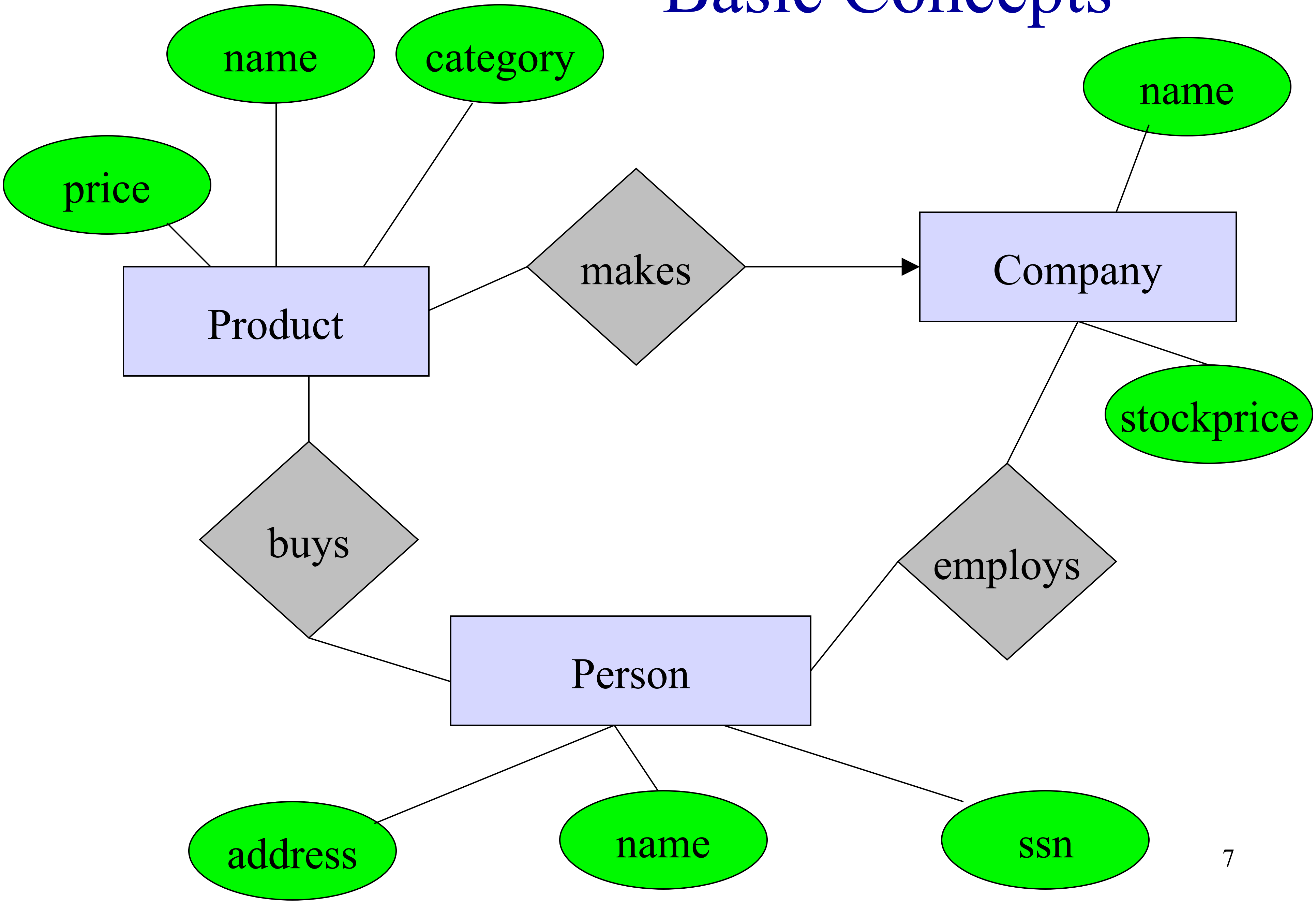
Steps in Building a DB Application

- Subsequent steps include
  - implement your relational DBMS using a "database programming language" called SQL
  - ordinary users cannot interact with the database directly
  - and the database also cannot do everything you want
  - hence write your application program in C++, Java, Perl, etc to handle the interaction and take care of things that the database cannot do
- So, the first thing we should start with is to learn ER model ...

ER Model

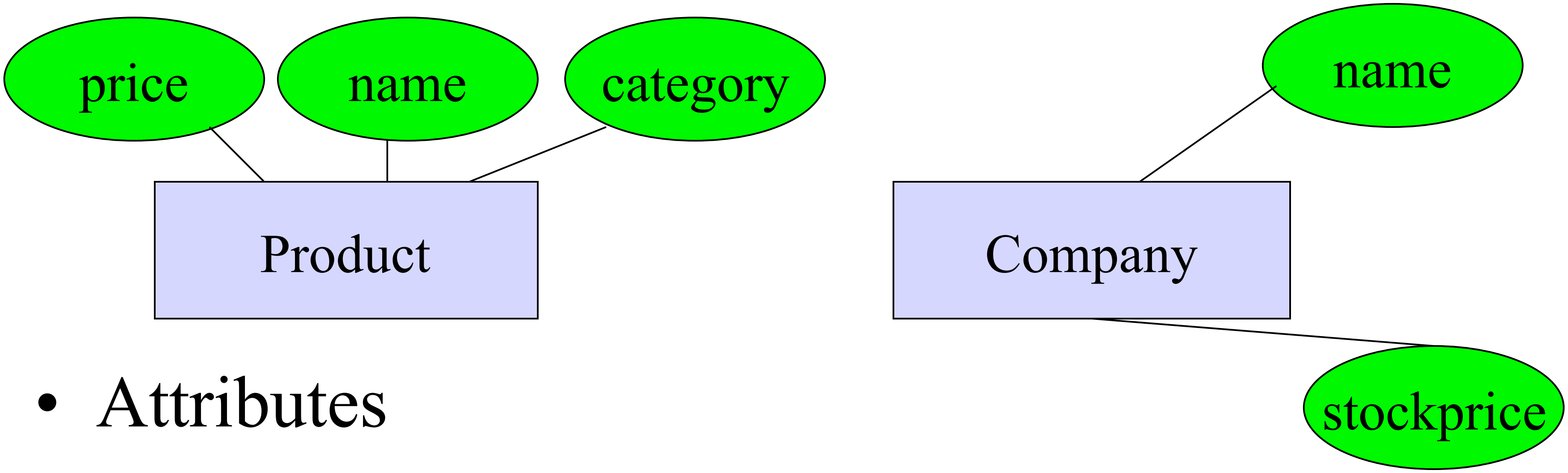
- Gives us a language to specify
  - what information the db must hold
  - what are the relationships among components of that information
- Proposed by Peter Chen in 1976
- What we will cover
  - basic stuff
  - constraints
  - weak entity sets
  - design principles

Basic Concepts



Entities and Attributes

- Entities
  - – real-world objects distinguishable from other objects
  - – described using a set of attributes

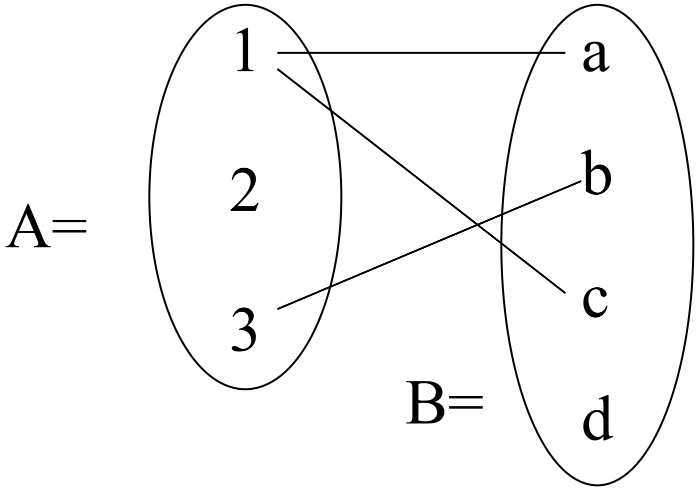


- Attributes
  - – each has an atomic domain: string, integers, reals, etc.
- Entity set: a collection of similar entities

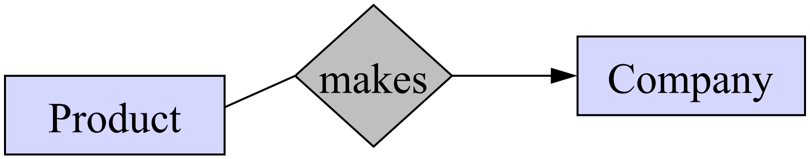
Relations

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

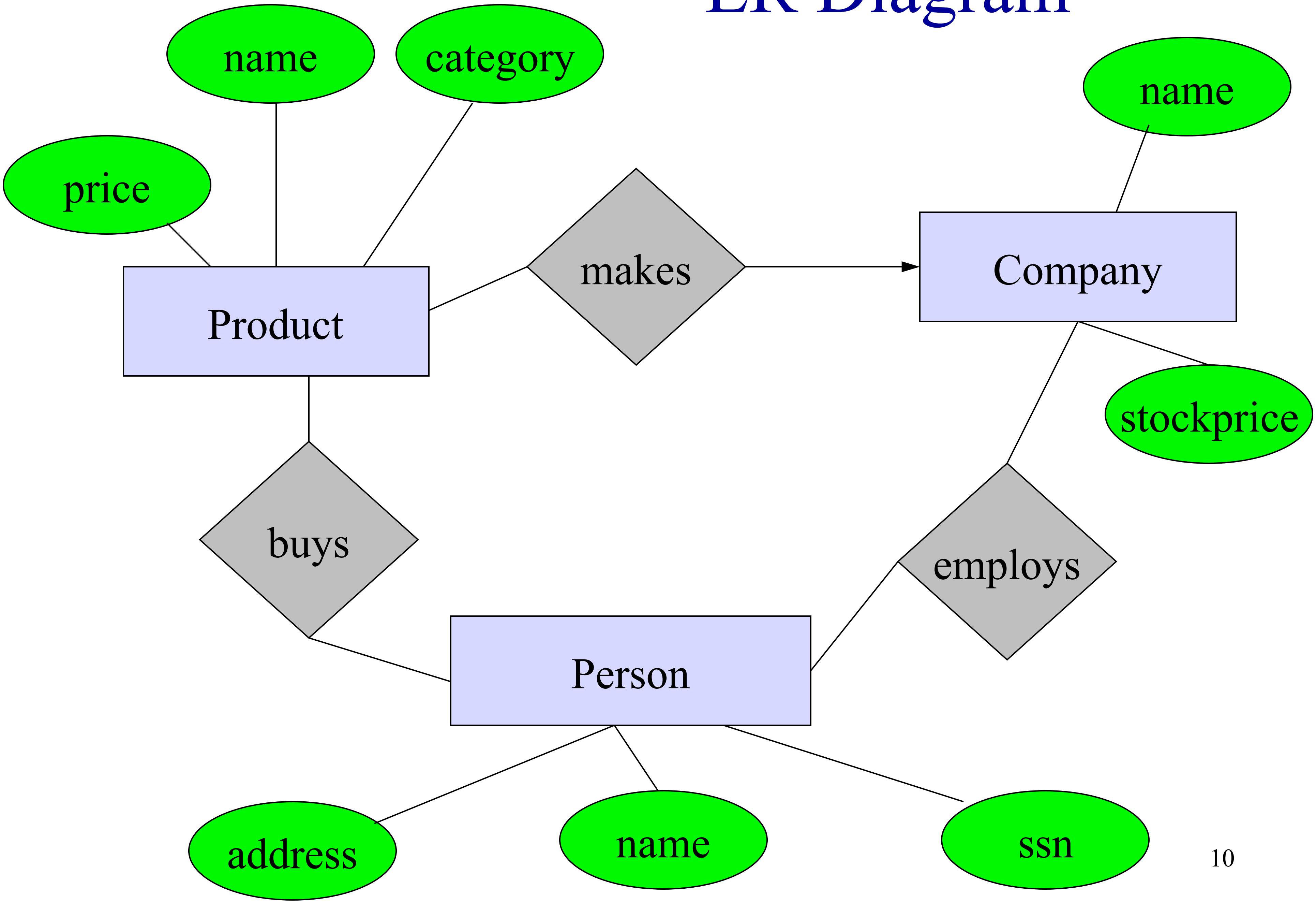
  R = {(1,a), (1,c), (3,b)}



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



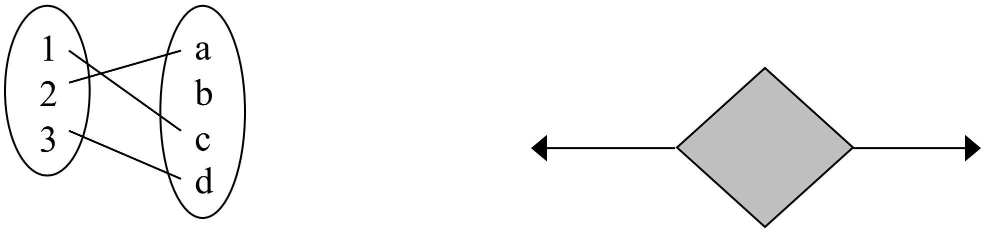
ER Diagram


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

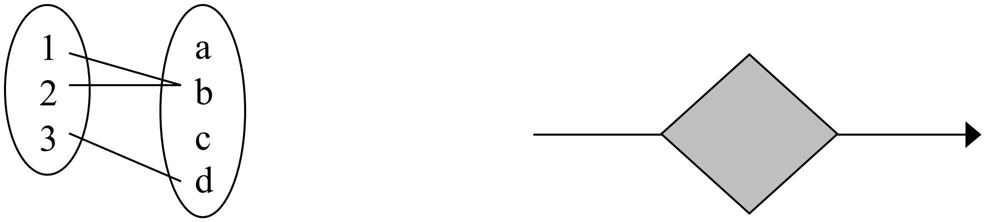
More about relationships ...

Multiplicity of E/R Relationships

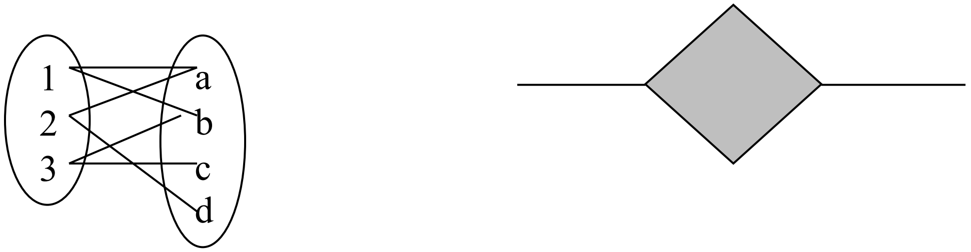
- one-one:



- many-one



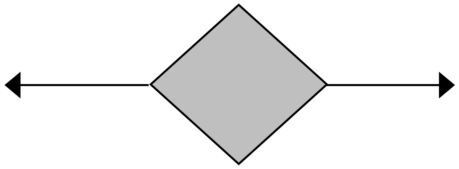
- many-many



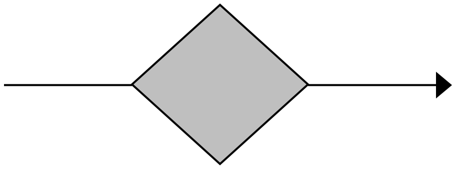
- Multiplicity can be shown with arrows

Q: Example scenarios for each case?

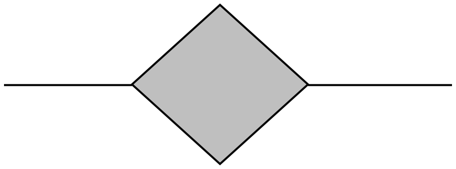
- one-one:



- many-one

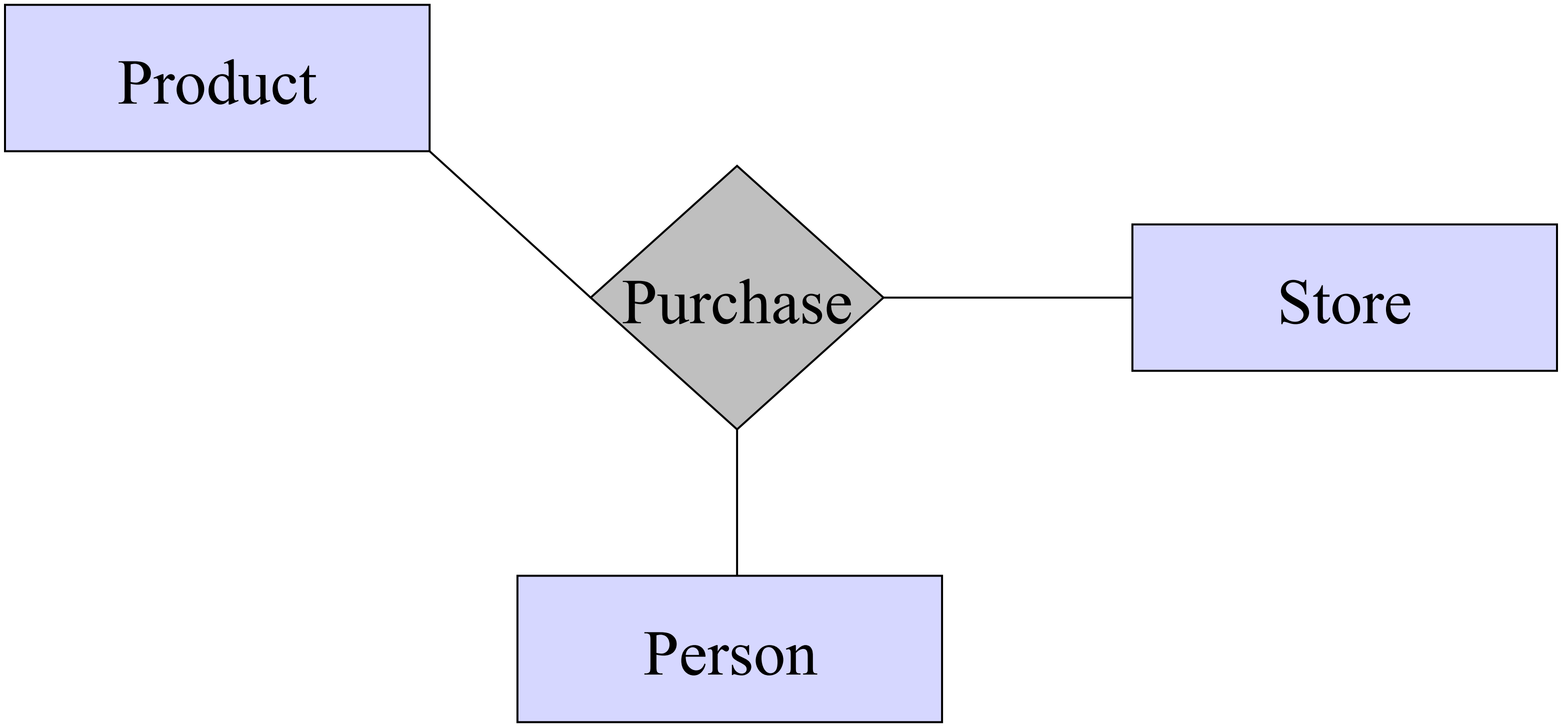


- many-many



Multiway Relationships

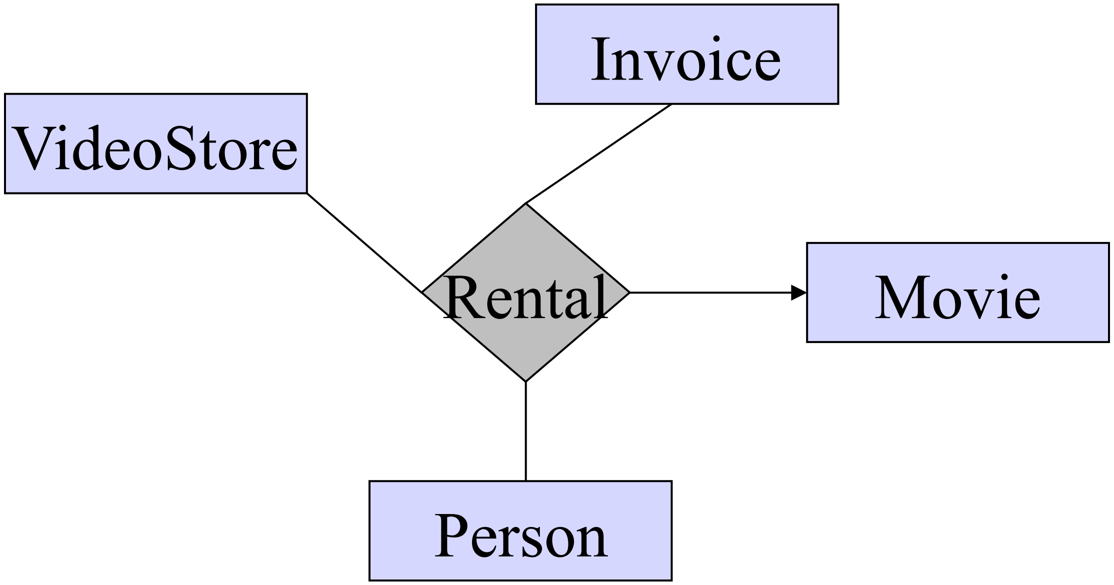
How do we model a purchase relationship between buyers, products and stores?



Can still model as a mathematical set (how ?)

Arrows in Multiway Relationships

Q: what does the arrow mean ?

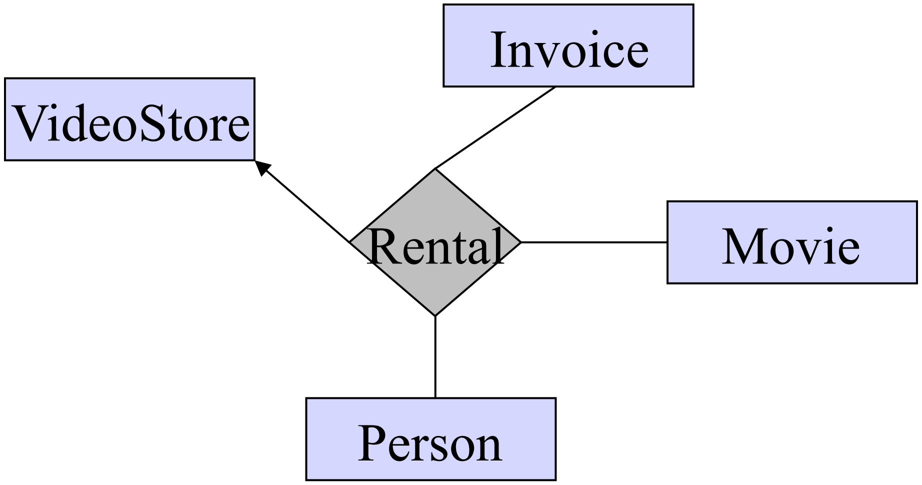


A:

Arrows in Multiway Relationships

Q: how do I say: “invoice determines store” ?

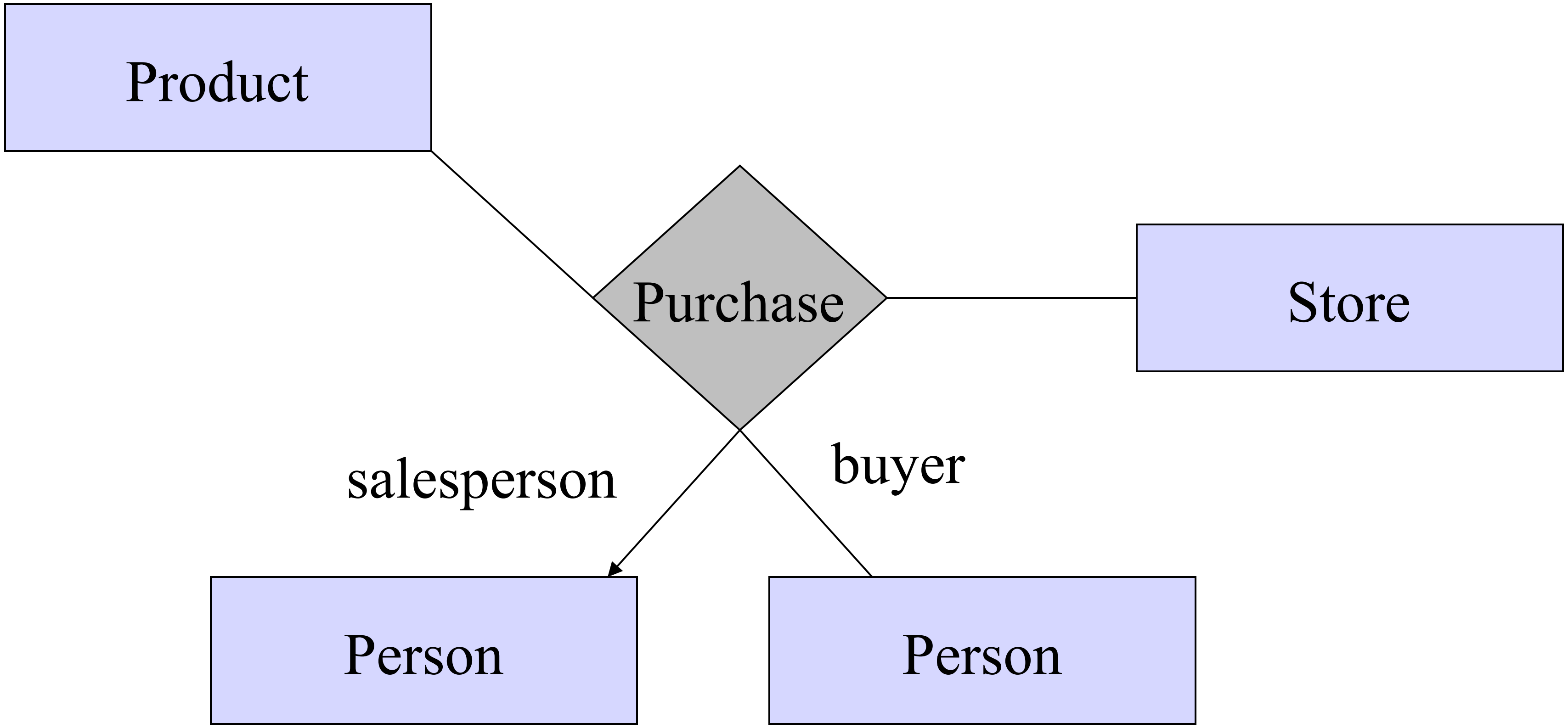
A: no good way; best approximation:



Q: Why is this incomplete ?

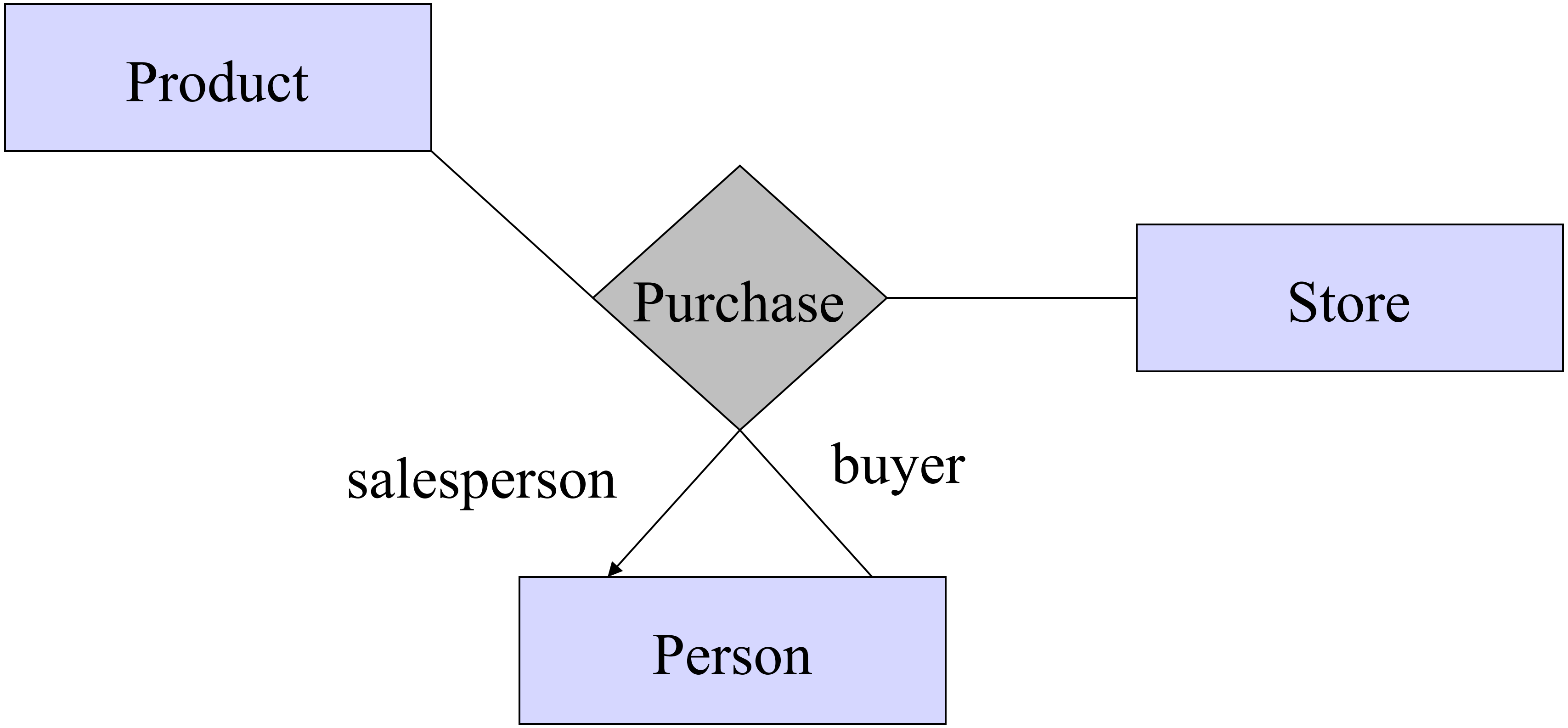
Roles in Relationships

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

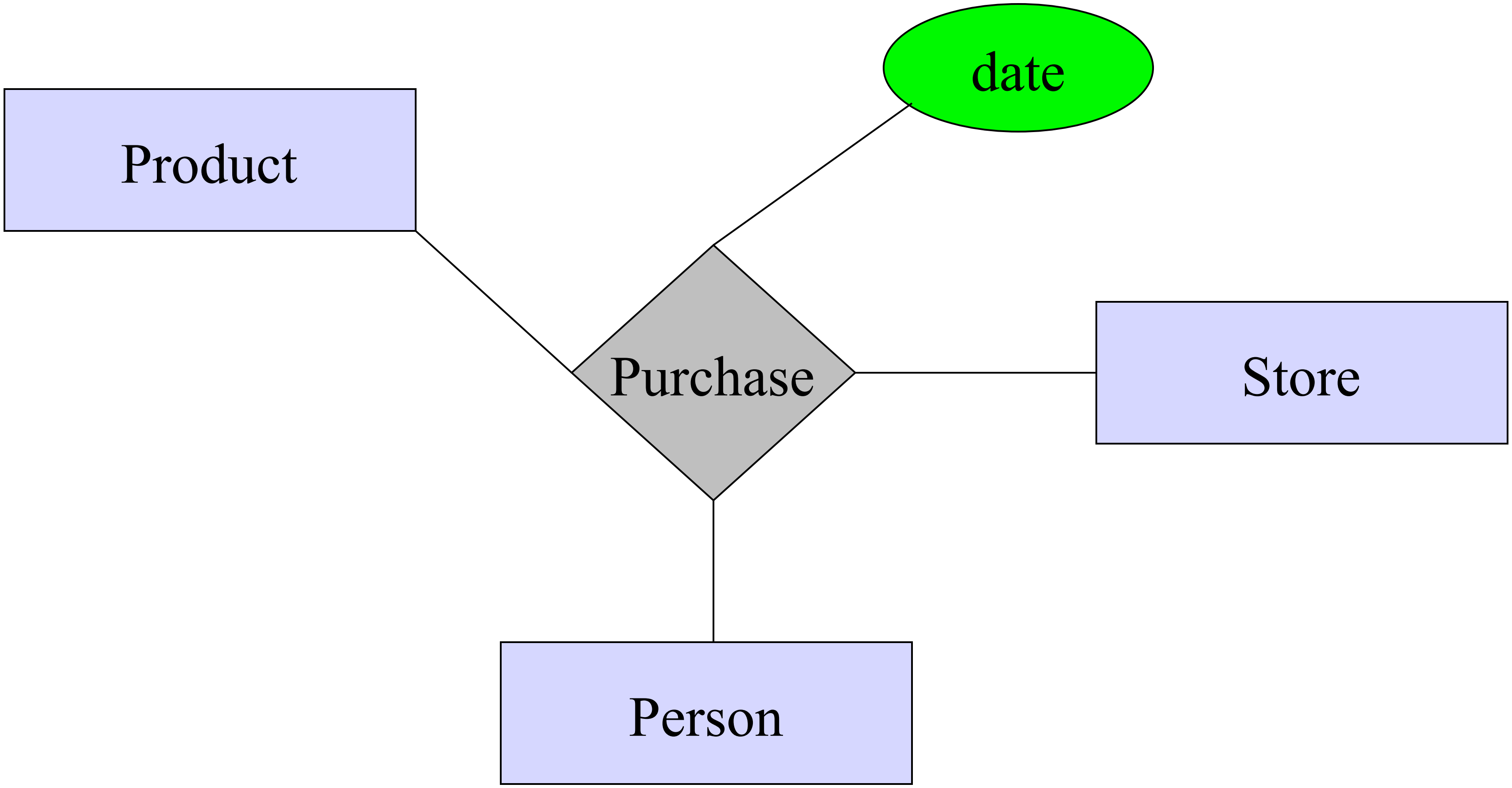


Roles in Relationships

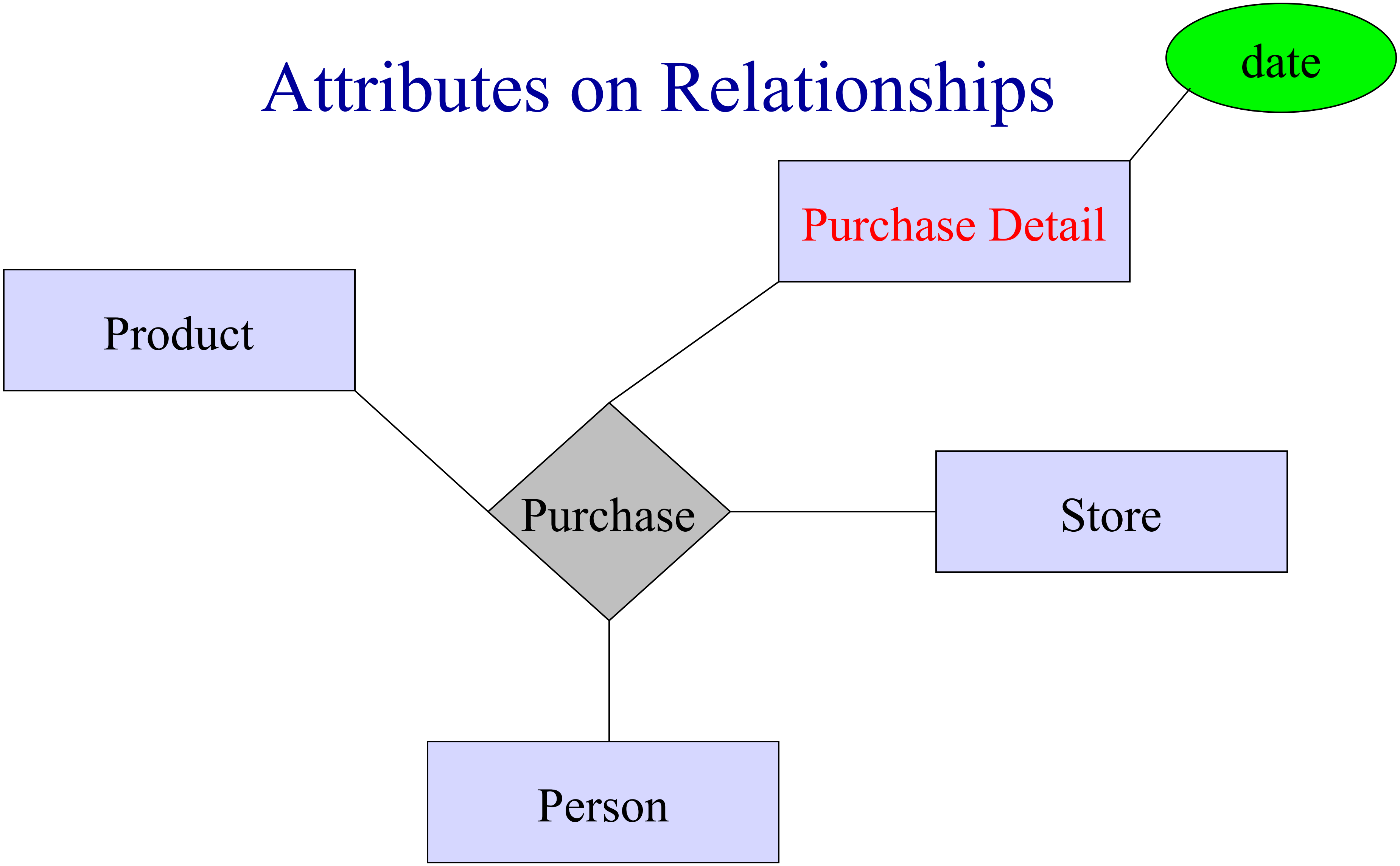
What if we need an entity set twice in one relationship?

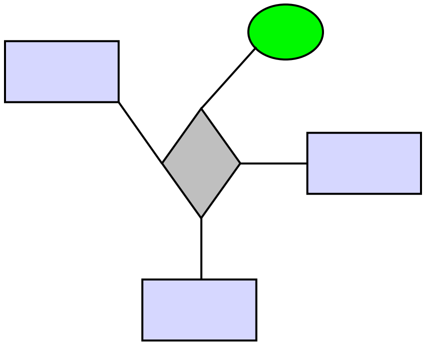


Attributes on Relationships

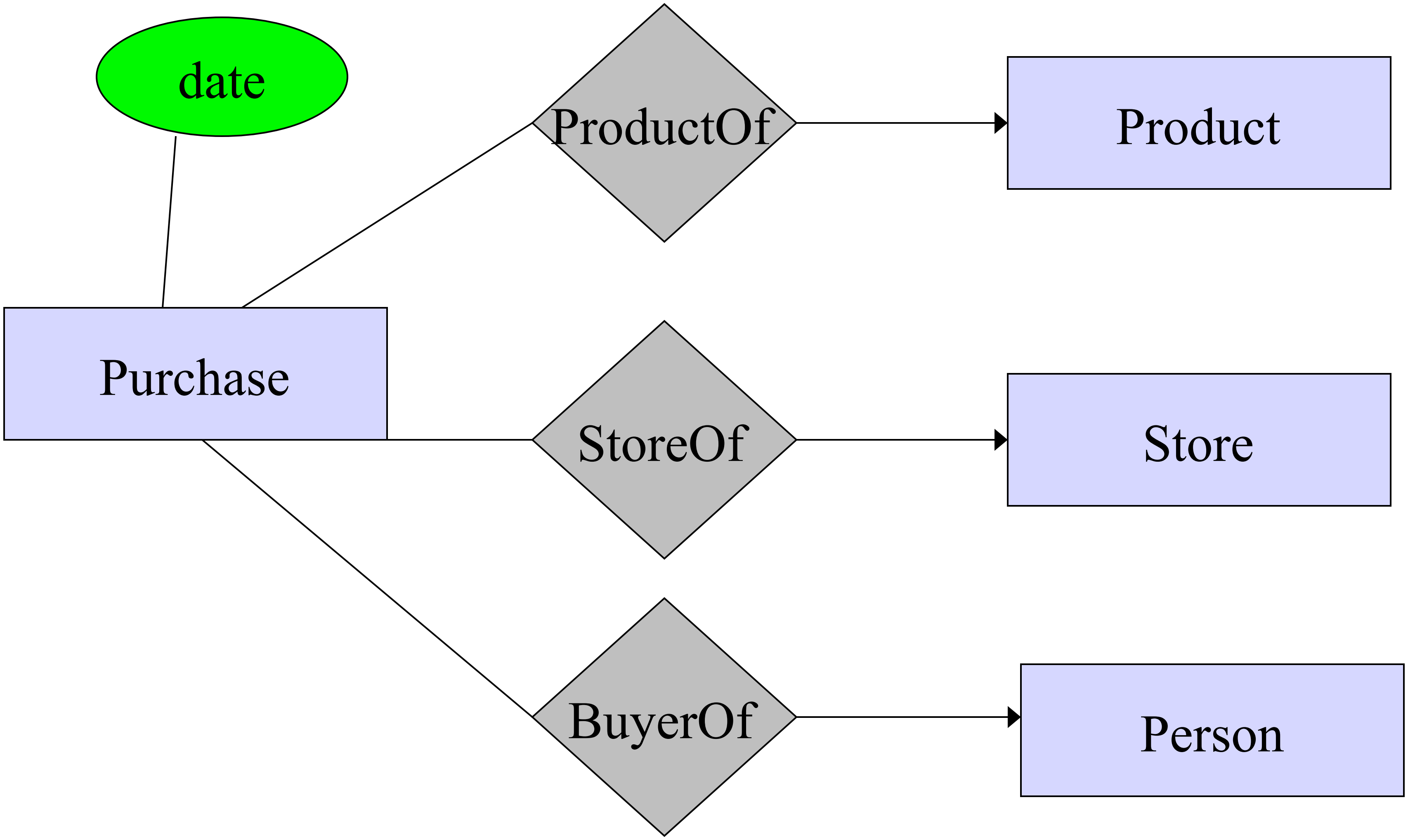


Attributes on Relationships





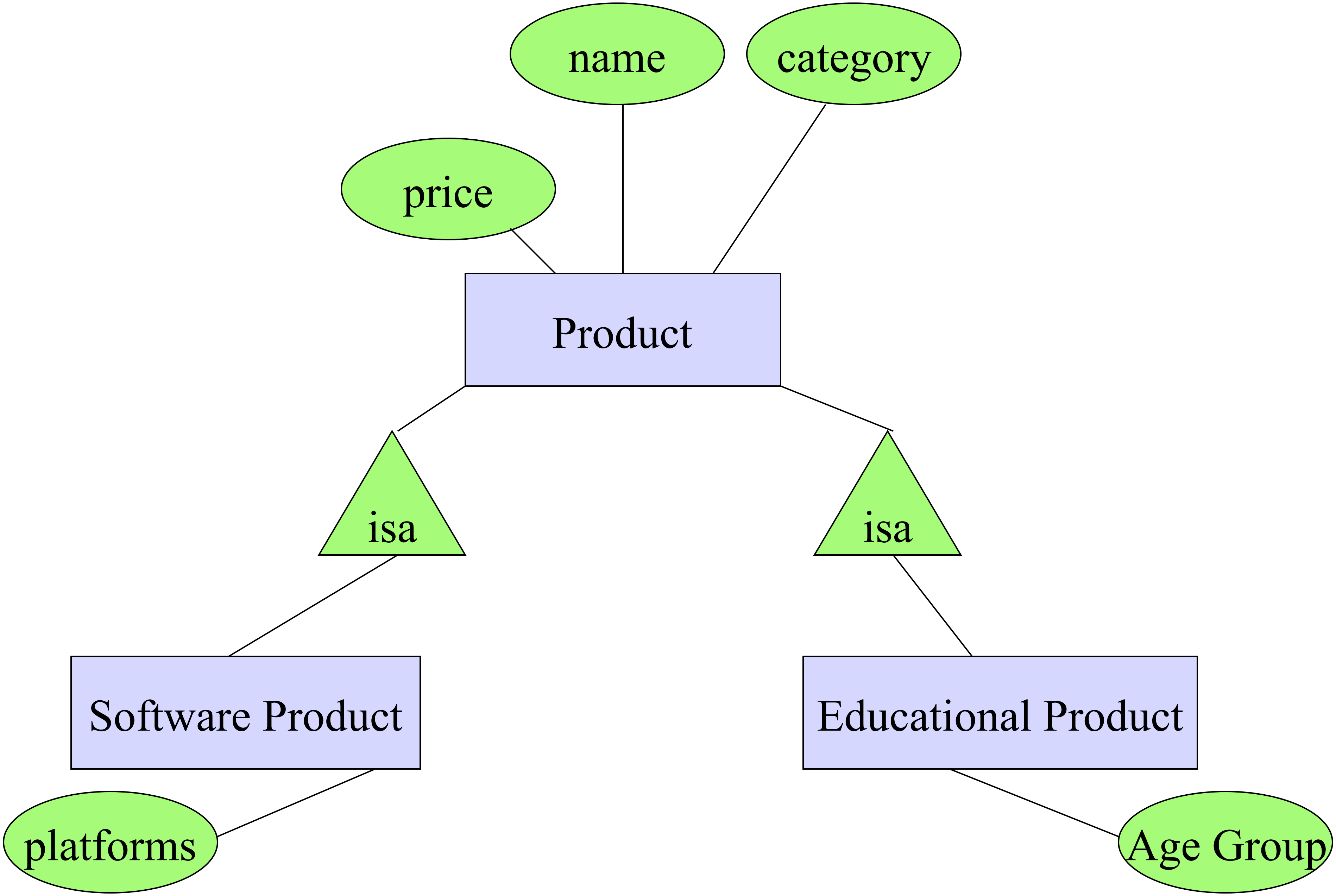
Converting Multiway Relationships to Binary



Relationships: Summary

- Modeled as a mathematical set
- Binary and multiway relationships
- Converting a multiway 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

Subclasses in ER Diagrams



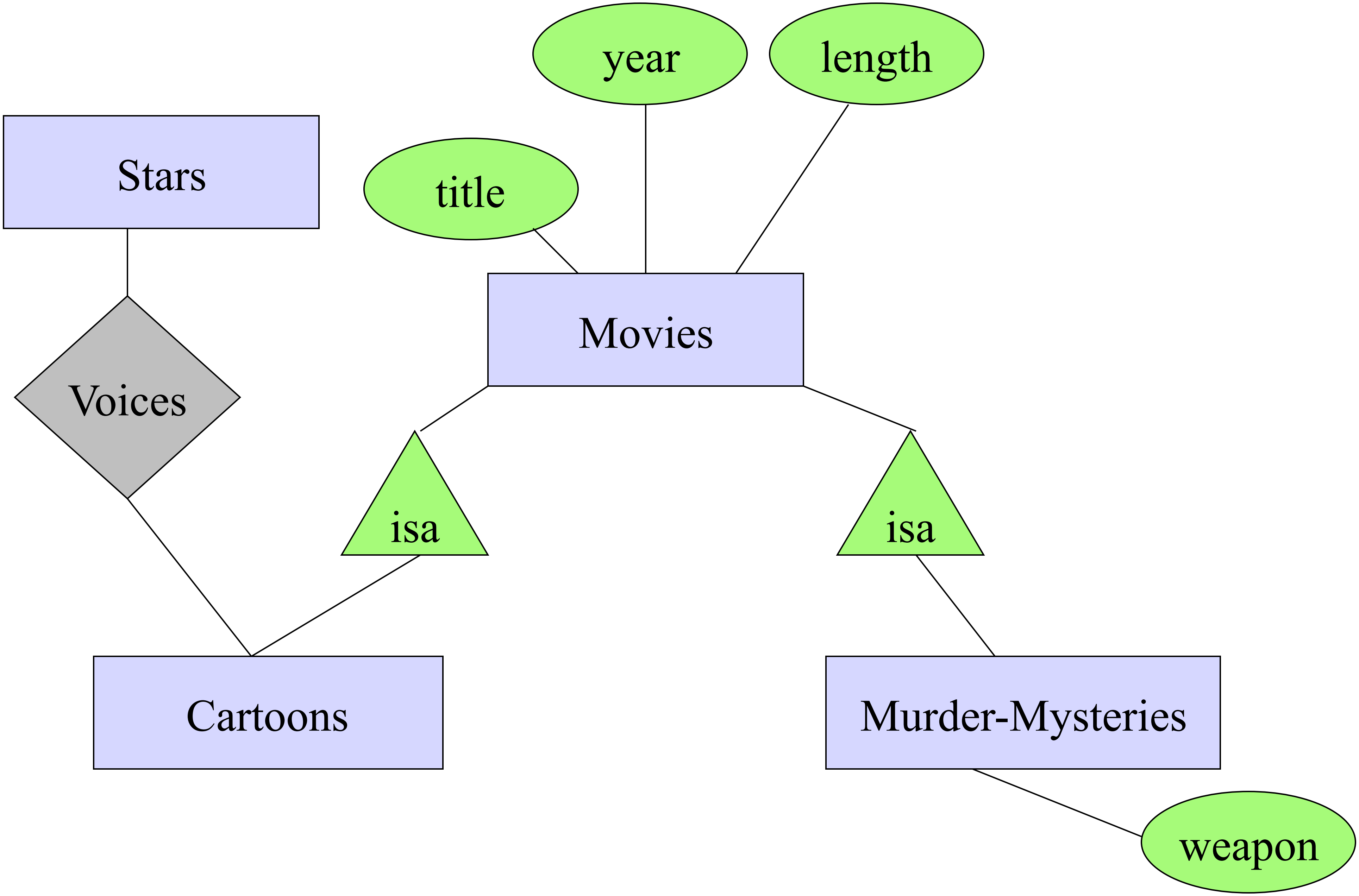
Subclasses

- Subclass = special case = fewer entities = more properties.
- Example: Ales are a kind of beer.
  - Not every beer is an ale, but some are.
  - Let us suppose that in addition to all the *properties* (attributes and relationships) of beers, ales also have the attribute *color*.

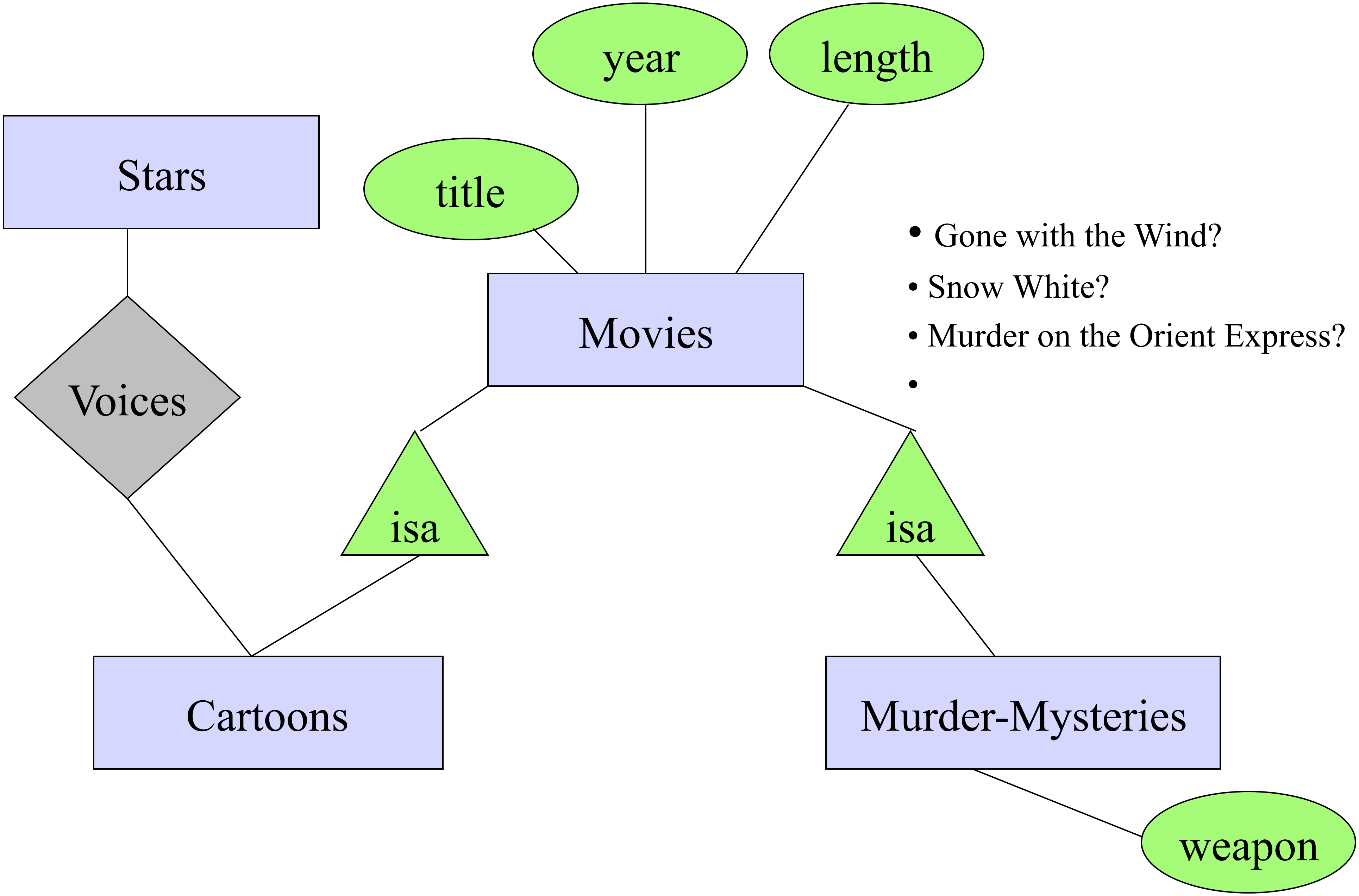
Subclasses in ER Diagrams

- Assume subclasses form a tree.
  - I.e., no "multiple inheritance".
- "Isa" triangles indicate the subclass relationship.
  - Point to the superclass.

Example


year
length
Stars
title
Movies
Voices
isa
isa
Cartoons
Murder-Mysteries
weapon

Example

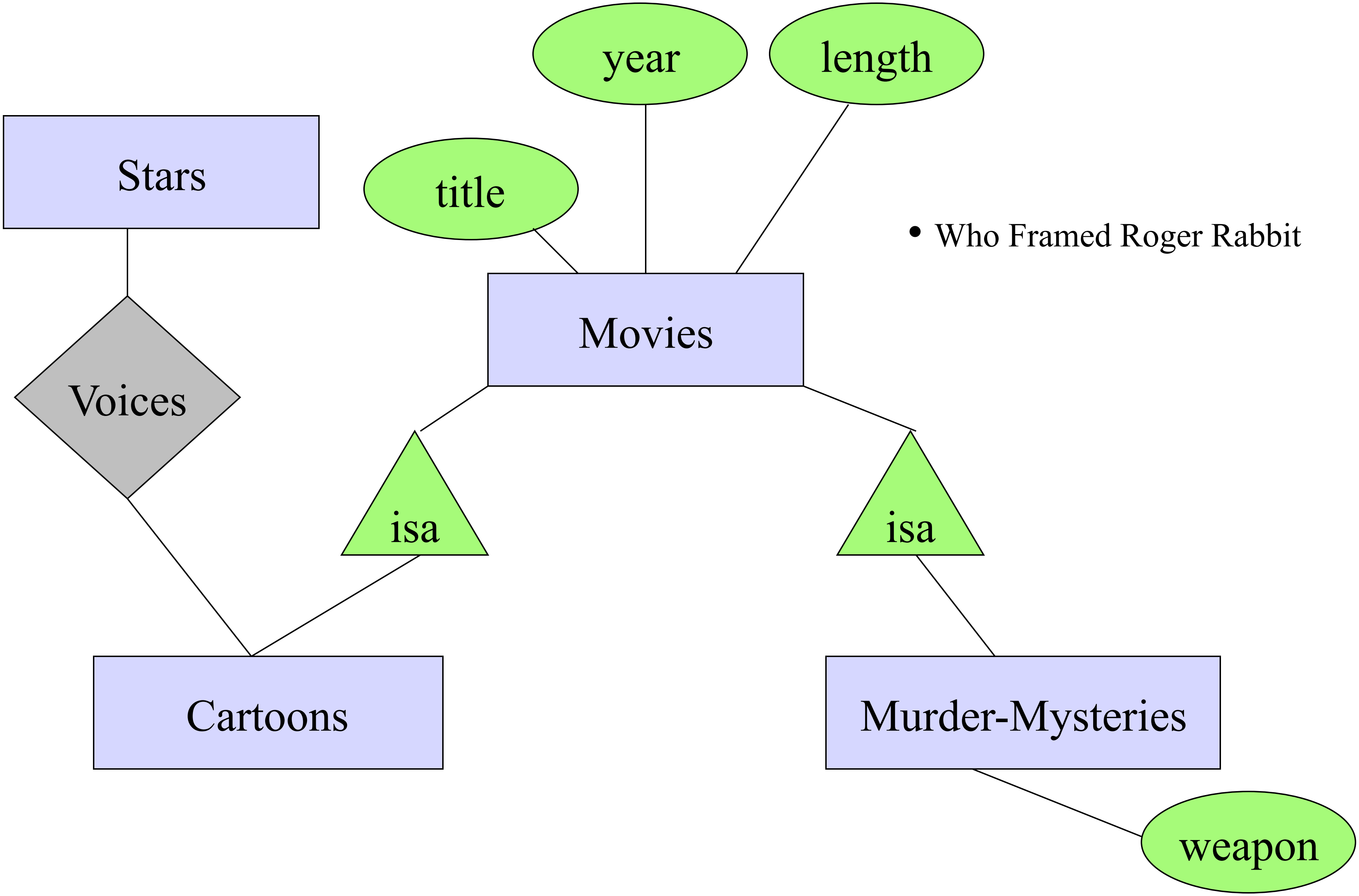


- Gone with the Wind?
- Snow White?
- Murder on the Orient Express?
-

ER Vs. Object Oriented Subclasses

- In the object-oriented world, objects are in one class only.
  - Subclasses inherit properties from superclasses.
- In contrast, E/R entities have components in all subclasses to which they belong.
  - Matters when we convert to relations.

Example



- Who Framed Roger Rabbit

Constraints in ER diagram

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

- Part of the database schema

- Very important in database design

Modeling Constraints

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.

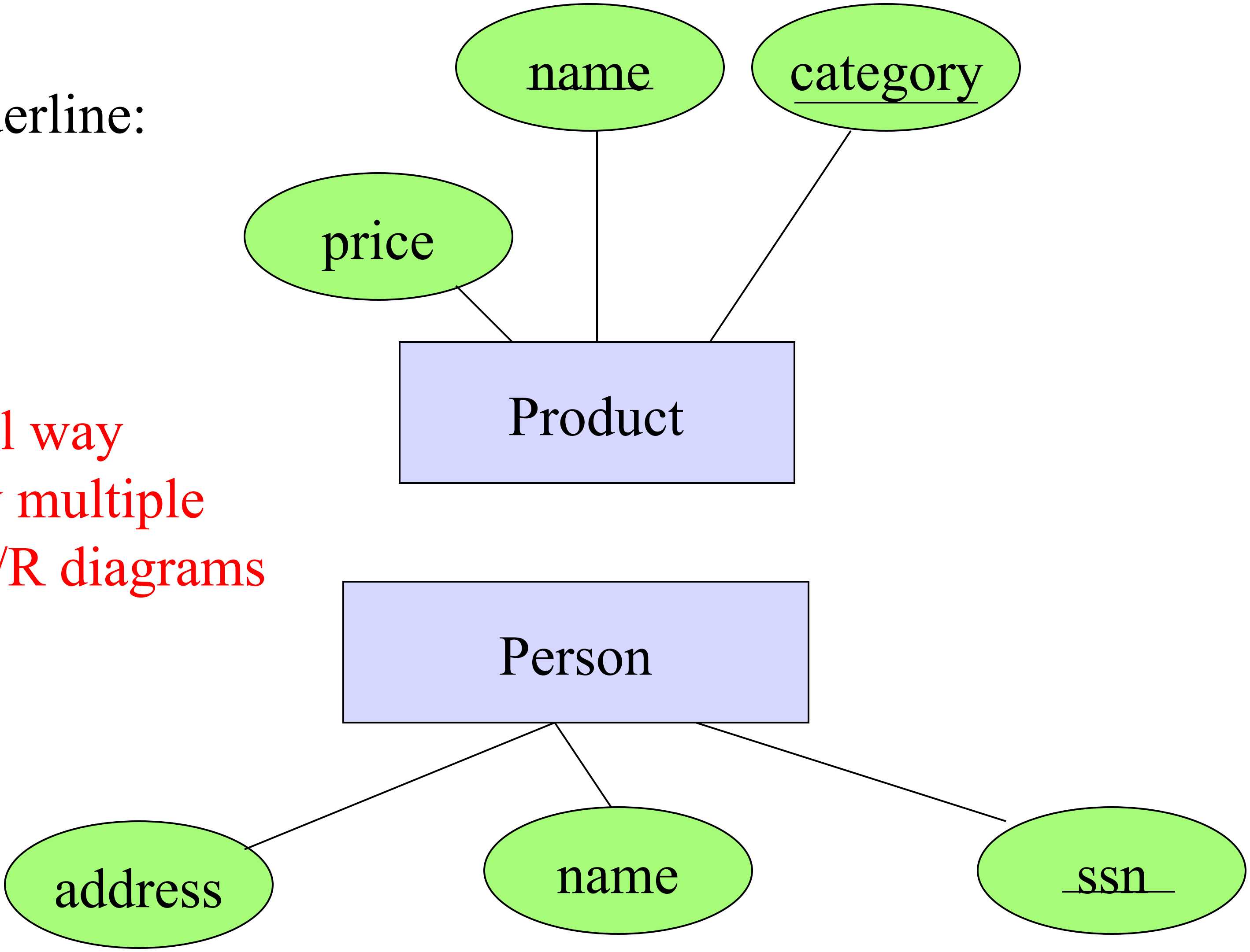
- Domain constraints: peoples' ages are between 0 and 150.
- General constraints: all others (at most 50 students enroll in a class)

Why Constraints are Important

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

Keys in E/R Diagrams

Underline:



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

More about Keys

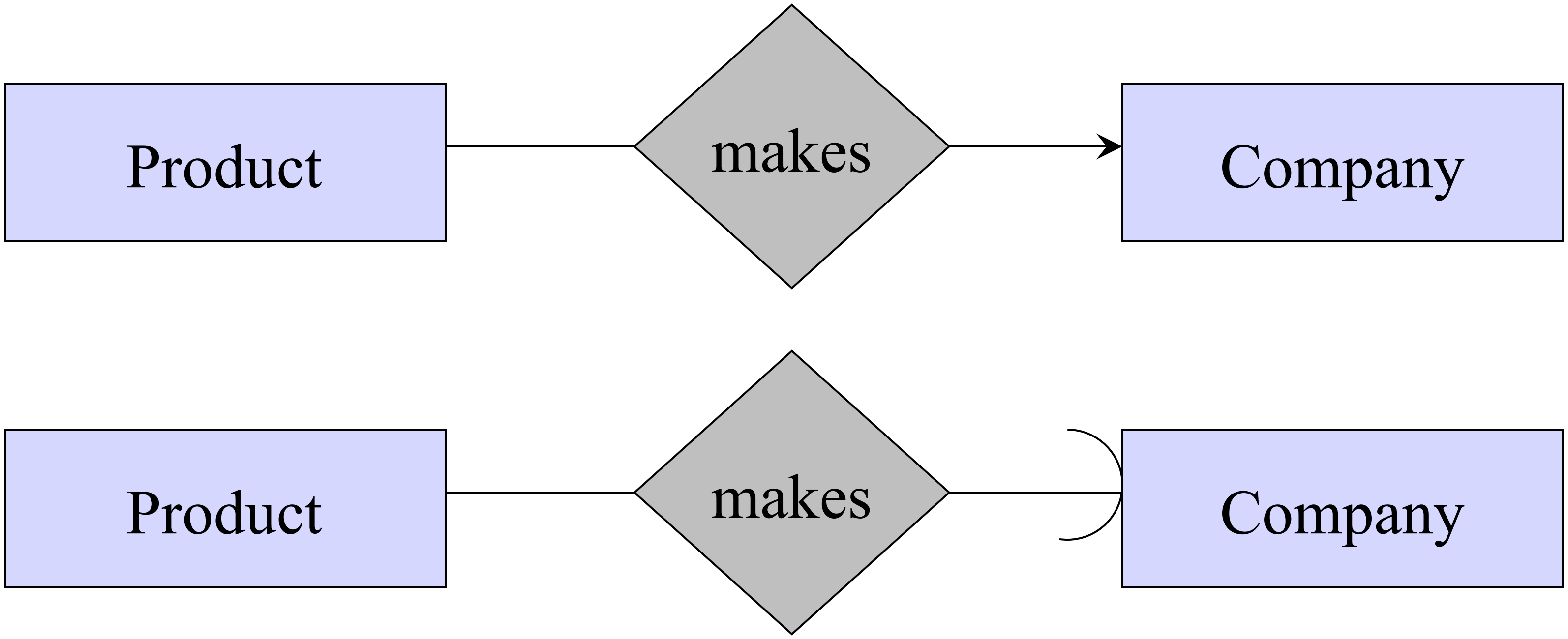
- Every entity set must have a key
  - why?
- A key can consist of more than one attribute
- There can be more than one key for an entity set
  - one key will be designated as primary key
- Requirement for key in an isa hierarchy
  - see text

Referential Integrity Constraints

- In some formalisms we may refer to other object but get garbage instead
  - e.g. a dangling pointer in C/C++

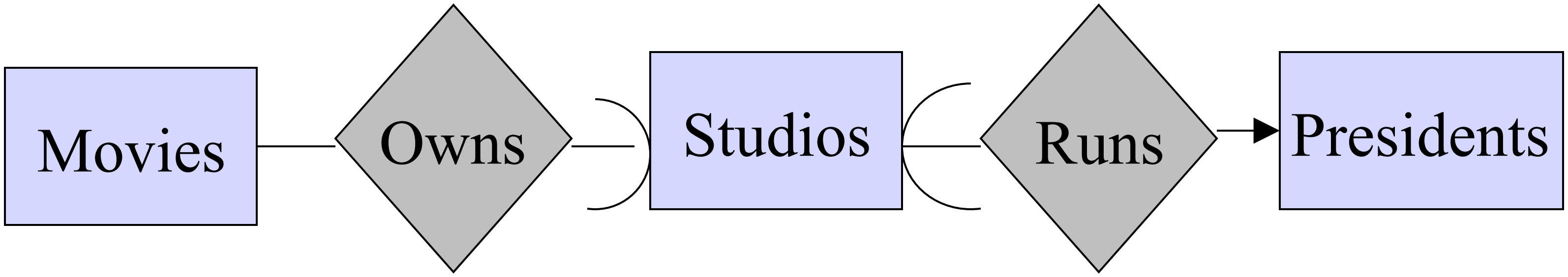
- the Referential Integrity Constraint on relationships explicitly requires a reference to exist

Referential Integrity Constraints



- This will be even clearer once we get to relational databases

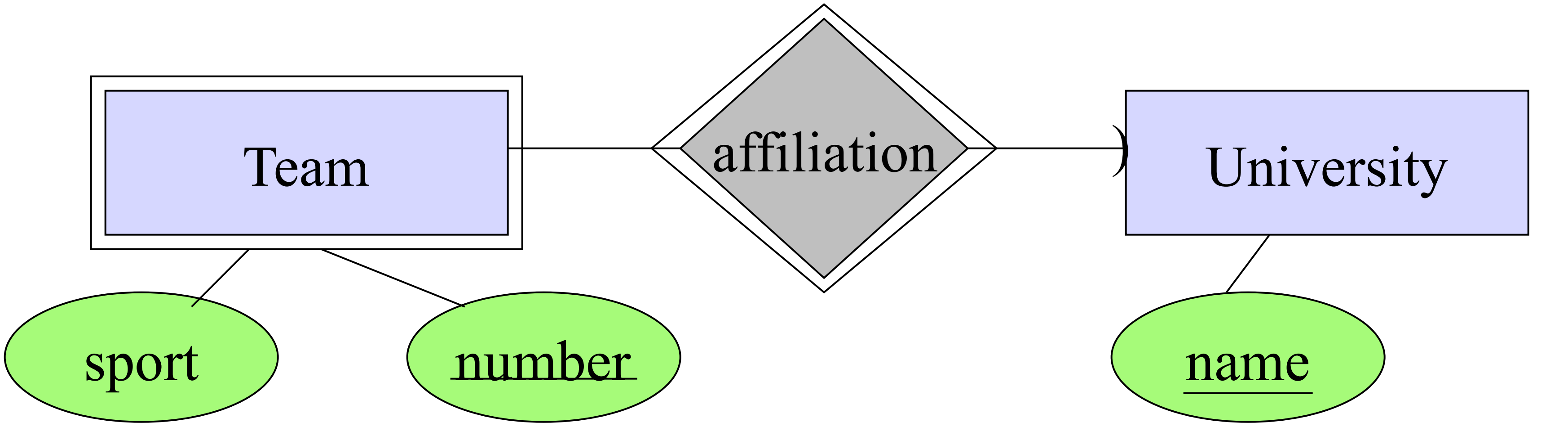
Referential Integrity Constraints



Weak Entity Sets

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

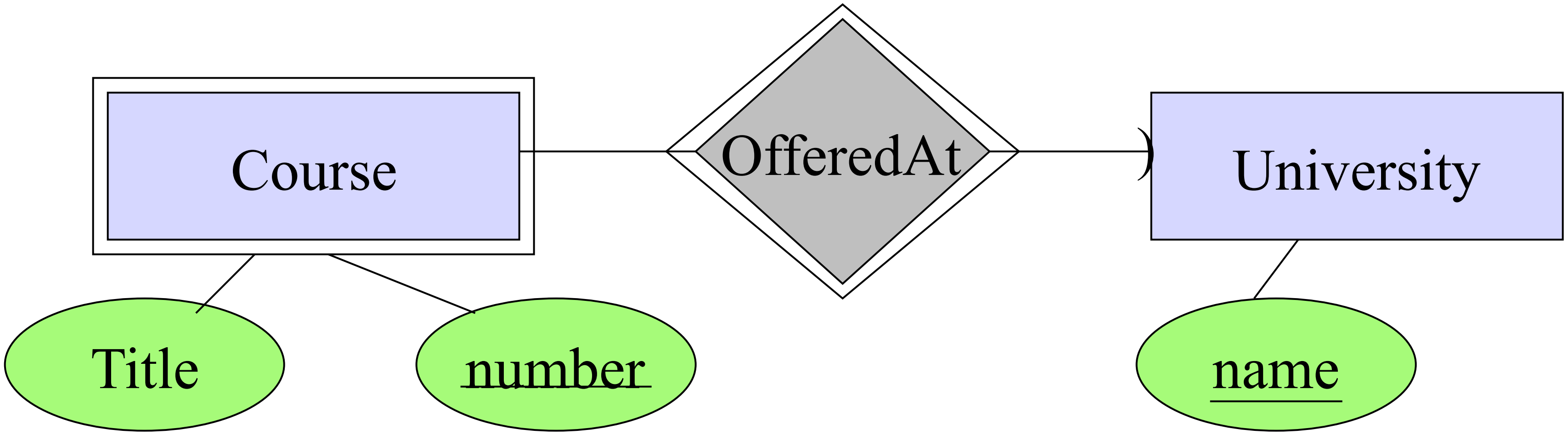
- Football Team 1 (UIUC)



Weak Entity Sets

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

- CS411, “Db systems” (UIUC)

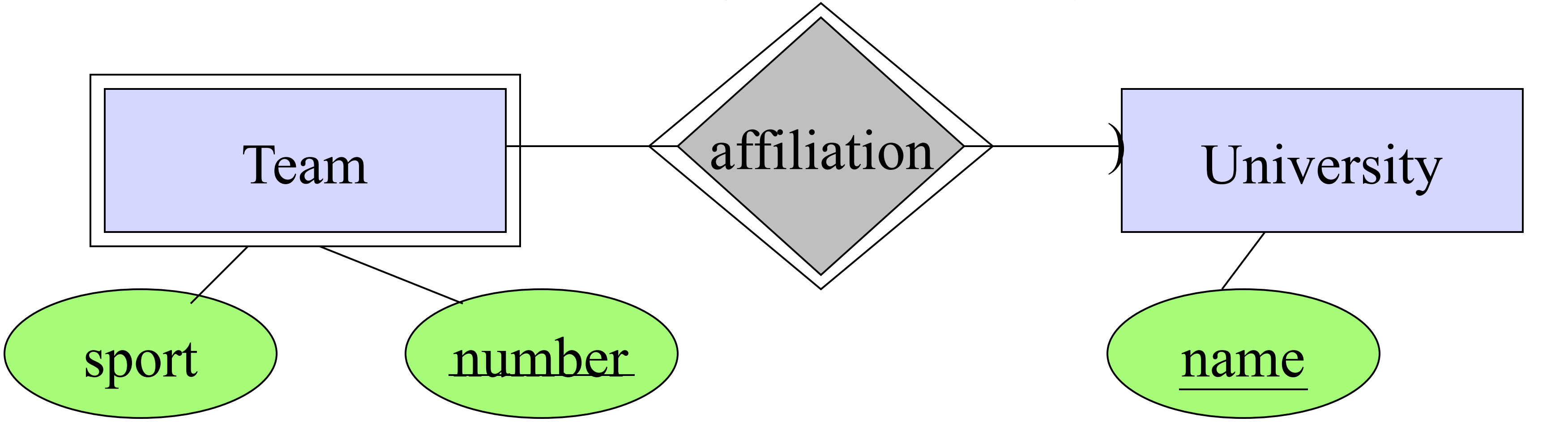


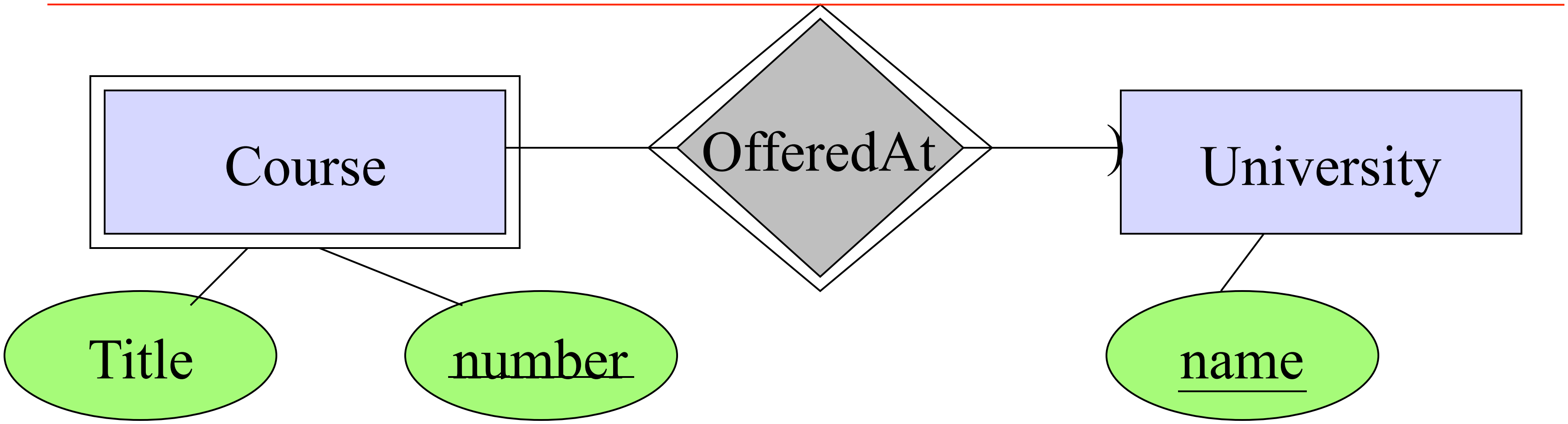
Weak Entity Sets

- Occasionally, entities of an entity set need “help” to identify them uniquely.

- Entity set *E* is said to be *weak* if in order to identify entities of *E* uniquely, we need to follow one or more many-one relationships from *E* and include the key of the related entities from the connected entity sets.

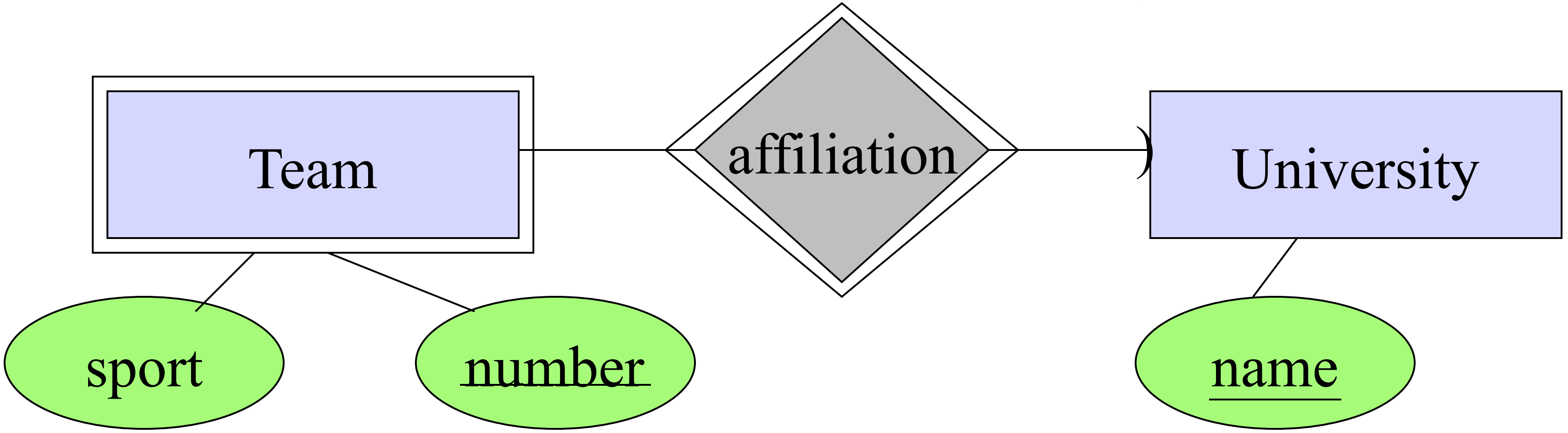
Q: Is this subclassing? Similarity? Difference?

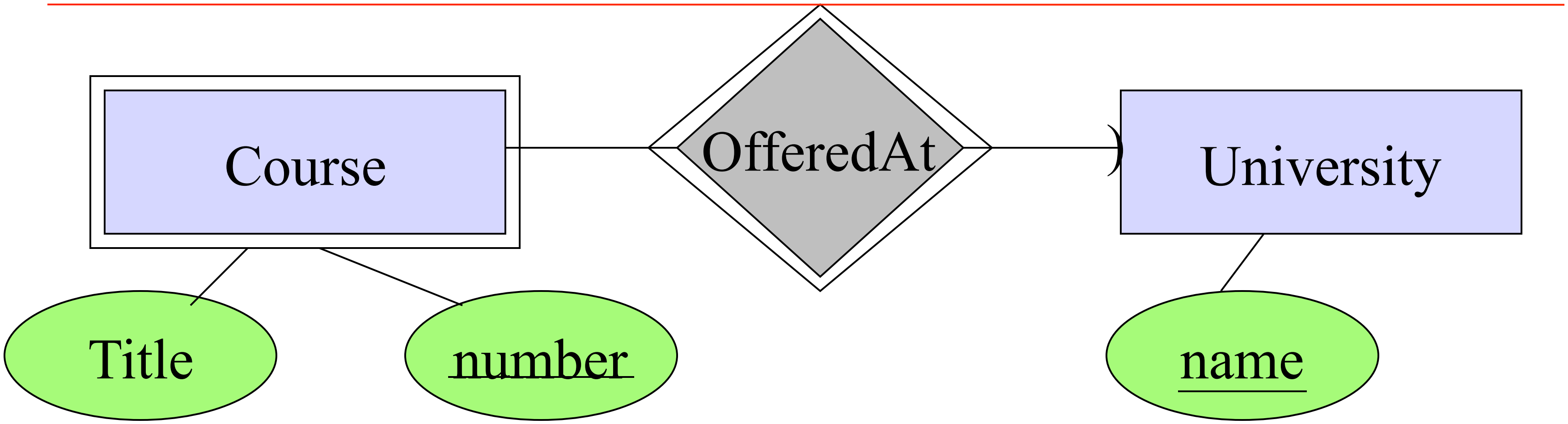




- Entities of “Team” (or “Course”) are subunits of entities in “University”; a “Team” entity is not unique until we take into account the “University” it belongs to.

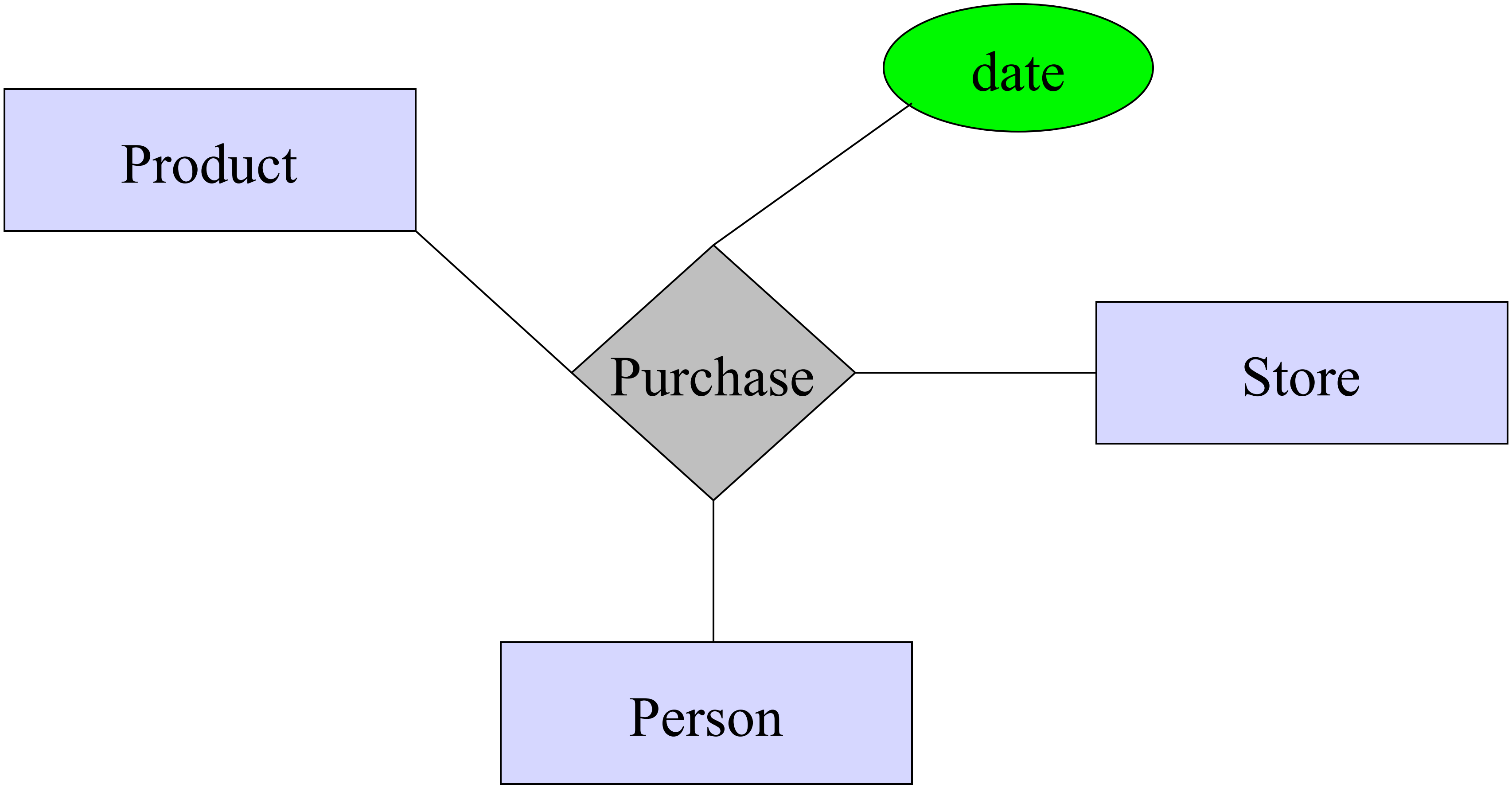
Notations for weak entity set





- “University” is a “supporting entity set” for “Team” (or “Course”).
- “Affiliation” (or “OfferedAt”) is a “supporting relationship”.

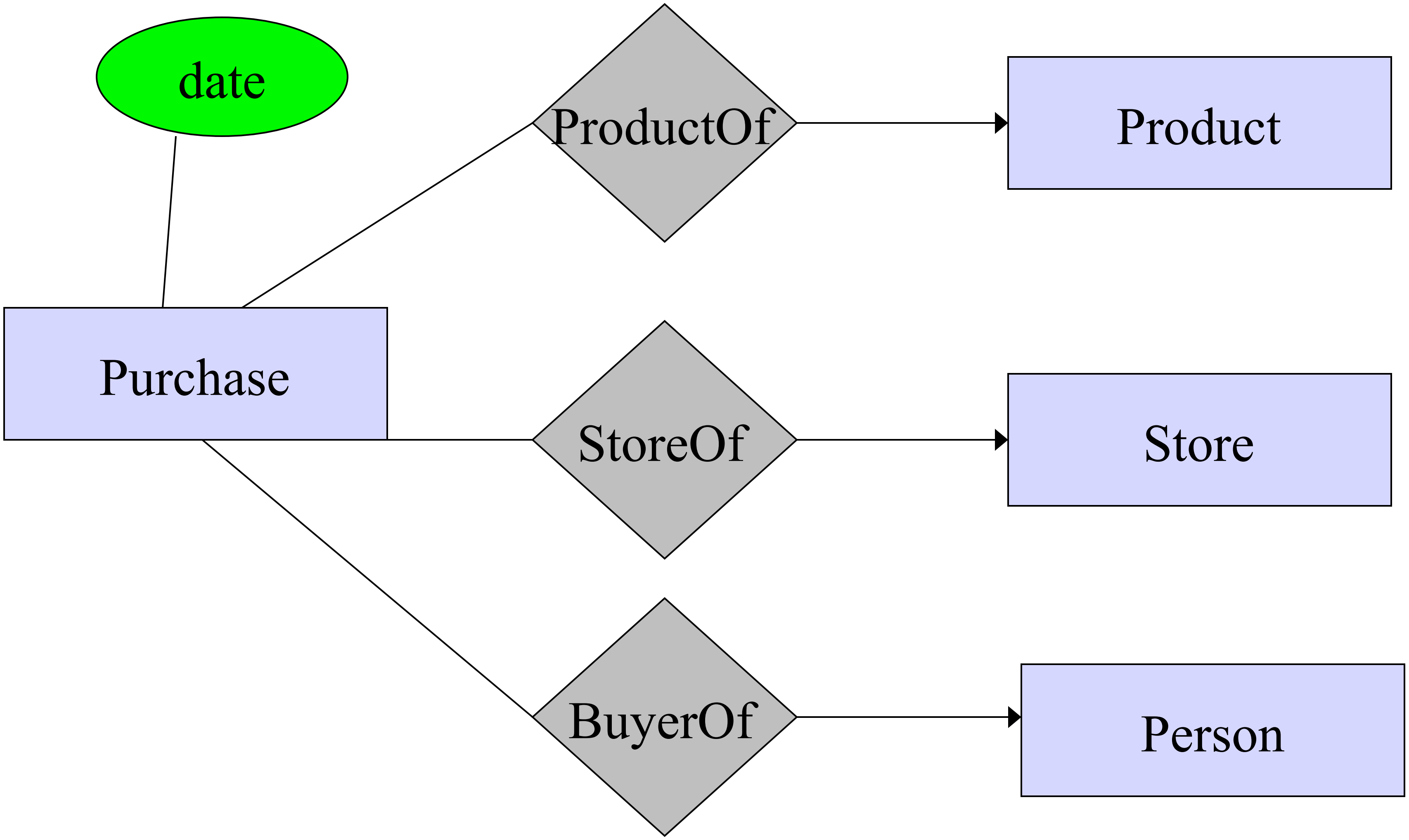
Another scenario where weak e.s. arises



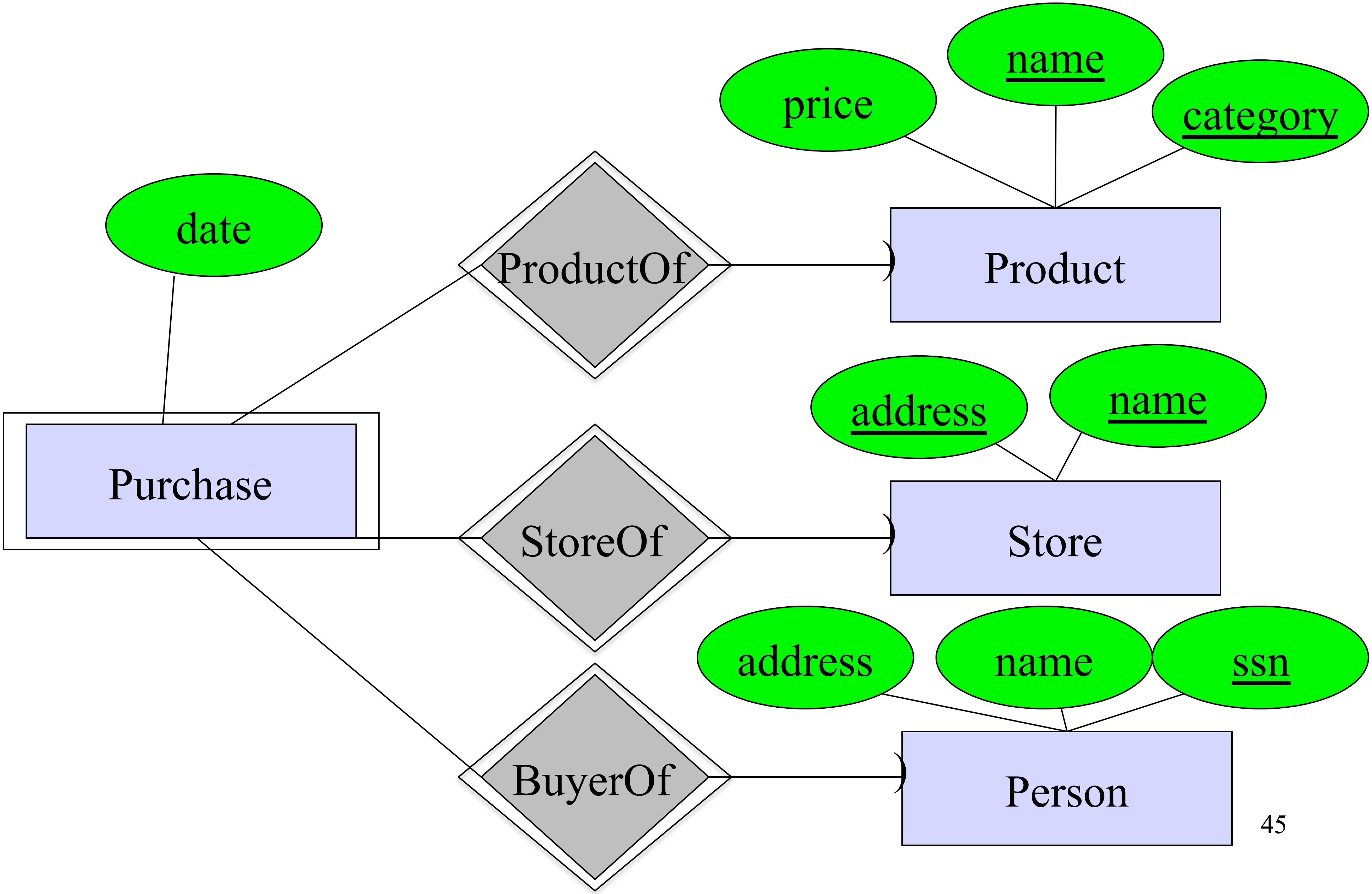
- A Multi-way relationship …

Another scenario where weak e.s. arises

- … converted to binary relationships

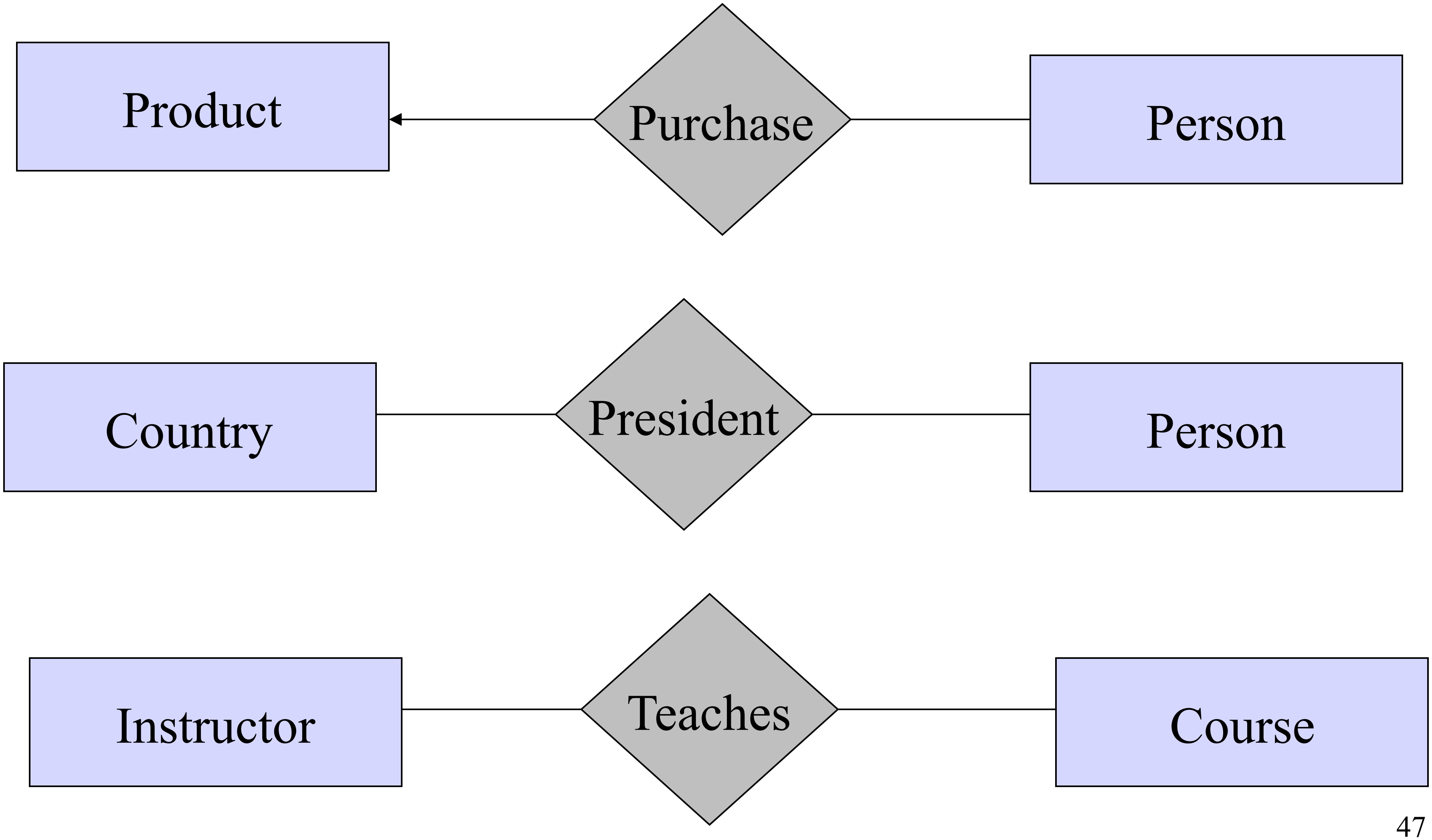


Another scenario where weak e.s. arises



Now, about design principles ...

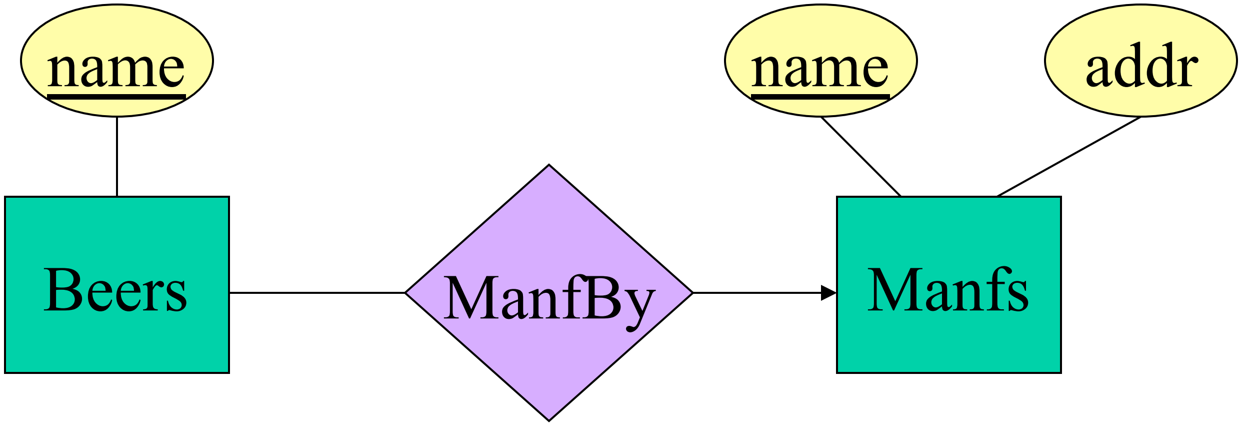
Design Principles: Be Faithful



Avoiding Redundancy

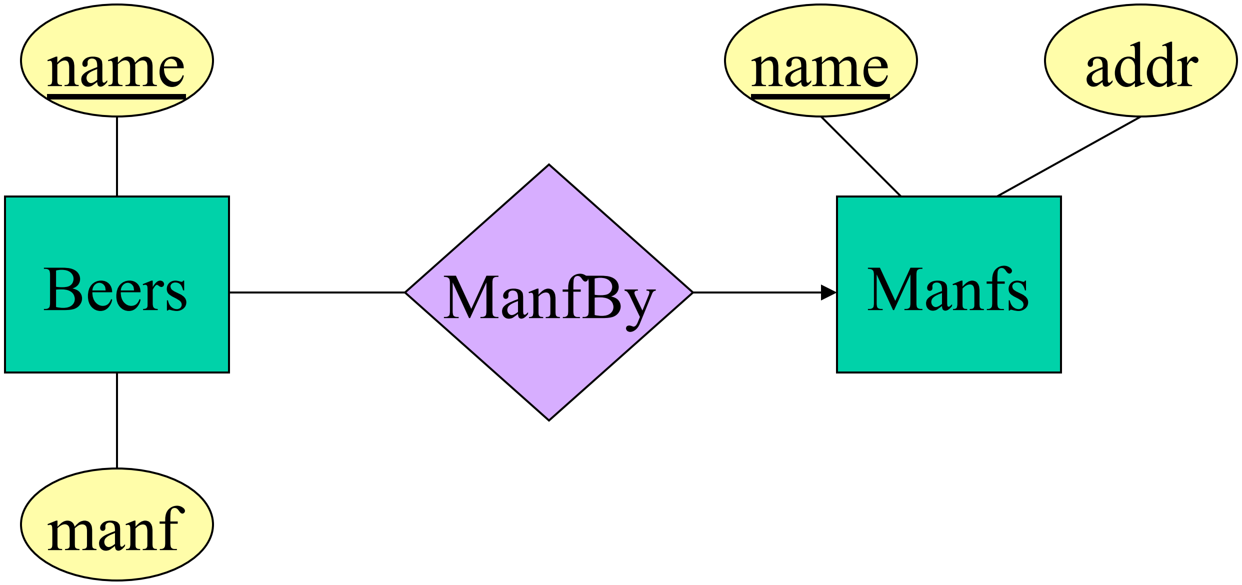
- Redundancy occurs when we say the same thing in two different ways.
- Redundancy wastes space and (more importantly) encourages inconsistency.
  - The two instances of the same fact may become inconsistent if we change one and forget to change the other, related version.

Example: Good



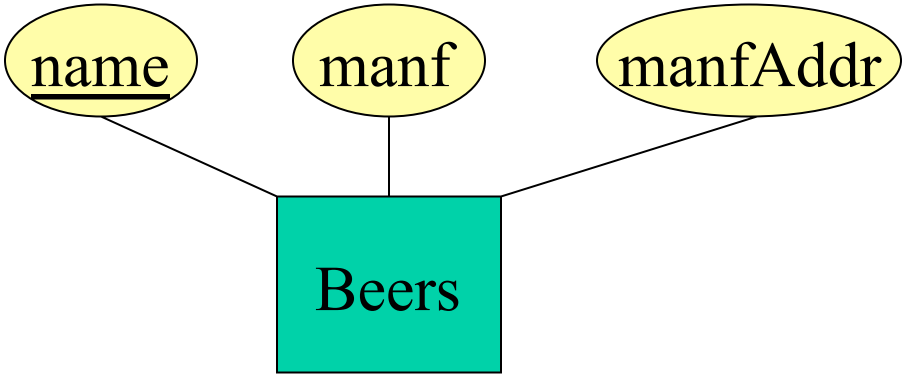
This design gives the address of each manufacturer exactly once.

Example: Bad



This design states the manufacturer of a beer twice: as an attribute and as a related entity.

Example: Bad



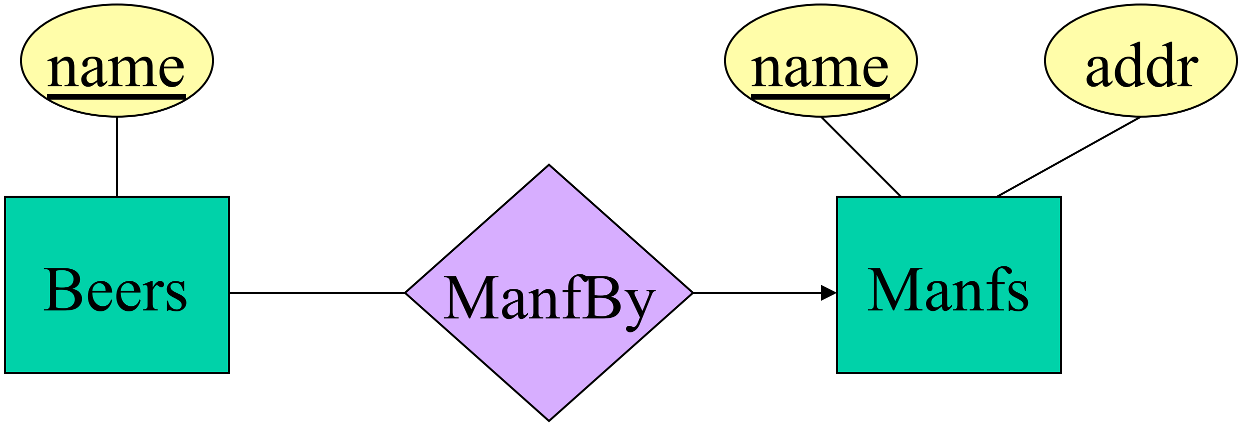
This design repeats the manufacturer’s address once for each beer; loses the address if there are temporarily no beers for a manufacturer.

Entity Sets Versus Attributes

- 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 nonkey attribute.

    or
  - It is the "many" in a many-one or many-many relationship.

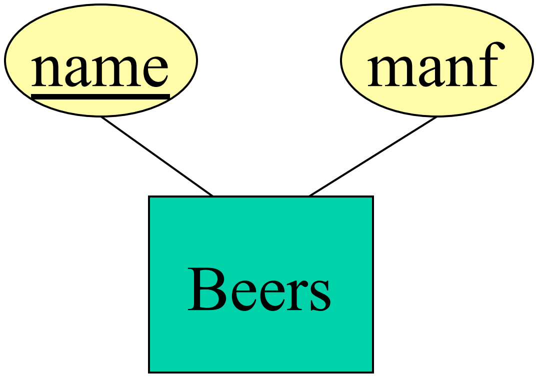
Example: Good



- *Manfs* deserves to be an entity set because of the nonkey attribute *addr*.

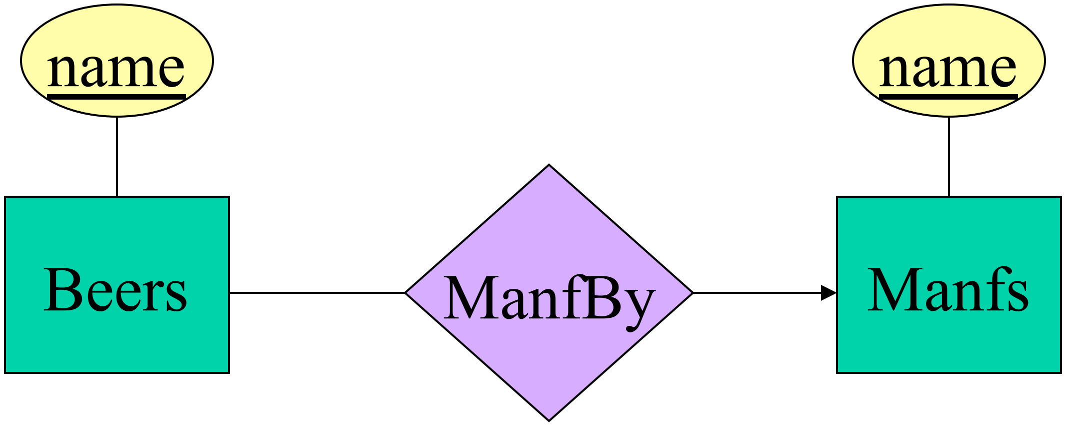
- *Beers* deserves to be an entity set because it is the “many” of the many-one relationship *ManfBy*.

Example: Good



There is no need to make the manufacturer an entity set, because we record nothing about manufacturers besides their name.

Example: Bad



Since the manufacturer is nothing but a name, and is not at the "many" end of any relationship, it should not be an entity set.

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 usually create unique ID’s for entity sets.
  - Examples include social-security numbers, automobile VIN’s etc.

When Do We Need Weak Entity Sets?

- The usual reason is that there is no global authority capable of creating unique ID's.
- Example: it is unlikely that there could be an agreement to assign unique player numbers across all football teams in the world.

ER Review

- Basic stuff
  - entity, attribute, entity set
  - relation: binary, multiway, converting from multiway
  - relationship roles, attributes on relationships
  - subclasses (is-a)
- Constraints
  - on relations
    - many-one, one-one, many-many
    - limitations of arrows
  - keys, single-valued, ref integrity, domain & general constraints

ER Review

- Weak entity set
- Design principles