Introduction to Data Management
CSE 414

Unit 6: Conceptual Design

E/R Diagrams

Integrity Constraints

BCNF

(3 lectures)

Introduction to Data Management CSE 414

E/R Diagrams

Announcements

- HW6 due on Friday. Turn instances off!!!
- WebQuiz 6 due on Saturday

Class Overview

- Unit 1: Intro
- Unit 2: Relational Data Models and Query Languages
- Unit 3: Non-relational data
- Unit 4: RDMBS internals and query optimization
- Unit 5: Parallel query processing
- Unit 6: DBMS usability, conceptual design
  - E/R diagrams
  - Constraints
  - Schema normalization
- Unit 7: Transactions
- Unit 8: Advanced topics (time permitting)

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 a very long time (years). Updating the schema while in production is very expensive (why?)

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
  - UML, model-driven architecture
- Reading: Sec. 4.1-4.6

Database Design Process

Conceptual Model:

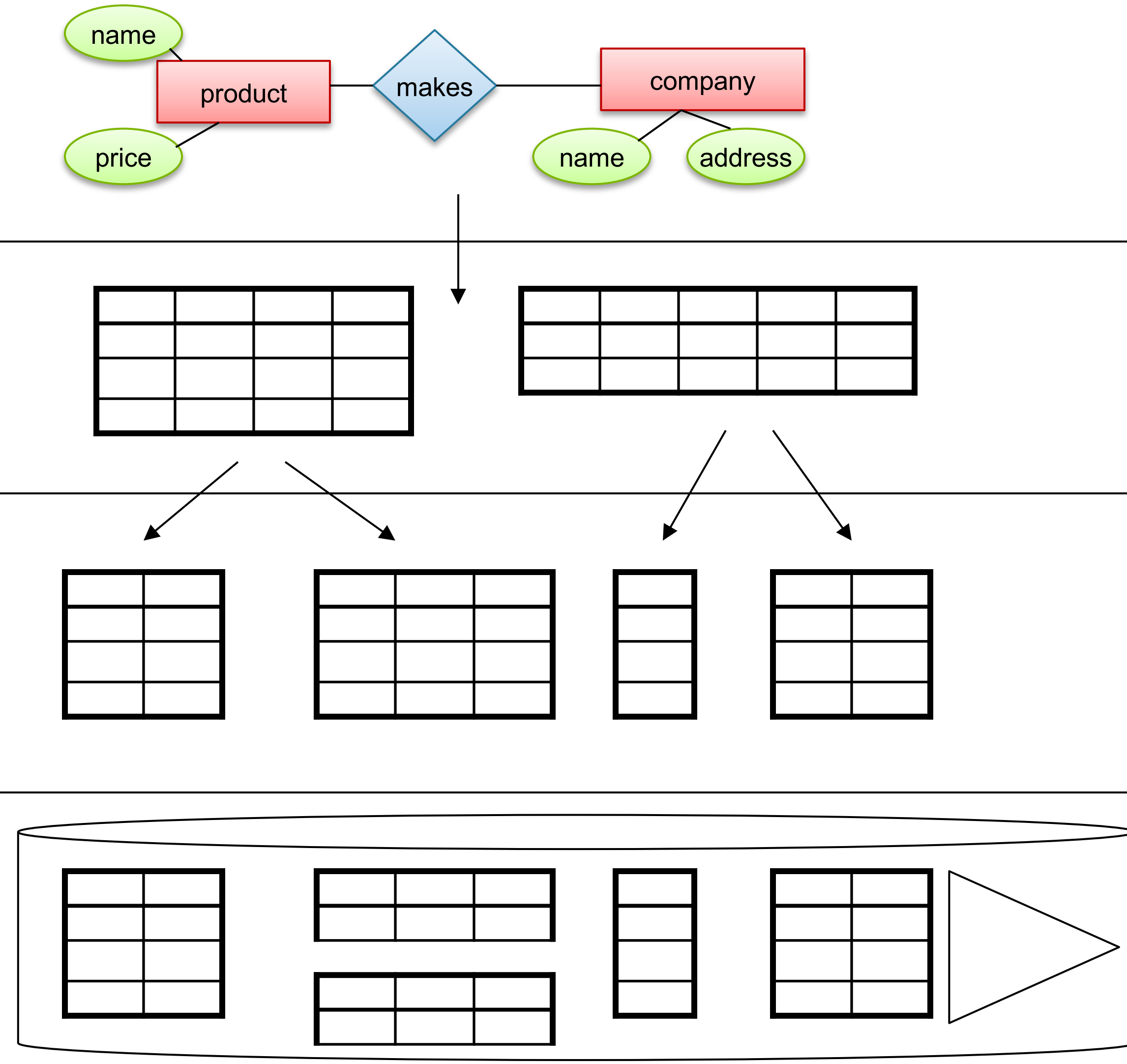
Relational Model:
Tables + constraints
And also functional dep.

Normalization:
Eliminates anomalies

Conceptual Schema

Physical storage details

Physical Schema

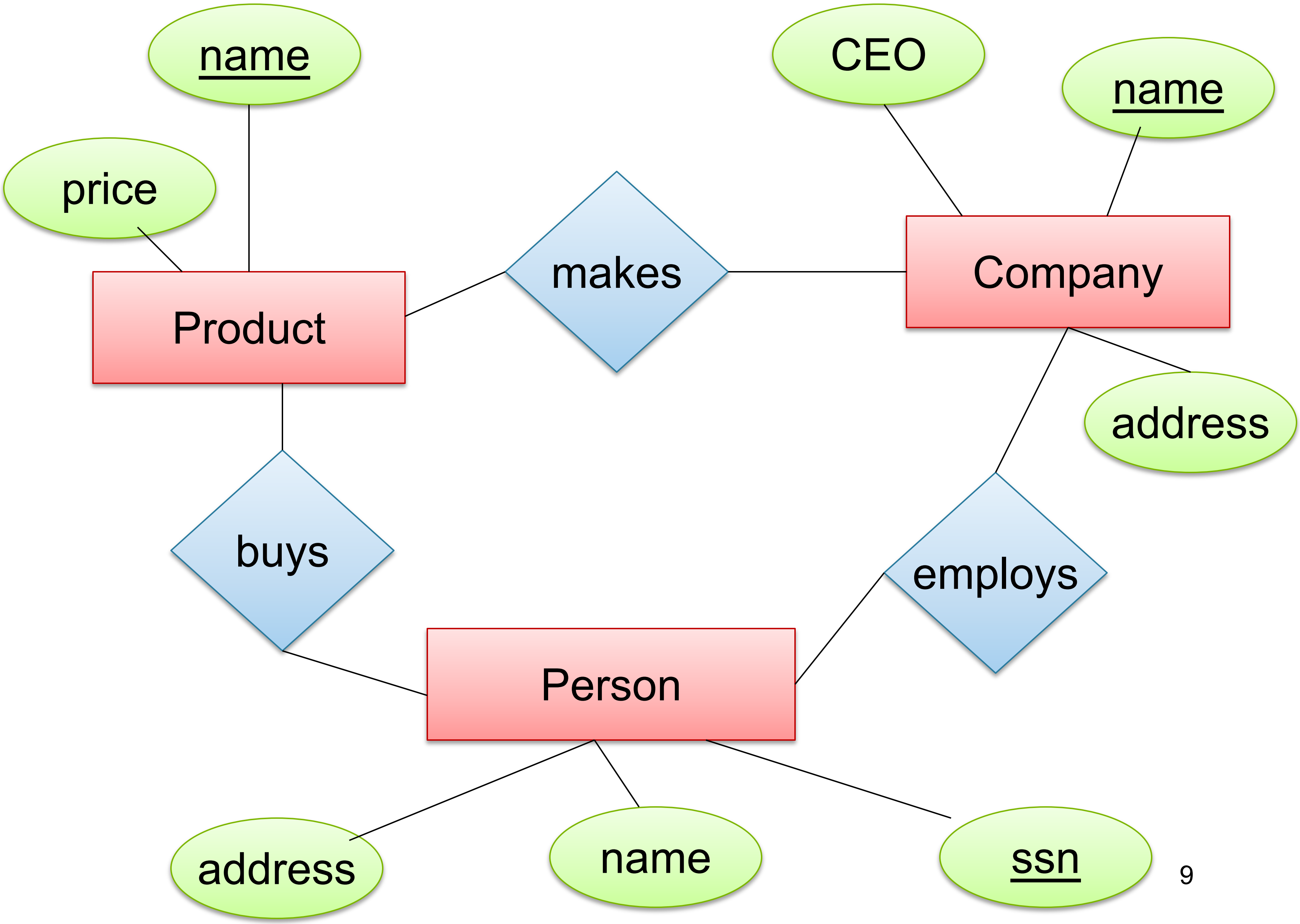


Entity / Relationship Diagrams

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

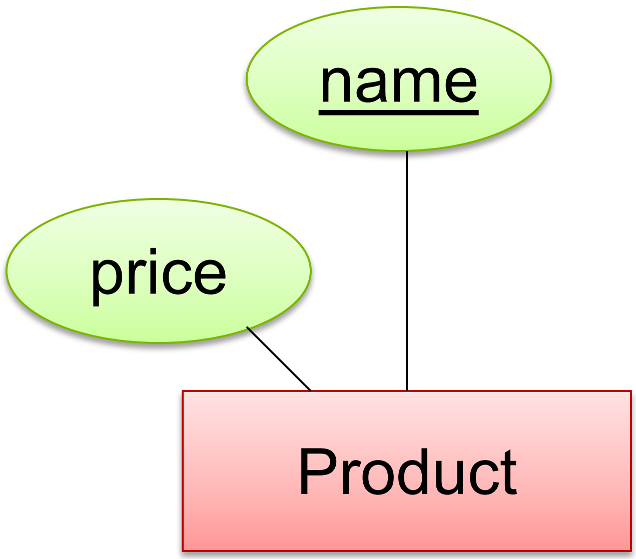
- Attribute

- Relationship


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

Keys in E/R Diagrams

- Every entity set must have a key

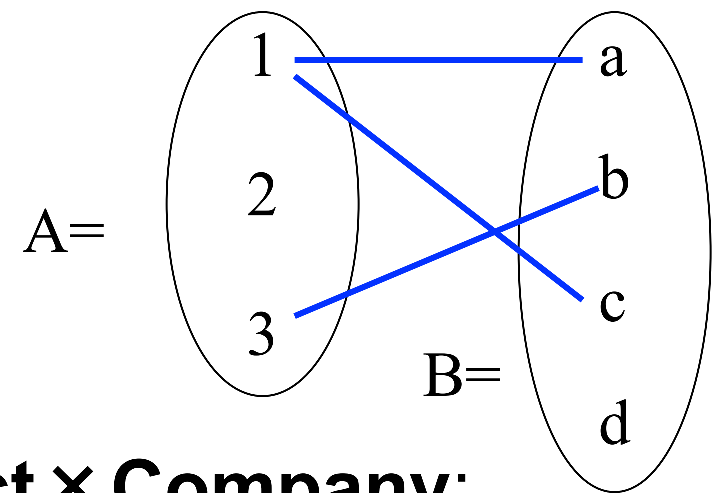


What is a Relation ?

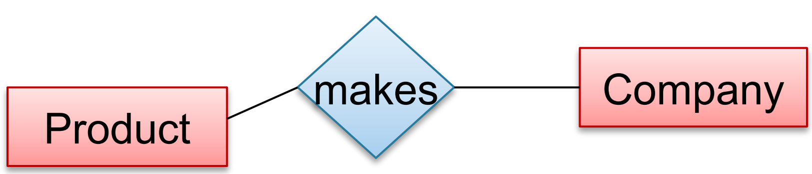
- A mathematical definition:
  - if A, B are sets, then a relation R is a subset of A × B
- A={1,2,3}, B={a,b,c,d},

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

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

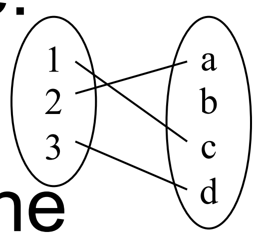


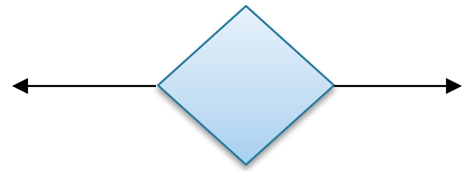
- **makes** is a subset of **Product × Company**:



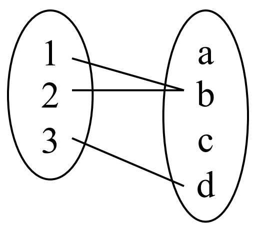
Multiplicity of E/R Relations

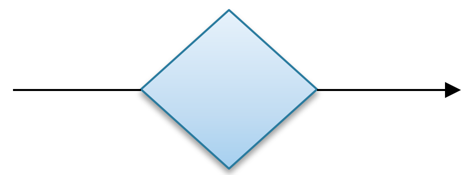
- one-one:



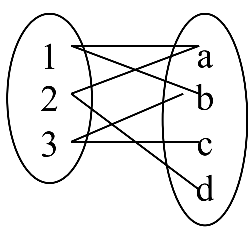


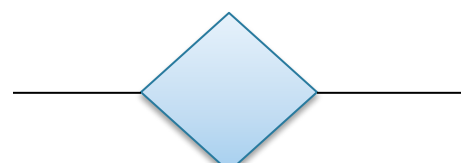
- many-one

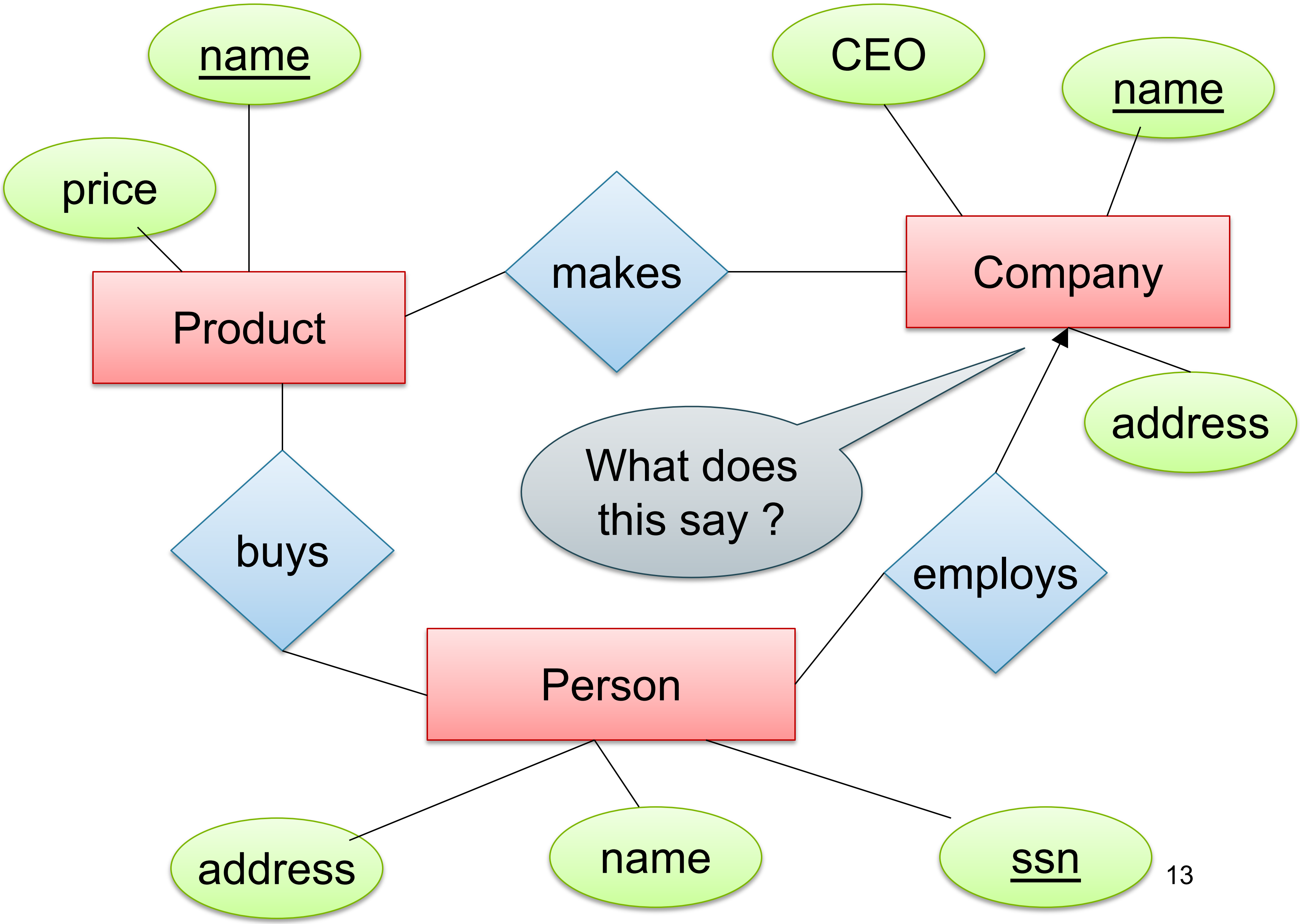




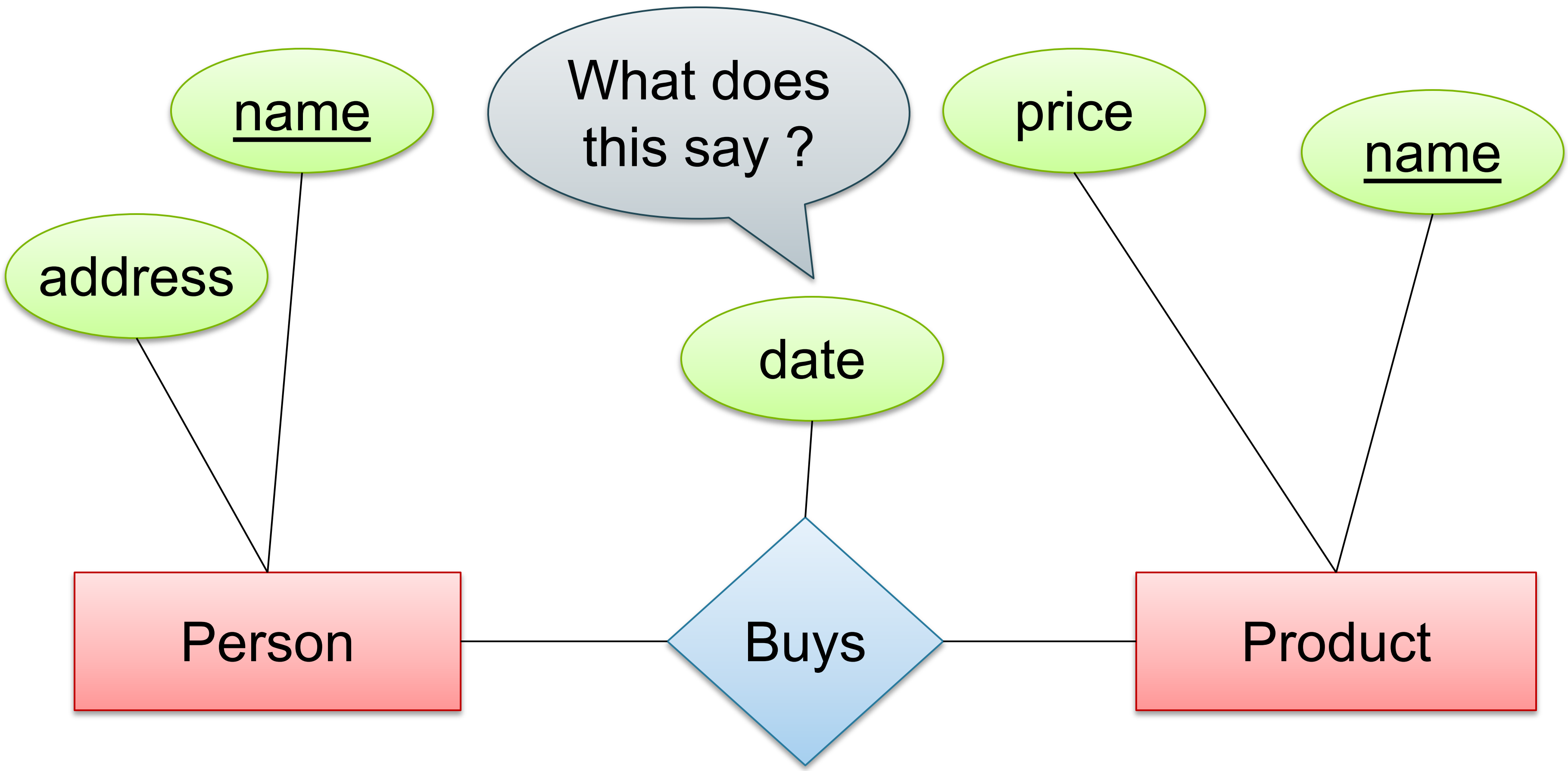
- many-many





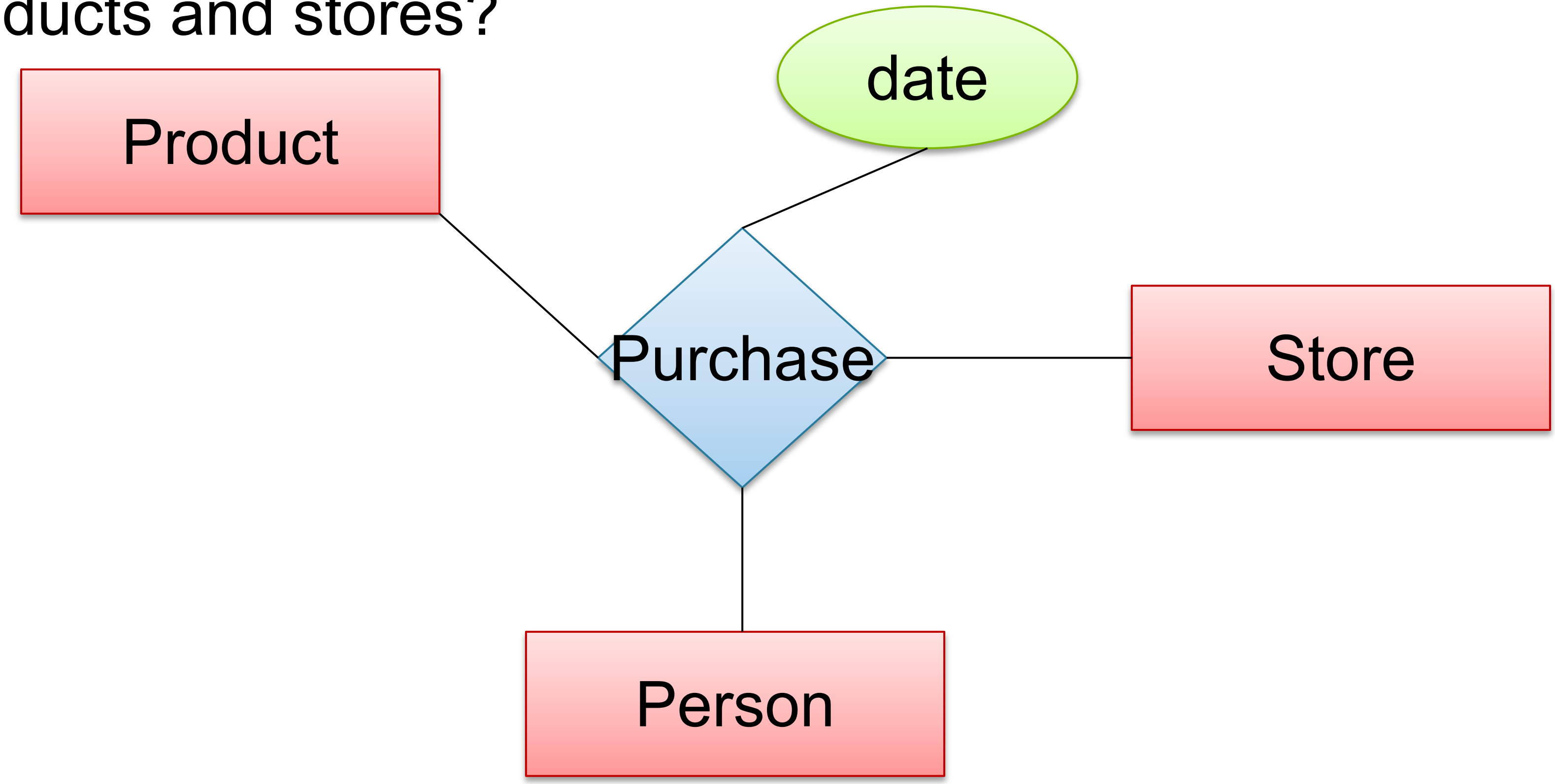

name
price
Product
makes
CEO
name
Company
address
What does this say ?
buys
employs
Person
address
name
ssn

Attributes on Relationships



Multi-way Relationships

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

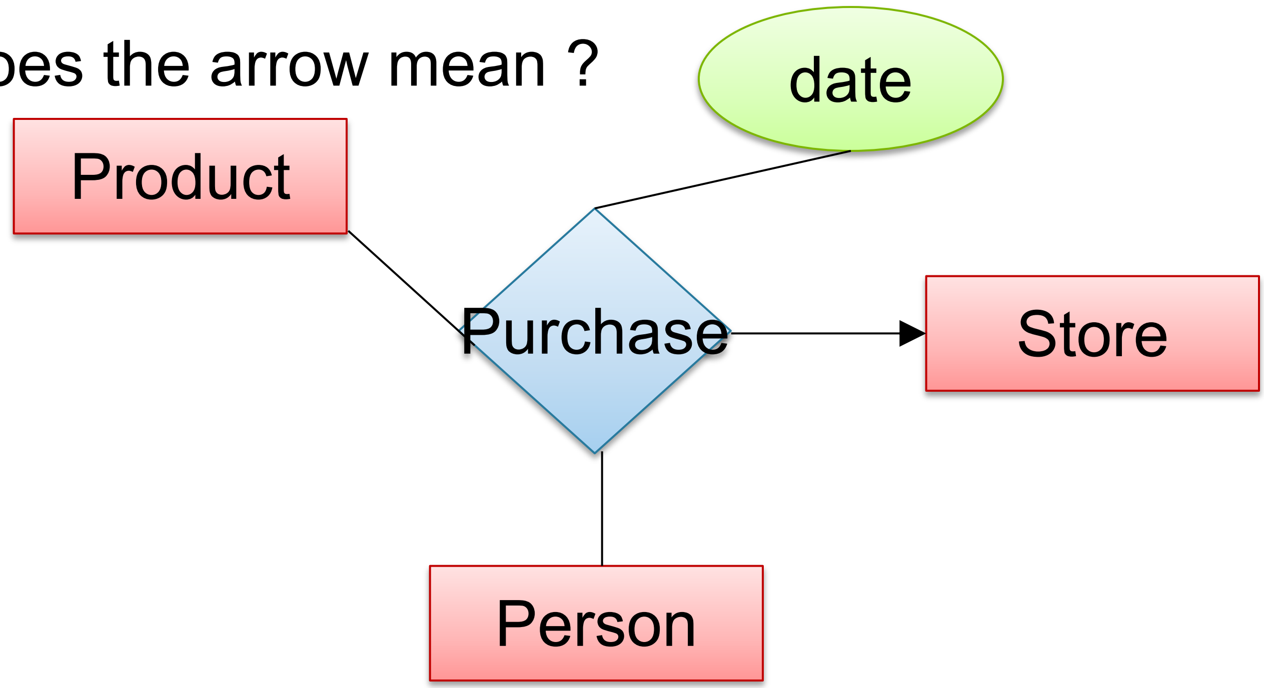


Can still model as a mathematical set (How?)

As a set of triples $\subseteq$ Product $\times$ Person $\times$ Store

Arrows in Multiway Relationships

Q: What does the arrow mean ?

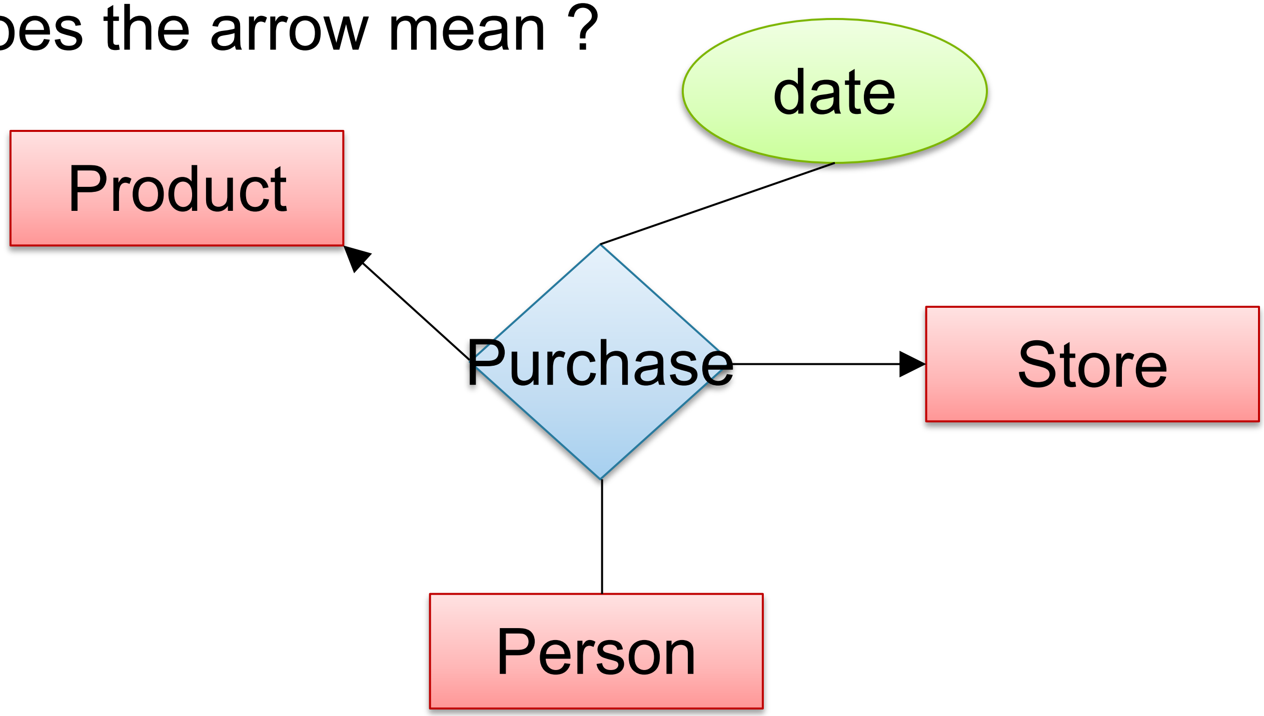


A: Any person buys a given product from at most one store

[Fine print: 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]

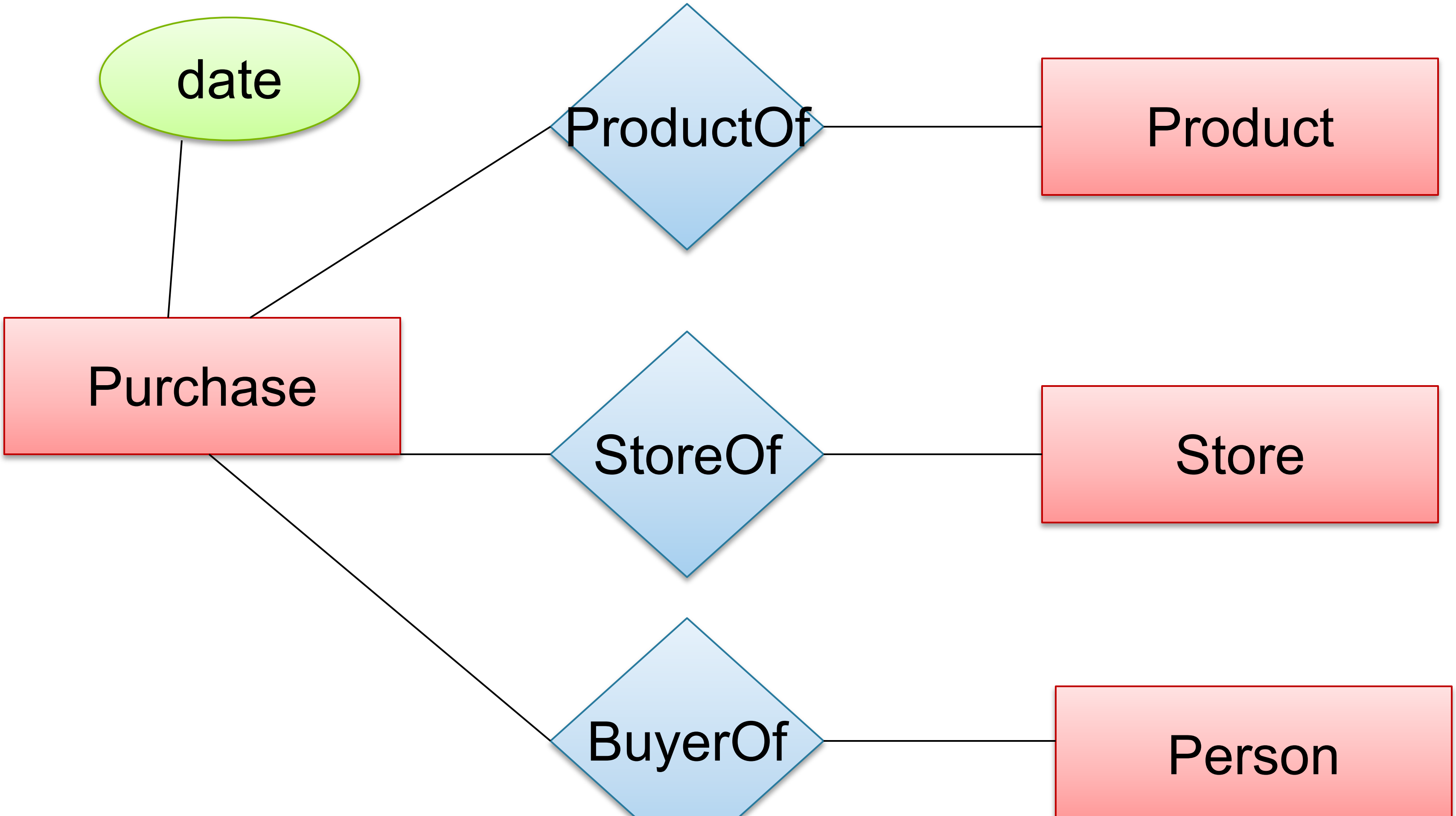
Arrows in Multiway Relationships

Q: What does the arrow mean ?



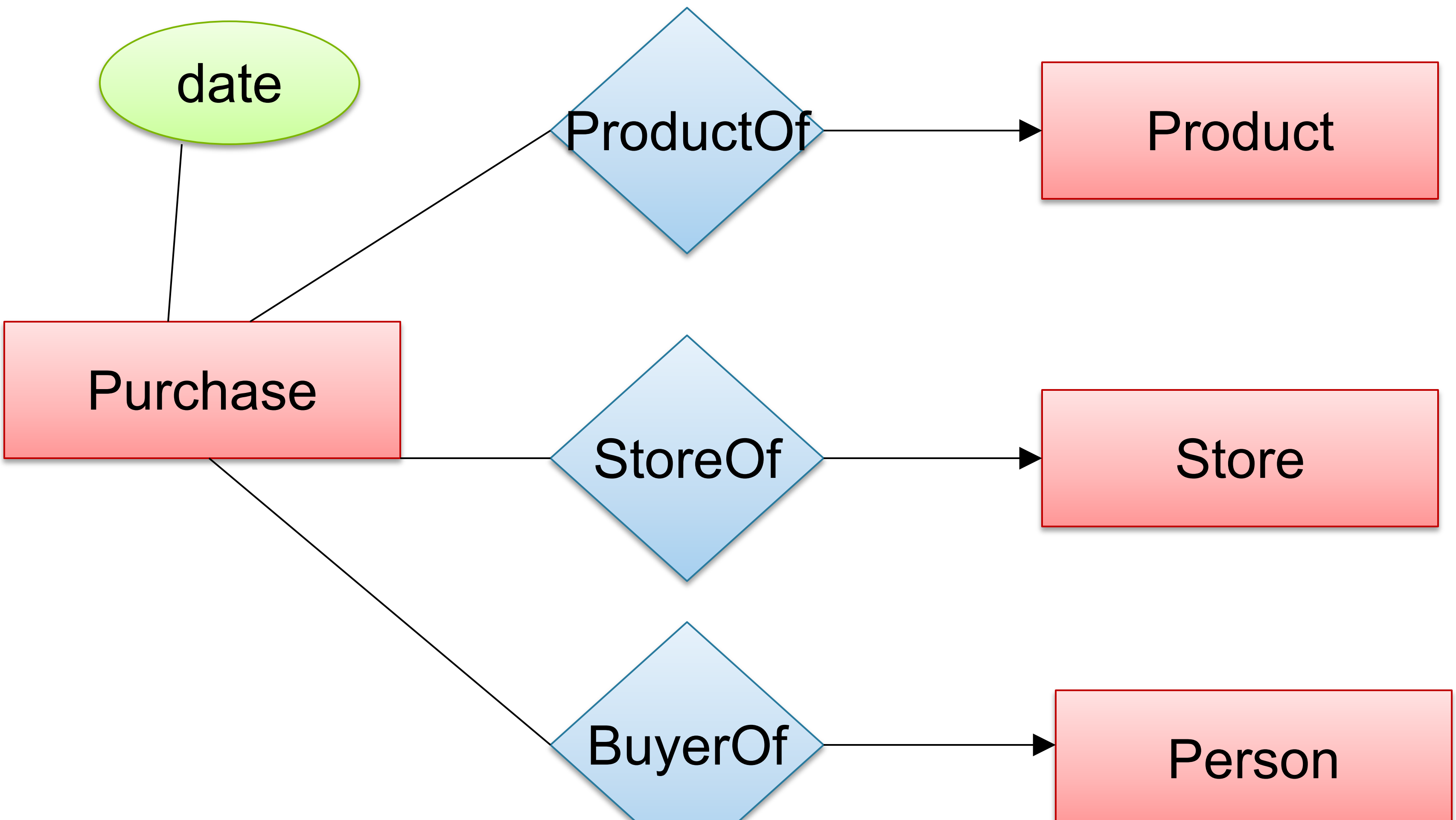
A: Any person buys a given product from at most one store AND every store sells to every person at most one product

Converting Multi-way Relationships to Binary



Arrows go in which direction?

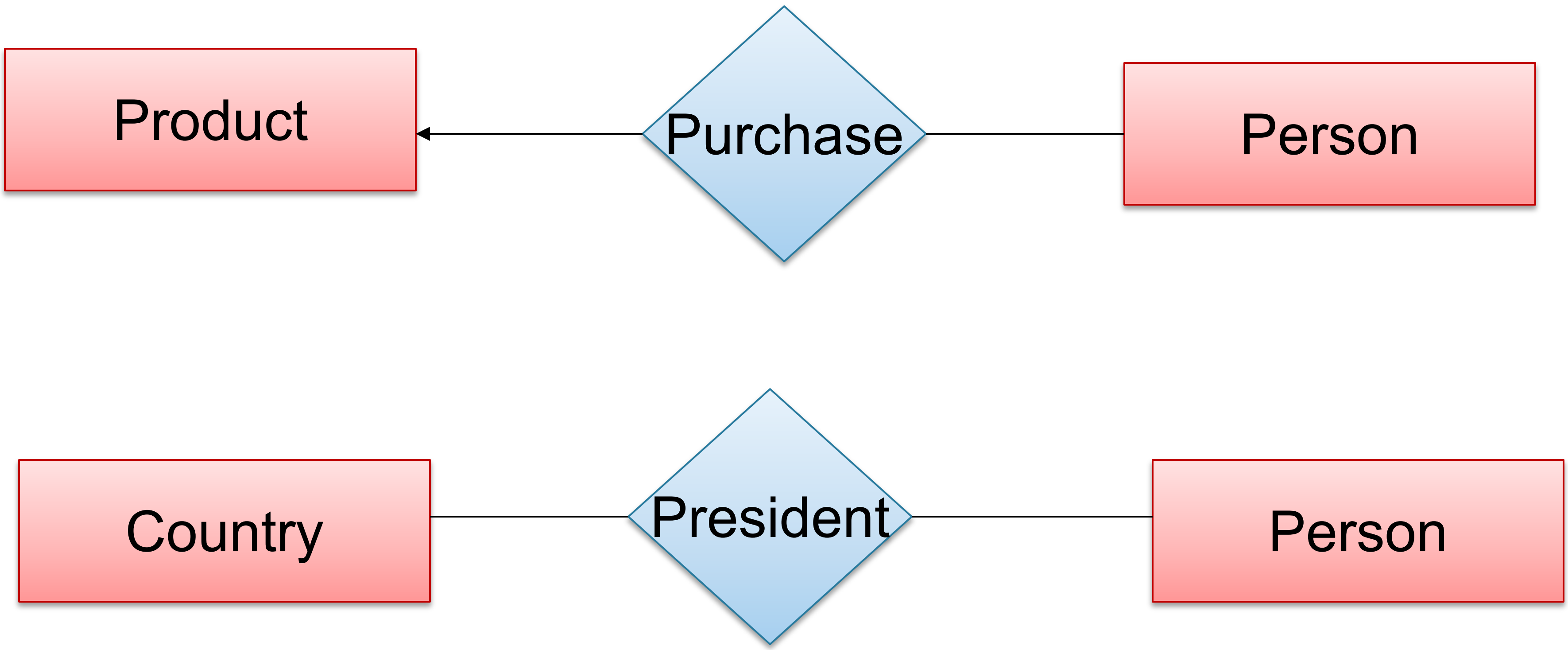
Converting Multi-way Relationships to Binary



Make sure you understand why!

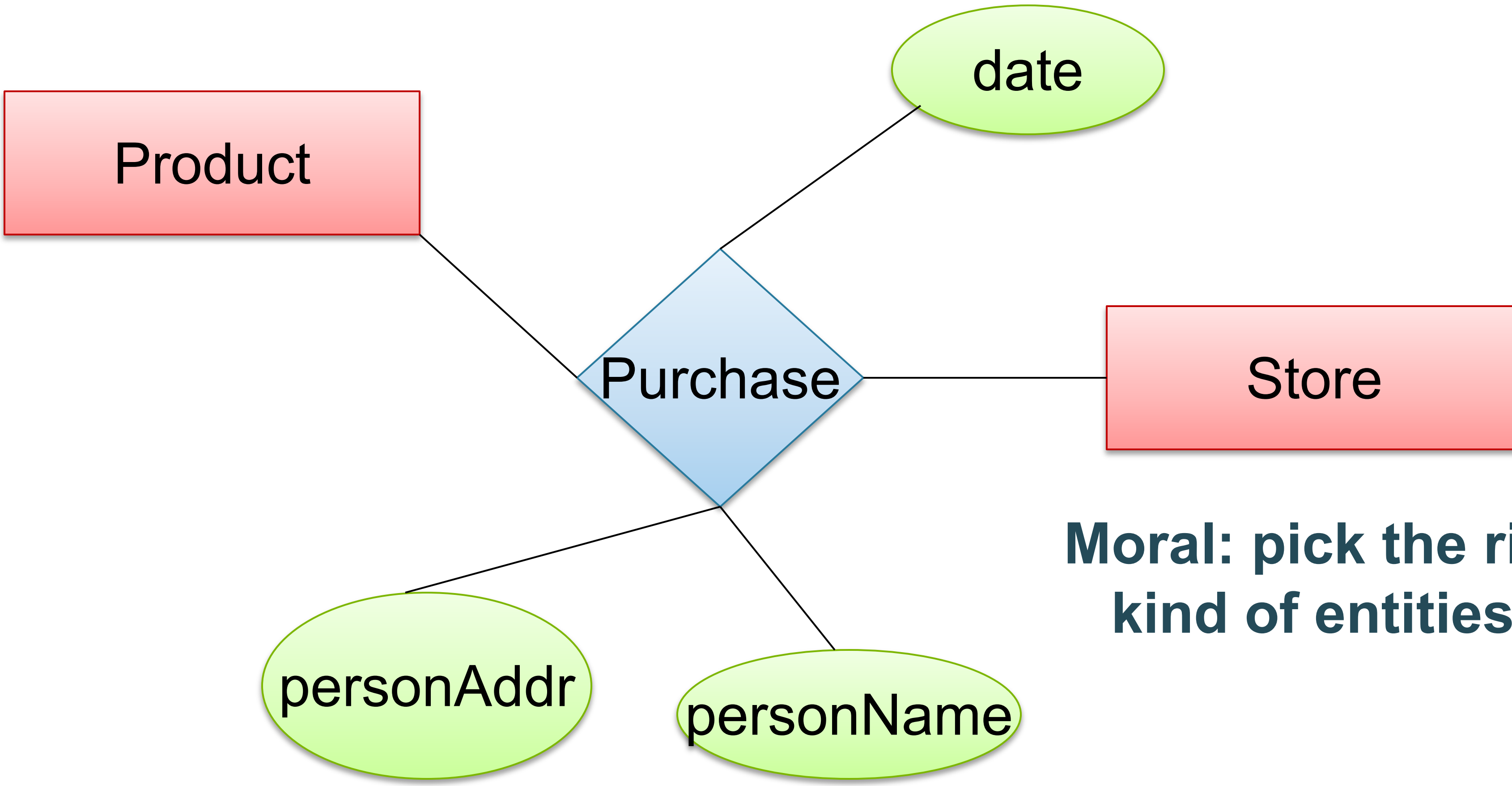
3. Design Principles

What's wrong?



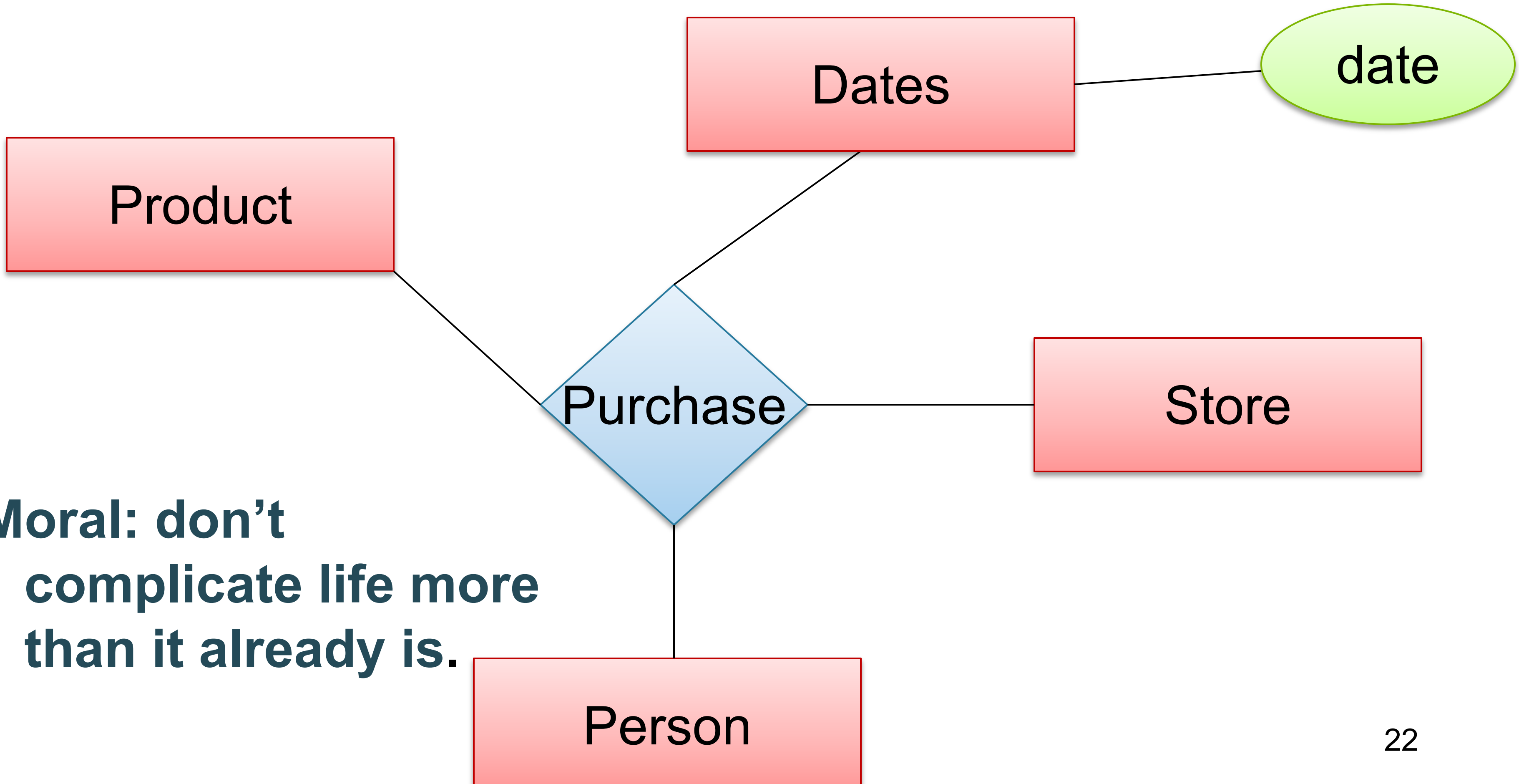
Moral: Be faithful to the specifications of the application!

Design Principles: What's Wrong?



Moral: pick the right kind of entities.

Design Principles: What's Wrong?

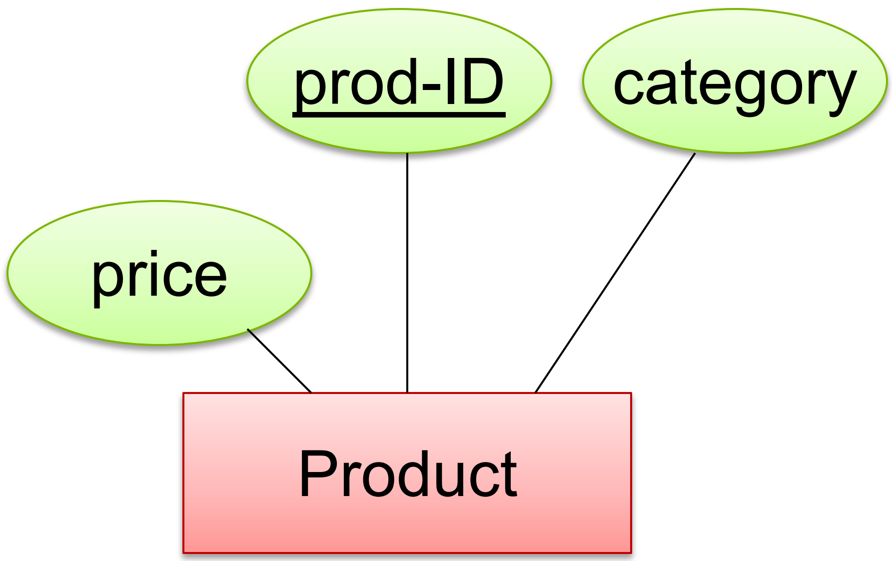


Moral: don't complicate life more than it already is.

From E/R Diagrams to Relational Schema

- Entity set → relation
- Relationship → relation

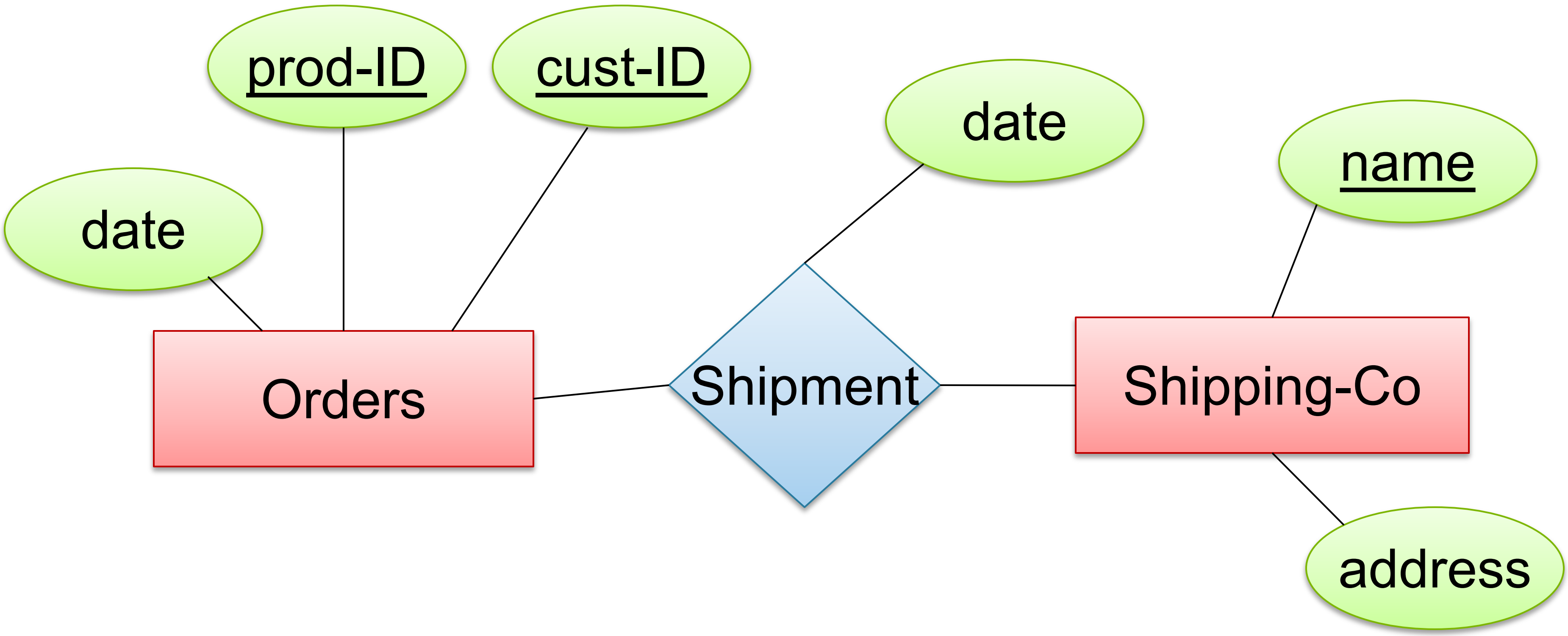
Entity Set to Relation



Product(prod-ID, category, price)

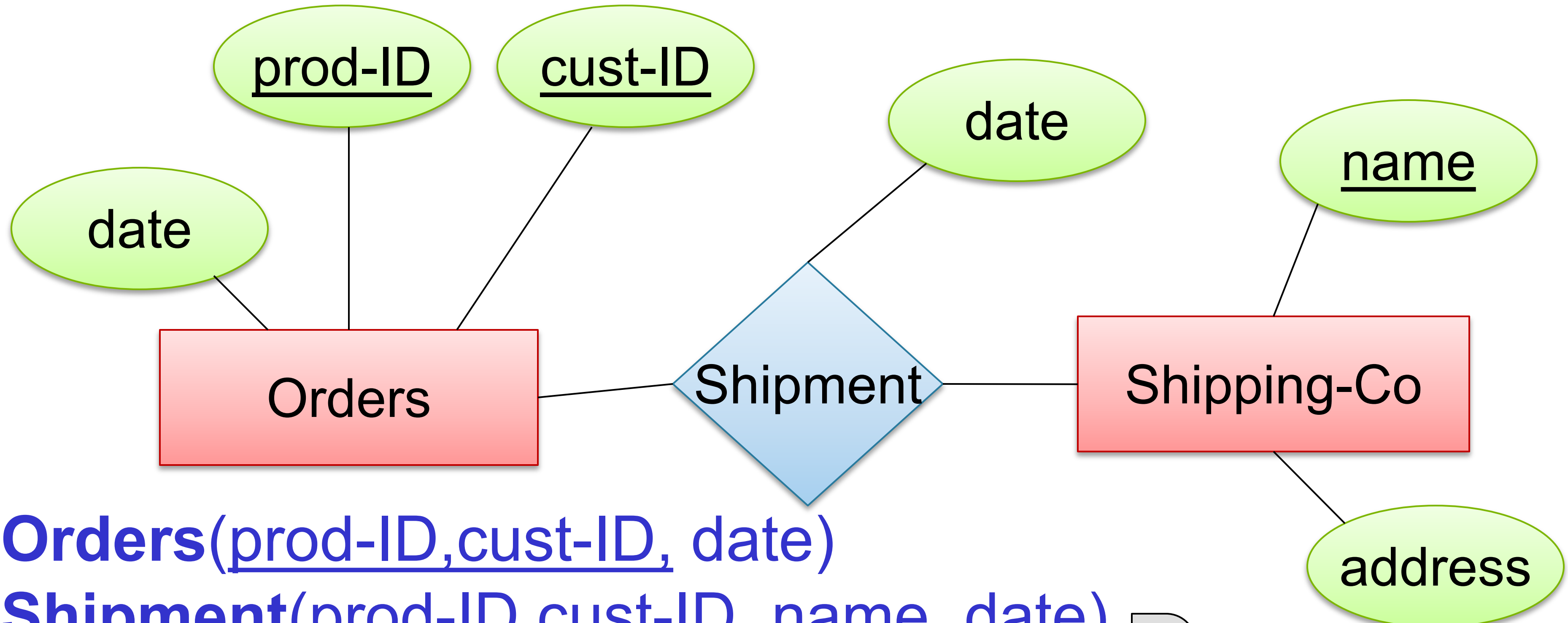
prod-ID	category	price
Gizmo55	Camera	99.99
Pokemn19	Toy	29.99

N-N Relationships to Relations



Represent this in relations

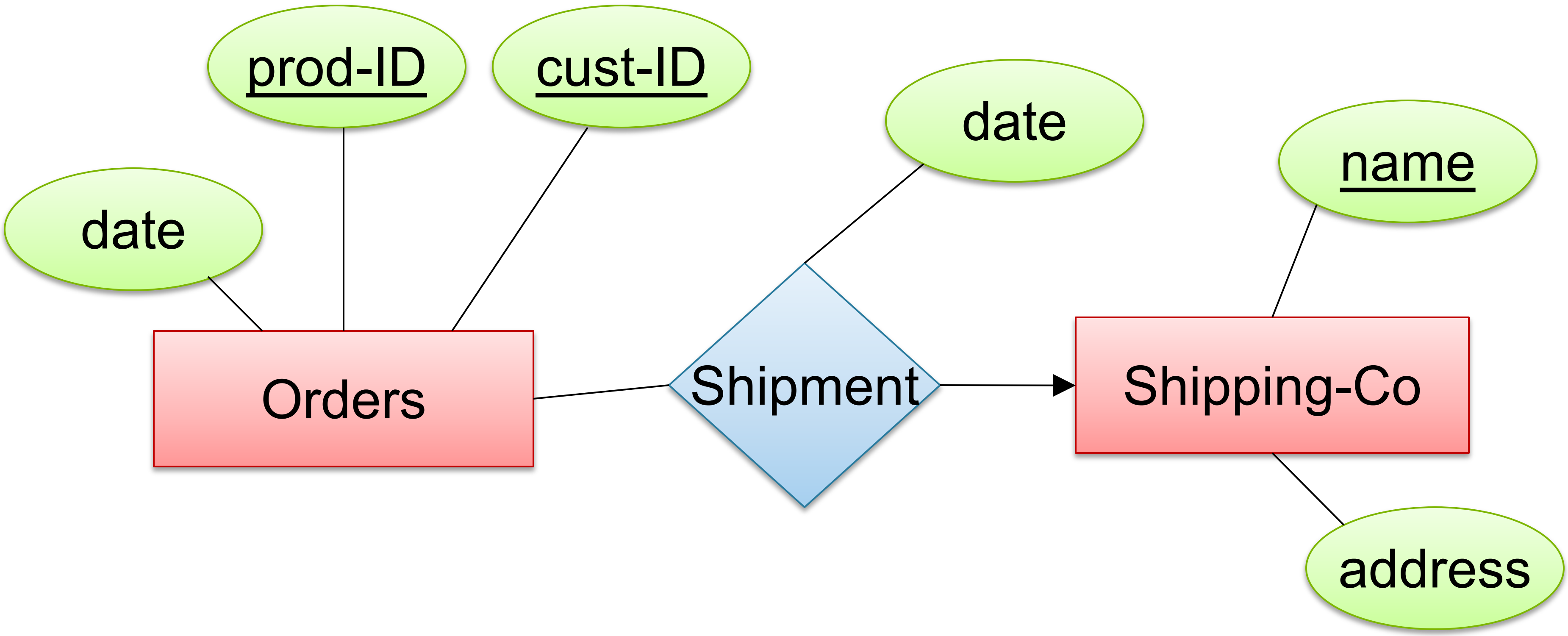
N-N Relationships to Relations



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

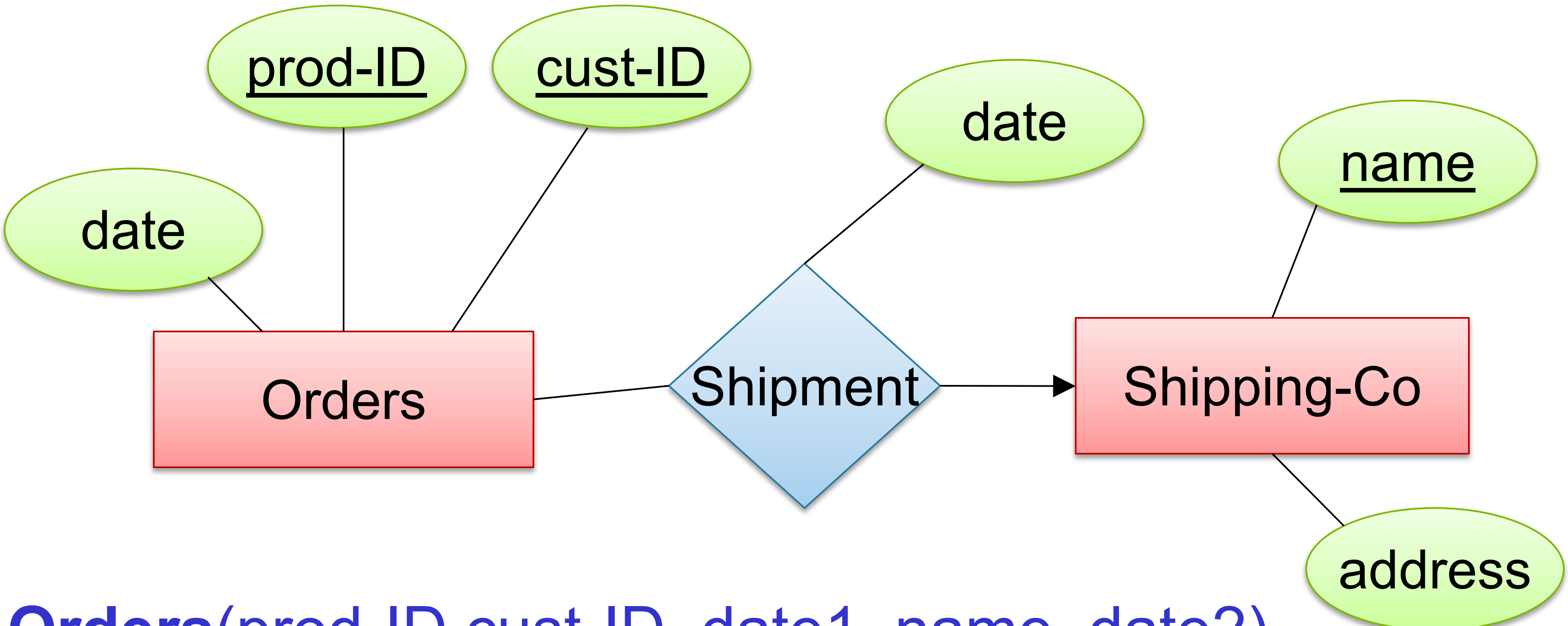
prod-ID	cust-ID	name	date
Gizmo55	Joe12	UPS	4/10/2011
Gizmo55	Joe12	FEDEX	4/9/2011

N-1 Relationships to Relations



Represent this in relations

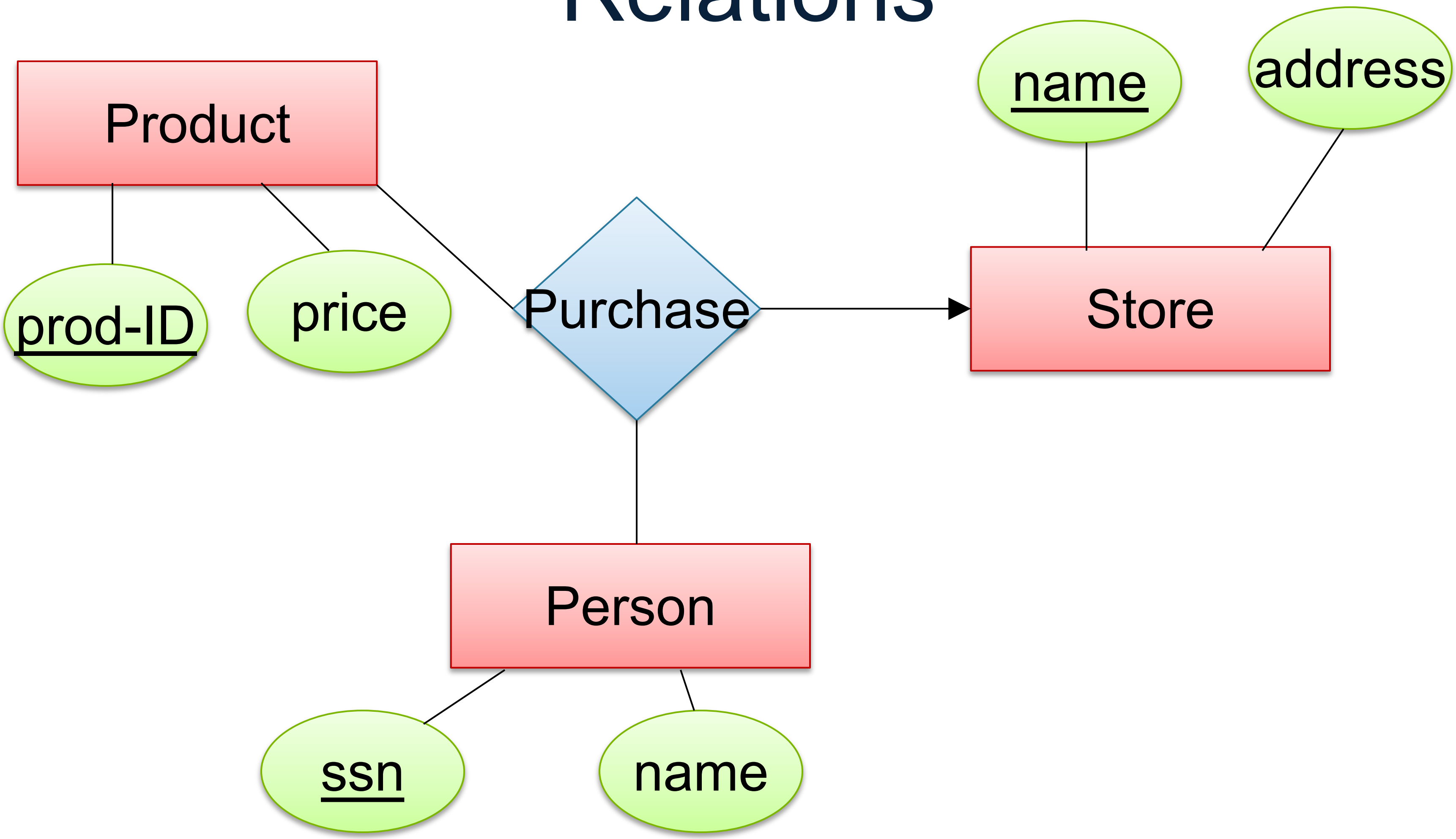
N-1 Relationships to Relations



Orders(<u>prod-ID,cust-ID,</u> date1, name, date2)
Shipping-Co(<u>name</u>, address)

Remember: no separate relations for many-one relationship

Multi-way Relationships to Relations



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

Modeling Subclasses

Some objects in a class may be special

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

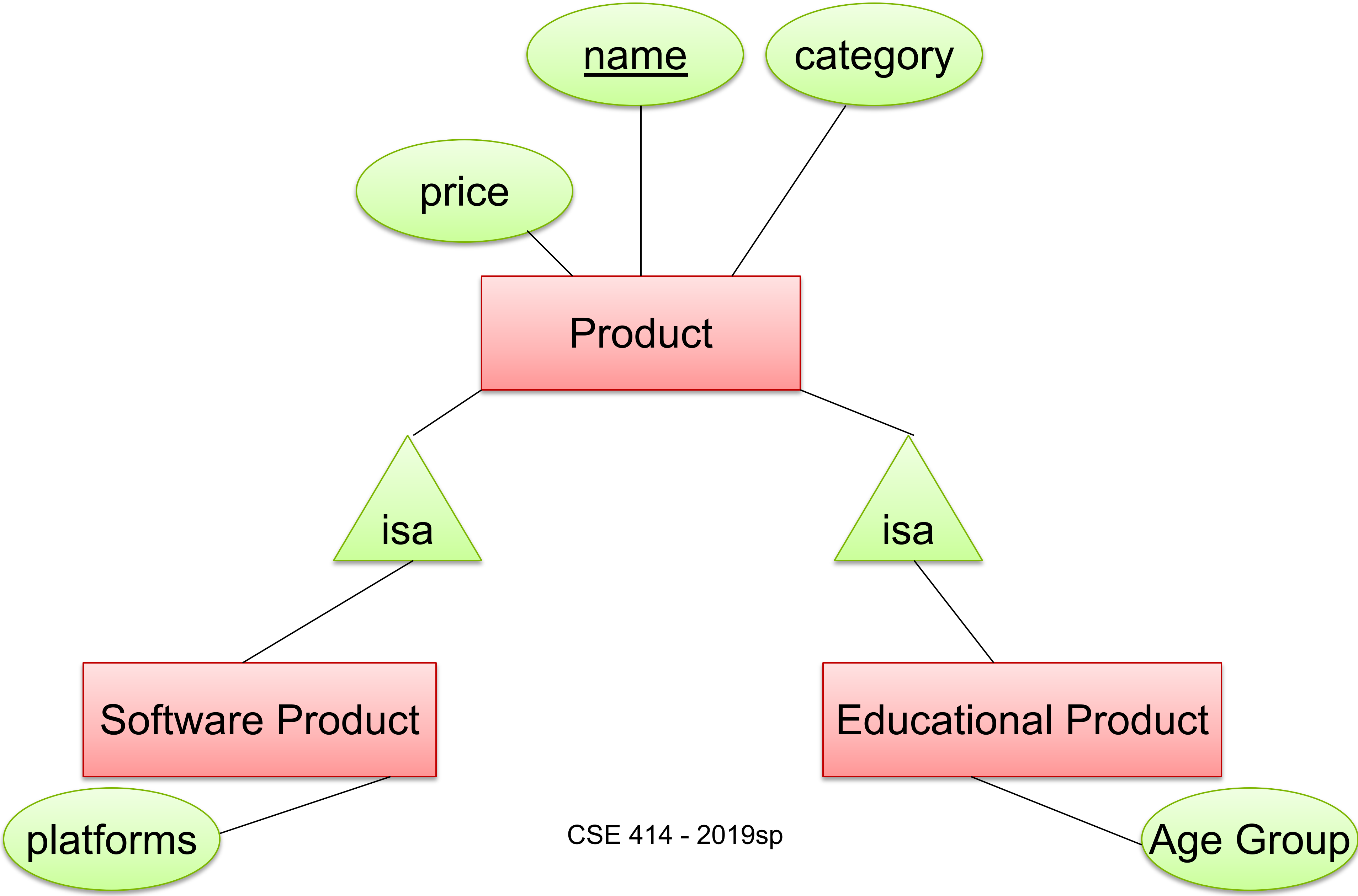
Products

Software products

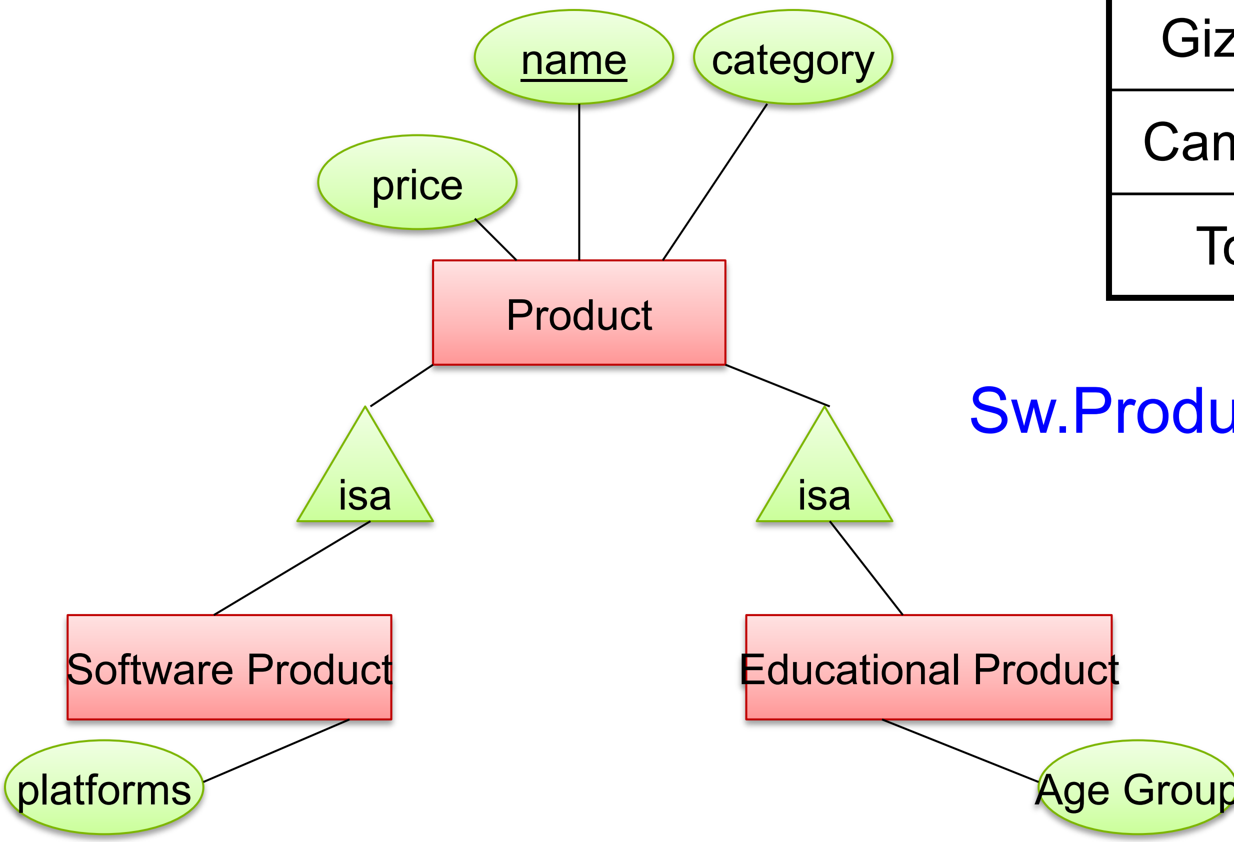
Educational products

So --- we define subclasses in E/R

Subclasses


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

Subclasses to Relations



Product

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

Sw.Product

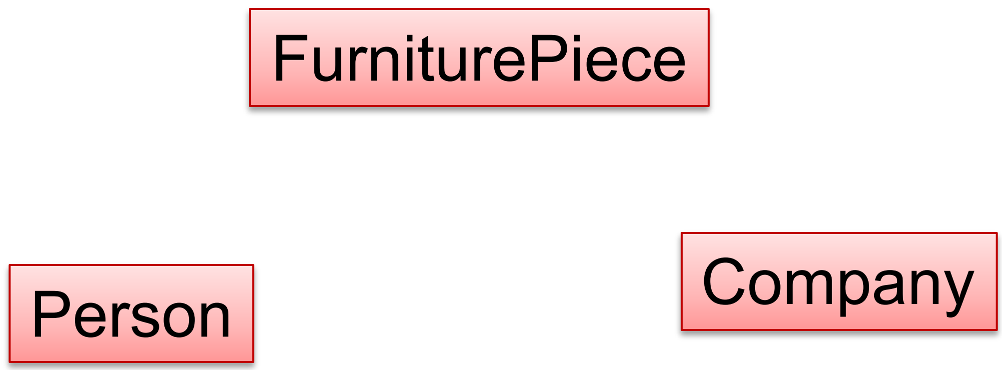
Name	platforms
Gizmo	unix

Ed.Product

Name	Age Group
Gizmo	toddler
Toy	retired

Other ways to convert are possible

Modeling Union Types with Subclasses

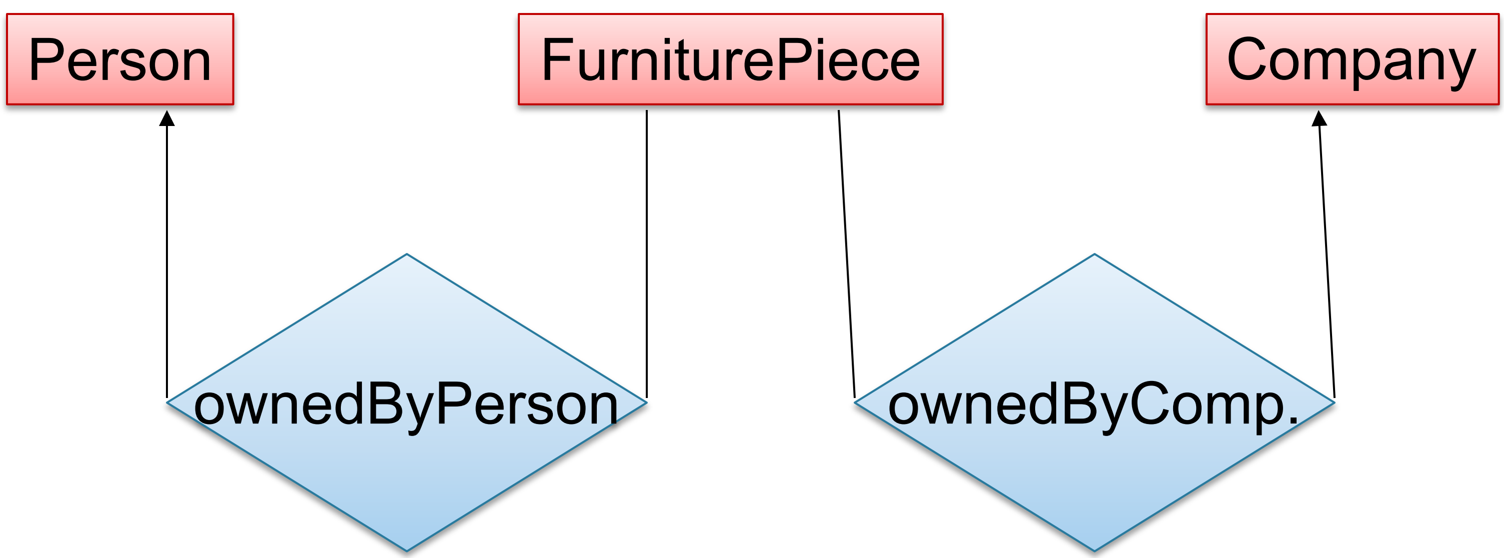


Say: each piece of furniture is owned either by a person or by a company

Modeling Union Types with Subclasses

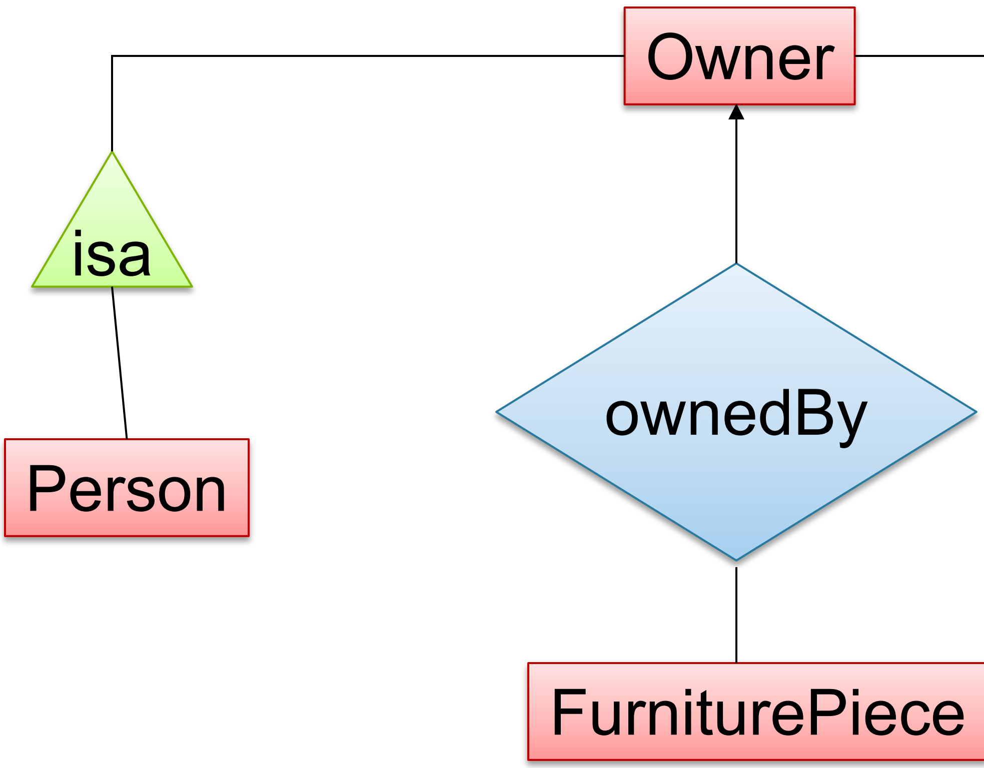
Say: each piece of furniture is owned either by a person or by a company

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



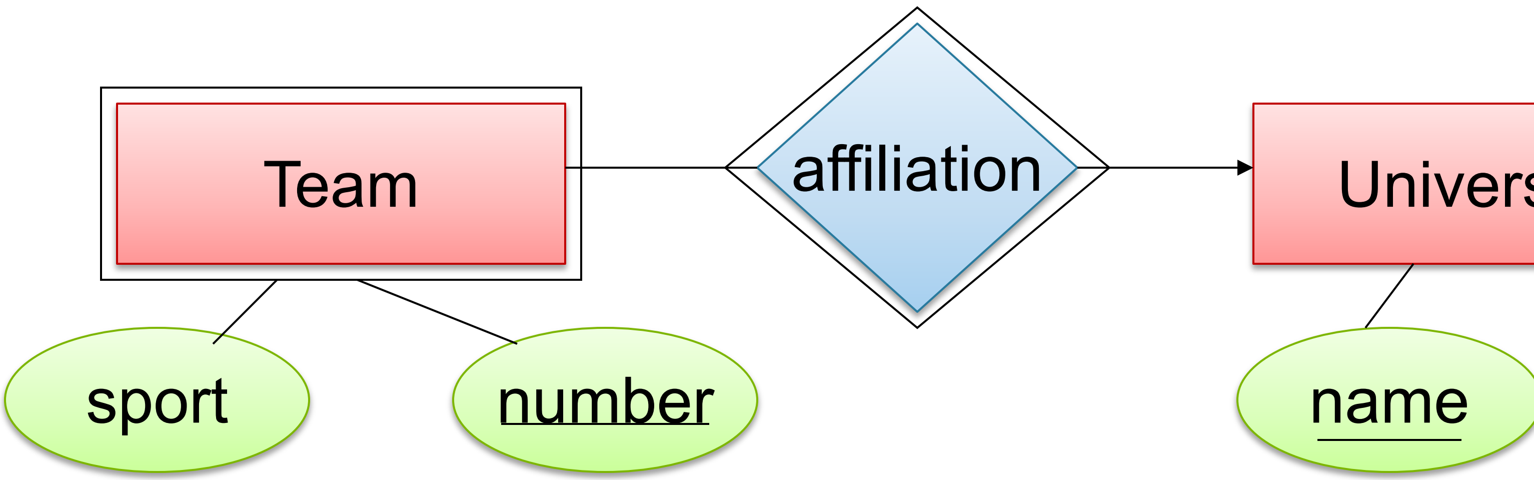
Modeling Union Types with Subclasses

Solution 2: better, more laborious



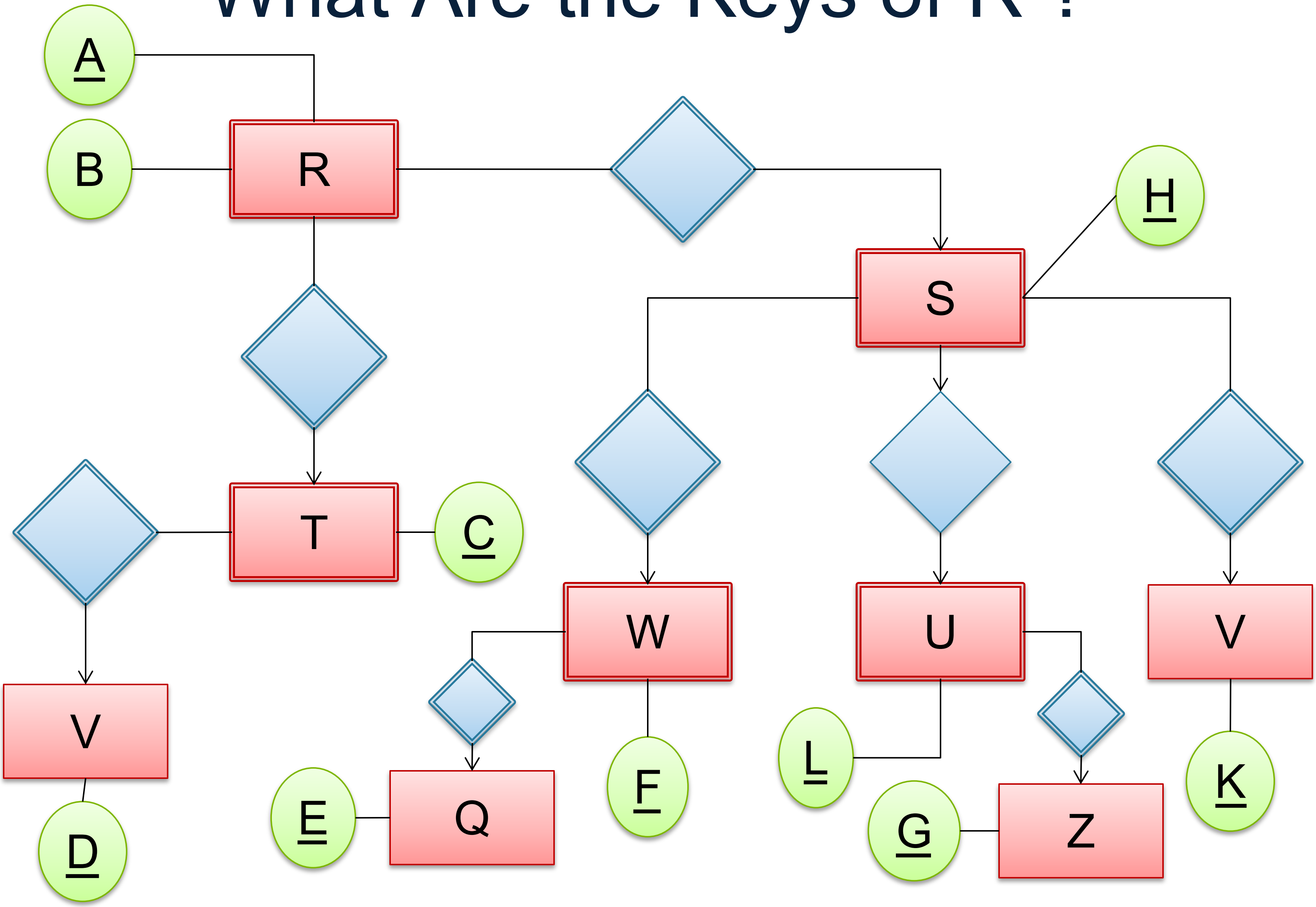
Weak Entity Sets

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



Team(sport, <u>number, universityName</u>)
University(<u>name</u>)

What Are the Keys of R ?



Introduction to Data Management CSE 414

Integrity Constraints

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.

Why?

How?

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.

Why? Because we want application data to be consistent

How?

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.

Why? Because we want application data to be consistent

How? The DBMS checks and enforces IC during updates

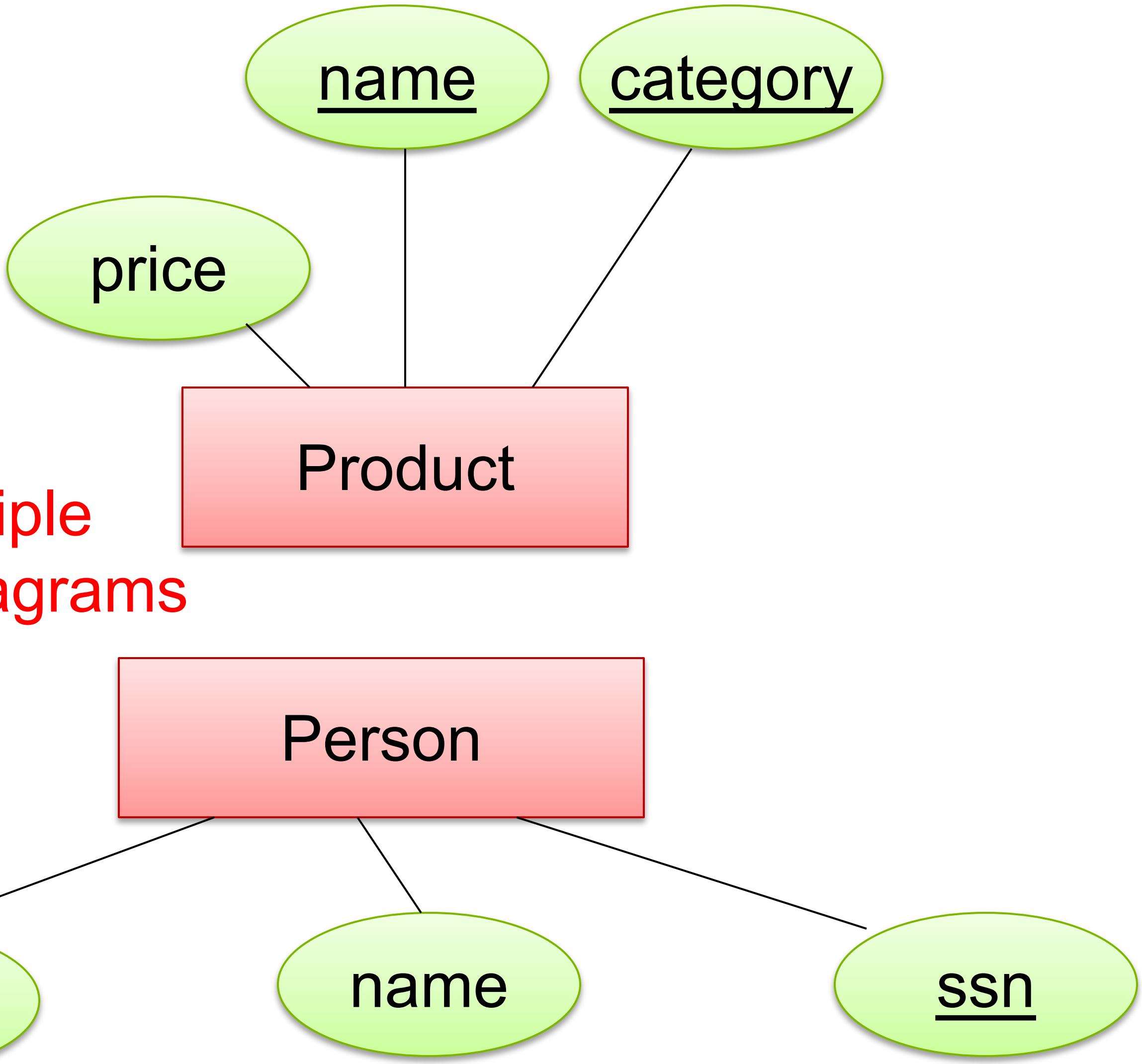
Constraints in E/R Diagrams

- Keys
- Single-value constraints
- Referential integrity constraints
- General constraints

Keys in E/R Diagrams

Underline:

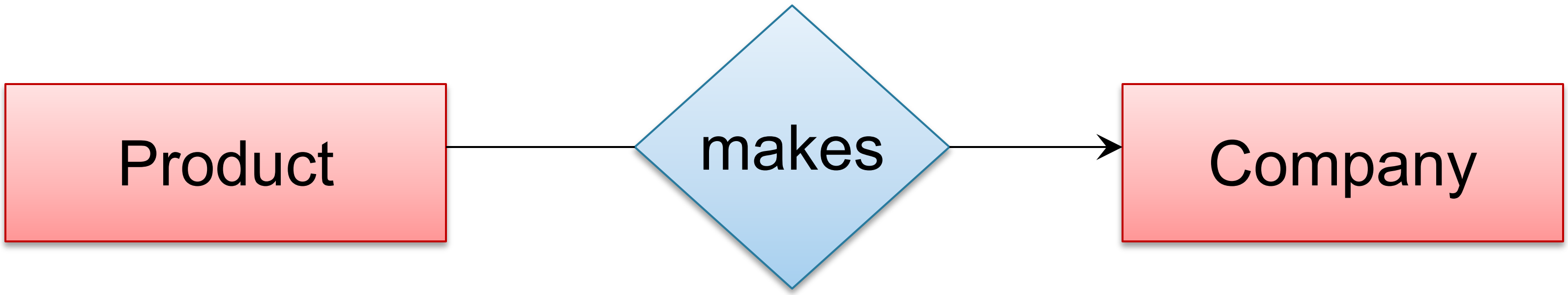
No formal way
to specify multiple
keys in E/R diagrams



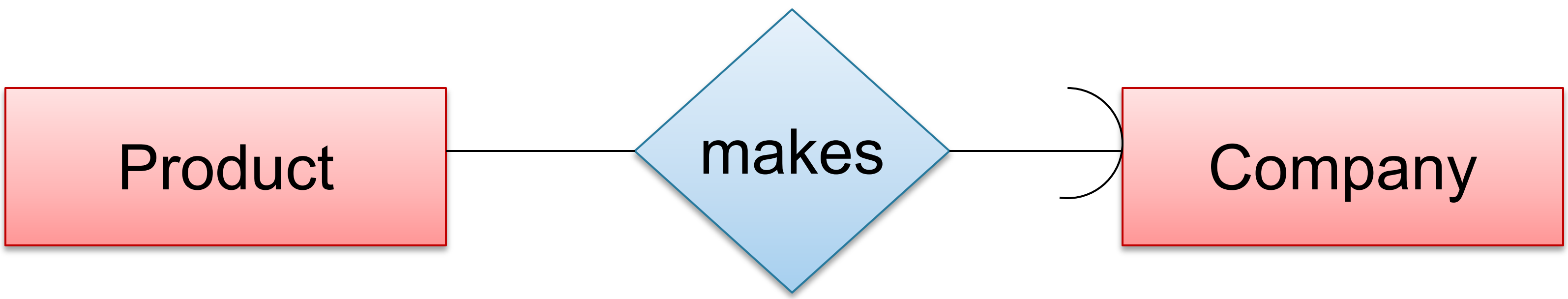
Single Value Constraints

vs.

Referential Integrity Constraints

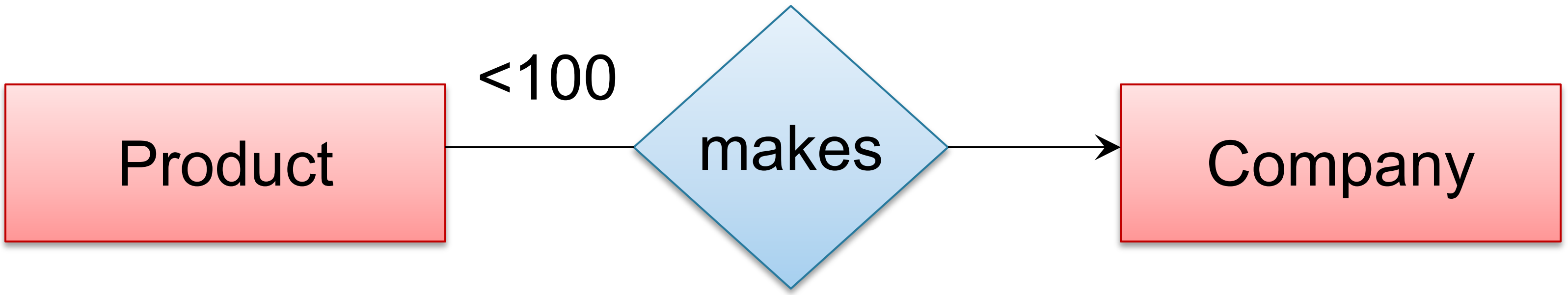


Each product made by at most one company.
Some products made by no company



Each product made by *exactly* one company.

Other Constraints



A Company entity is connected to at most 99 Product entities

Constraints in SQL

- Keys
- Attribute-level, tuple-level constraints
- General (complex) constraints

The more complex the constraint, the harder it is to check and to enforce

Key Constraints

Product(<u>name</u>, 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, 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

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 policies for maintaining referential integrity:

- NO ACTION reject modifications (default)
- CASCADE after delete/update do delete/update
- SET NULL set foreign-key field to NULL
- SET DEFAULT

```
CREATE TABLE …
   (pid int DEFAULT 42 REFERENCES…)
```

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  User (
        uid int primary key,
        firstName text,
        lastName text NOT NULL,
        age int CHECK (age > 12 and age < 120),
        email text,
        phone text,
        CHECK (email is not NULL or phone is not NULL)
)
```

Constraints on Attributes and Tuples

What does this constraint do?

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

What is the difference from Foreign-Key ?

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, they provide triggers

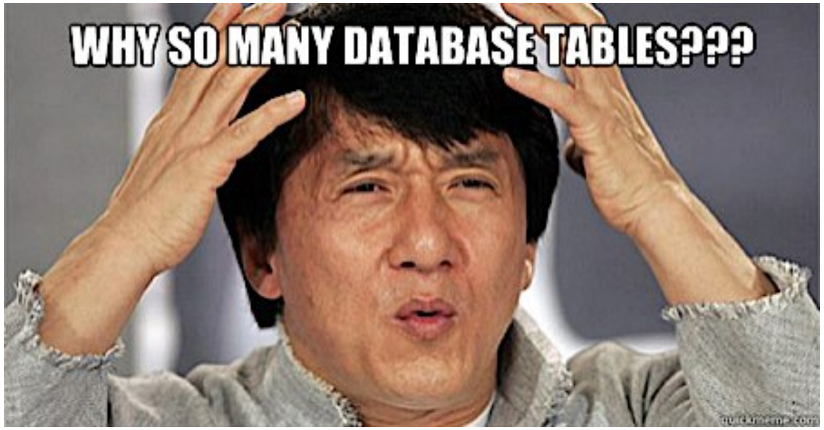
Introduction to Data Management
CSE 414

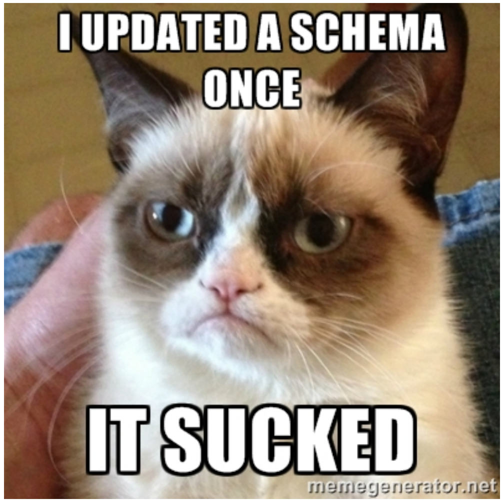
Design Theory and BCNF

Announcements

- Monday is Memorial day – no lecture
- Webquiz 6 is due tomorrow
- HW6 is due tonight
- HW7 is posted, due next Friday.

What makes good schemas?





Relational Schema Design

Name	SSN	PhoneNumber	City
Fred	123-45-6789	206-555-1234	Seattle
Fred	123-45-6789	206-555-6543	Seattle
Joe	987-65-4321	908-555-2121	Westfield

One person may have multiple phones, but lives in only one city

Primary key is thus (SSN, PhoneNumber)

What is the problem with this schema?

Relational Schema Design

Name	SSN	PhoneNumber	City
Fred	123-45-6789	206-555-1234	Seattle
Fred	123-45-6789	206-555-6543	Seattle
Joe	987-65-4321	908-555-2121	Westfield

Anomalies:

- Redundancy = repeat data
- Update anomalies = what if Fred moves to “Bellevue”?
- Deletion anomalies = what if Joe deletes his phone number?

Relation Decomposition

Break the relation into two:

Name	SSN	PhoneNumber	City
Fred	123-45-6789	206-555-1234	Seattle
Fred	123-45-6789	206-555-6543	Seattle
Joe	987-65-4321	908-555-2121	Westfield

Name	<u>SSN</u>	City
Fred	123-45-6789	Seattle
Joe	987-65-4321	Westfield

<u>SSN</u>	<u>PhoneNumber</u>
123-45-6789	206-555-1234
123-45-6789	206-555-6543
987-65-4321	908-555-2121

Anomalies have gone:

- No more repeated data
- Easy to move Fred to “Bellevue” (how ?)
- Easy to delete all Joe’s phone numbers (how ?)

Relational Schema Design (or Logical Design)

How do we do this systematically?

- Start with some relational schema
- Find out its ***functional dependencies*** (FDs)
- Use FDs to ***normalize*** the relational schema

Functional Dependencies (FDs)

Definition

If two tuples agree on the attributes

$A_1, A_2, \ldots, A_n$

then they must also agree on the attributes

$B_1, B_2, \ldots, B_m$

Formally:

$A_1 \ldots A_n$ **determines** $B_1 .. B_m$

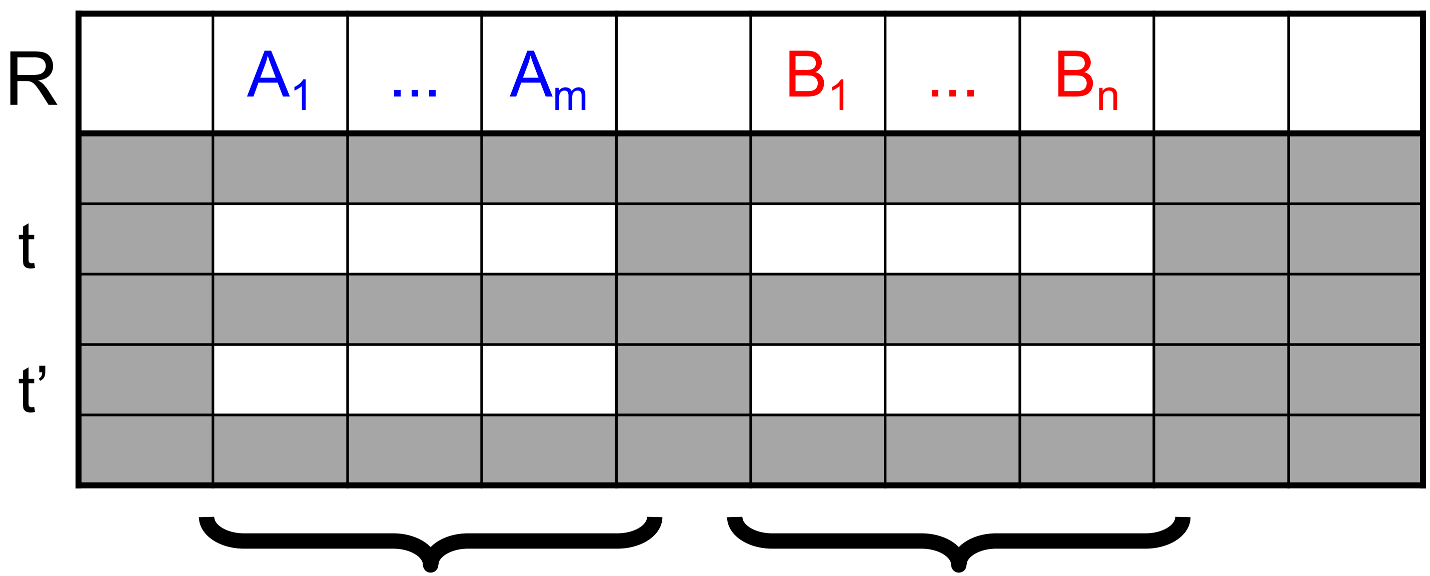
$A_1, A_2, \ldots, A_n \rightarrow B_1, B_2, \ldots, B_m$

Functional Dependencies (FDs)

Definition $A_1, \ldots, A_m \rightarrow B_1, \ldots, B_n$ **holds** in R if:

$\forall t, t' \in R,$

$(t.A_1 = t'.A_1 \wedge \ldots \wedge t.A_m = t'.A_m \rightarrow t.B_1 = t'.B_1 \wedge \ldots \wedge t.B_n = t'.B_n)$



if t, t' agree here then t, t' agree here

Example

An FD holds, or does not hold on an instance:

EmpID	Name	Phone	Position
E0045	Smith	1234	Clerk
E3542	Mike	9876	Salesrep
E1111	Smith	9876	Salesrep
E9999	Mary	1234	Lawyer

EmpID → Name, Phone, Position

Position → Phone

but not Phone → Position

Example

EmpID	Name	Phone	Position
E0045	Smith	1234	Clerk
E3542	Mike	9876 ←	Salesrep
E1111	Smith	9876 ←	Salesrep
E9999	Mary	1234	Lawyer

Position → Phone

Example

EmpID	Name	Phone	Position
E0045	Smith	1234 →	Clerk
E3542	Mike	9876	Salesrep
E1111	Smith	9876	Salesrep
E9999	Mary	1234 →	Lawyer

But not Phone → Position

Example

name → color
category → department
color, category → price
department → price

name	category	color	department	price
Gizmo	Gadget	Green	Toys	49
Tweaker	Gadget	Red	Toys	49
Gizmo	Stationary	Green	Office-supp.	59

Which FD's hold?

Buzzwords

- FD **holds** or **does not hold** on an instance
- If we can be sure that *every instance of R* will be one in which a given FD is true, then we say that **R satisfies the FD**
- If we say that R satisfies an FD, we are **stating a constraint on R**

Why bother with FDs?

Name	SSN	PhoneNumber	City
Fred	123-45-6789	206-555-1234	Seattle
Fred	123-45-6789	206-555-6543	Seattle
Joe	987-65-4321	908-555-2121	Westfield

Anomalies:

- Redundancy = repeat data
- Update anomalies = what if Fred moves to "Bellevue"?
- Deletion anomalies = what if Joe deletes his phone number?

An Interesting Observation

If all these FDs are true:

name → color
category → department
color, category → price

Then this FD also holds:

name, category → price

If we find out from application domain that a relation satisfies some FDs, it doesn't mean that we found all the FDs that it satisfies!
There could be more FDs implied by the ones we have.

Closure of a set of Attributes

Given a set of attributes $A_1, \ldots, A_n$

The **closure** is the set of attributes B, notated $\{A_1, \ldots, A_n\}^+$,
s.t. $A_1, \ldots, A_n \rightarrow B$

Example:

1. name → color
2. category → department
3. color, category → price

Closures:

name$^+$ = {name, color}

{name, category}$^+$ = {name, category, color, department, price}

color$^+$ = {color}

Closure Algorithm

X={A1, …, An}.

Repeat until X doesn't change **do**:
  if $B_1, \ldots, B_n \rightarrow C$ is a FD **and**
  $B_1, \ldots, B_n$ are all in X
  then add C to X.

Example:

1. name → color
2. category → department
3. color, category → price

{name, category}$^+$ =
  { name, category, color, department, price }

Hence: name, category → color, department, price

Why do we care?

- The closure allows us to compute all FDs implied by a given FD; Here is how:

- To check if the FD implies $A \rightarrow B$
  - Compute A^+
  - Check if $B \subseteq A^+$

Example

In class:

R(A,B,C,D,E,F)

$A, B \rightarrow C$
$A, D \rightarrow E$
$B \rightarrow D$
$A, F \rightarrow B$

Compute $\{A,B\}^+$ X = {A, B, }

Compute $\{A, F\}^+$ X = {A, F, }

Example

In class:

R(A,B,C,D,E,F)

A, B	$\rightarrow$	C
A, D	$\rightarrow$	E
B	$\rightarrow$	D
A, F	$\rightarrow$	B

Compute $\{A,B\}^+$ X = {A, B, C, D, E }

Compute $\{A, F\}^+$ X = {A, F, }

Example

In class:

R(A,B,C,D,E,F)

A, B	$\rightarrow$	C
A, D	$\rightarrow$	E
B	$\rightarrow$	D
A, F	$\rightarrow$	B

Compute $\{A,B\}^+$ X = {A, B, C, D, E }

Compute $\{A, F\}^+$ X = {A, F, B, C, D, E }

Example

In class:

R(A,B,C,D,E,F)

$A, B \rightarrow C$
$A, D \rightarrow E$
$B \rightarrow D$
$A, F \rightarrow B$

Compute $\{A,B\}^+$ X = {A, B, C, D, E }

Compute $\{A, F\}^+$ X = {A, F, B, C, D, E }

What is the key of R?

Practice at Home

Find all FD's implied by:

$$A, B \rightarrow C$$
$$A, D \rightarrow B$$
$$B \rightarrow D$$

Practice at Home

Find all FD's implied by:

$A, B \rightarrow C$
$A, D \rightarrow B$
$B \rightarrow D$

Step 1: Compute X^+, for every X:

$A^+ = A$, $B^+ = BD$, $C^+ = C$, $D^+ = D$

$AB^+ = ABCD$, $AC^+ = AC$, $AD^+ = ABCD$,
$BC^+ = BCD$, $BD^+ = BD$, $CD^+ = CD$

$ABC^+ = ABD^+ = ACD^+ = ABCD$ (no need to compute– why ?)

$BCD^+ = BCD$, $ABCD^+ = ABCD$

Practice at Home

Find all FD's implied by:

$A, B \rightarrow C$
$A, D \rightarrow B$
$B \rightarrow D$

Step 1: Compute X^+, for every X:

$A^+ = A$, $B^+ = BD$, $C^+ = C$, $D^+ = D$

$AB^+ = ABCD$, $AC^+ = AC$, $AD^+ = ABCD$,
$BC^+ = BCD$, $BD^+ = BD$, $CD^+ = CD$

$ABC^+ = ABD^+ = ACD^+ = ABCD$ (no need to compute– why ?)

$BCD^+ = BCD$, $ABCD^+ = ABCD$

Step 2: Enumerate all FD's $X \rightarrow Y$, s.t. $Y \subseteq X^+$ and $X \cap Y = \emptyset$:

$AB \rightarrow CD$, $AD \rightarrow BC$, $ABC \rightarrow D$, $ABD \rightarrow C$, $ACD \rightarrow B$

Keys

- A **superkey** is a set of attributes $A_1, ..., A_n$ s.t. for any other attribute B, we have $A_1, ..., A_n \rightarrow B$

- A **key** is a minimal superkey
  - A superkey and for which no subset is a superkey

Computing (Super)Keys

- For all sets X, compute X^+
- If X^+ = [all attributes], then X is a superkey
- Try reducing to the minimal X's to get the key

Example

Product(name, price, category, color)

name, category → price
category → color

What is the key ?

Example

Product(name, price, category, color)

name, category $\rightarrow$ price
category $\rightarrow$ color

What is the key ?

(name, category) + = { name, category, price, color }

Hence (name, category) is a key

Key or Keys ?

We can we have more than one key!

What are the keys here ?

$A \rightarrow B$
$B \rightarrow C$
$C \rightarrow A$

$A \rightarrow BC$
$B \rightarrow AC$

Eliminating Anomalies

Main idea:

- $X \rightarrow A$ is OK if X is a (super)key
- $X \rightarrow A$ is not OK otherwise
  - Need to decompose the table, but how?

Boyce-Codd Normal Form

There are no "bad" FDs:

Definition. A relation R is in BCNF if:

Whenever $X \rightarrow B$ is a non-trivial dependency, then X is a superkey.

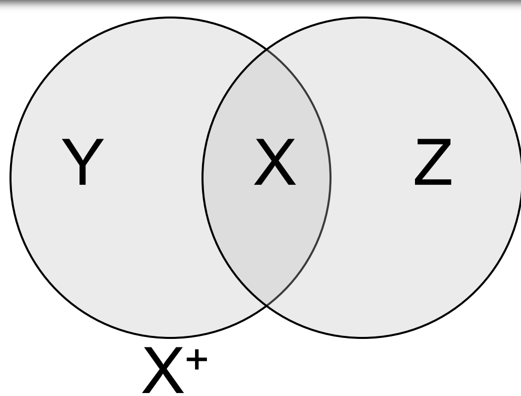
Equivalently:

Definition. A relation R is in BCNF if:

$\forall$ X, either $X^+ = X$ (i.e., X is not in any FDs)
or X^+ = [all attributes] (computed using FDs)

BCNF Decomposition Algorithm

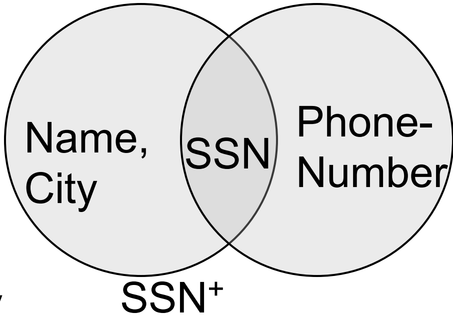
Normalize(R)
  find X s.t.: $X \neq X^+$ and $X^+ \neq$ [all attributes]
  <u>if</u> (not found) **<u>then</u>** "R is in BCNF"
  <u>let</u> $Y = X^+ - X$; $Z =$ [all attributes] $- X^+$
  decompose R into R1($X \cup Y$) and R2($X \cup Z$)
  Normalize(R1); Normalize(R2);



Example

Name	SSN	PhoneNumber	City
Fred	123-45-6789	206-555-1234	Seattle
Fred	123-45-6789	206-555-6543	Seattle
Joe	987-65-4321	908-555-2121	Westfield
Joe	987-65-4321	908-555-1234	Westfield

$\boxed{\text{SSN} \rightarrow \text{Name, City}}$



The only key is: {SSN, PhoneNumber}

Hence SSN → Name, City is a "bad" dependency

In other words:

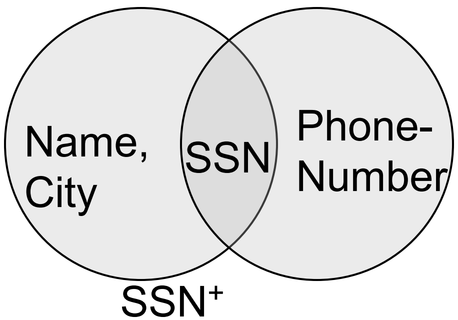
SSN+ = SSN, Name, City and is neither SSN nor All Attributes

Example BCNF Decomposition

Name	SSN	City
Fred	123-45-6789	Seattle
Joe	987-65-4321	Westfield

SSN	PhoneNumber
123-45-6789	206-555-1234
123-45-6789	206-555-6543
987-65-4321	908-555-2121
987-65-4321	908-555-1234

SSN → Name, City



Let's check anomalies:

- Redundancy ?
- Update ?
- Delete ?

Find X s.t.: $X \neq X^+$ and $X^+ \neq$ [all attributes]

Example BCNF Decomposition

Person(name, SSN, age, hairColor, phoneNumber)

SSN → name, age

age → hairColor

Find X s.t.: $X \neq X^+$ and $X^+ \neq$ [all attributes]

Example BCNF Decomposition

Person(name, SSN, age, hairColor, phoneNumber)

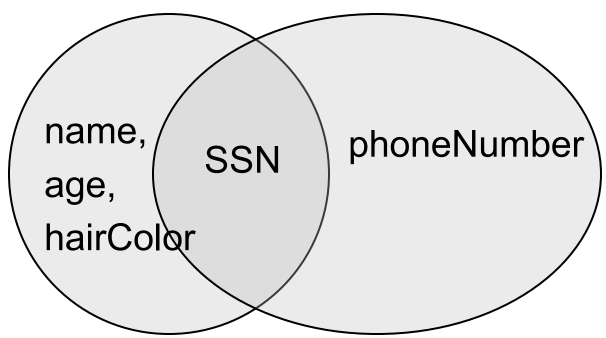
SSN → name, age

age → hairColor

Iteration 1: Person: SSN+ = SSN, name, age, hairColor

Decompose into: P(SSN, name, age, hairColor)

Phone(SSN, phoneNumber)



Find X s.t.: X $\neq X^+$ and $X^+ \neq$ [all attributes]

Example BCNF Decomposition

Person(name, SSN, age, hairColor, phoneNumber)

SSN → name, age

age → hairColor

What are the keys ?

Iteration 1: Person: SSN+ = SSN, name, age, hairColor

Decompose into: P(SSN, name, age, hairColor)

Phone(SSN, phoneNumber)

Iteration 2: P: age+ = age, hairColor

Decompose: People(SSN, name, age)

Hair(age, hairColor)

Phone(SSN, phoneNumber)

Find X s.t.: X $\neq X^+$ and $X^+ \neq$ [all attributes]

Example BCNF Decomposition

Person(name, SSN, age, hairColor, phoneNumber)

SSN → name, age

age → hairColor

Note the keys!

Iteration 1: Person: SSN+ = SSN, name, age, hairColor

Decompose into: P(SSN, name, age, hairColor)
Phone(SSN, phoneNumber)

Iteration 2: P: age+ = age, hairColor

Decompose: People(SSN, name, age)
Hair(age, hairColor)
Phone(SSN, phoneNumber)

R(A,B,C,D)

A → B
B → C

Example: BCNF

R(A,B,C,D)

R(A,B,C,D)

$A \rightarrow B$
$B \rightarrow C$

Example: BCNF

Recall: find X s.t.
$X \subsetneq X^+ \subsetneq$ [all-attrs]

R(A,B,C,D)

R(A,B,C,D)

$A \rightarrow B$
$B \rightarrow C$

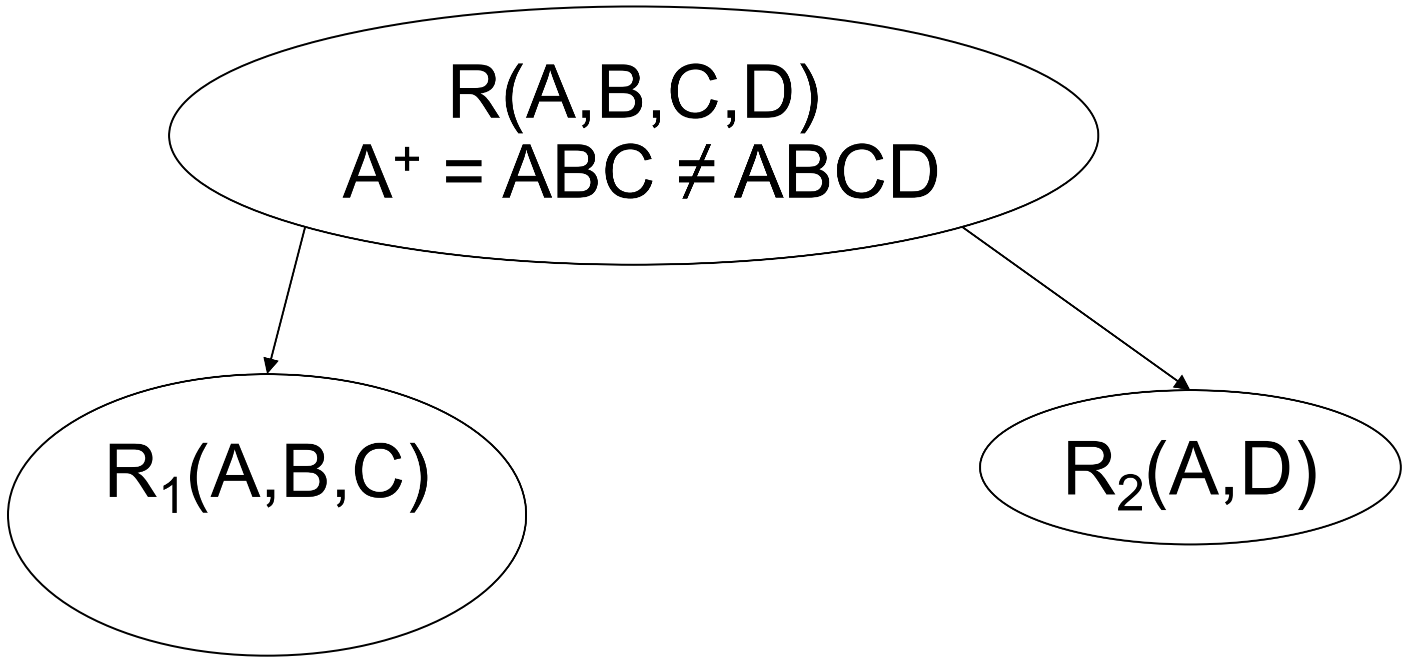
Example: BCNF

R(A,B,C,D)
$A^+ = ABC \neq ABCD$

R(A,B,C,D)

$A \rightarrow B$
$B \rightarrow C$

Example: BCNF



R(A,B,C,D)

$A \rightarrow B$
$B \rightarrow C$

Example: BCNF

R(A,B,C,D)
$A^+ = ABC \neq ABCD$

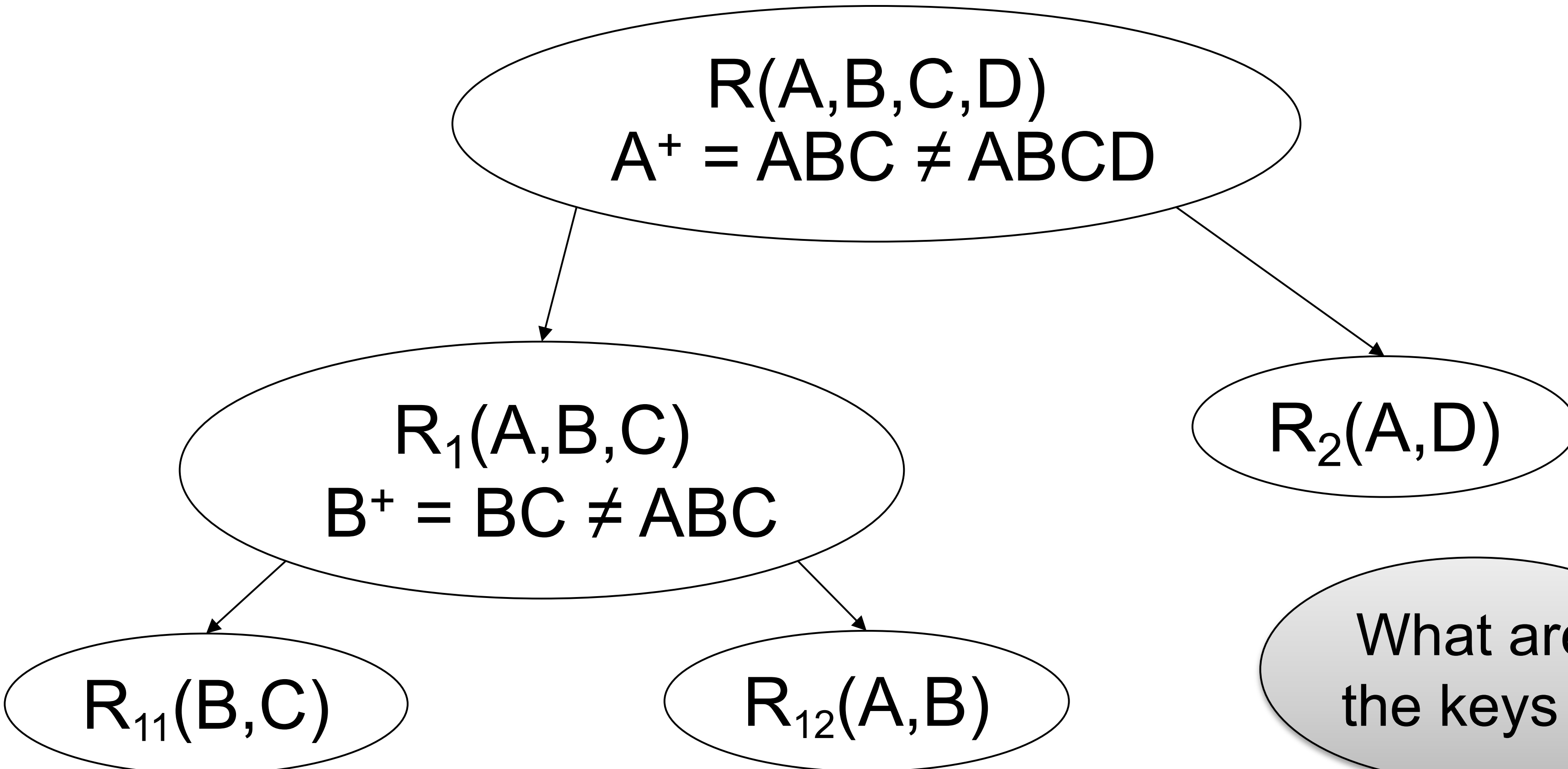
$R_1(A,B,C)$
$B^+ = BC \neq ABC$

$R_2(A,D)$

R(A,B,C,D)

$A \rightarrow B$
$B \rightarrow C$

Example: BCNF



What are the keys ?

What happens if in R we first pick B^+ ? Or AB^+ ?

Decompositions in General

$R(A_1, \ldots, A_n, B_1, \ldots, B_m, C_1, \ldots, C_p)$

$S_1(A_1, \ldots, A_n, B_1, \ldots, B_m)$ $S_2(A_1, \ldots, A_n, C_1, \ldots, C_p)$

S_1 = projection of R on $A_1, \ldots, A_n, B_1, \ldots, B_m$
S_2 = projection of R on $A_1, \ldots, A_n, C_1, \ldots, C_p$

Lossless Decomposition

Name	Price	Category
Gizmo	19.99	Gadget
OneClick	24.99	Camera
Gizmo	19.99	Camera

Name	Price
Gizmo	19.99
OneClick	24.99
~~Gizmo~~	~~19.99~~

Name	Category
Gizmo	Gadget
OneClick	Camera
Gizmo	Camera

Lossy Decomposition

What is lossy here?

Name	Price	Category
Gizmo	19.99	Gadget
OneClick	24.99	Camera
Gizmo	19.99	Camera

Name	Category
Gizmo	Gadget
OneClick	Camera
Gizmo	Camera

Price	Category
19.99	Gadget
24.99	Camera
19.99	Camera

Lossy Decomposition

Name	Price	Category
Gizmo	19.99	Gadget
OneClick	24.99	Camera
Gizmo	19.99	Camera

Name	Category
Gizmo	Gadget
OneClick	Camera
Gizmo	Camera

Price	Category
19.99	Gadget
24.99	Camera
19.99	Camera

Lossy Decomposition

Name	Price	Category
Gizmo	19.99	Gadget
OneClick	24.99	Camera
Gizmo	19.99	Camera

Name	Category
Gizmo	Gadget
OneClick	Camera
Gizmo	Camera

Price	Category
19.99	Gadget
24.99	Camera
19.99	Camera

Decomposition in General

$$R(A_1, \ldots, A_n, B_1, \ldots, B_m, C_1, \ldots, C_p)$$

$$S_1(A_1, \ldots, A_n, B_1, \ldots, B_m) \qquad S_2(A_1, \ldots, A_n, C_1, \ldots, C_p)$$

Let: S_1 = projection of R on $A_1, \ldots, A_n, B_1, \ldots, B_m$
S_2 = projection of R on $A_1, \ldots, A_n, C_1, \ldots, C_p$

The decomposition is called *lossless* if $R = S_1 \bowtie S_2$

Fact: If $A_1, \ldots, A_n \rightarrow B_1, \ldots, B_m$ then the decomposition is lossless

It follows that every BCNF decomposition is lossless

Testing for Lossless Join

If we decompose R into $\Pi_{S1}(R)$, $\Pi_{S2}(R)$, $\Pi_{S3}(R)$, …
Is it true that $S1 \bowtie S2 \bowtie S3 \bowtie \ldots = R$?

To check "=" we need to check "$\subseteq$" and "$\supseteq$"

$R \subseteq S1 \bowtie S2 \bowtie S3 \bowtie \ldots$ always holds (why?)

$R \supseteq S1 \bowtie S2 \bowtie S3 \bowtie \ldots$ neet to check

Example from textbook Ch. 3.4.2

The Chase Test for Lossless Join

$R(A,B,C,D) = S1(A,D) \bowtie S2(A,C) \bowtie S3(B,C,D)$
R satisfies: $A \rightarrow B$, $B \rightarrow C$, $CD \rightarrow A$

Lossless?

$S1 = \Pi_{AD}(R)$, $S2 = \Pi_{AC}(R)$, $S3 = \Pi_{BCD}(R)$

Example from textbook Ch. 3.4.2

The Chase Test for Lossless Join

$R(A,B,C,D) = S1(A,D) \bowtie S2(A,C) \bowtie S3(B,C,D)$
R satisfies: $A \rightarrow B$, $B \rightarrow C$, $CD \rightarrow A$

Lossless?

$S1 = \Pi_{AD}(R)$, $S2 = \Pi_{AC}(R)$, $S3 = \Pi_{BCD}(R)$

$R \subseteq S1 \bowtie S2 \bowtie S3$

To check: $R \supseteq S1 \bowtie S2 \bowtie S3$

Example from textbook Ch. 3.4.2

The Chase Test for Lossless Join

R(A,B,C,D) = S1(A,D) ⋈ S2(A,C) ⋈ S3(B,C,D)
R satisfies: A→B, B→C, CD→A

Lossless?

$S1 = \Pi_{AD}(R)$, $S2 = \Pi_{AC}(R)$, $S3 = \Pi_{BCD}(R)$

$R \subseteq S1 \bowtie S2 \bowtie S3$

To check: $R \supseteq S1 \bowtie S2 \bowtie S3$

Suppose $(a,b,c,d) \in S1 \bowtie S2 \bowtie S3$ Is it also in R?

Example from textbook Ch. 3.4.2

The Chase Test for Lossless Join

$R(A,B,C,D) = S1(A,D) \bowtie S2(A,C) \bowtie S3(B,C,D)$

R satisfies: $A \rightarrow B$, $B \rightarrow C$, $CD \rightarrow A$

Lossless?

$S1 = \Pi_{AD}(R)$, $S2 = \Pi_{AC}(R)$, $S3 = \Pi_{BCD}(R)$

$R \subseteq S1 \bowtie S2 \bowtie S3$

To check: $R \supseteq S1 \bowtie S2 \bowtie S3$

Suppose $(a,b,c,d) \in S1 \bowtie S2 \bowtie S3$ Is it also in R?

R must contain the following tuples:

A	B	C	D
a	b1	c1	d

Why ?

$(a,d) \in S1 = \Pi_{AD}(R)$

Example from textbook Ch. 3.4.2

The Chase Test for Lossless Join

R(A,B,C,D) = S1(A,D) ⋈ S2(A,C) ⋈ S3(B,C,D)
R satisfies: A→B, B→C, CD→A

Lossless?

S1 = Π_{AD}(R), S2 = Π_{AC}(R), S3 = Π_{BCD}(R)

R⊆ S1 ⋈ S2 ⋈ S3

To check: R ⊇ S1 ⋈ S2 ⋈ S3

Suppose (a,b,c,d) ∈ S1 ⋈ S2 ⋈ S3 Is it also in R?

R must contain the following tuples:

A	B	C	D
a	b1	c1	d
a	b2	c	d2

Why ?

(a,d) ∈S1 = Π_{AD}(R)

(a,c) ∈S2 = Π_{BD}(R)

Example from textbook Ch. 3.4.2

The Chase Test for Lossless Join

$R(A,B,C,D) = S1(A,D) \bowtie S2(A,C) \bowtie S3(B,C,D)$
R satisfies: $A \rightarrow B$, $B \rightarrow C$, $CD \rightarrow A$

Lossless?

$S1 = \Pi_{AD}(R)$, $S2 = \Pi_{AC}(R)$, $S3 = \Pi_{BCD}(R)$

$R \subseteq S1 \bowtie S2 \bowtie S3$

To check: $R \supseteq S1 \bowtie S2 \bowtie S3$

Suppose $(a,b,c,d) \in S1 \bowtie S2 \bowtie S3$ Is it also in R?

R must contain the following tuples:

A	B	C	D	Why ?
a	b1	c1	d	$(a,d) \in S1 = \Pi_{AD}(R)$
a	b2	c	d2	$(a,c) \in S2 = \Pi_{BD}(R)$
a3	b	c	d	$(b,c,d) \in S3 = \Pi_{BCD}(R)$

Example from textbook Ch. 3.4.2

The Chase Test for Lossless Join

R(A,B,C,D) = S1(A,D) ⋈ S2(A,C) ⋈ S3(B,C,D)
R satisfies: A→B, B→C, CD→A

Lossless?

S1 = Π_{AD}(R), S2 = Π_{AC}(R), S3 = Π_{BCD}(R)

R⊆ S1 ⋈ S2 ⋈ S3

To check: R ⊇ S1 ⋈ S2 ⋈ S3

Suppose (a,b,c,d) ∈ S1 ⋈ S2 ⋈ S3 Is it also in R?

R must contain the following tuples:

A	B	C	D	Why ?
a	b1	c1	d	(a,d) ∈S1 = Π_{AD}(R)
a	b2	c	d2	(a,c) ∈S2 = Π_{BD}(R)
a3	b	c	d	(b,c,d) ∈S3 = Π_{BCD}(R)

"Chase" them (apply FDs):

A→B

A	B	C	D
a	b1	c1	d
a	b1	c	d2
a3	b	c	d

Example from textbook Ch. 3.4.2

The Chase Test for Lossless Join

R(A,B,C,D) = S1(A,D) ⋈ S2(A,C) ⋈ S3(B,C,D)
R satisfies: A→B, B→C, CD→A

Lossless?

S1 = Π_{AD}(R), S2 = Π_{AC}(R), S3 = Π_{BCD}(R)

R⊆ S1 ⋈ S2 ⋈ S3

To check: R ⊇ S1 ⋈ S2 ⋈ S3

Suppose (a,b,c,d) ∈ S1 ⋈ S2 ⋈ S3 Is it also in R?

R must contain the following tuples:

A	B	C	D	Why ?
a	b1	c1	d	(a,d) ∈S1 = Π_{AD}(R)
a	b2	c	d2	(a,c) ∈S2 = Π_{BD}(R)
a3	b	c	d	(b,c,d) ∈S3 = Π_{BCD}(R)

"Chase" them (apply FDs):

A→B

A	B	C	D
a	b1	c1	d
a	b1	c	d2
a3	b	c	d

B→C

A	B	C	D
a	b1	c	d
a	b1	c	d2
a3	b	c	d

Example from textbook Ch. 3.4.2

The Chase Test for Lossless Join

$R(A,B,C,D) = S1(A,D) \bowtie S2(A,C) \bowtie S3(B,C,D)$
R satisfies: $A \rightarrow B$, $B \rightarrow C$, $CD \rightarrow A$

Lossless?

$S1 = \Pi_{AD}(R)$, $S2 = \Pi_{AC}(R)$, $S3 = \Pi_{BCD}(R)$

$R \subseteq S1 \bowtie S2 \bowtie S3$

To check: $R \supseteq S1 \bowtie S2 \bowtie S3$

Suppose $(a,b,c,d) \in S1 \bowtie S2 \bowtie S3$ Is it also in R?

R must contain the following tuples:

A	B	C	D
a	b1	c1	d
a	b2	c	d2
a3	b	c	d

Why ?

$(a,d) \in S1 = \Pi_{AD}(R)$

$(a,c) \in S2 = \Pi_{BD}(R)$

$(b,c,d) \in S3 = \Pi_{BCD}(R)$

"Chase" them (apply FDs):

$A \rightarrow B$

A	B	C	D
a	b1	c1	d
a	b1	c	d2
a3	b	c	d

$B \rightarrow C$

A	B	C	D
a	b1	c	d
a	b1	c	d2
a3	b	c	d

$CD \rightarrow A$

A	B	C	D
a	b1	c	d
a	b1	c	d2
a	b	c	d

Hence R contains (a,b,c,d)

Example from textbook Ch. 3.4.2

The Chase Test for Lossless Join

$R(A,B,C,D) = S1(A,D) \bowtie S2(A,C) \bowtie S3(B,C,D)$
R satisfies: $A \to B$, $B \to C$, $CD \to A$

Lossless?

$S1 = \Pi_{AD}(R)$, $S2 = \Pi_{AC}(R)$, $S3 = \Pi_{BCD}(R)$

$R \subseteq S1 \bowtie S2 \bowtie S3$

YES!

To check: $R \supseteq S1 \bowtie S2 \bowtie S3$

Suppose $(a,b,c,d) \in S1 \bowtie S2 \bowtie S3$ Is it also in R?

R must contain the following tuples:

A	B	C	D
a	b1	c1	d
a	b2	c	d2
a3	b	c	d

Why ?

$(a,d) \in S1 = \Pi_{AD}(R)$

$(a,c) \in S2 = \Pi_{BD}(R)$

$(b,c,d) \in S3 = \Pi_{BCD}(R)$

"Chase" them (apply FDs):

$A \to B$

A	B	C	D
a	b1	c1	d
a	b1	c	d2
a3	b	c	d

$B \to C$

A	B	C	D
a	b1	c	d
a	b1	c	d2
a3	b	c	d

$CD \to A$

A	B	C	D
a	b1	c	d
a	b1	c	d2
a	b	c	d

Hence R contains (a,b,c,d)

Schema Refinements = Normal Forms

- 1st Normal Form = all tables are flat
- 2nd Normal Form = obsolete
- Boyce Codd Normal Form = no bad FDs
- 3rd Normal Form = see book
  - BCNF removes anomalies, but my lose some FDs (see book 3.4.4)
  - 3NF preserves all FD's, but may still have some anomalies

Conclusion

- E/R diagrams are means to structurally visualize and design relational schemas
- Normalization is a principled way of converting schemas into a form that avoid such redundancies.
- BCNF and 3NF are the most widely used normalized form in practice