Database Systems
CSE 414

Lecture 22: E/R Diagrams (4.1-6)
and Constraints (7.1-2)

Announcements

- HW7 will be posted on Tuesday and due on Dec. 1st 11pm

- WQ6 will be posted on Tuesday and due on Nov. 30th 11pm

Database Design

What it is:

- Starting from scratch, design the database schema: relation, attributes, keys, foreign keys, constraints etc.

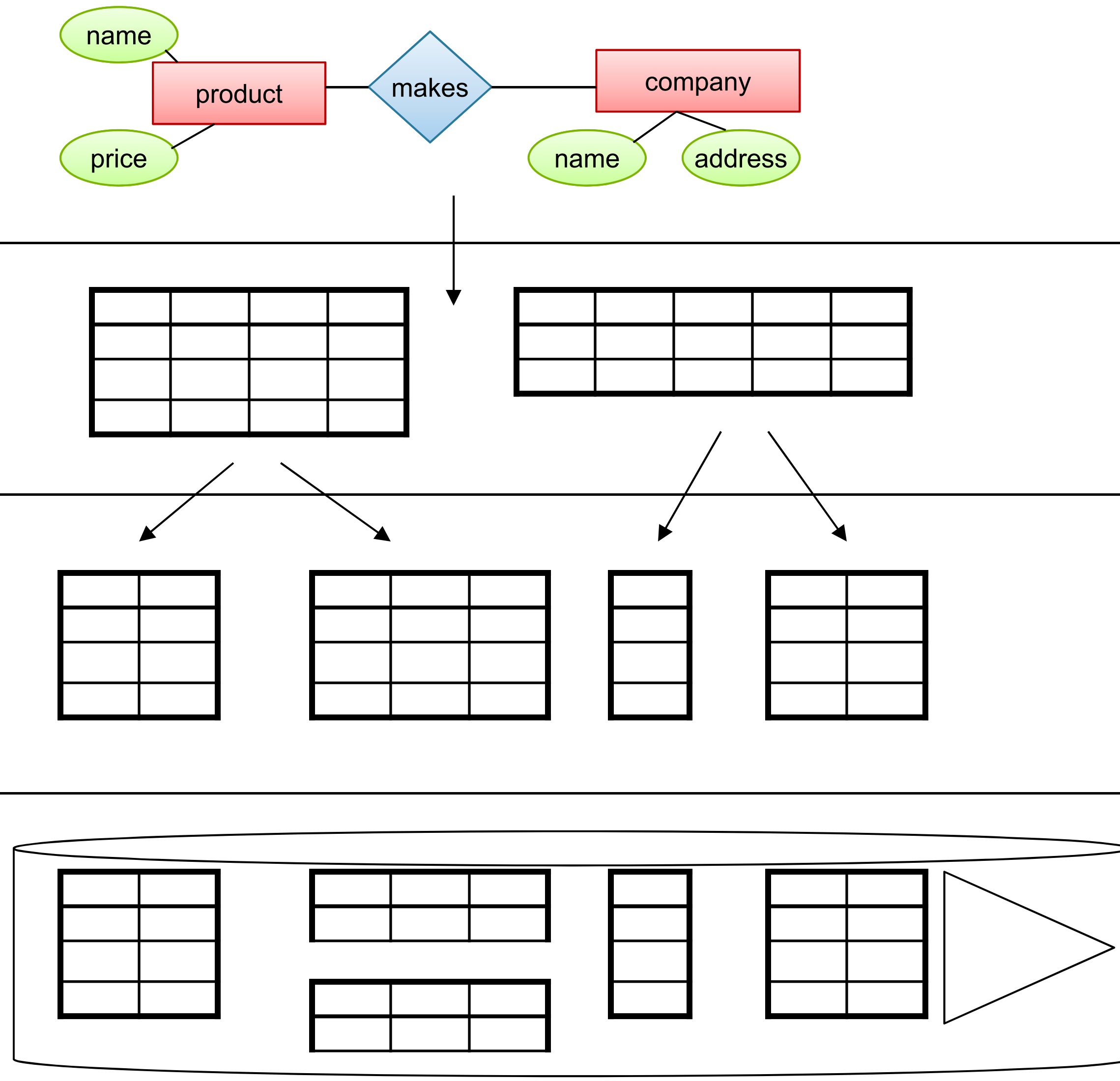
Why it's hard:

- The database will be in operation for years.
- Updating the schema in production is very hard:
  - schema change modifications are expensive (why?)
  - making the change without introducing any bugs is hard
    - this part is, by far, the most important consideration in practice

Database Design

- Consider issues such as:
  - What entities to model
  - How entities are related
  - What constraints exist in the domain
- Several formalisms exists
  - We discuss E/R diagrams
- Reading: Sec. 4.1-4.6

Database Design Process



Conceptual Model:

Relational Model:
Tables + constraints
And also functional dep.

Normalization:
Eliminates anomalies

Conceptual Schema

Physical storage details

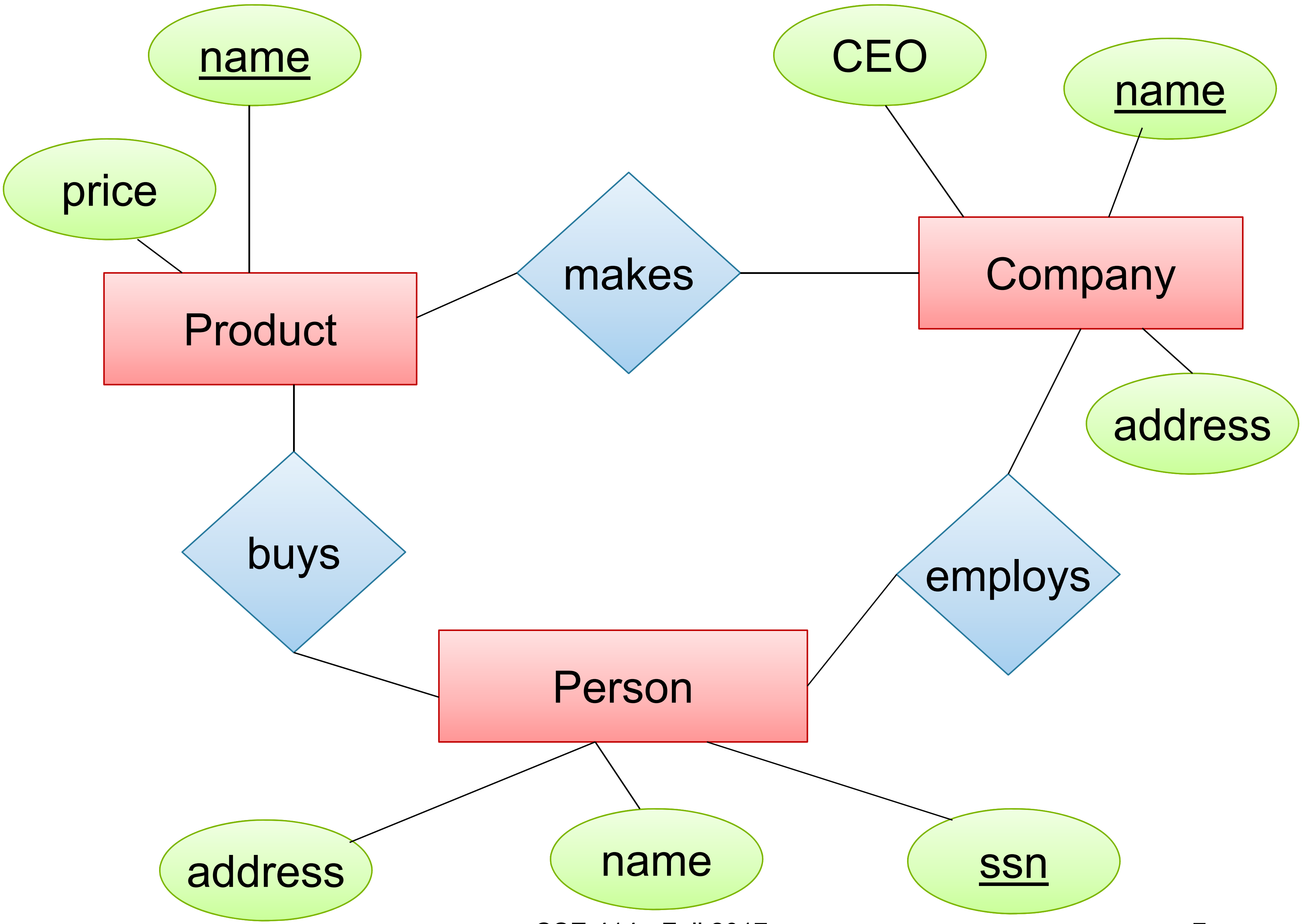
Physical Schema

Entity / Relationship Diagrams

- Entity set = a class
  - An entity = an object

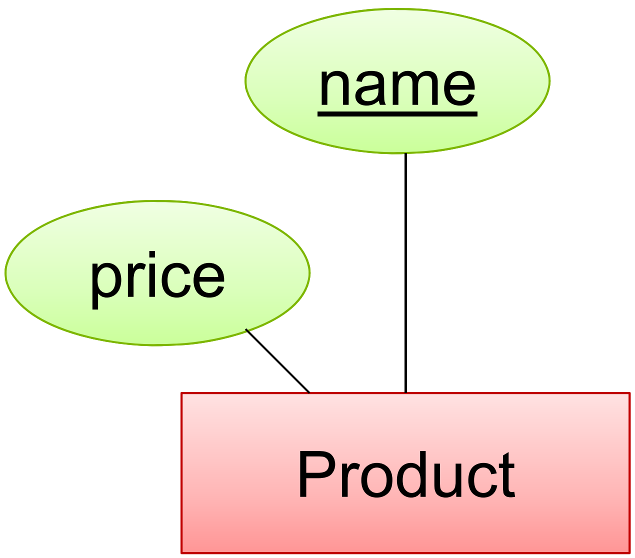
- Attribute

- Relationship


name
price
CEO
name
makes
Company
Product
address
buys
employs
Person
address
name
ssn

Keys in E/R Diagrams

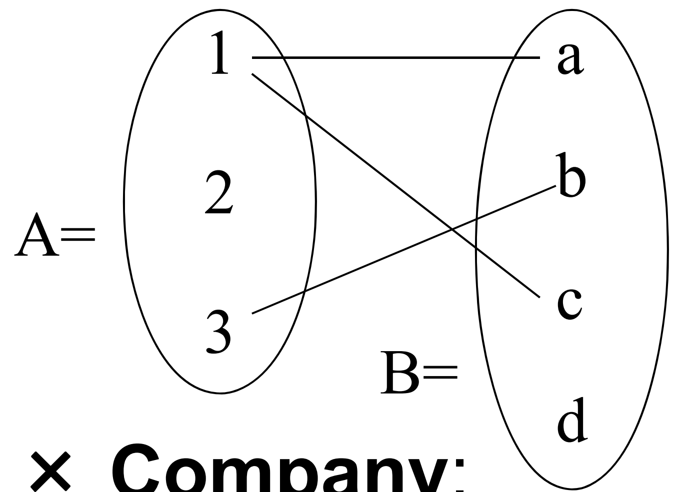
- Every entity set must have a key



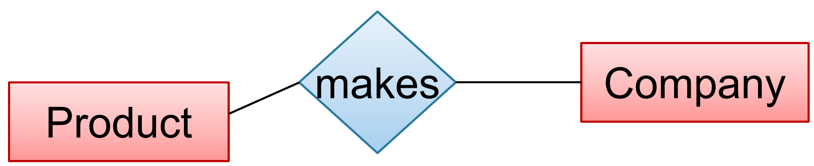
What is a Relation(ship)?

- A mathematical definition:
  – if A, B are sets, then a relation 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)}

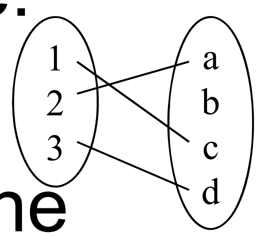


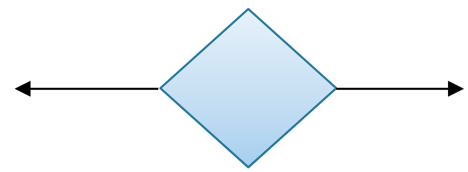
- **makes** is a subset of **Product × Company**:



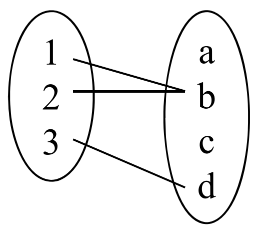
Multiplicity of E/R Relations

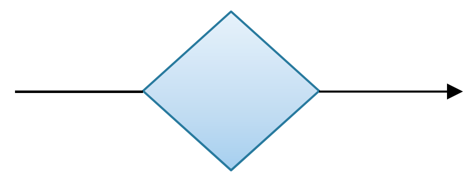
- one-one:



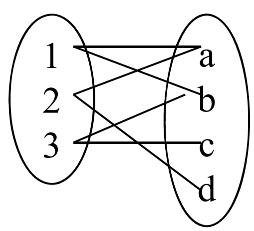


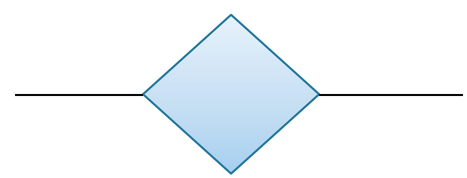
- many-one

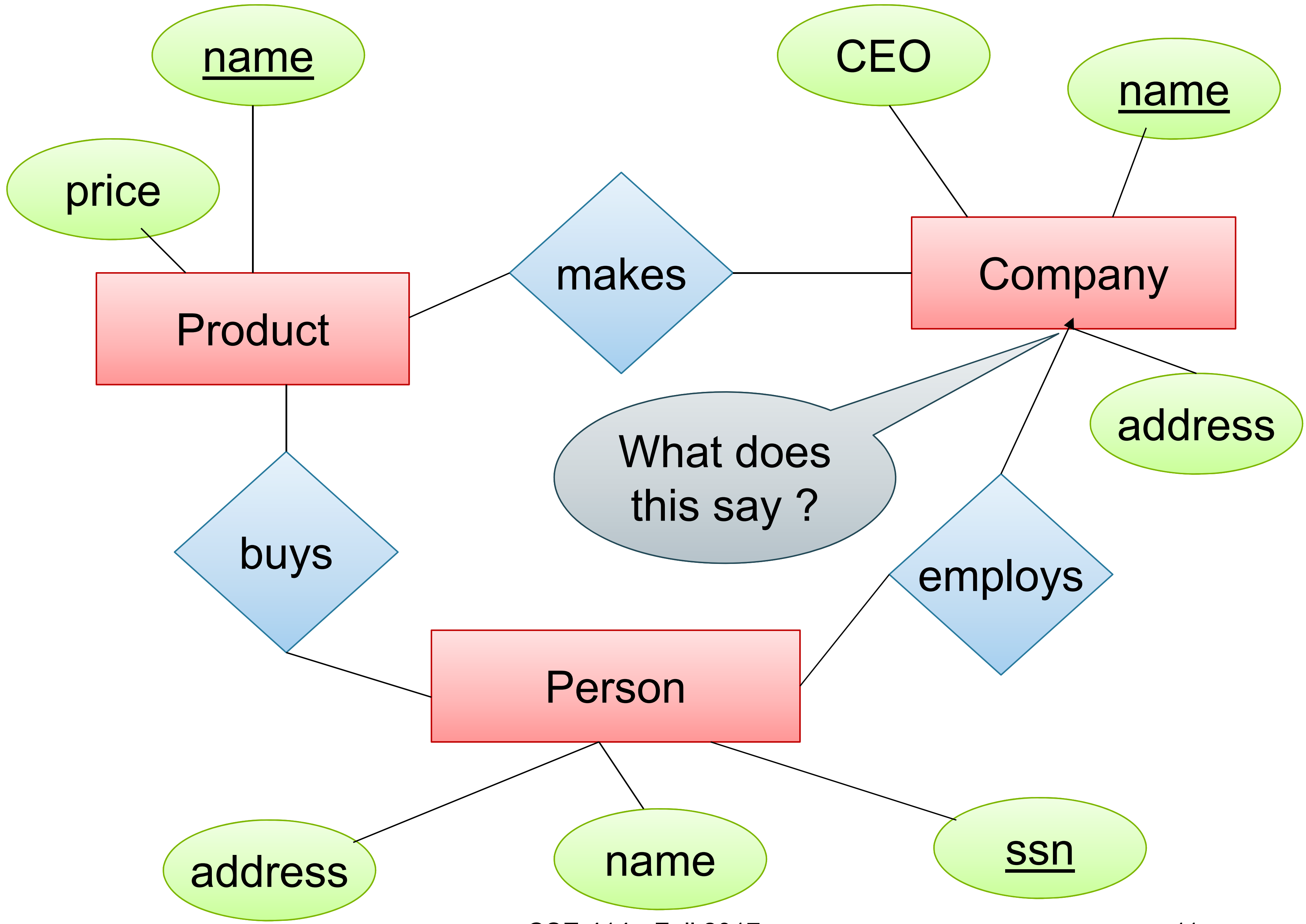




- many-many

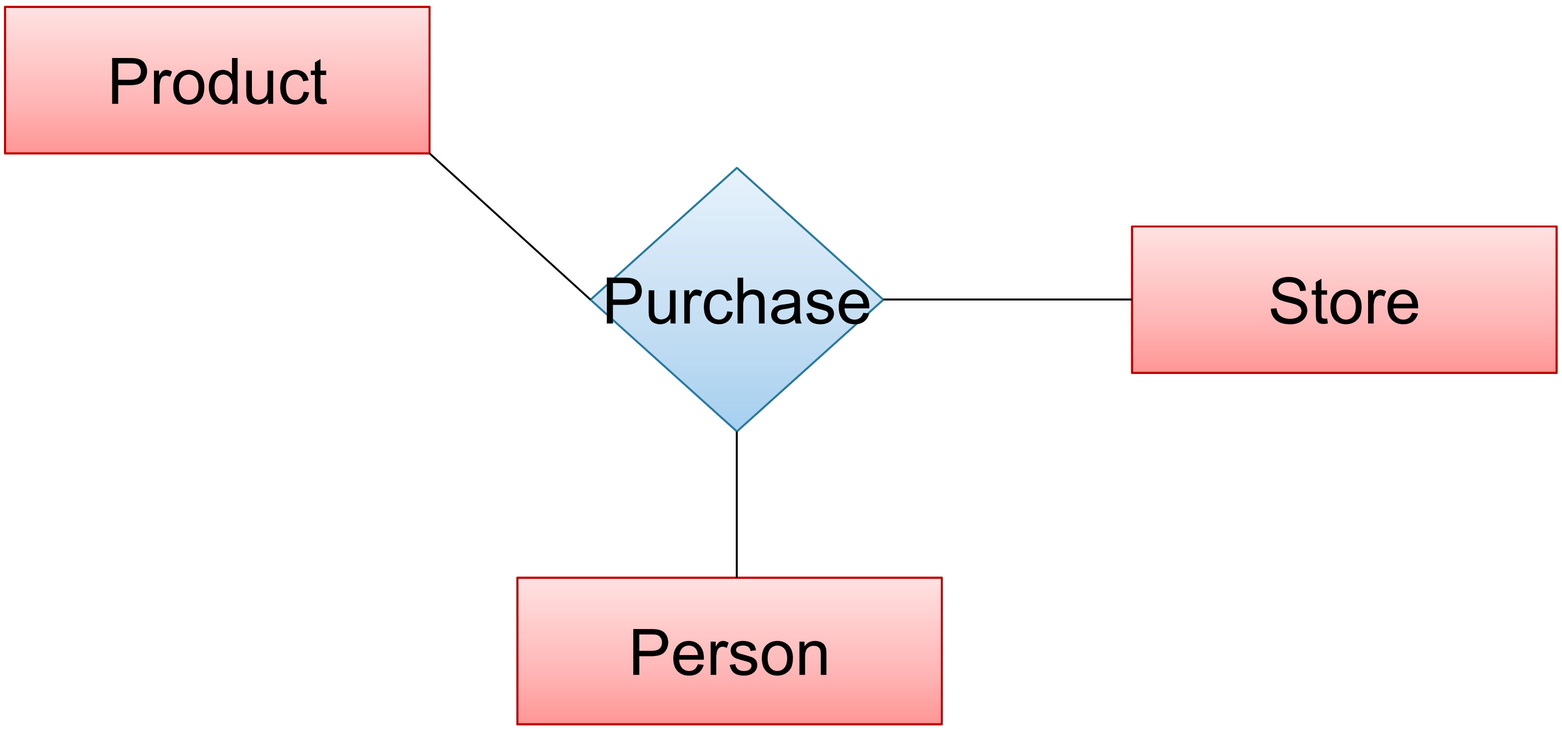





name
price
Product
makes
CEO
name
Company
address
What does this say ?
buys
employs
Person
address
name
ssn

Multi-way Relationships

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

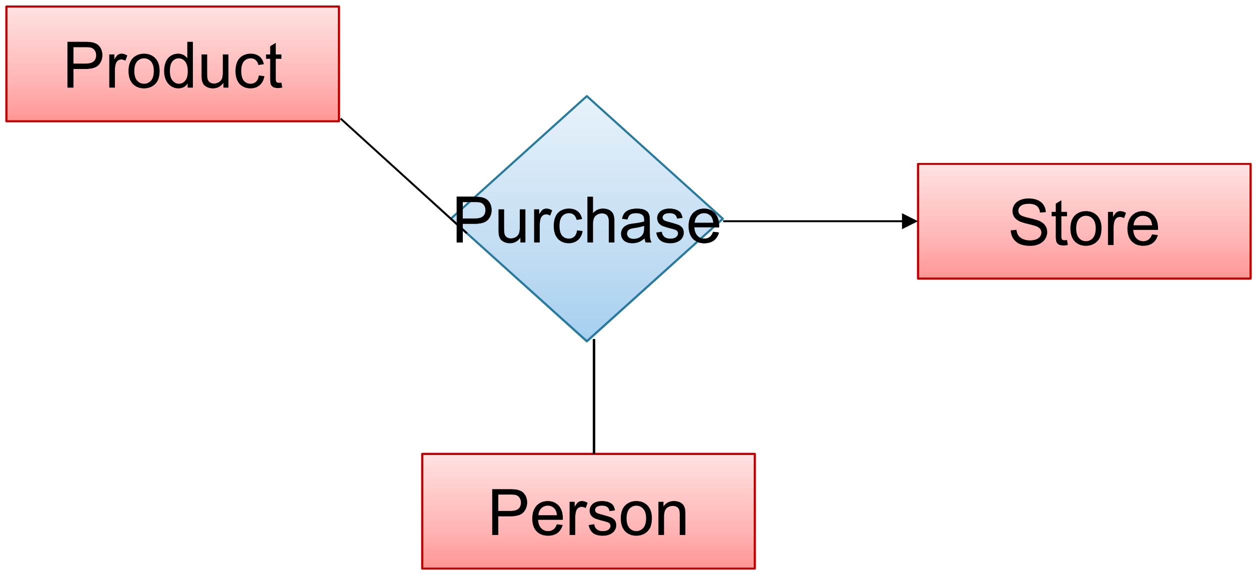


Can still model as a mathematical set (Q. how ?)

A. As a set of triples $\subseteq$ Person × Product × Store

Arrows in Multiway Relationships

Q: What does the arrow mean ?

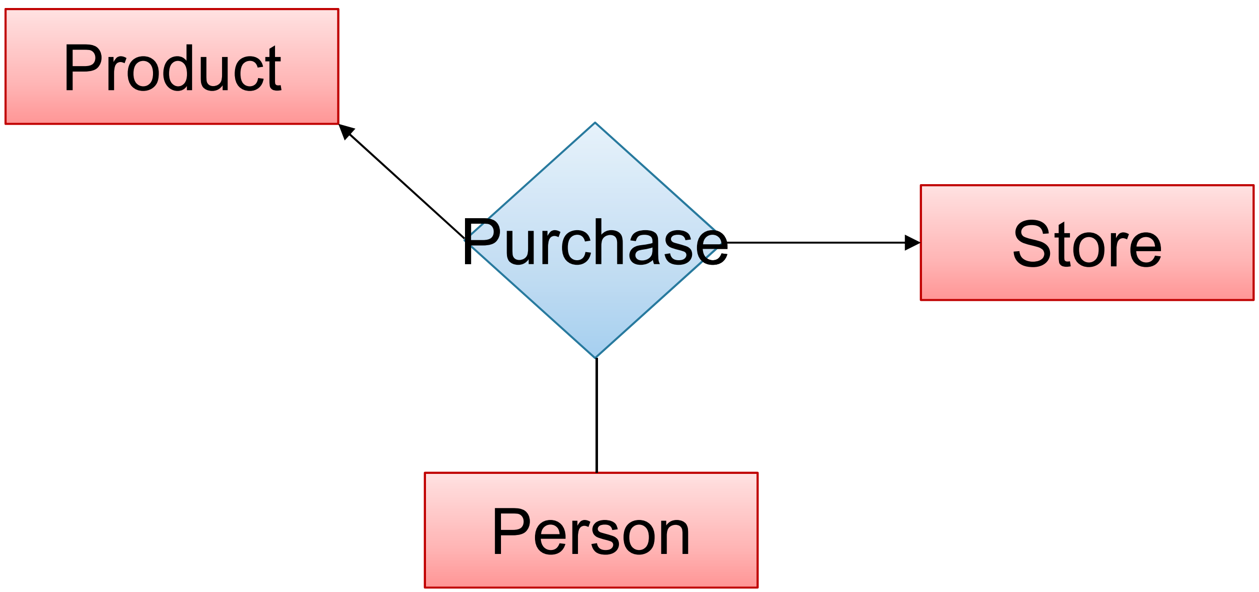


A: A given person buys a given product from at most one store

[Arrow pointing to E means that if we select one entity from each of the other entity sets in the relationship, those entities are related to at most one entity in E]

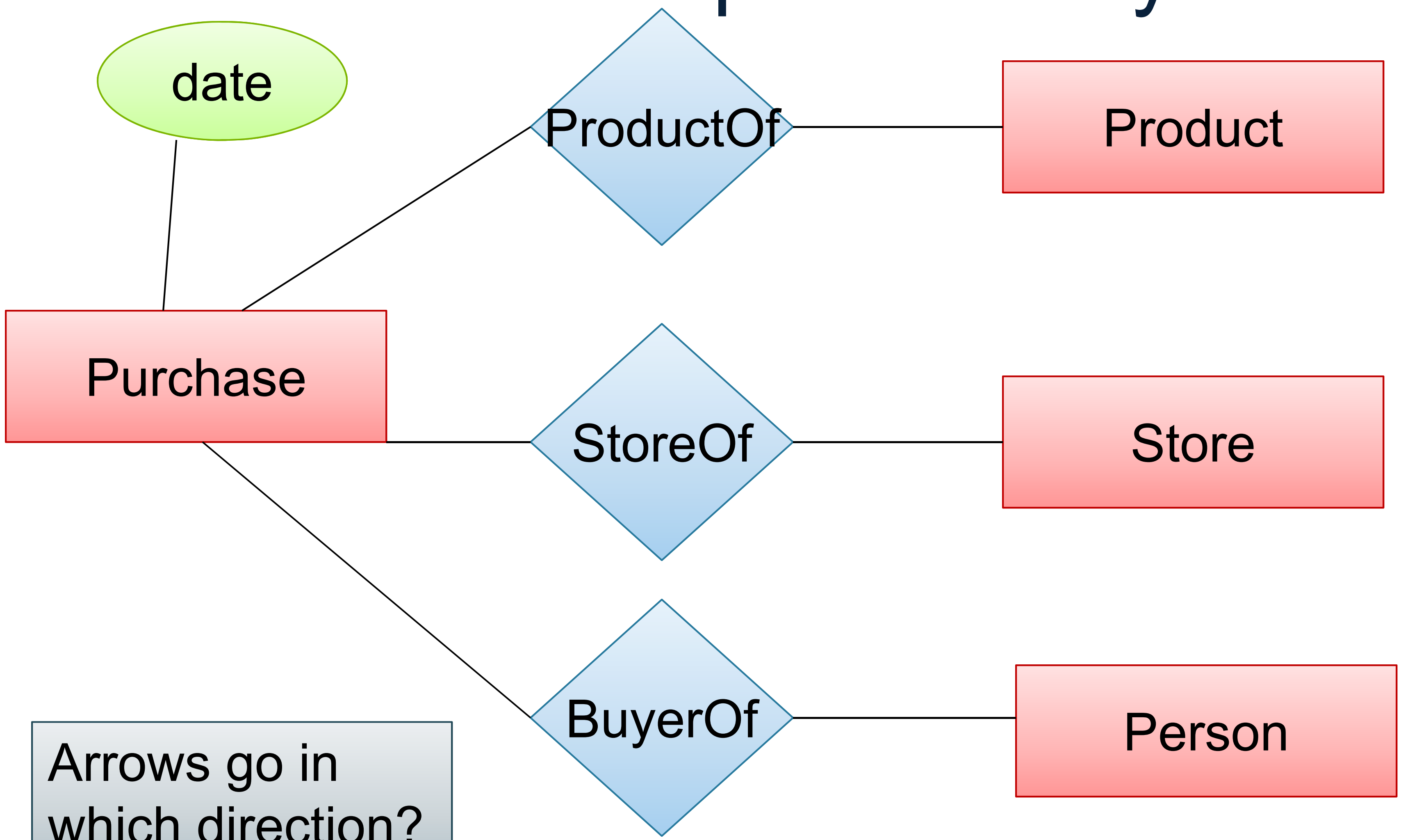
Arrows in Multiway Relationships

Q: What does the arrow mean ?



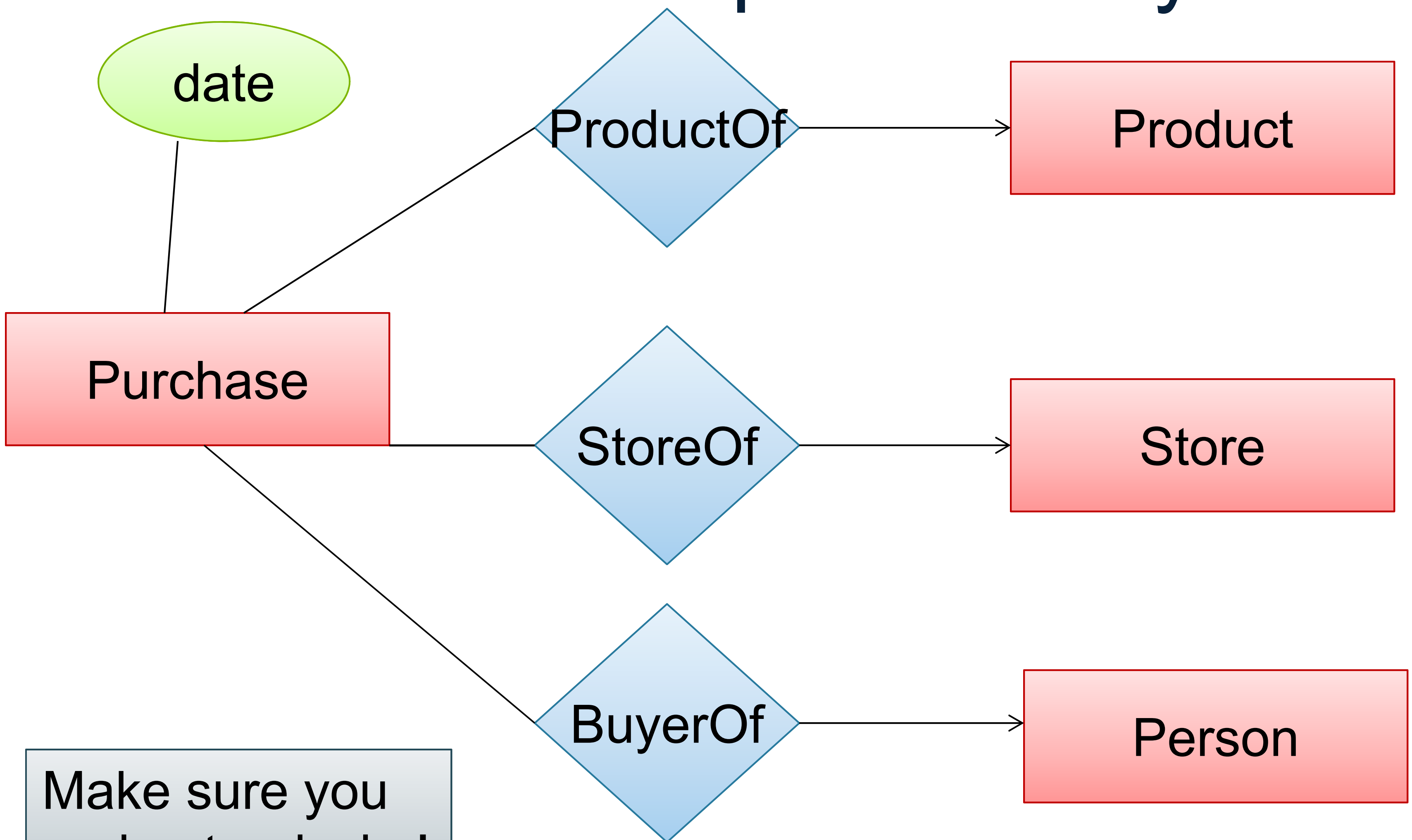
A: A given person buys a given product from at most one store AND every store sells to every person at most one product

Converting Multi-way Relationships to Binary



Arrows go in which direction?

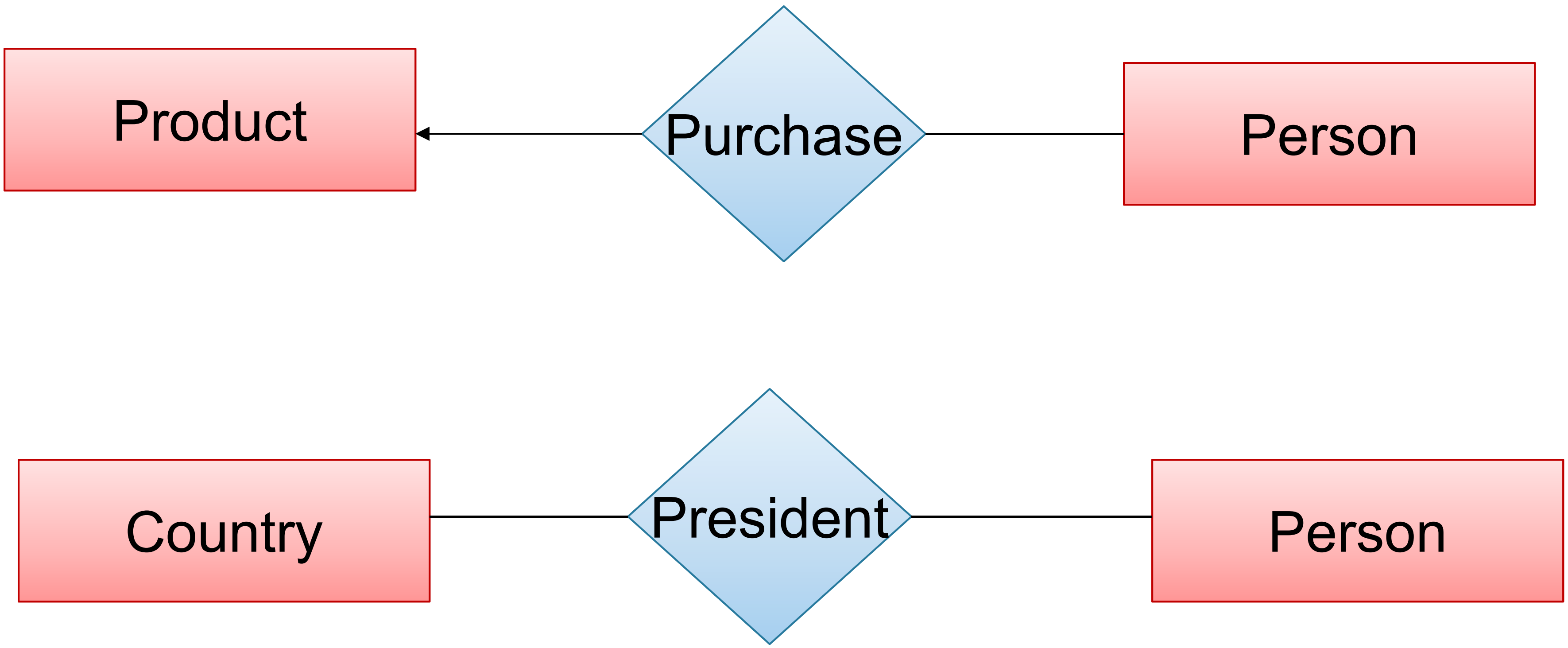
Converting Multi-way Relationships to Binary



Make sure you understand why!

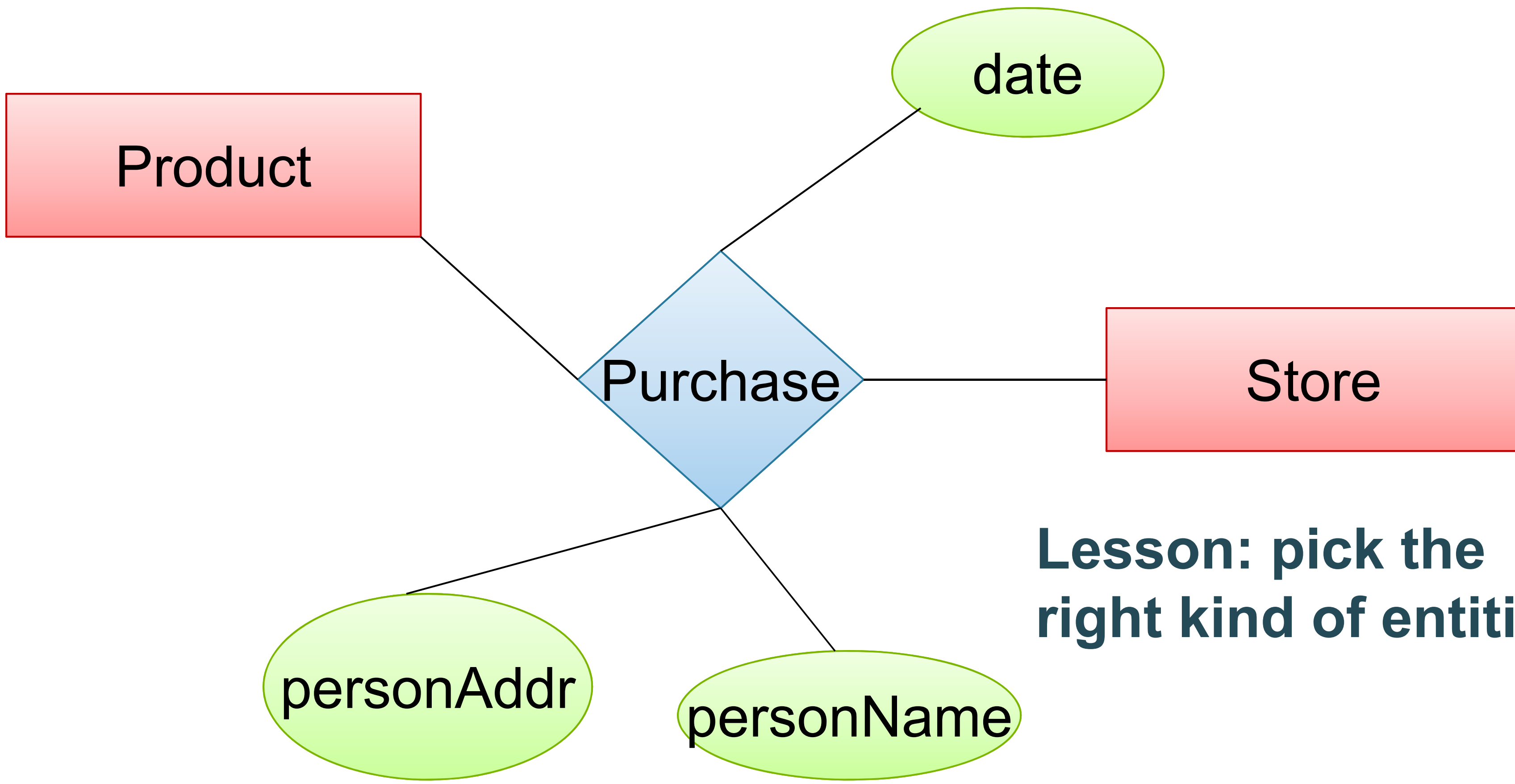
3. Design Principles

What's wrong?



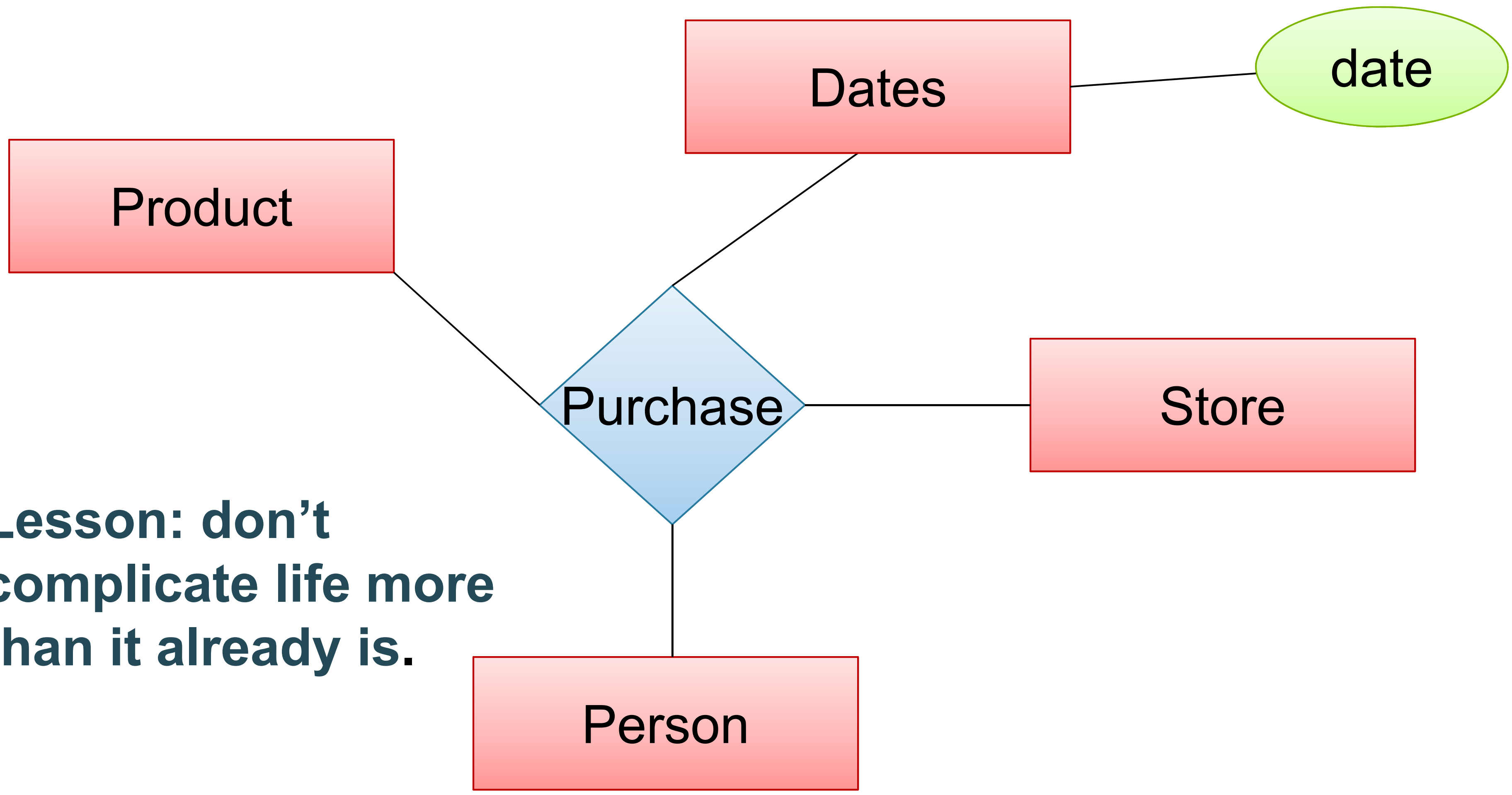
Lesson: be faithful to the specifications of the app!

Design Principles: What's Wrong?



Lesson: pick the right kind of entities.

Design Principles: What's Wrong?

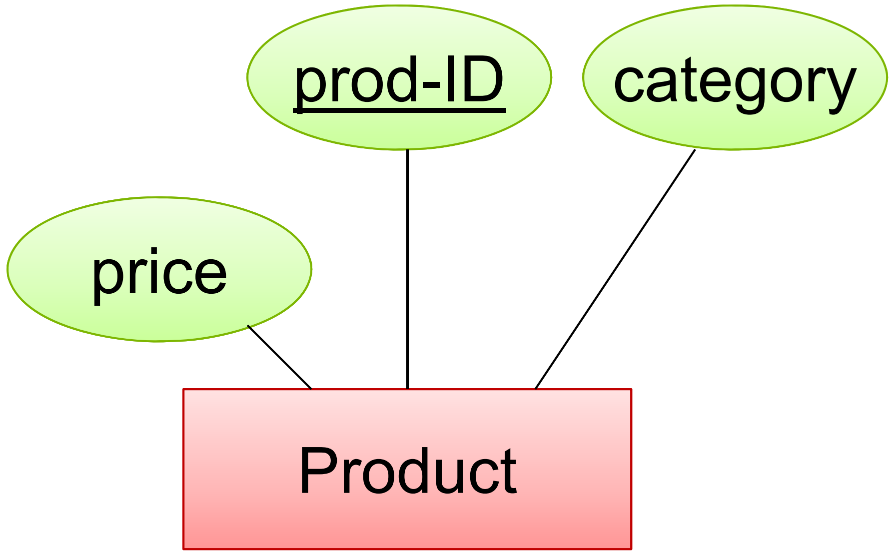


Lesson: don't complicate life more than it already is.

From E/R Diagrams to Relational Schema

- Entity set → relation
- Relationship → relation

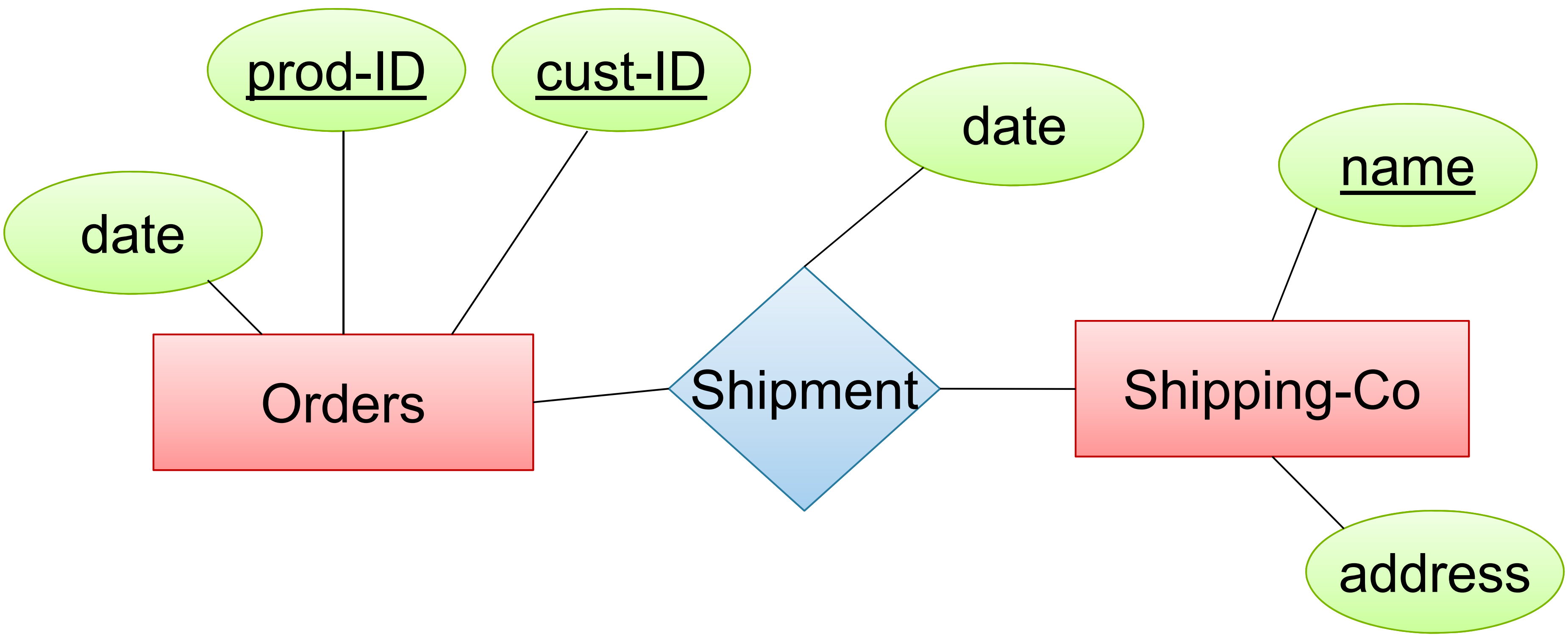
Entity Set to Relation



Product(<u>prod-ID</u>, category, price)

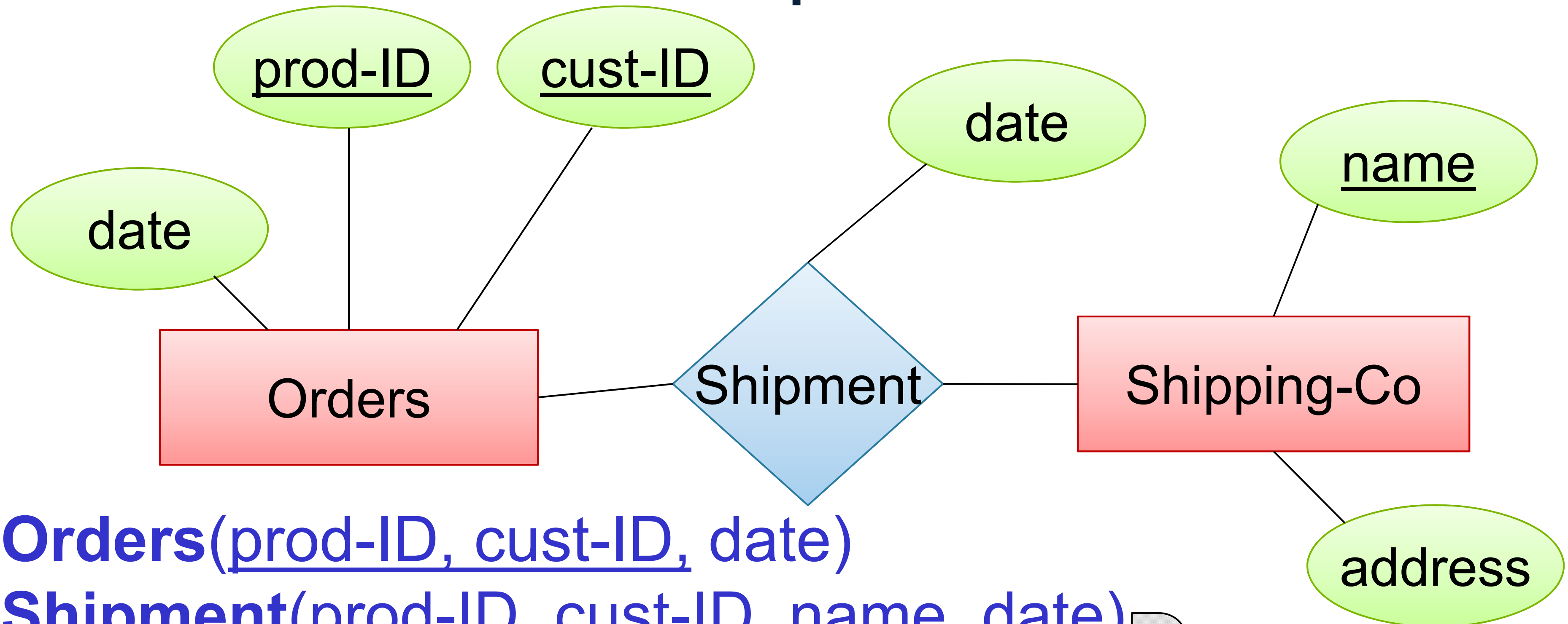
<u>prod-ID</u>	category	price
Gizmo55	Camera	99.99
Pokemn19	Toy	29.99

N-N Relationships to Relations



Represent this in relations

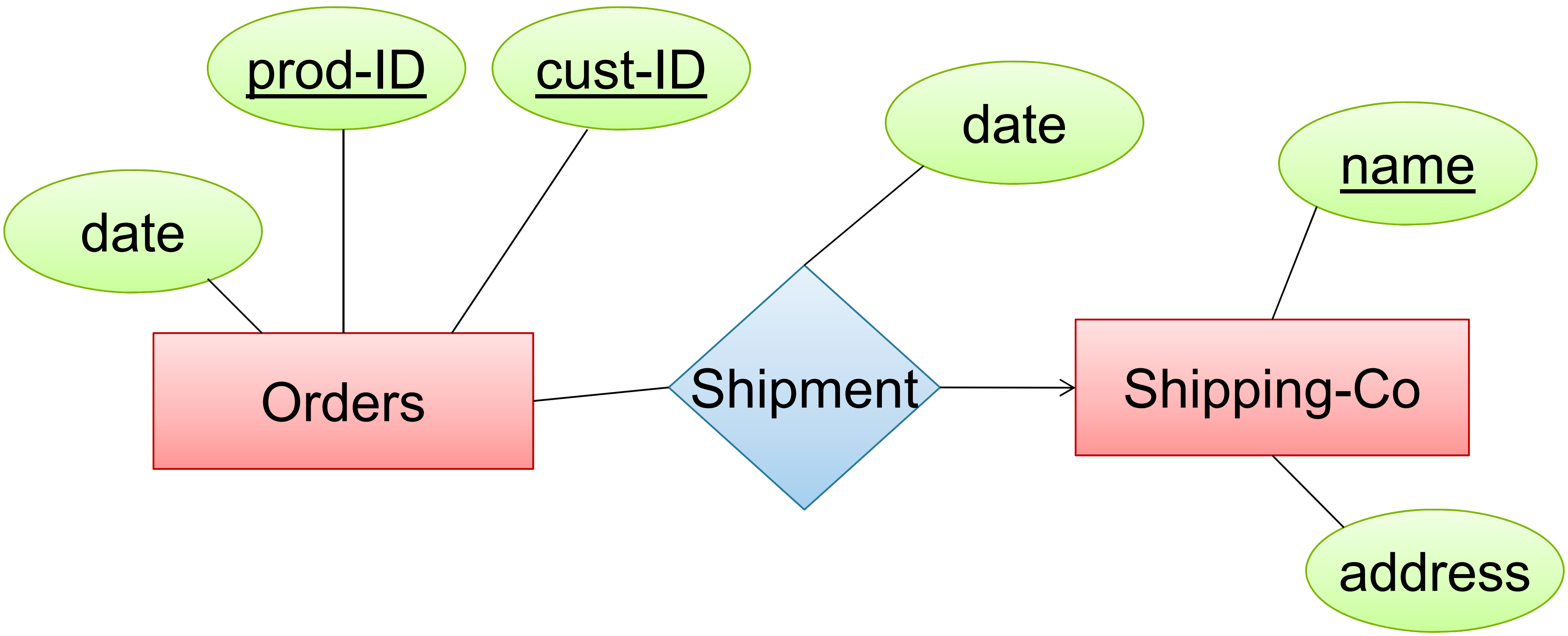
N-N Relationships to Relations



Orders(prod-ID, cust-ID, date)
Shipment(prod-ID, cust-ID, name, date)
Shipping-Co(name, address)

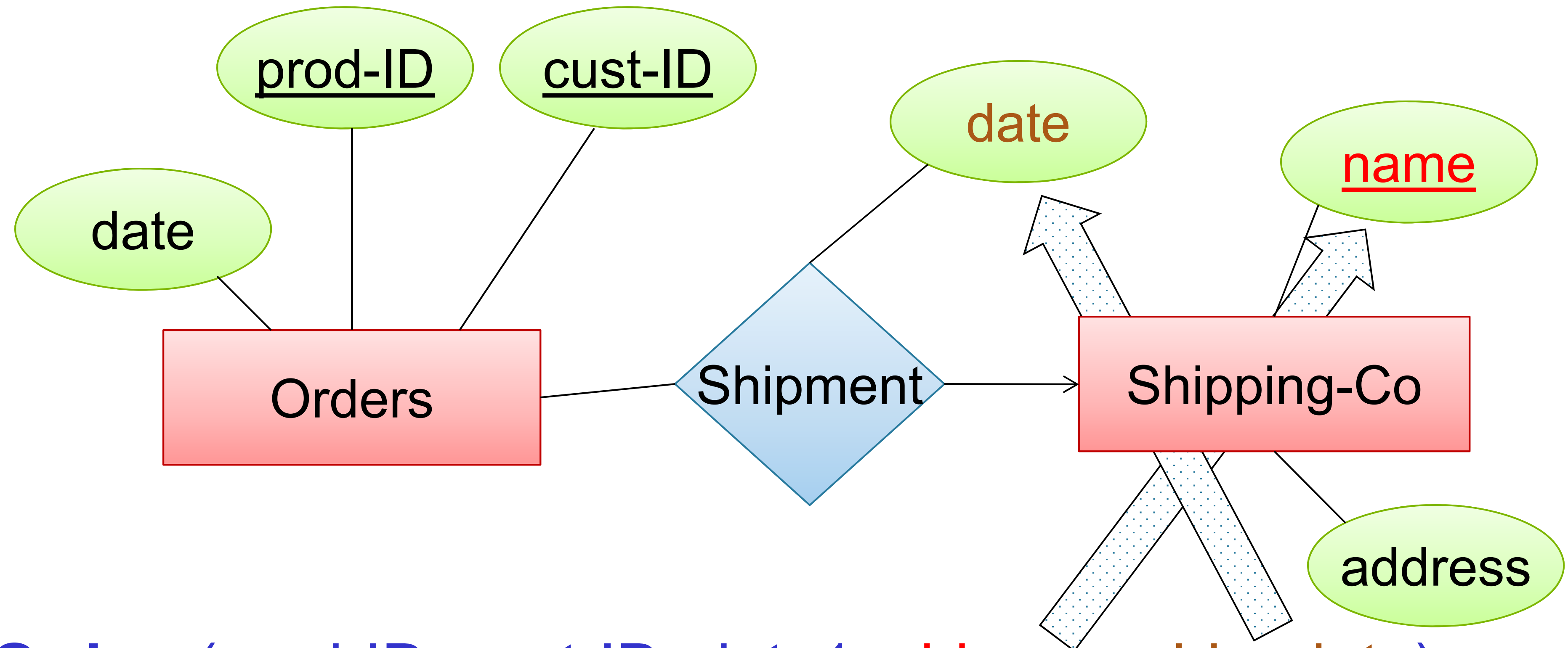
prod-ID	cust-ID	name	date
Gizmo55	Joe12	UPS	4/10/2011
Gizmo55	Joe12	FEDEX	4/9/2011

N-1 Relationships to Relations



Represent this in relations

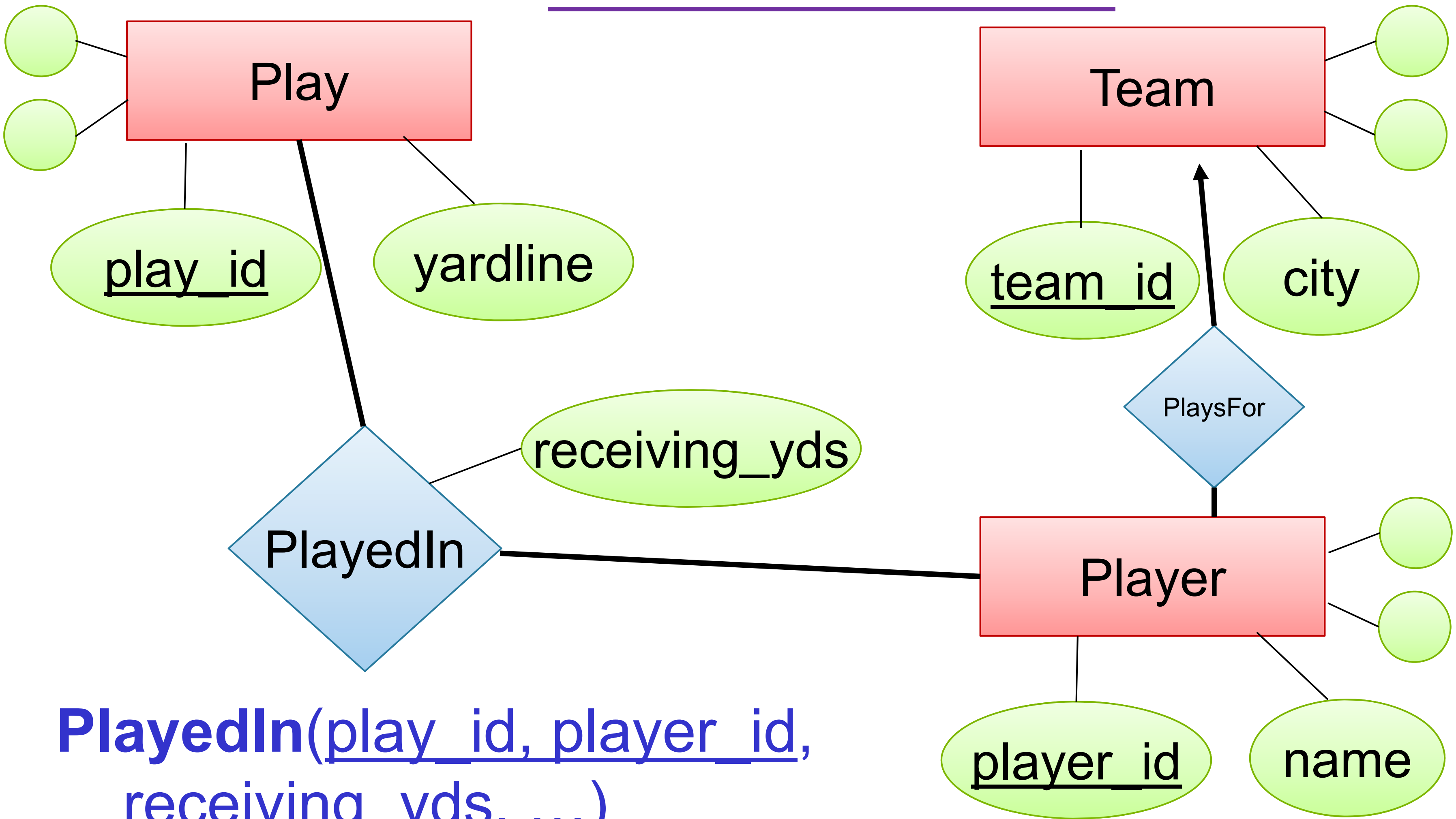
N-1 Relationships to Relations



Orders(prod-ID, cust-ID, date1, ship_co, ship_date)
Shipping-Co(name, address)

Remember: many-one relationship becomes FK, not relation

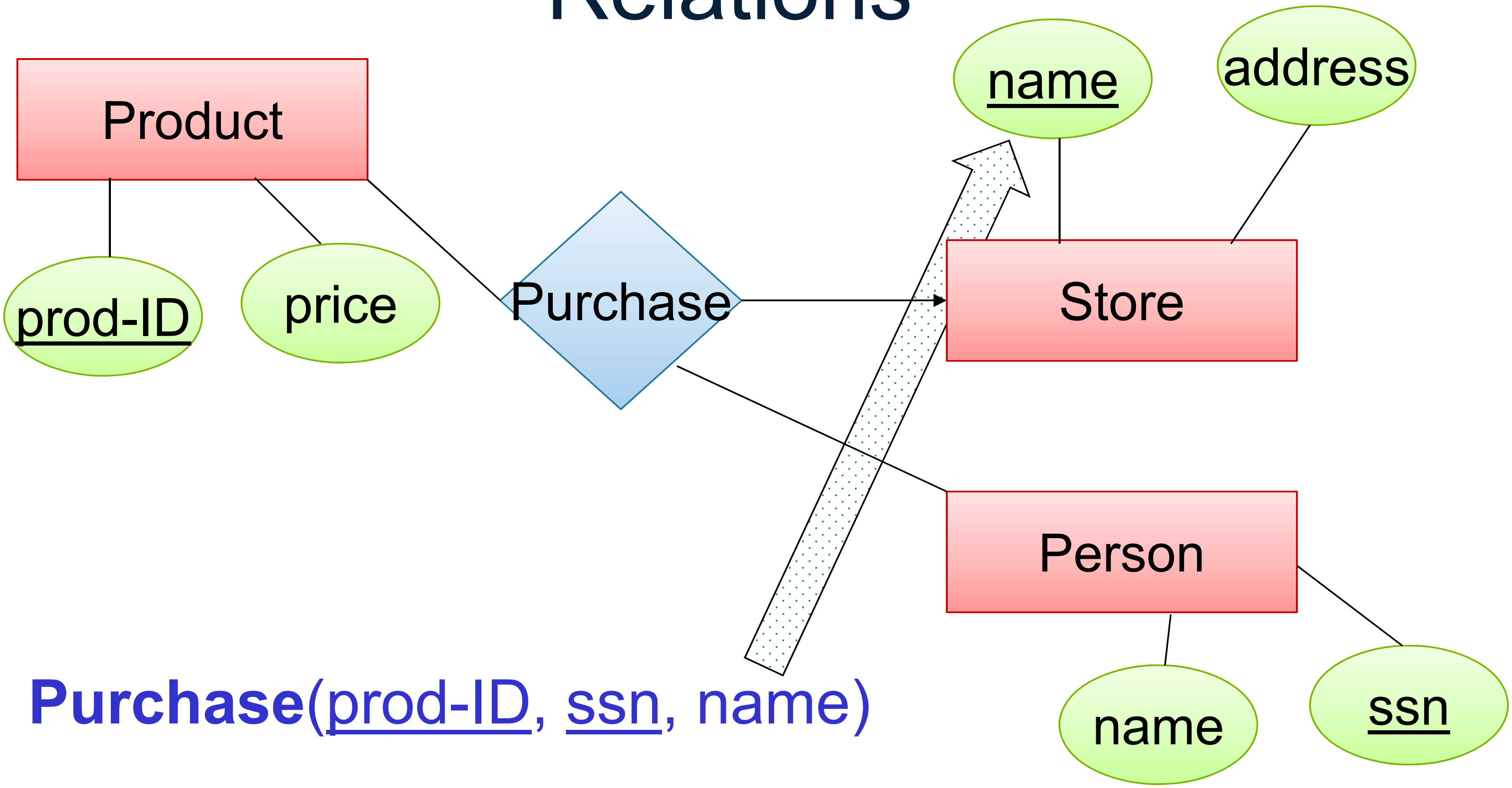
Ex: NFL Game DB



PlayedIn(<u>play_id, player_id</u>, receiving_yds, …)

(Actually, the key of Play is not play_id. More on this later...)

Multi-way Relationships to Relations



Purchase(<u>prod-ID</u>, <u>ssn</u>, name)

Modeling Subclasses

Some objects in a class may be special

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

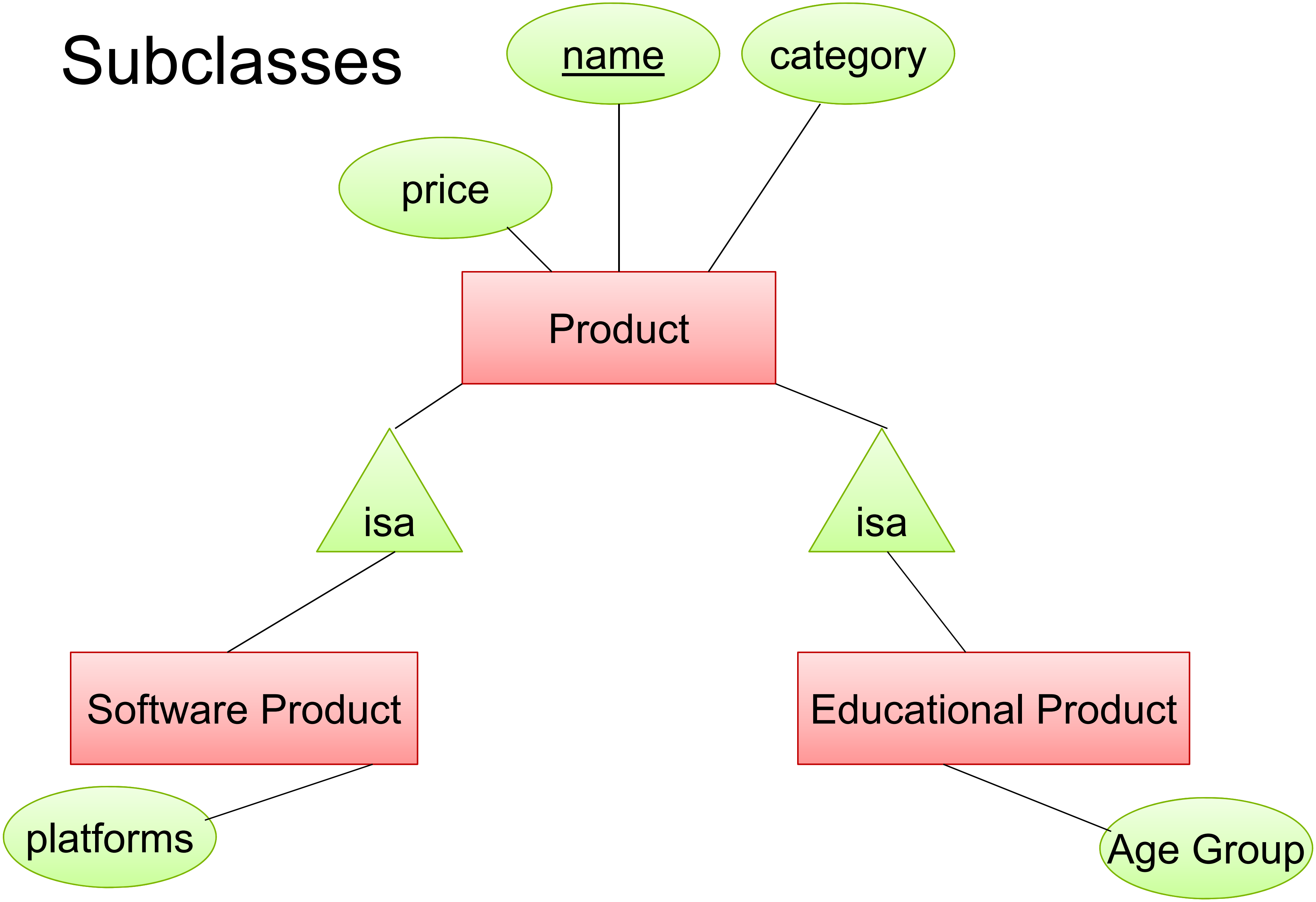
Products

Software products

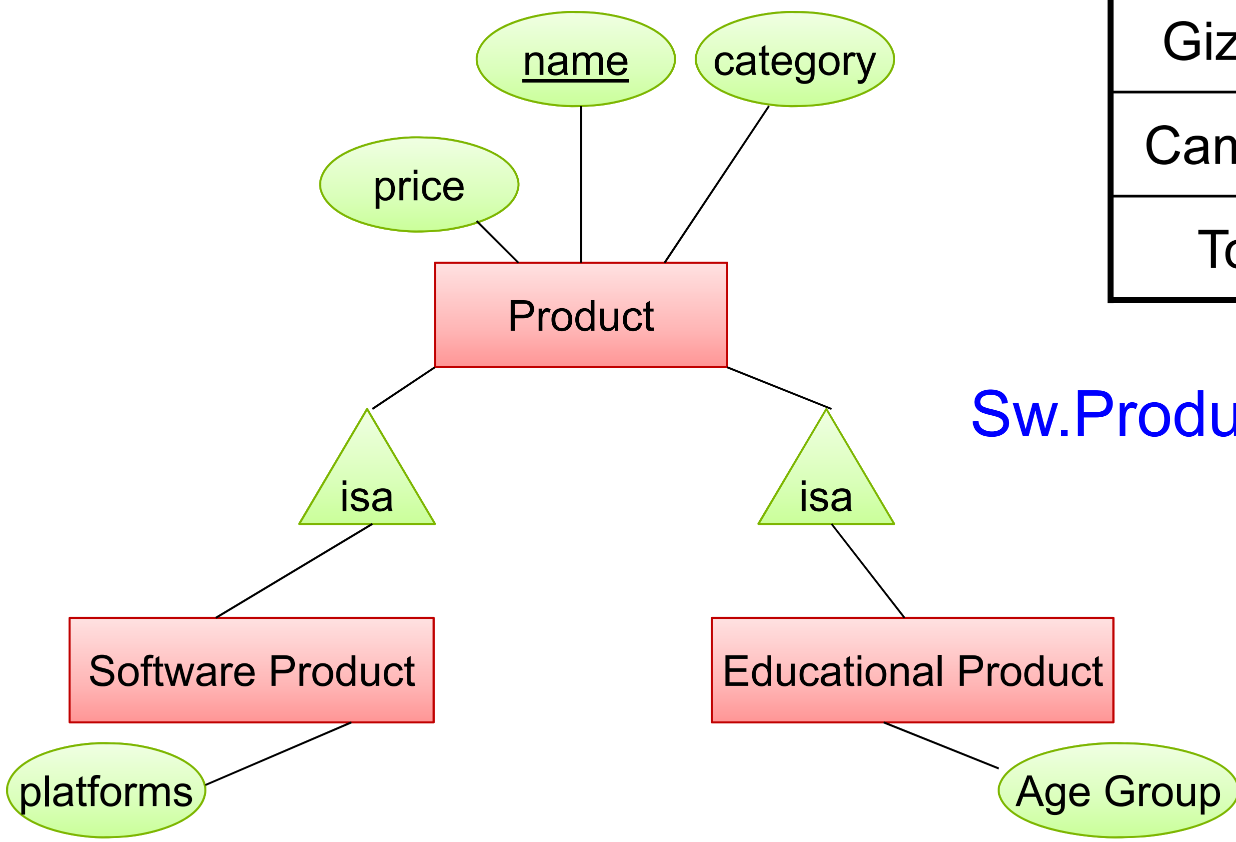
Educational products

So --- we define subclasses in E/R

Subclasses


name
category
price
Product
isa
isa
Software Product
Educational Product
platforms
Age Group

Subclasses to Relations (one option)



Product

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

Sw.Product

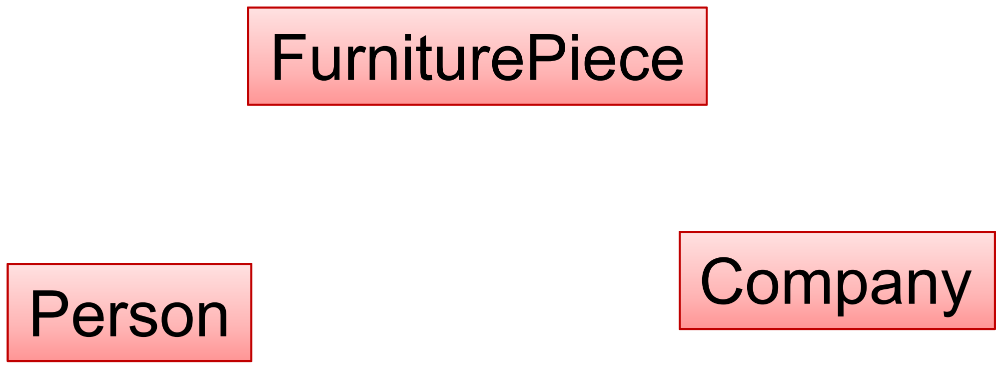
Name	platforms
Gizmo	unix

Ed.Product

Name	Age Group
Gizmo	toddler
Toy	retired

Other ways to convert are possible…

Modeling Union Types with Subclasses

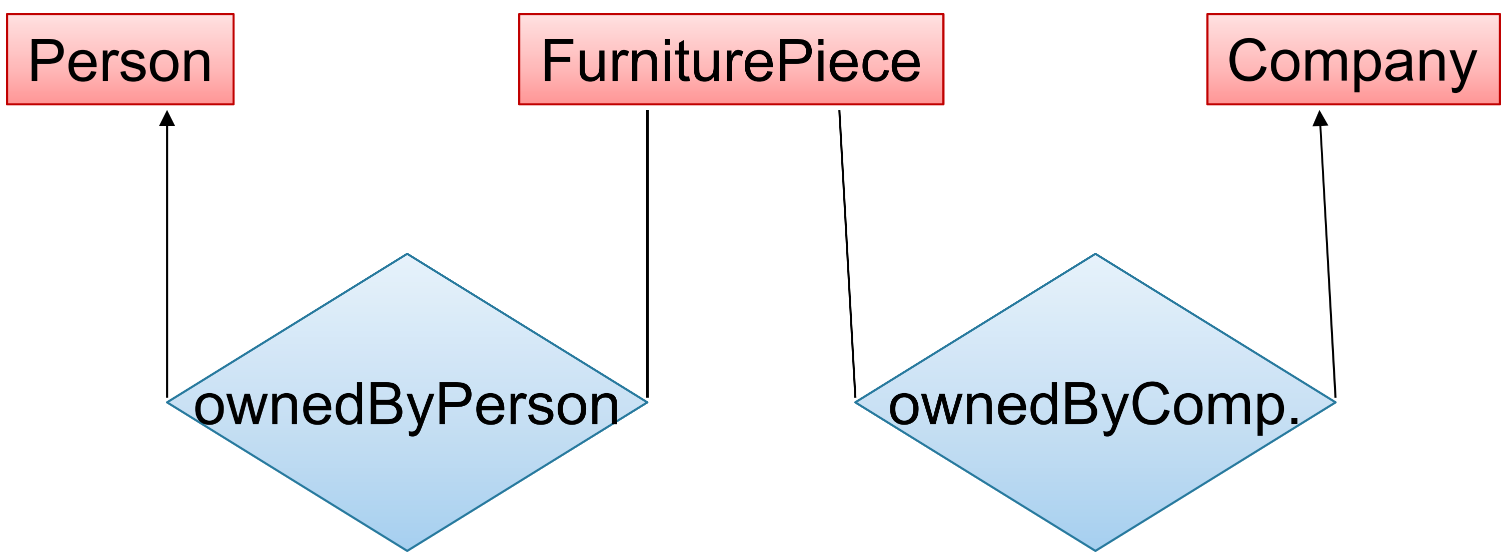


Say: each piece of furniture is owned either by a person or by a company

Modeling Union Types with Subclasses

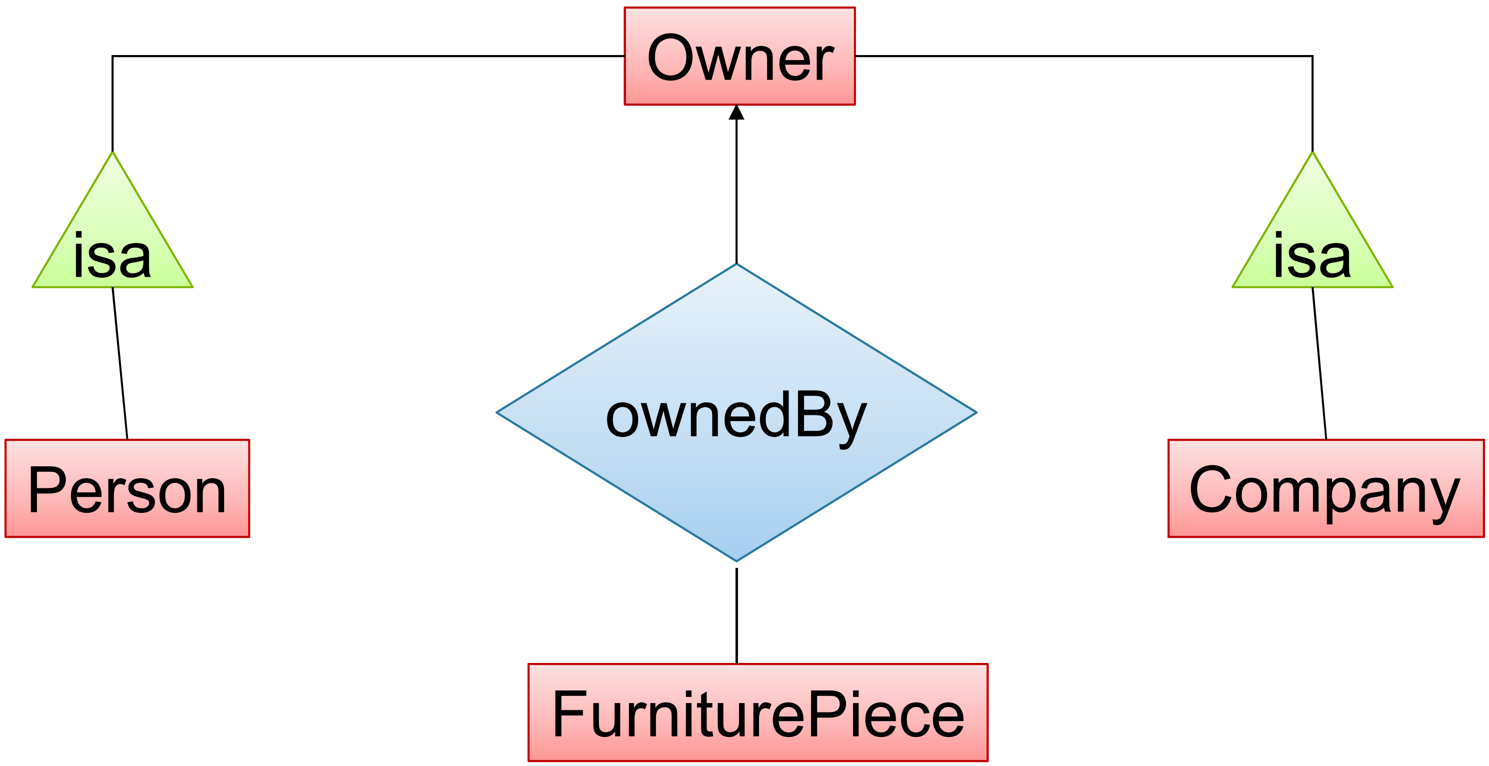
Say: each piece of furniture is owned either by a person or by a company

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



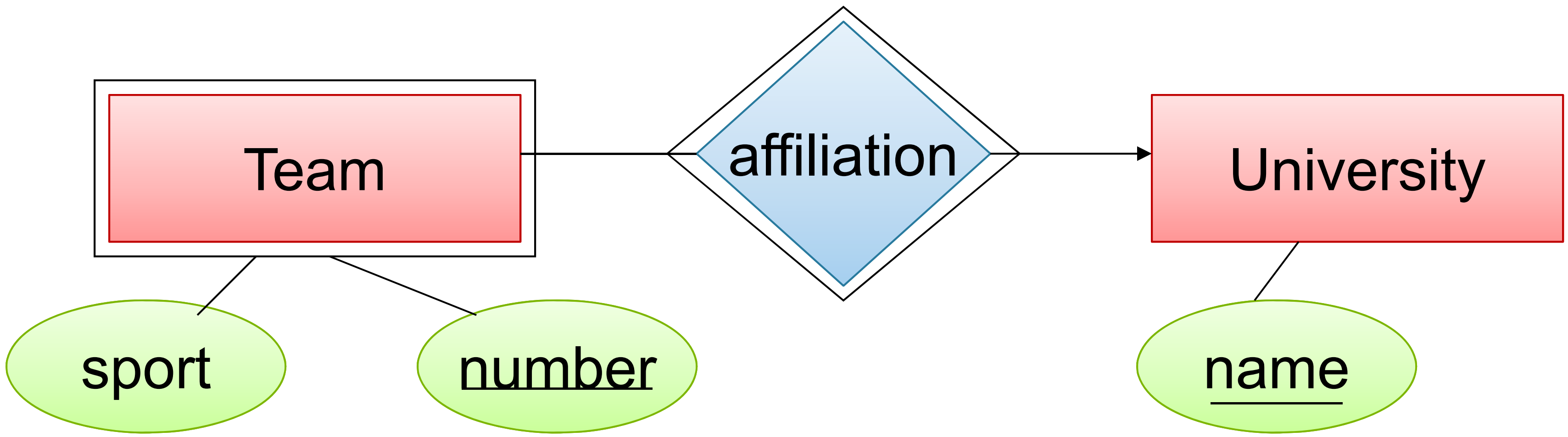
Modeling Union Types with Subclasses

Solution 2: better, more laborious



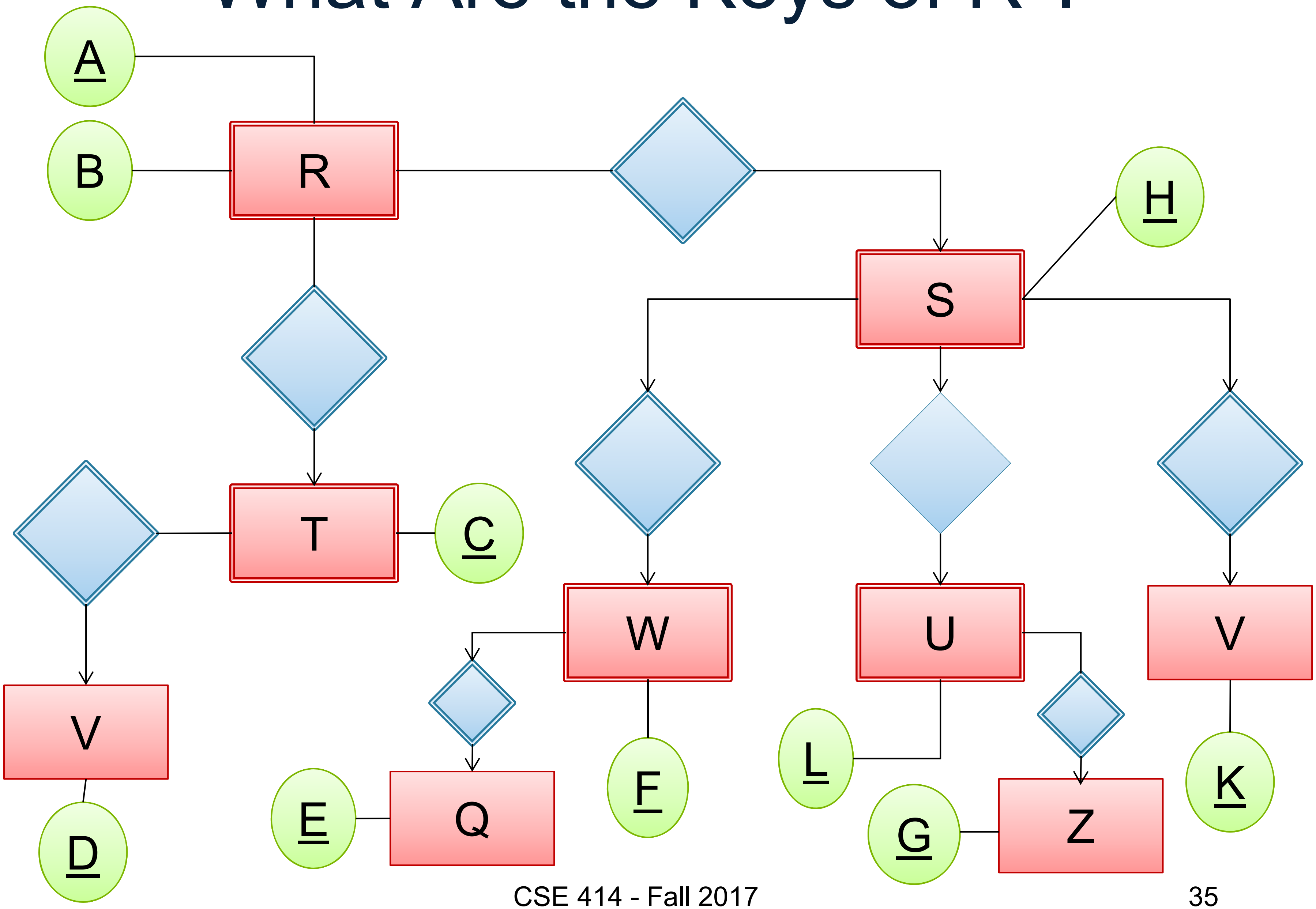
Weak Entity Sets

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

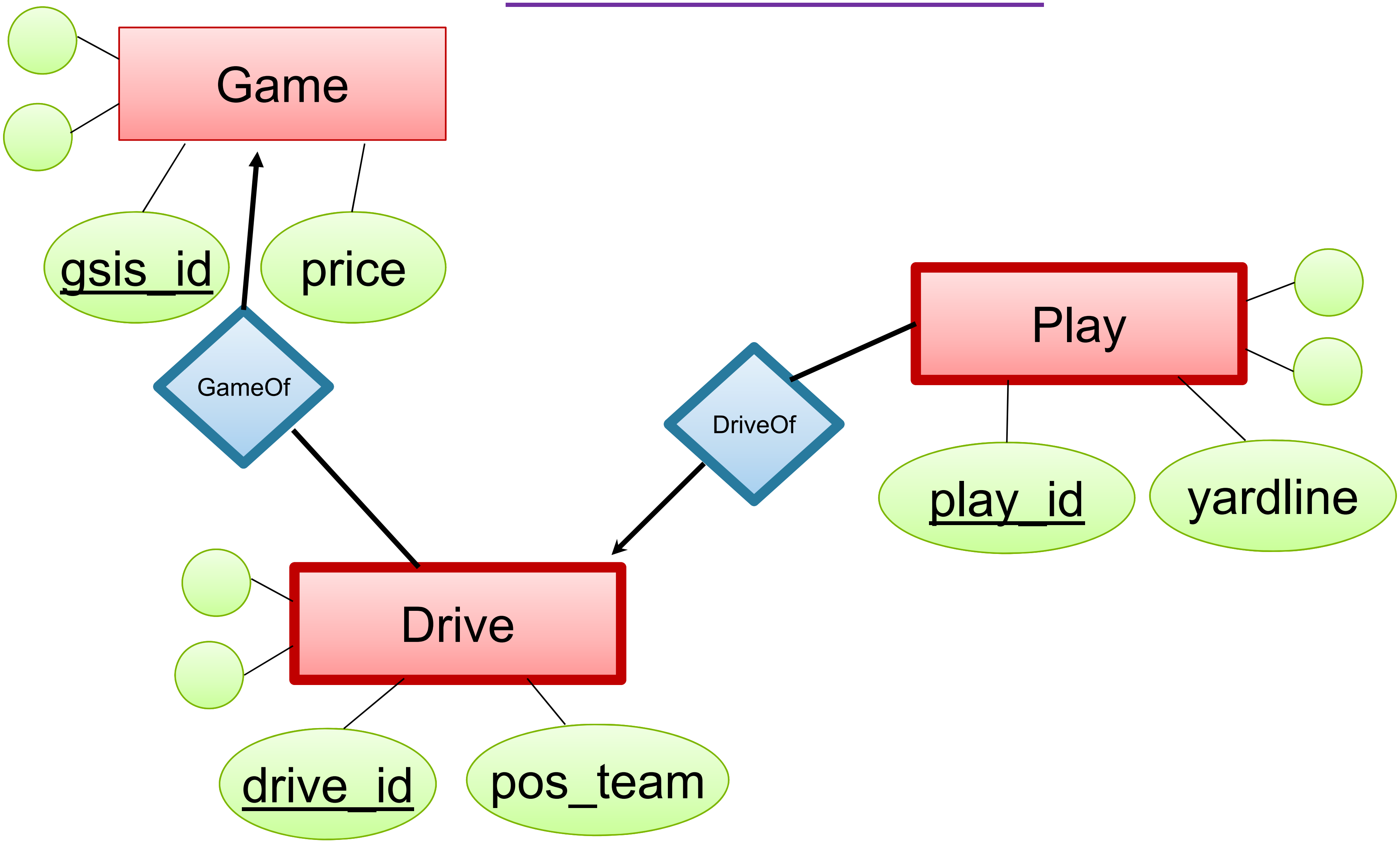


Team(sport, number, universityName)
University(name)

What Are the Keys of R ?



Ex: NFL Game DB



Integrity Constraints Motivation

An integrity constraint is a condition specified on a database schema that restricts the data that can be stored in an instance of the database.

Most important issue in practice

- ICs help prevent entry of incorrect information
- How? DBMS enforces integrity constraints
  - Allows only legal database instances (i.e., those that satisfy all constraints) to exist
  - Ensures that all necessary checks are always performed and avoids duplicating the verification logic in each application

Constraints in E/R Diagrams

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

Keys: social security number uniquely identifies a person.

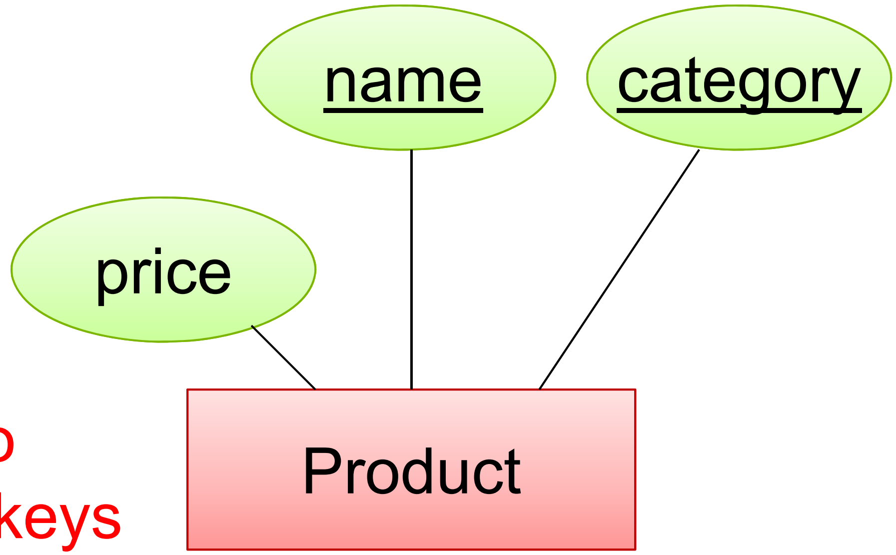
Single-value constraints: can have only one genetic father

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

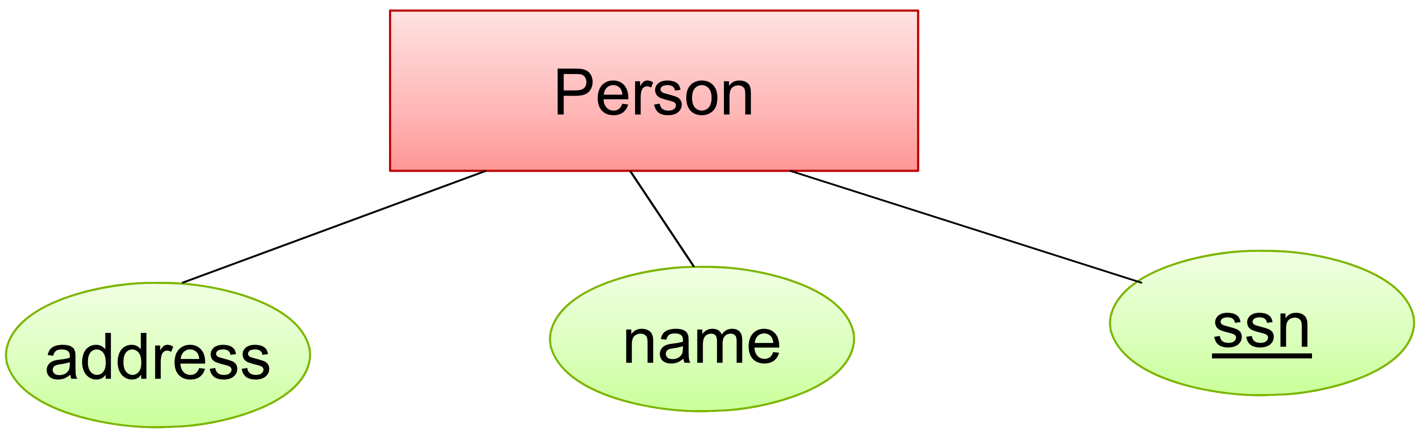
Other constraints: peoples' ages are between 0 and 150.
some values should not be NULL

Keys in E/R Diagrams

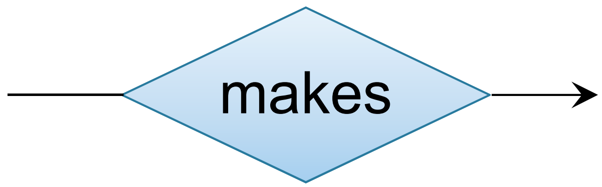
Underline:



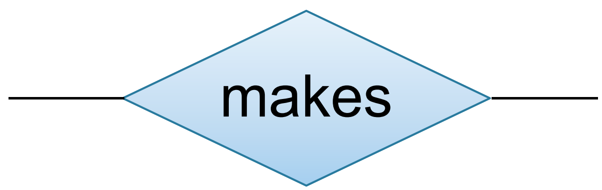
No formal way to specify multiple keys in E/R diagrams



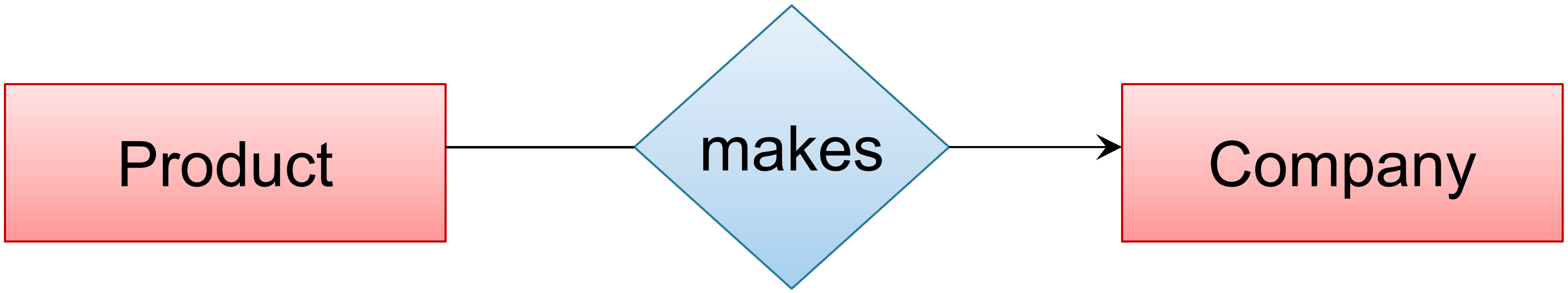
Single Value Constraints



vs.

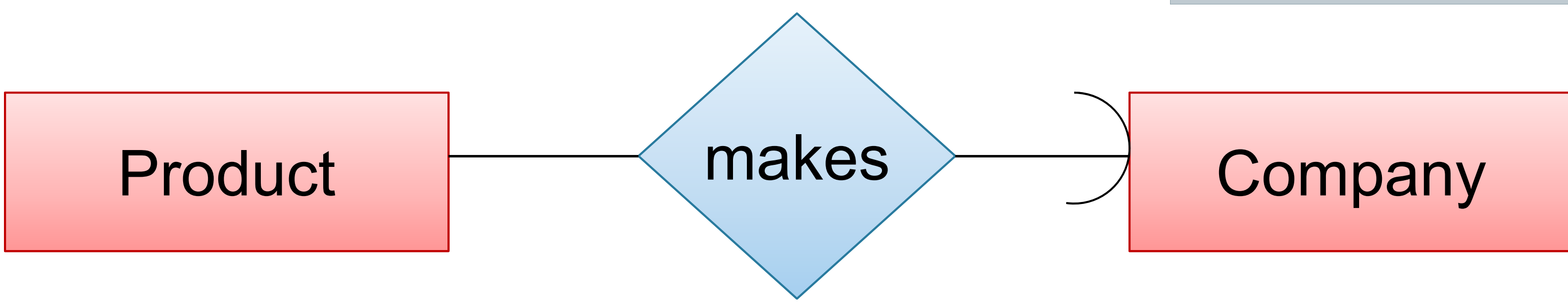


Referential Integrity Constraints



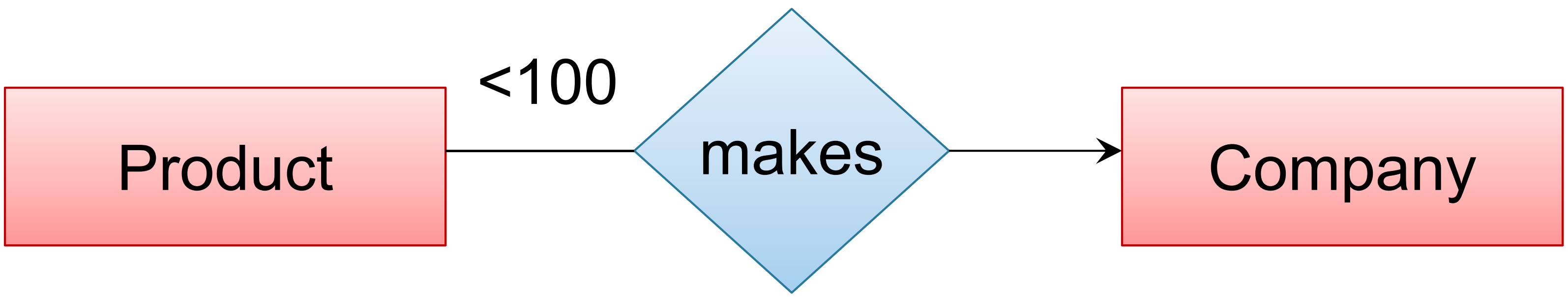
Each product made by at most one company.
Some products made by no company

Which one is FK?



Each product made by *exactly* one company.

Other Constraints



Q: What does this mean ?
A: A Company entity cannot be connected
by relationship to more than 99 Product entities

Constraints in SQL

Constraints in SQL:

- Keys, foreign keys
- Attribute-level constraints
- Tuple-level constraints
- Global constraints: assertions

simplest

Most complex

- The more complex the constraint, the harder it is to check and to enforce…
  - (Still, performance is secondary to correctness.)

Key Constraints

Product(name, category)

```
CREATE TABLE Product (
        name CHAR(30) PRIMARY KEY,
        category VARCHAR(20))
```

OR:

```
CREATE TABLE Product (
        name CHAR(30),
        category VARCHAR(20),
        PRIMARY KEY (name))
```

Keys with Multiple Attributes

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

```
CREATE TABLE Product (
        name CHAR(30),
        category VARCHAR(20),
        price INT,
        PRIMARY KEY (name, category))
```

Name	Category	Price
Gizmo	Gadget	10
Camera	Photo	20
Gizmo	Photo	30
~~Gizmo~~	~~Gadget~~	~~40~~

Other Keys

```
CREATE TABLE Product (
        productID  CHAR(10),
        name CHAR(30),
        category VARCHAR(20),
        price INT,
        PRIMARY KEY (productID),
        UNIQUE (name, category))
```

There is at most one PRIMARY KEY;
there can be many UNIQUE

Foreign Key Constraints

```
CREATE TABLE Purchase (
        prodName CHAR(30)
            REFERENCES Product(name),
        date DATETIME)
```

Referential integrity constraints

May write just Product if name is PK

prodName is a **foreign key** to Product(name)

name must be a **key** in Product
(i.e., PK or UNIQUE)

Foreign Key Constraints

- Example with multi-attribute primary key

```
CREATE TABLE Purchase (
        prodName CHAR(30),
        category VARCHAR(20),
        date DATETIME,
        FOREIGN KEY (prodName, category)
            REFERENCES  Product(name, category))
```

- (name, category) must be a KEY in Product

What happens when data changes?

Types of updates:

- In Purchase: insert/update
- In Product: delete/update

Product

Name	Category
Gizmo	gadget
Camera	Photo
OneClick	Photo

Purchase

ProdName	Store
Gizmo	Wiz
Camera	Ritz
Camera	Wiz

What happens when data changes?

SQL has three options for maintaining referential integrity on changes:

- NO ACTION reject bad modifications (default)
- CASCADE after delete/update do delete/update
- SET NULL set foreign-key field to NULL
- SET DEFAULT set FK field to default value
  - need to be declared with column, e.g.,
    CREATE TABLE Product (pid INT DEFAULT 42)

Maintaining Referential Integrity

```
CREATE TABLE Purchase (
        prodName CHAR(30),
        category VARCHAR(20),
        date DATETIME,
        FOREIGN KEY (prodName, category)
          REFERENCES  Product(name, category)
          ON UPDATE CASCADE
          ON DELETE SET NULL)
```

Product

Name	Category
Gizmo	gadget
Camera	Photo
OneClick	Photo

Purchase

ProdName	Category
Gizmo	gadget
Snap	Camera
EasyShoot	Camera

Constraints on Attributes and Tuples

- Constraints on attributes:
  - NOT NULL -- obvious meaning...
  - CHECK condition -- any condition !
- Constraints on tuples
  - CHECK condition

Constraints on Attributes and Tuples

```
CREATE TABLE Product (
        productID  CHAR(10),
        name CHAR(30),
        category VARCHAR(20),
        price INT CHECK (price > 0),
        PRIMARY KEY (productID))
```

Constraints on Attributes and Tuples

```
CREATE TABLE Product (
        productID  CHAR(10),
        name CHAR(30),
        category VARCHAR(20)
          CHECK (category in (‘toy’, ‘gadget’, ‘apparel’)),
        price INT CHECK (price > 0),
        PRIMARY KEY (productID))
```

Constraints on Attributes and Tuples

```
CREATE TABLE Product (
        productID  CHAR(10),
        name CHAR(30) NOT NULL,
        category VARCHAR(20)
          CHECK (category in ('toy', 'gadget', 'apparel')),
        price INT CHECK (price > 0),
        PRIMARY KEY (productID))
```

Constraints on Attributes and Tuples

```
CREATE TABLE R (
        A int NOT NULL,
        B int CHECK (B > 50 and B < 100),
        C varchar(20),
        D int,
        CHECK (C >= 'd' or D > 0))
```

Constraints on Attributes and Tuples

What does this constraint do?

What is the difference from Foreign Key?

```
CREATE TABLE Purchase (
        prodName CHAR(30)
                CHECK (prodName IN
                        (SELECT Product.name
                         FROM Product)),
        date DATETIME NOT NULL)
```

General Assertions

```
CREATE ASSERTION myAssert CHECK
  (NOT EXISTS(
        SELECT Product.name
        FROM Product, Purchase
        WHERE Product.name = Purchase.prodName
        GROUP BY Product.name
        HAVING count(*) > 200) )
```

But most DBMSs do not implement assertions, because it is hard to support them efficiently. Instead, DBMSs provide triggers