Lecture 2: E/R Diagrams and the Relational Model

Wednesday, April 3rd, 2002

Outline

- E/R Diagrams
  - Reading assignment: 2.4 (except 2.4.2, 2.4.5), 2.5 (except 2.5.4)
- The Relational Model
  - Reading assignment: 3.1, 3.2, 3.3
- E/R to Relational:
  - Reading assignment: 3.5
- SQL
  - Reading assignment: 5.1, 5.2

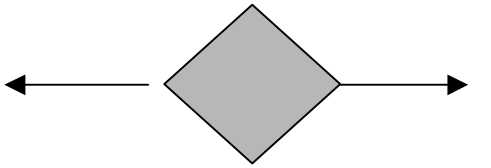
NOTE: lots of material in this lecture. Please read the book. Will review some key issues in the next lecture

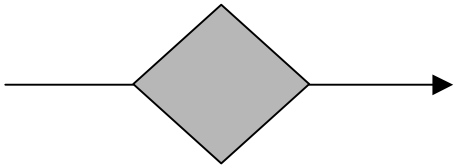
E/R Diagrams: Review

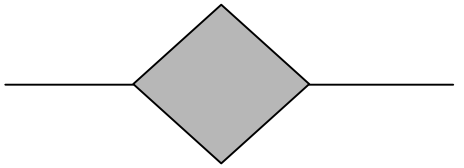
Entity sets: Product

Properties: address

Relationships: buys

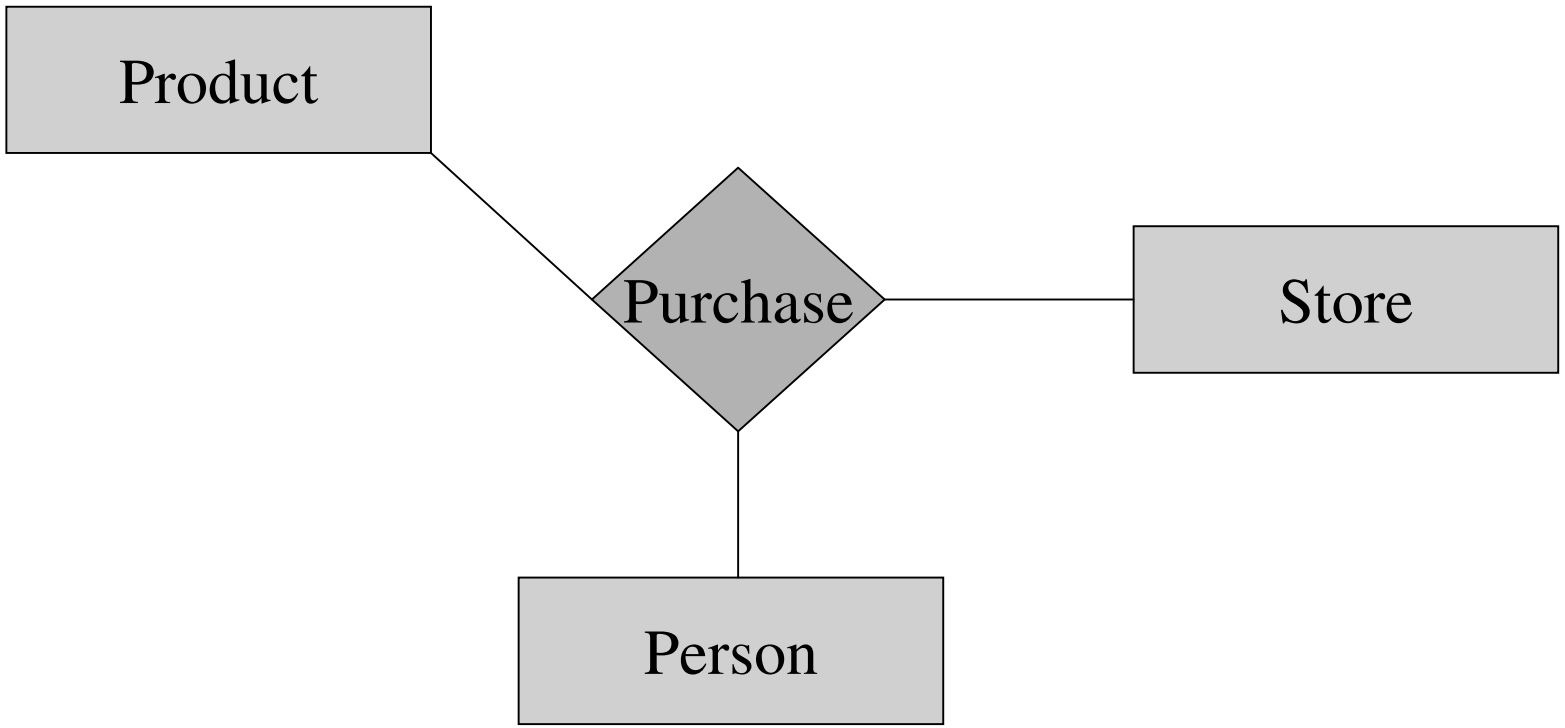






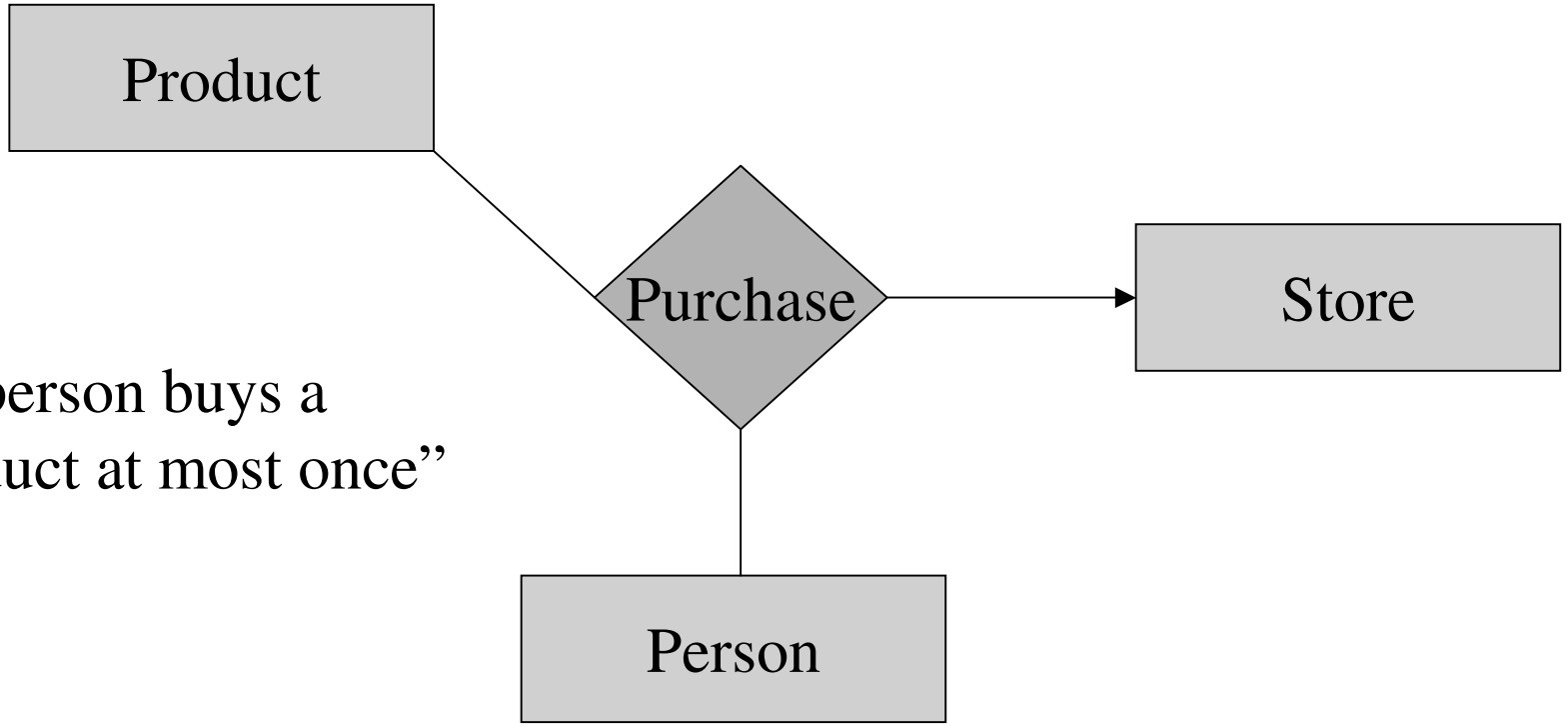
Multi-way Relationships in E/R

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



Can still model as a mathematical set (how ?)

Arrows in Multi-way Relationships (not in the book)

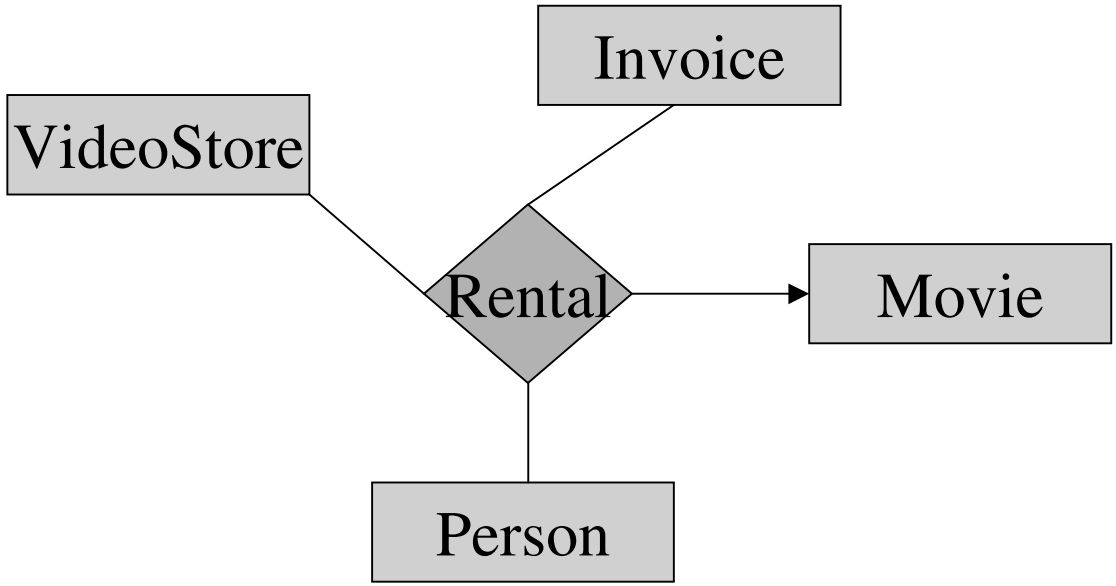


"A person buys a product at most once"

Limited expressive power.
Cannot say: "a person buys at most one product"

Arrows in Multiway Relationships

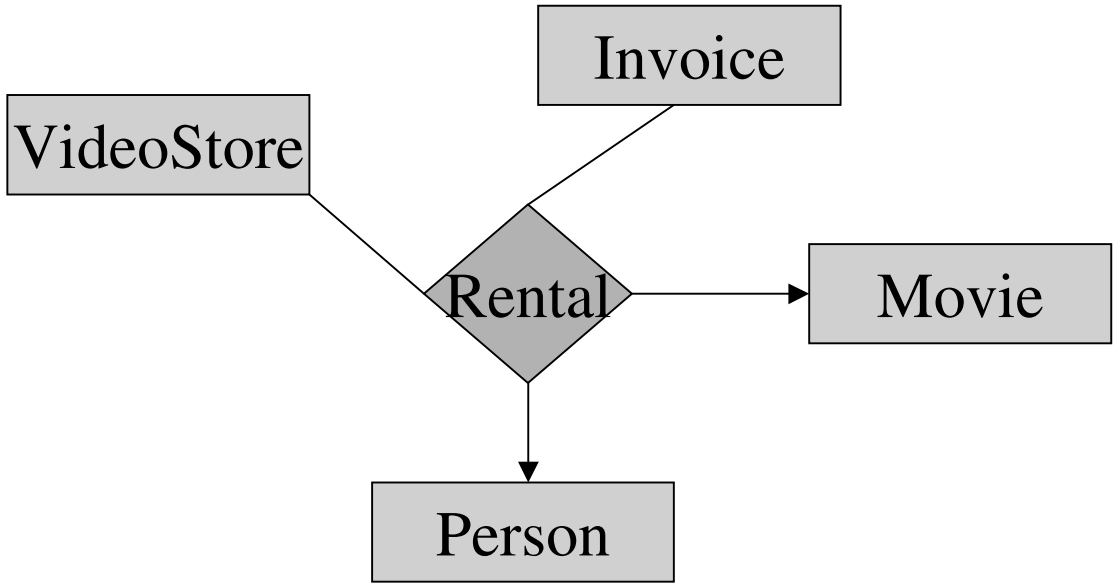
- Q: what does the arrow mean ?



- A: if I know the store, person, invoice, I know the movie too

Arrows in Multiway Relationships

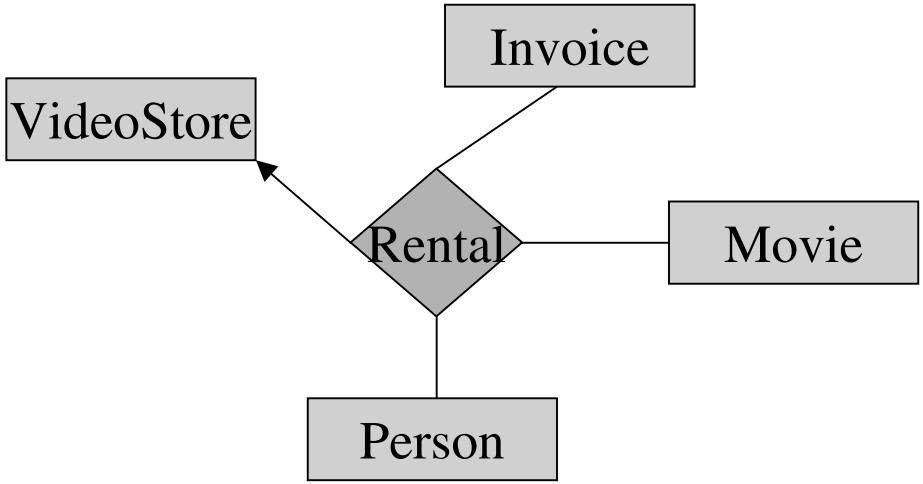
- Q: what do these arrow mean ?



- A: store, person, invoice determines movie and store, invoice, movie determines person

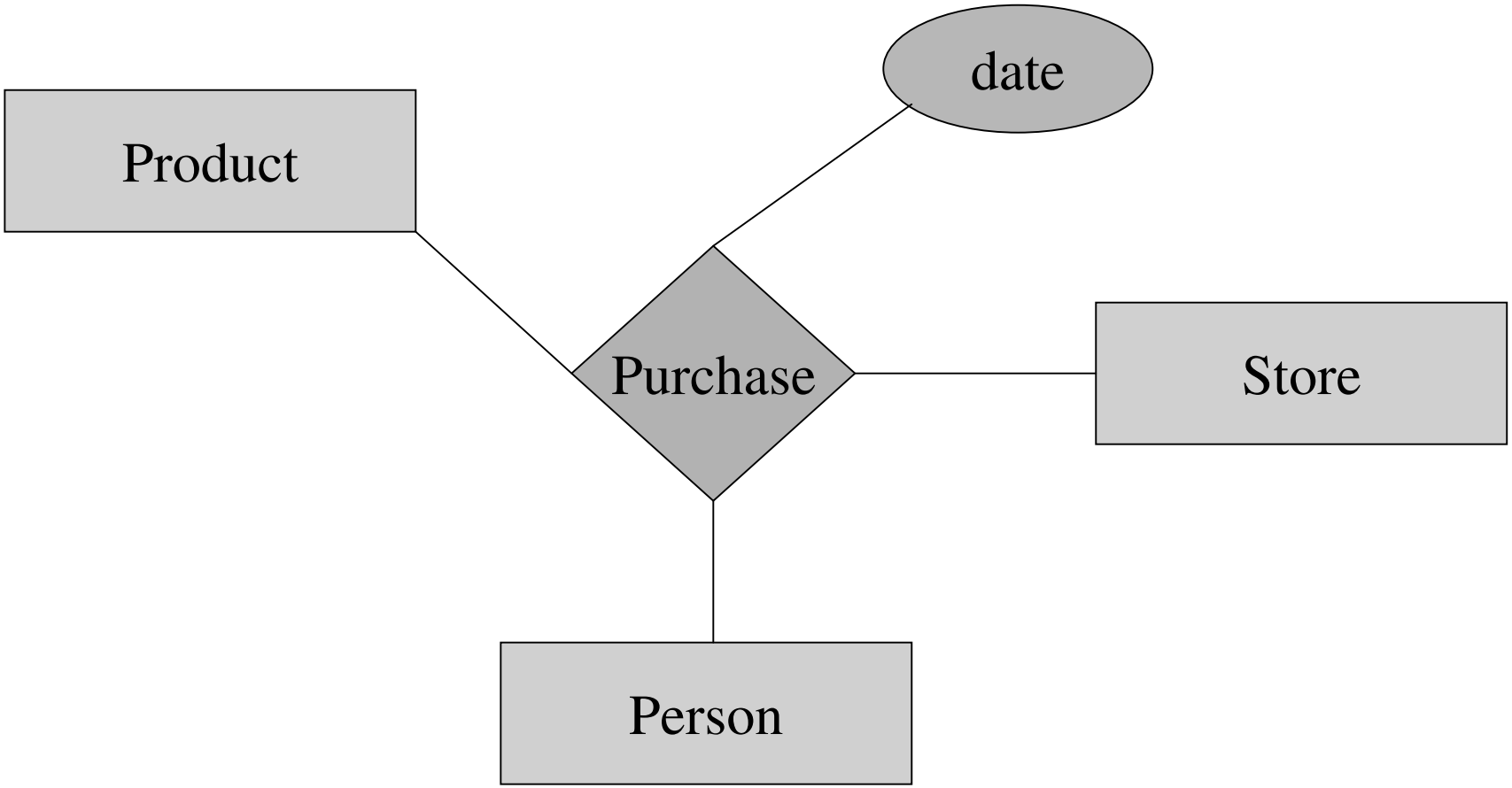
Arrows in Multiway Relationships

- Q: how do I say: “invoice determines store” ?
- A: no good way; best approximation:

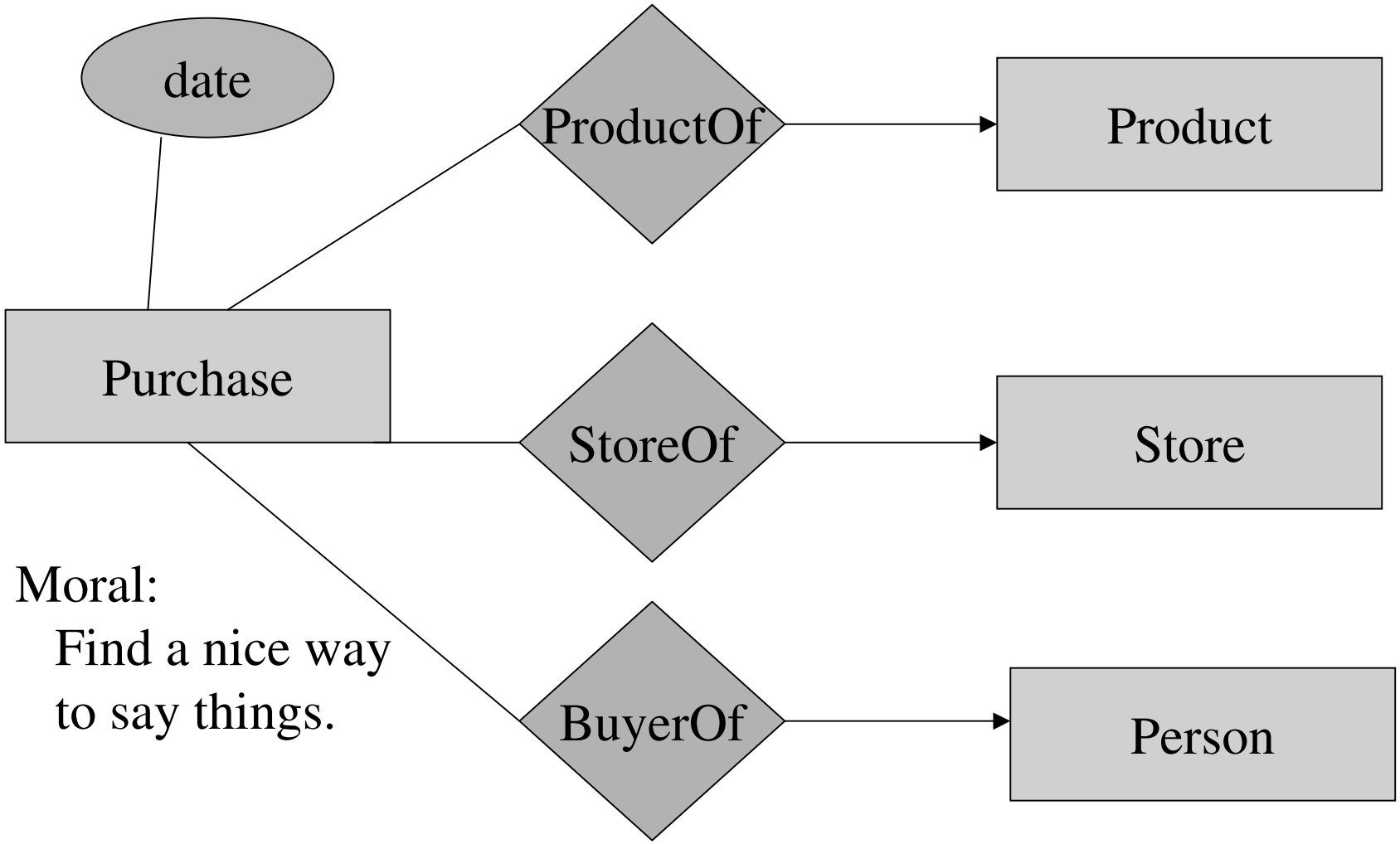


- Why is this incomplete ?

Converting Multi-way Relationships to Binary



Converting Multi-way Relationships to Binary

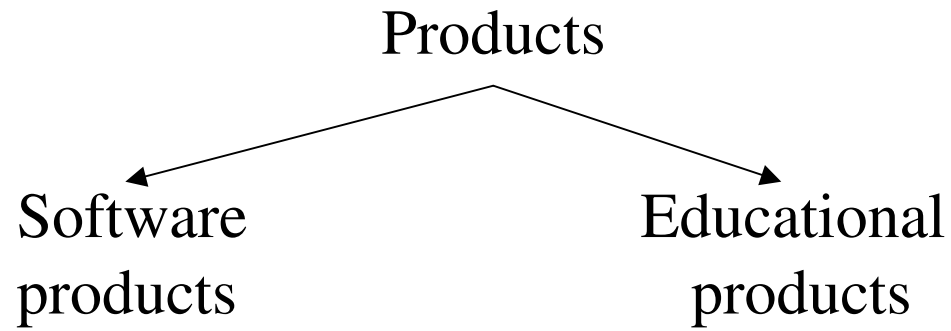


Moral:
Find a nice way to say things.

Subclasses and Inheritance in E/R

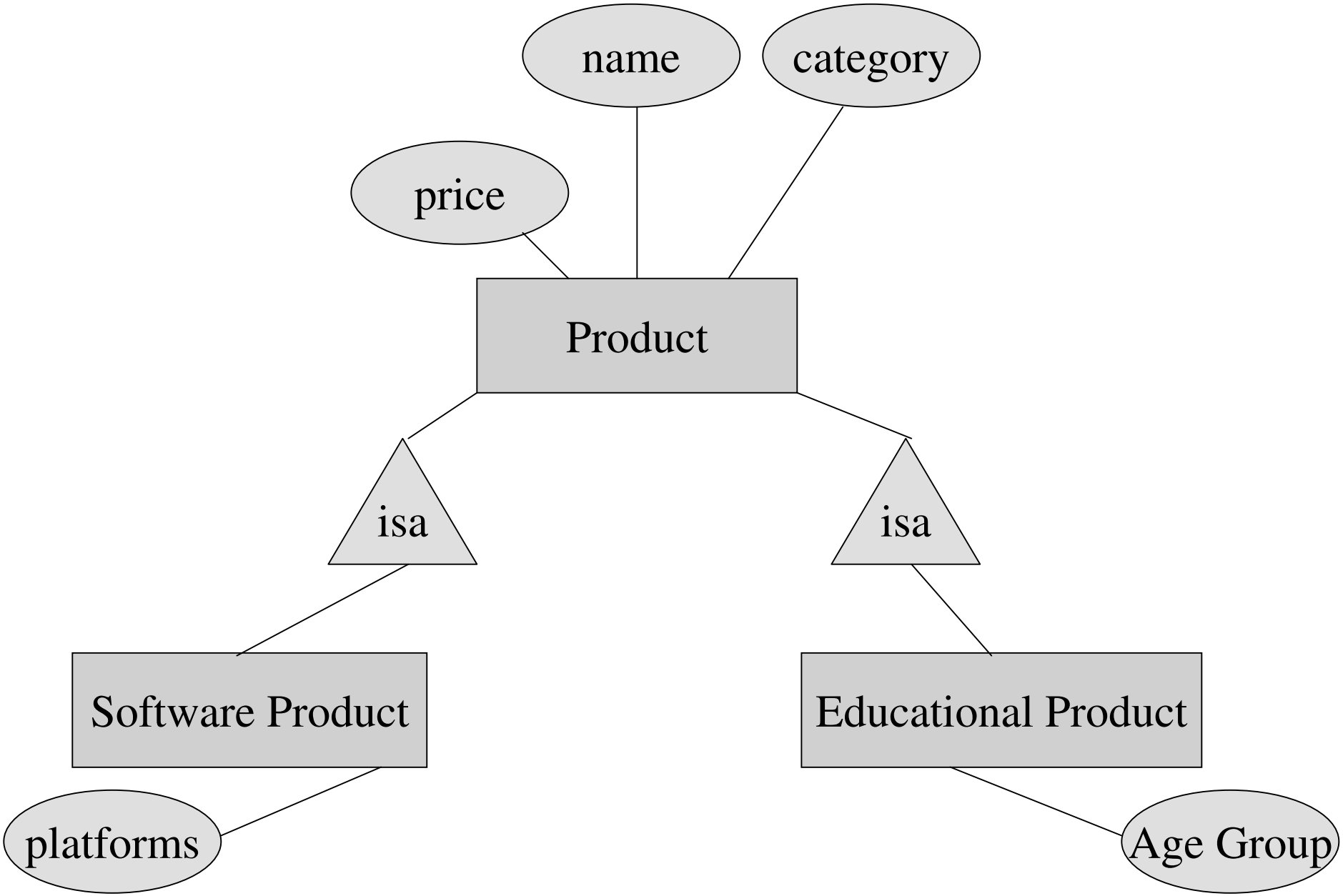
Some objects (entities) in a class may be special

- define a new class
- better: define a *subclass* that *inherits* the properties of the superclass



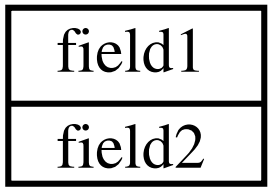
We define subclasses in E/R

Subclasses in E/R

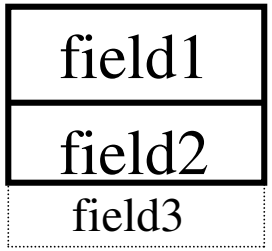


Understanding Subclasses

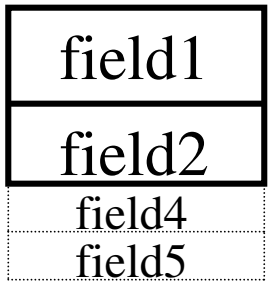
- Think in terms of records:
  - Product



  - SoftwareProduct

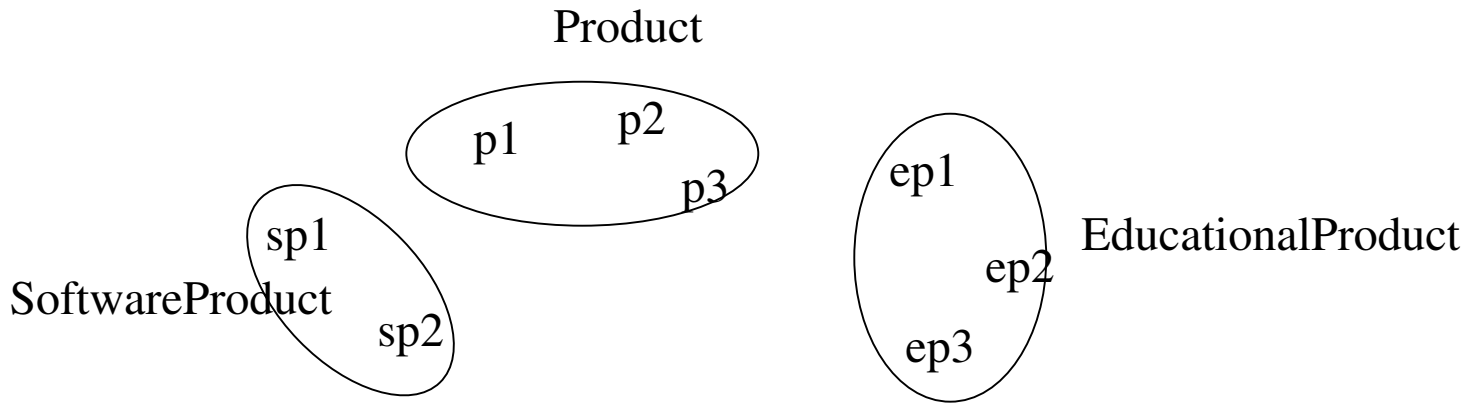


  - EducationalProduct



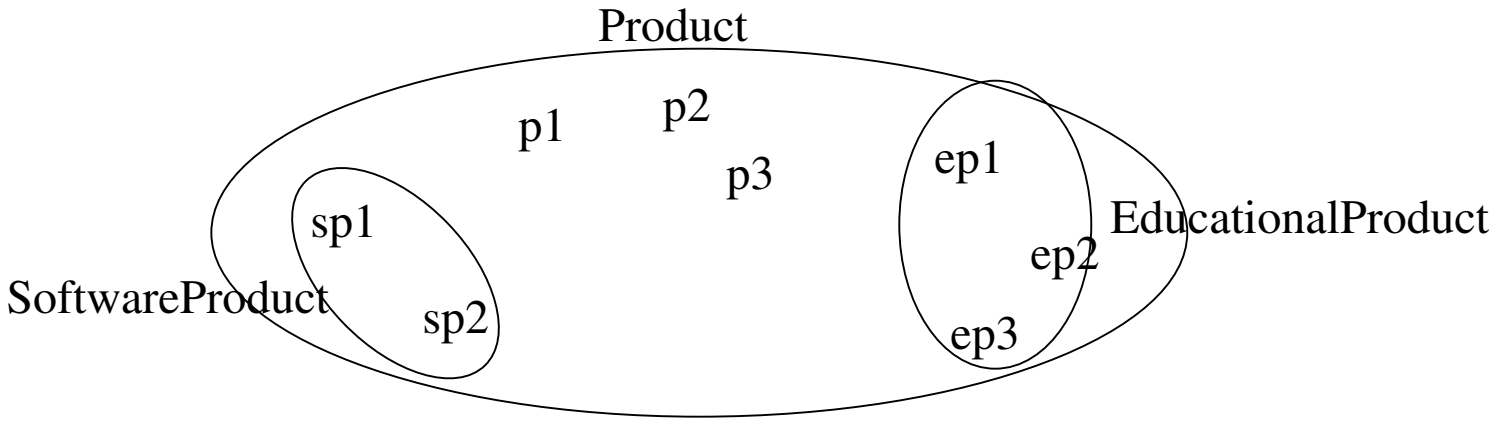
Difference between C++ and E/R inheritance

- C++: classes are disjoint



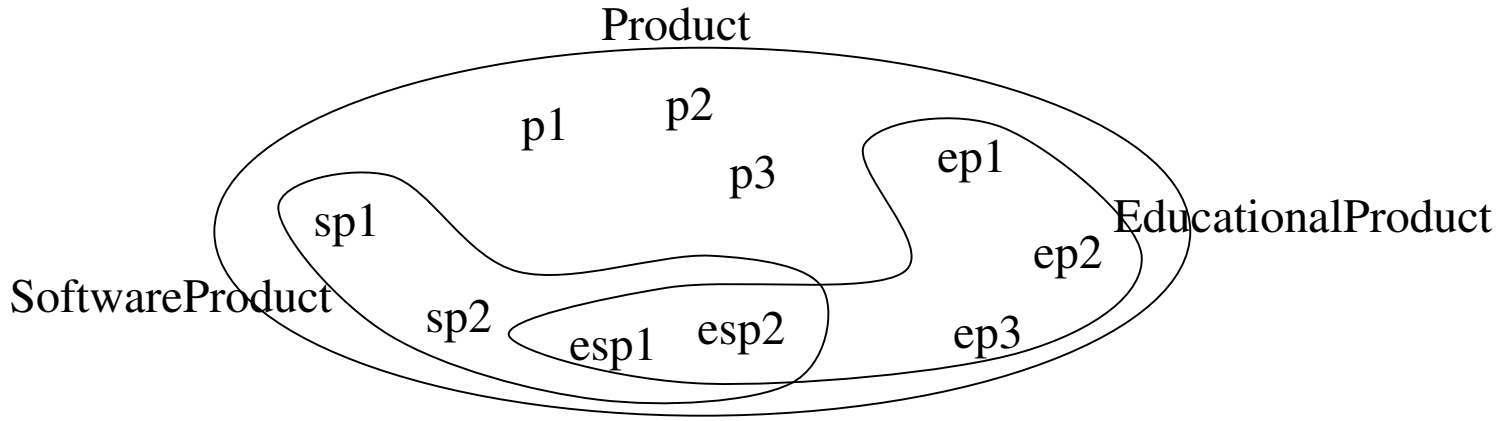
Difference between C++ and E/R inheritance

- E/R: entity sets overlap



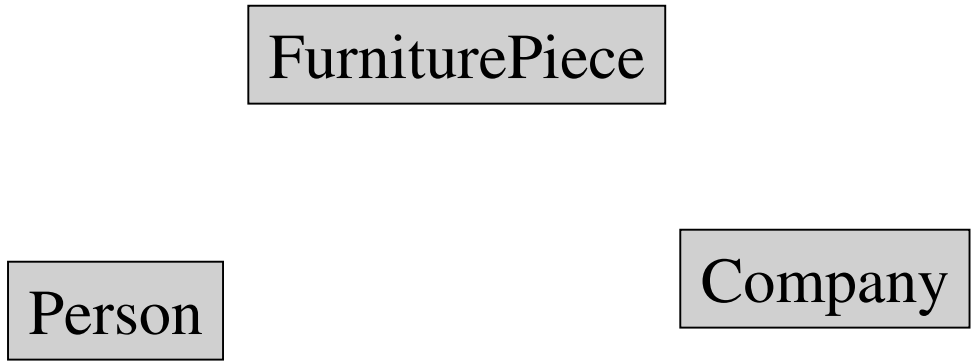
Difference between C++ and E/R inheritance

- No need for multiple inheritance in E/R



- we have three entity sets, but four different kinds of objects

Modeling Union Types in E/R

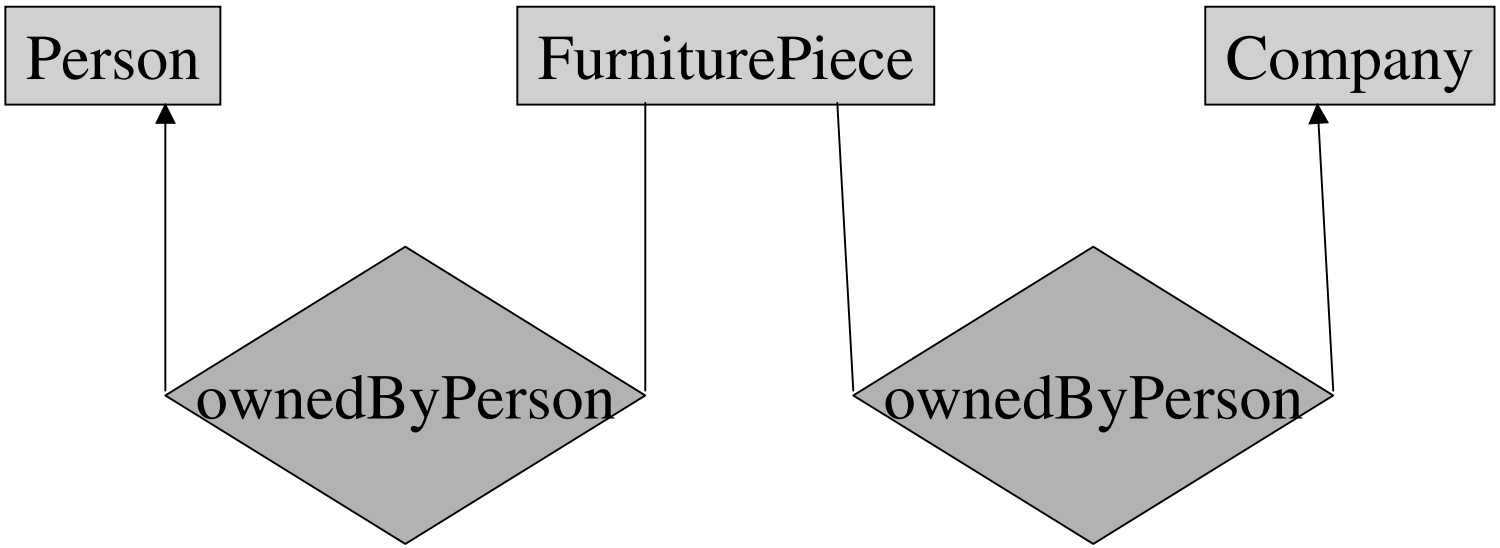


Say: each piece of furniture is owned either by a person, or by a company

Modeling Union Types in E/R

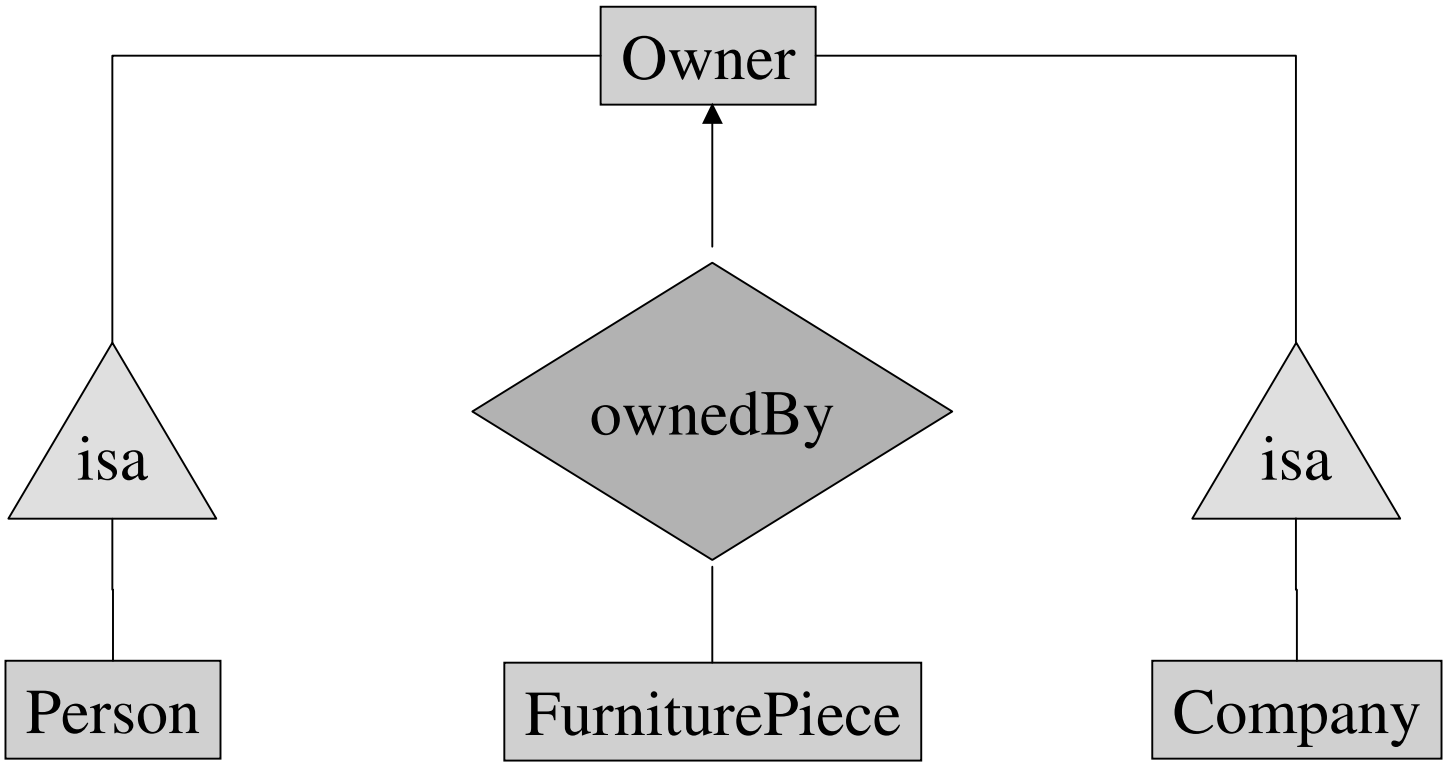
Say: each piece of furniture is owned either by a person, or by a company

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



Modeling Union Types in E/R

Solution 2: better, more laborious



Database Constraints

- A constraint = an assertion about the database that must be true at all times
- part of the db schema
- types in programming languages do not have anything similar
- correspond to *invariants* in programming languages

PLEASE READ THE TEXTBOOK

Database Constraints

Very important in databases

Keys: social security number uniquely identifies a person.

Single-value constraints: e.g. one-one, one-many, many-one

Participation constrain: total many-one

Domain constraints: peoples' ages are between 0 and 150.

General constraints: all others (at most 50 students enroll in a class)

Keys

A set of attributes that uniquely identify an object or entity:

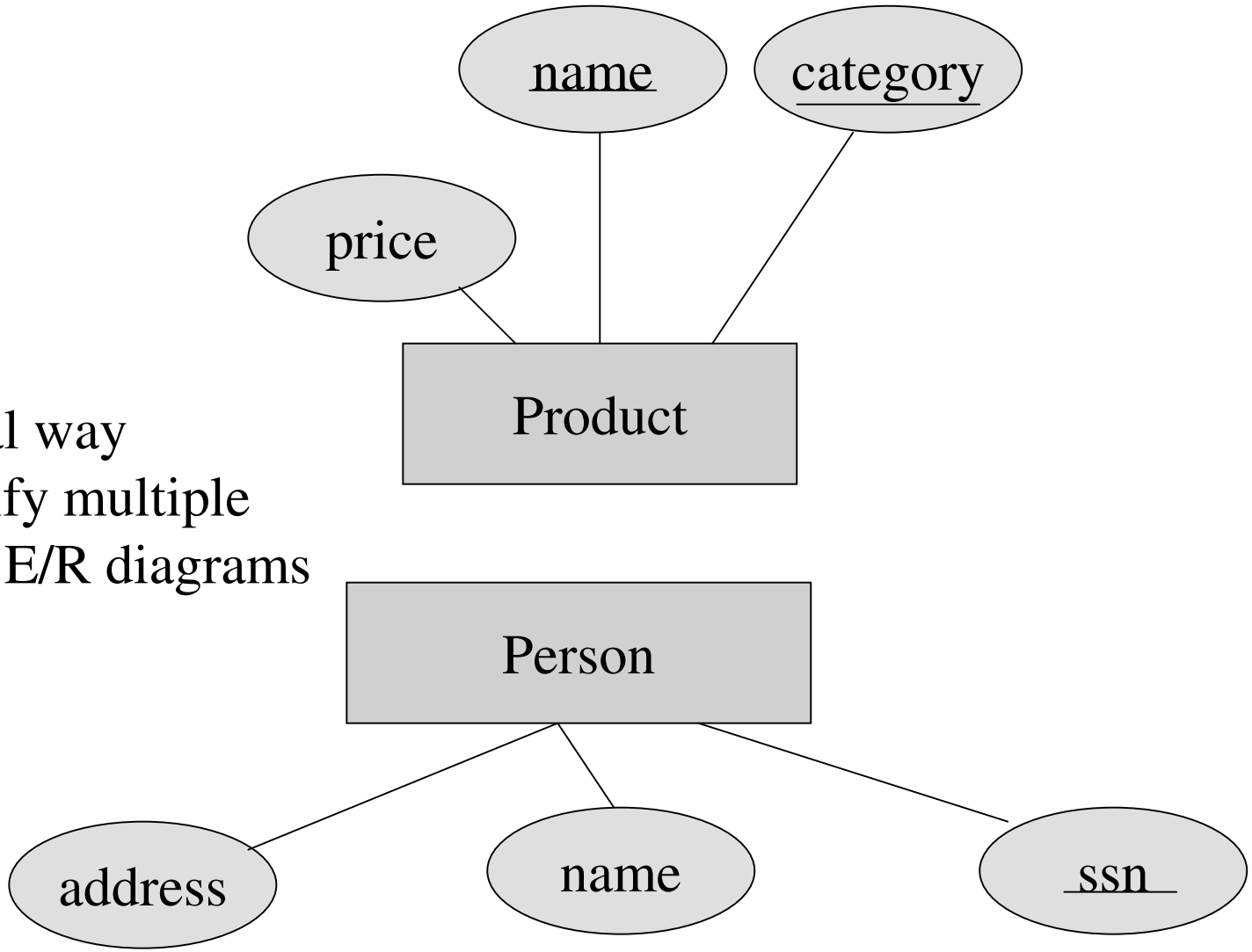
Person: ssn
name
name + address
name + address + age

Perfect keys are often hard to find, so organizations usually invent something.

An object may have multiple keys:

employee number, ssn

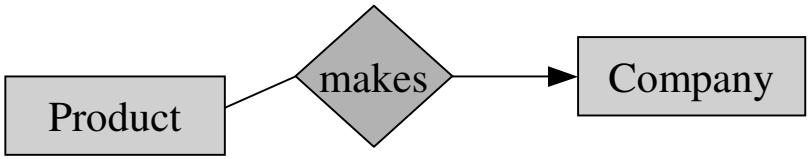
Keys in E/R Diagrams



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

Single Value Constraints in E/R

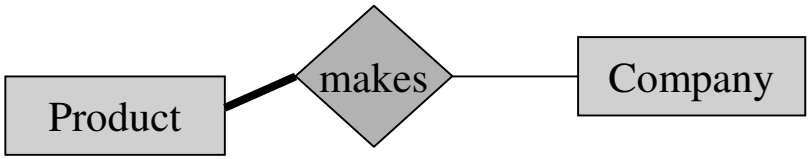
A product is made by at most one company:



Notice: some products are not made by any company

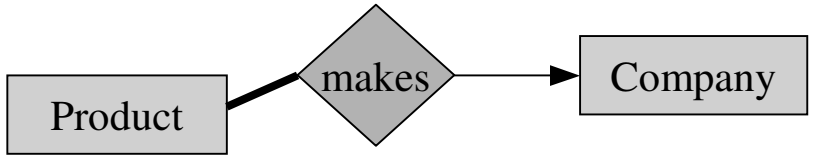
Participation Constraint

Each product is made by a lest one company



(notation from the book)

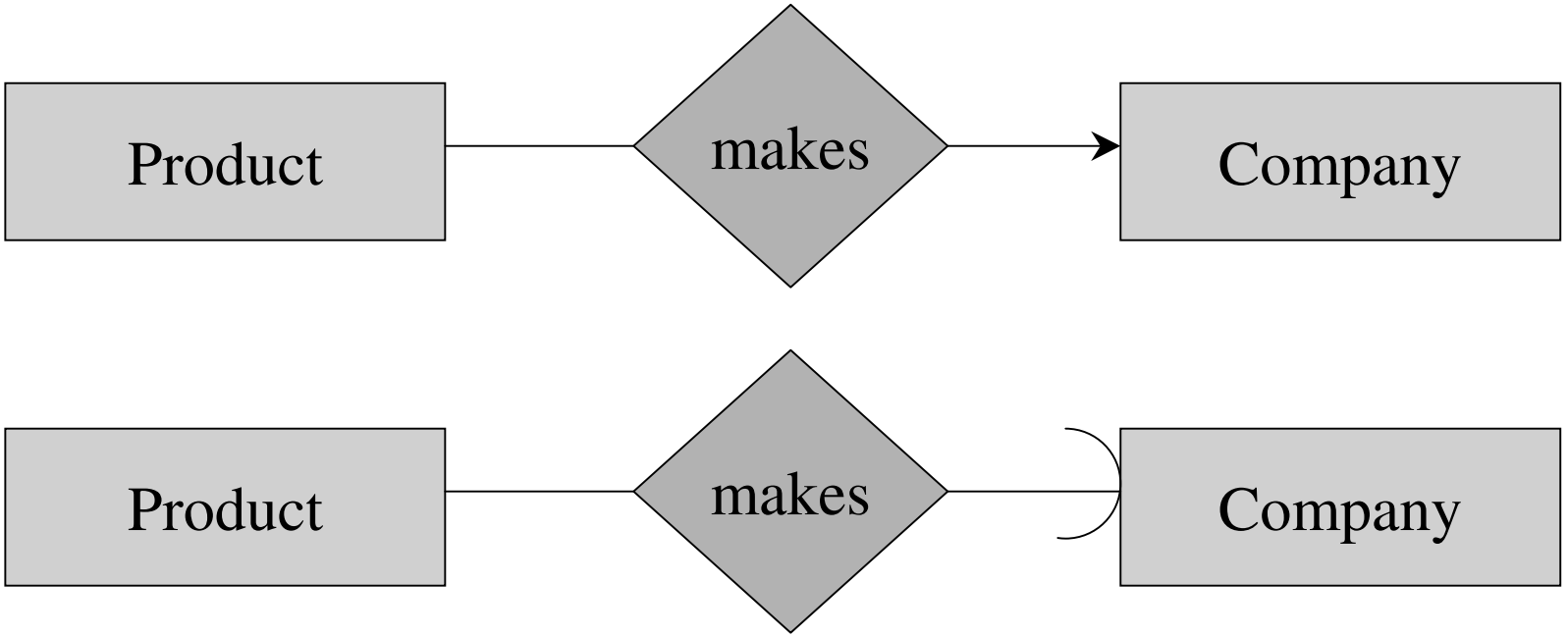
Each product is made by exactly one company



This: also called *referential integrity constraint*

Referential Integrity Constraint

- Another notation (in Ullman's book):

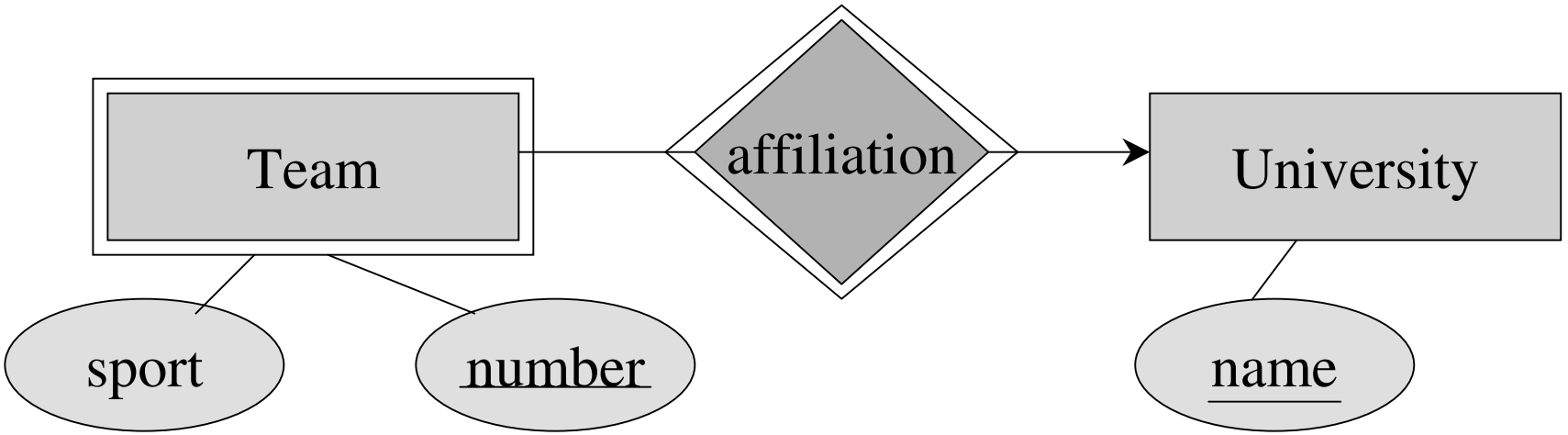


Weak Entity Sets

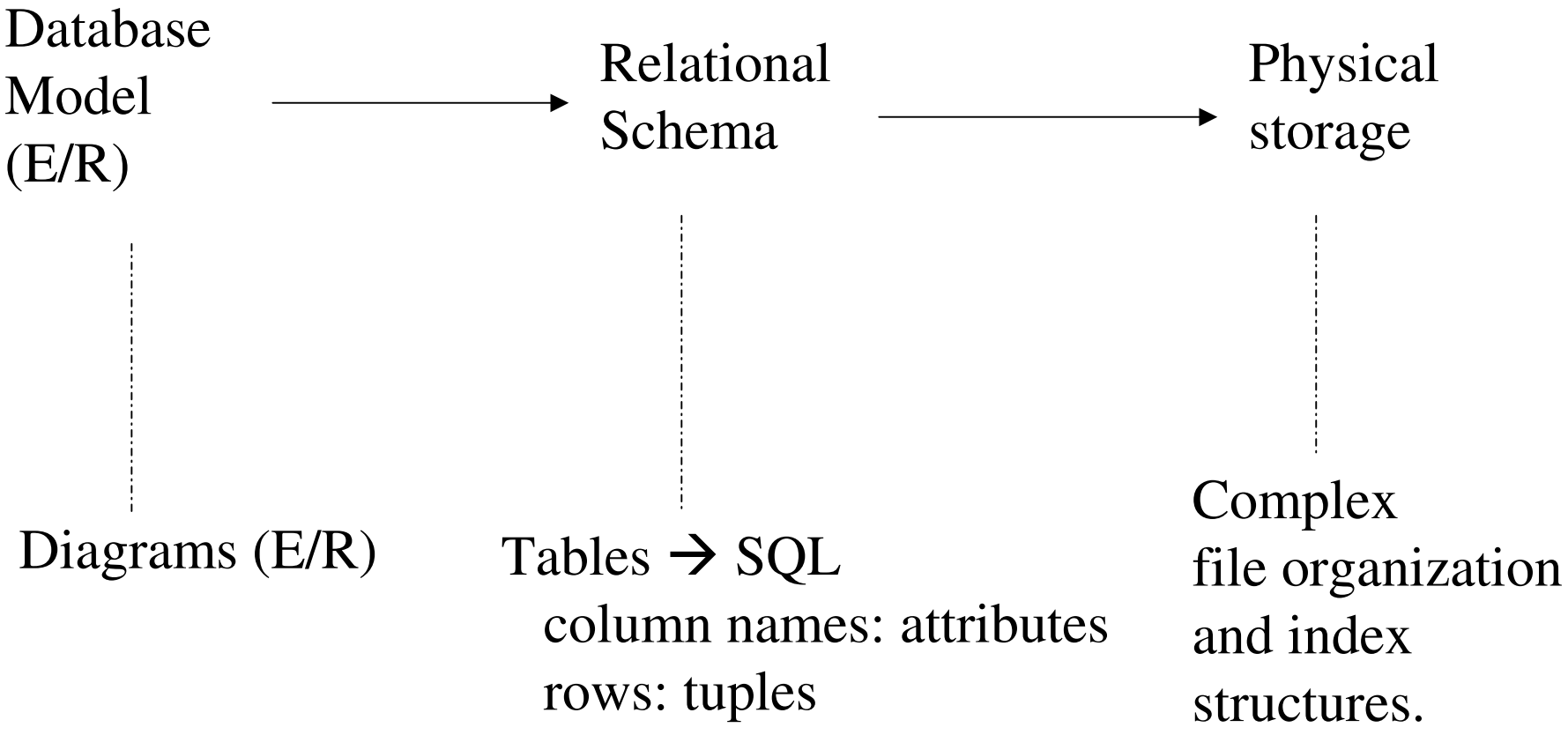
Entity sets are weak when their key attributes come from other classes to which they are related.

This happens if:

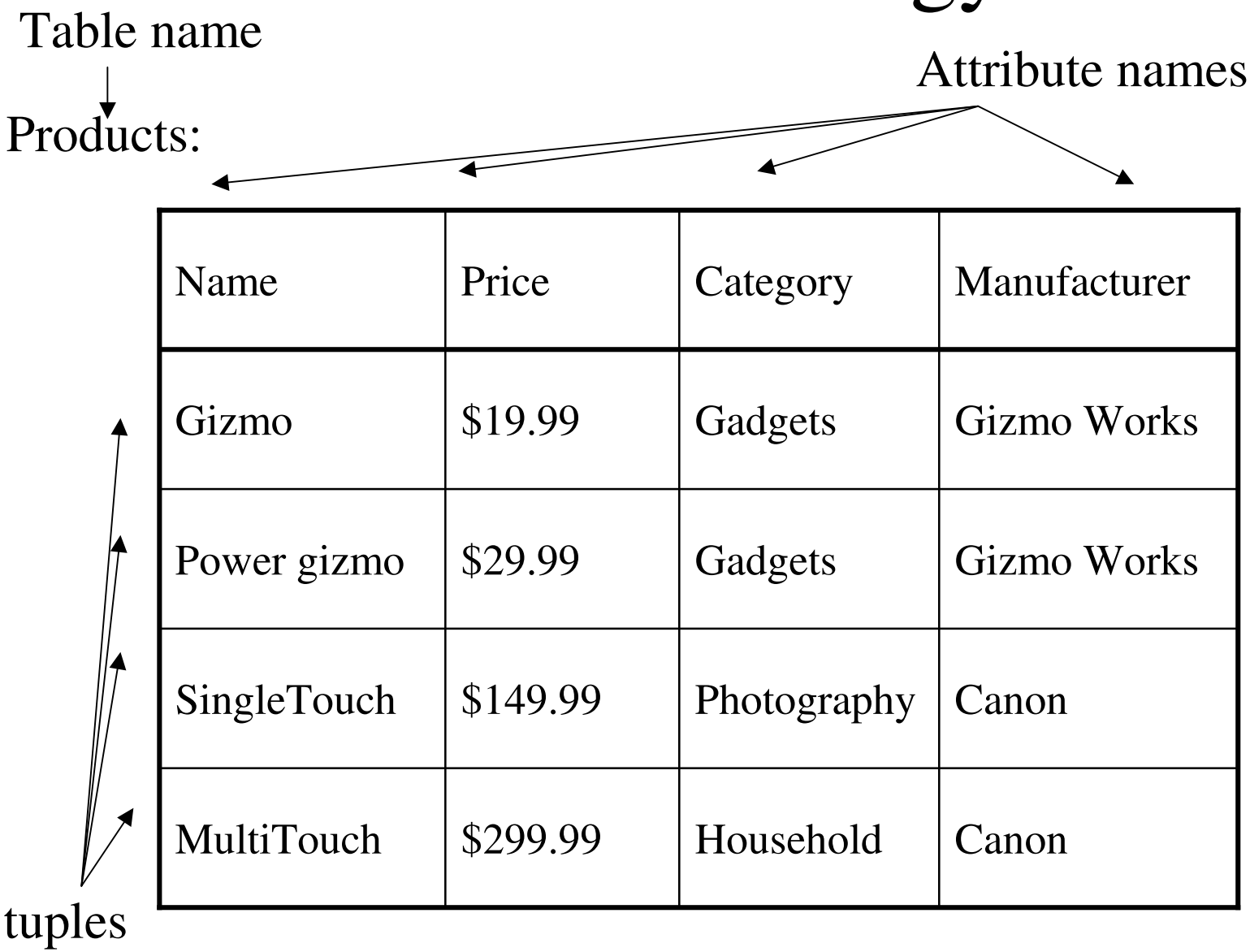
- part-of hierarchies
- splitting n-ary relations to binary.



The Relational Data Model



Terminology



Name	Price	Category	Manufacturer
Gizmo	$19.99	Gadgets	Gizmo Works
Power gizmo	$29.99	Gadgets	Gizmo Works
SingleTouch	$149.99	Photography	Canon
MultiTouch	$299.99	Household	Canon

Creating Tables in SQL

```
CREATE     TABLE Person(

            name                    VARCHAR(30),
            price                   REAL,
            category                VARCHAR(20),
            manufacturer            VARCHAR(30)
  );
```

Domains: CHAR, VARCHAR, INTEGER, REAL, etc, etc

Inserting Tuples SQL

```
INSERT
INTO      Person(name, price, category, manufacturer)
VALUES    ("Gizmo", 19.99,  "Gadgets", "Gizmo Works")
```

```
INSERT
INTO      Person(name)
VALUES    ("Power gizmo")
```

What happens with the missing fields ?

Tables with Default Values

Specifying default values:

```
CREATE    TABLE Person(

        name                    VARCHAR(30),
        price                   REAL  DEFAULT 99.99,
        category                VARCHAR(20) DEFAULT “General”,
        manufacturer            VARCHAR(30)
 );
```

The default of defaults: NULL

Foundations of the Relational Model

- ***Relational Schema:***
  - Relation name plus attribute names
  - E.g. Product(Name, Price, Category, Manufacturer)
  - In practice we add the domain for each attribute

- ***Database Schema:***
  - Set of relational schemas
  - E.g. Product(Name, Price, Category, Manufacturer)
    Vendor(Name, Address, Phone)

Foundations of the Relational Model

- An instance of a relational schema R(A1,…,Ak), is a relation with k attributes with values of corresponding domains

- An instance of a database schema R1(…), R2(…), …, Rn(…), consists of n relations, each an instance of the corresponding relational schema.

Example

Relational schema: Product(Name, Price, Category, Manufacturer)

Relational instance:

Name	Price	Category	Manufacturer
Gizmo	$19.99	Gadgets	Gizmo Works
Power gizmo	$29.99	Gadgets	Gizmo Works
SingleTouch	$149.99	Photography	Canon
MultiTouch	$299.99	Household	Canon

Schemas and Instances

- Analogy with programming languages:
  - Schema = type
  - Instance = value
- Important distinction:
  - Database Schema = stable over long periods of time
  - Database Instance = changes constantly, as data is inserted/updated/deleted

Two Mathematical Definitions of Relations

Definition A relation $R(dom_1, ..., dom_n)$ is a subset of the cartesian product: $R \subseteq dom_1 \times ... \times dom_n$

Example

- Product $\subseteq$ **string** x **real** x **string** x **string**
- Order in the tuple is important !
  - (gizmo, 19, gadgets, GizmoWorks)
  - (gizmo, 19 , GizmoWorks, gadgets)
- No attributes

Two Mathematical Definitions of Relations

Definition A relation $R(A_1:dom_1, ..., A_n:dom_n)$ is a set of functions $t: \{A_1, ..., A_n\} \rightarrow dom_1 \cup ... \cup dom_n$ s.t. $t(A_1) \in A_1, ..., t(A_n) \in A_n$

Example

- A={name , price, category, manufacturer}
- Example of a tuple:

{name → gizmo,
price → 19,
category → gadgets,
manufacturer → gizmoWorks}

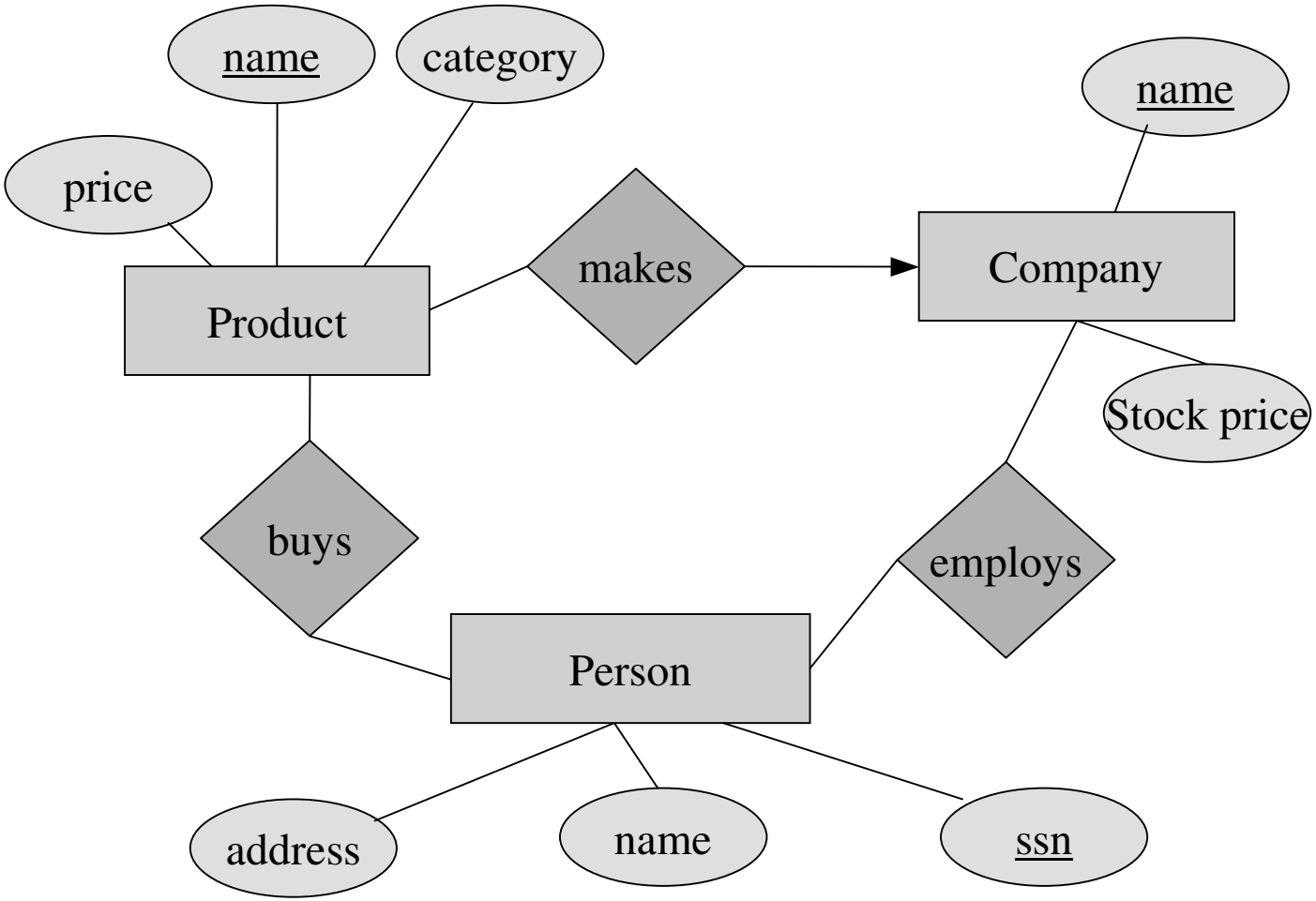
- Order in a tuple is not important
- Attribute names are important

Two Definitions of Relations

- We will switch back and forth between these two:
  - Relational schemas with attribute names
  - Positional tuples, without attribute names

From E/R Diagrams to Relational Schema

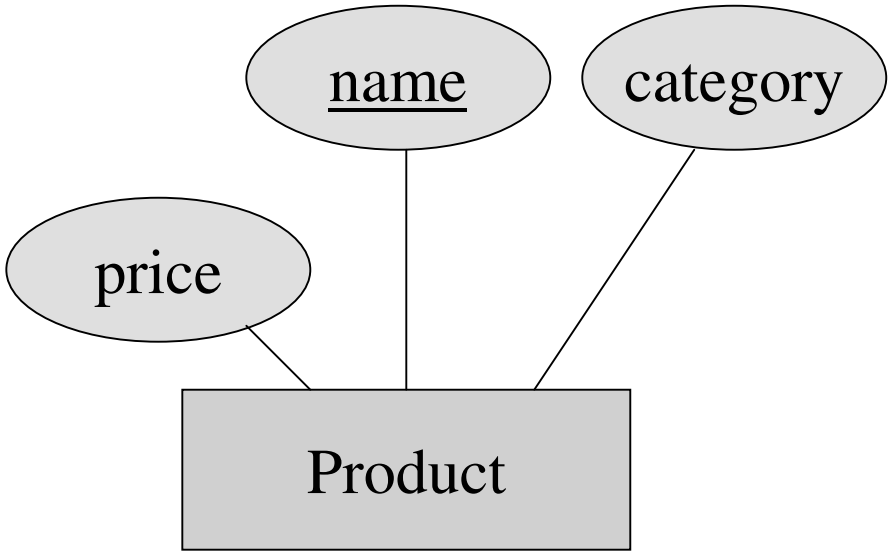
- Each entity set → relation
- Each relationship → relation
- Special cases:
  - one-one, many-one relationships
  - subclasses
  - weak entity sets



Convert to
Five tables (why ?):

Product(<u>name</u>, price, category, cname)
Company(<u>name</u>, stockPrice)
Person(<u>ssn</u>, name, address)
Buys(<u>pname, ssn</u>)
Employs(<u>cname, ssn</u>)

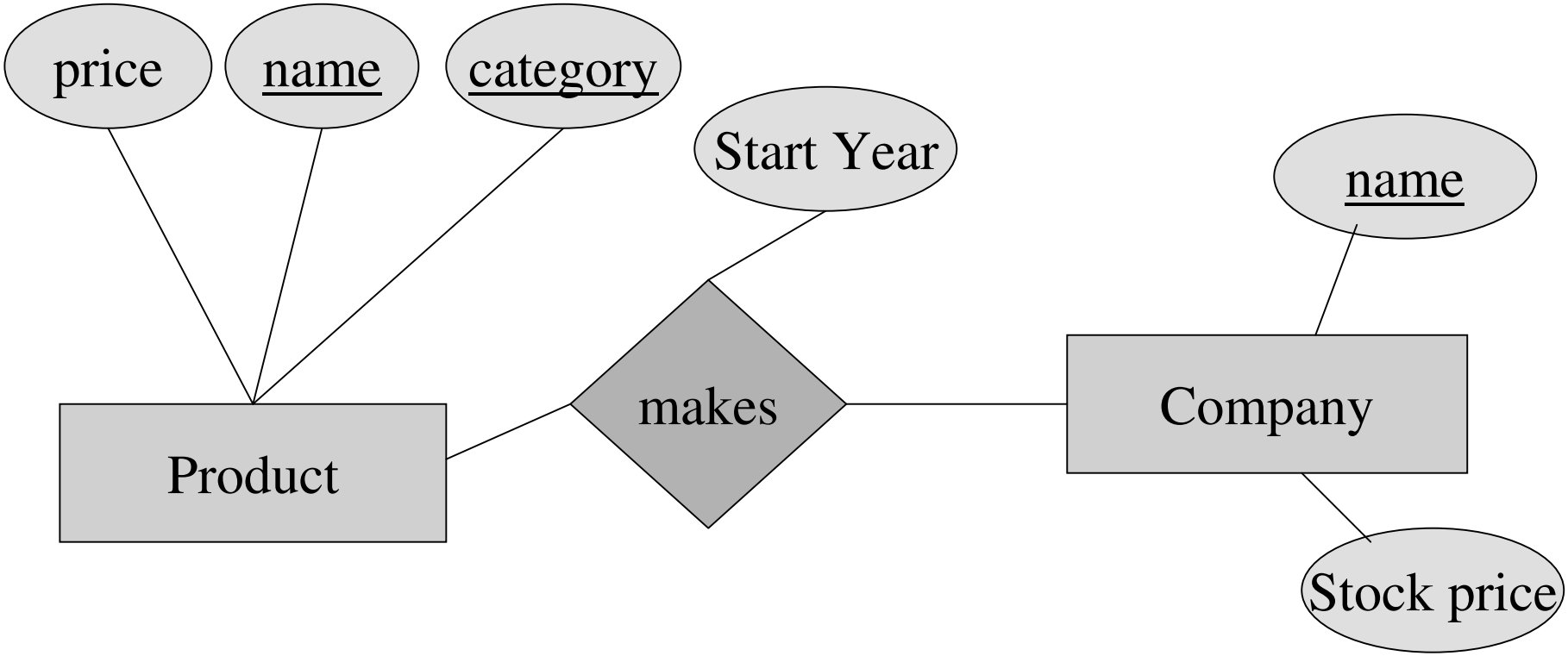
Entity Sets to Relations



Product:

Name	Category	Price
gizmo	gadgets	$19.99

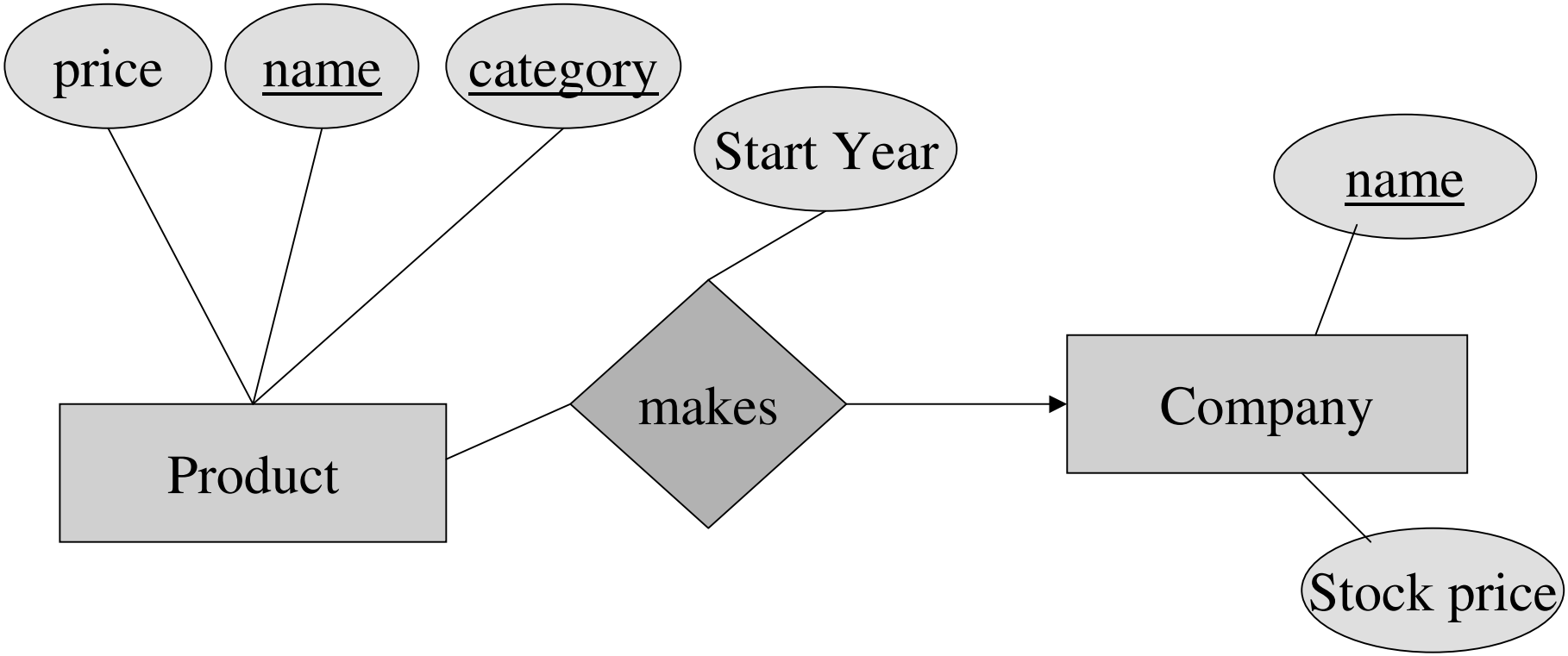
Relationships to Relations



Relations: Product, Makes, Company:

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

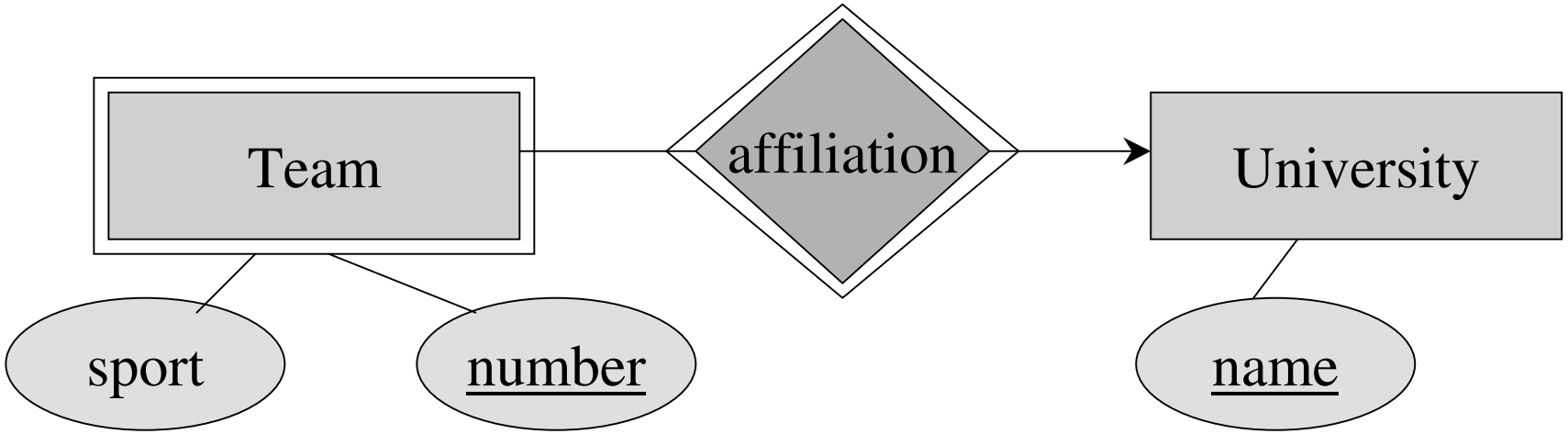
Many-one Relationships



No need for Makes. Just modify Product:

name	category	price	StartYear	companyName
gizmo	gadgets	19.99	1963	gizmoWorks

Handling Weak Entity Sets

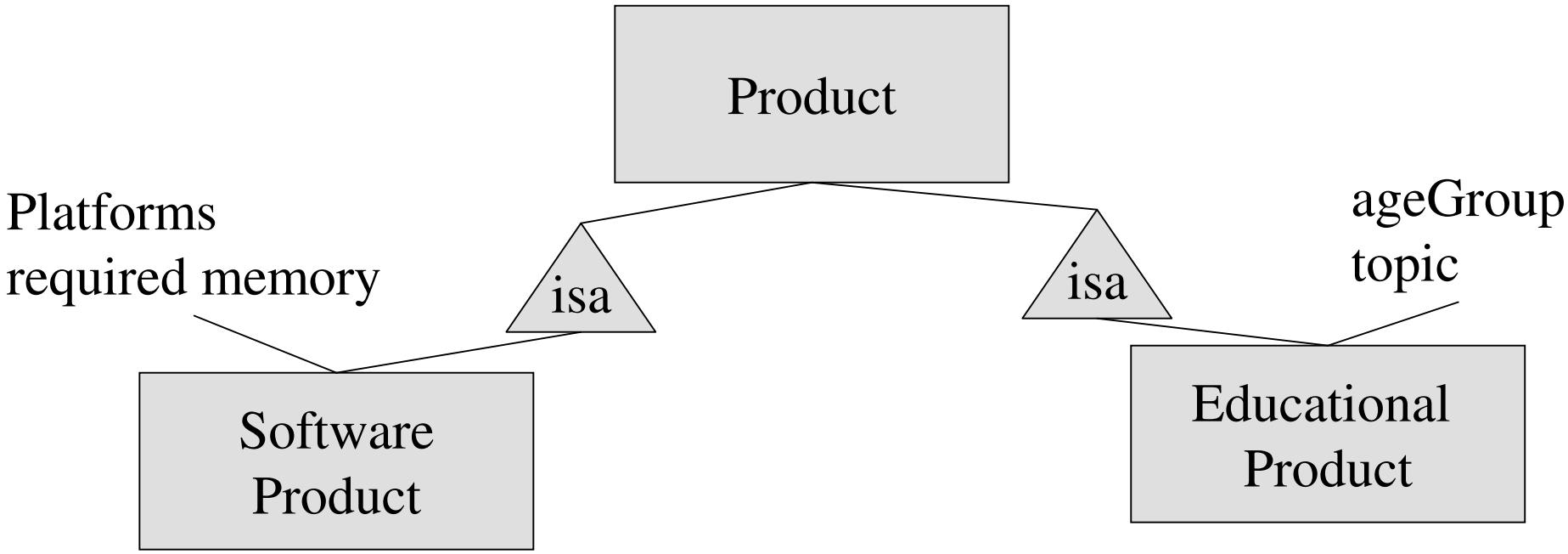


Relation Team:

Sport	Number	Affiliated University
mud wrestling	15	Montezuma State U.

- need all the attributes that contribute to the key of Team
- don’t need a separate relation for Affiliation. (why ?)

Modeling Subclass Structure



The right way

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

EducationalProduct(<u>name</u>, ageGroup, topic)

SoftwareProduct(<u>name</u>, platforms, requiredMemory)

Notice: each subclass stores only the *additional* attributes

Option #2: The Null Value Approach

Have one table:

Product (name, price, manufacturer, age-group, topic, platforms, required-memory, educational-method)

Some values in the table will be NULL, meaning that the attribute not make sense for the specific product.

SQL Introduction

Standard language for querying and manipulating data

Structured **Q**uery **L**anguage

Many standards out there: SQL92, SQL2, SQL3, SQL99
Vendors support various subsets of these, but all of what we'll be talking about.

SQL Introduction

Basic form: (many many more bells and whistles in addition)

```
Select   [attributes]
From     [relations]
Where    [conditions]
```

Selections

Company(sticker, name, country, stockPrice)

Find all US companies whose stock is > 50:

```
SELECT   *
FROM     Company
WHERE    country="USA" AND stockPrice > 50
```

Output schema: R(sticker, name, country, stockPrice)

Selections

What you can use in WHERE:

- attribute names of the relation(s) used in the FROM.
- comparison operators: =, <>, <, >, <=, >=
- apply arithmetic operations: stockprice*2
- operations on strings (e.g., "||" for concatenation).
- Lexicographic order on strings.
- Pattern matching: s LIKE p
- Special stuff for comparing dates and times.

The **LIKE** operator

- s **LIKE** p: pattern matching on strings
- p may contain two special symbols:
  - % = any sequence of characters
  - _ = any single character

Company(sticker, name, address, country, stockPrice)
Find all US companies whose address contains "Mountain":

```
SELECT   *
FROM     Company
WHERE    country="USA" AND
         address LIKE "%Mountain%"
```

Projections

Select only a subset of the attributes

```
SELECT   name, stockPrice
FROM     Company
WHERE    country="USA" AND stockPrice > 50
```

Input schema: Company(sticker, name, country, stockPrice)
Output schema: R(name, stock price)

Projections

Rename the attributes in the resulting table

```
SELECT   name AS company, stockprice AS price
FROM     Company
WHERE    country="USA" AND stockPrice > 50
```

Input schema: Company(sticker, name, country, stockPrice)
Output schema: R(company, price)

Ordering the Results

```
SELECT   name, stockPrice
FROM     Company
WHERE    country="USA" AND stockPrice > 50
ORDERBY  country, name
```

Ordering is ascending, unless you specify the DESC keyword.

Ties are broken by the second attribute on the ORDERBY list, etc.

Joins

Product (pname, price, category, maker)
Purchase (buyer, seller, store, product)
Company (cname, stockPrice, country)
Person(pname, phoneNumber, city)

Find names of people living in Seattle that bought gizmo products, and the names of the stores they bought from

```
SELECT   pname, store
FROM     Person, Purchase
WHERE    pname=buyer AND city="Seattle"
         AND product="gizmo"
```

Disambiguating Attributes

Find names of people buying telephony products:

Product (name, price, category, maker)
Purchase (buyer, seller, store, product)
Person(name, phoneNumber, city)

```
SELECT   Person.name
FROM     Person, Purchase, Product
WHERE    Person.name=Purchase.buyer
     AND  Product=Product.name
     AND  Product.category="telephony"
```

Tuple Variables

Find pairs of companies making products in the same category

```
SELECT  product1.maker, product2.maker
FROM    Product AS product1, Product AS product2
WHERE  product1.category=product2.category
   AND  product1.maker <> product2.maker
```

Product (name, price, category, maker)

Tuple Variables

Tuple variables introduced automatically by the system:

Product (name, price, category, maker)

```
SELECT  name
FROM    Product
WHERE  price > 100
```

Becomes:

```
SELECT Product.name
FROM   Product AS Product
WHERE Product.price > 100
```

Doesn't work when Product occurs more than once:
In that case the user needs to define variables explicitely.