

Fundamentals of Database Systems

Unit 4: Conceptual Design
E/R Diagrams
Integrity Constraints
BCNF

(3 lectures)
174A - 21Wi

Fundamentals of Database Systems

E/R Diagrams

Class Overview

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

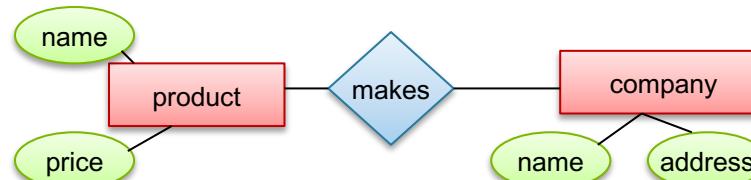
- The database will be in operation for a very long time (years). Updating the schema while in production is very expensive (why?)

Database Design

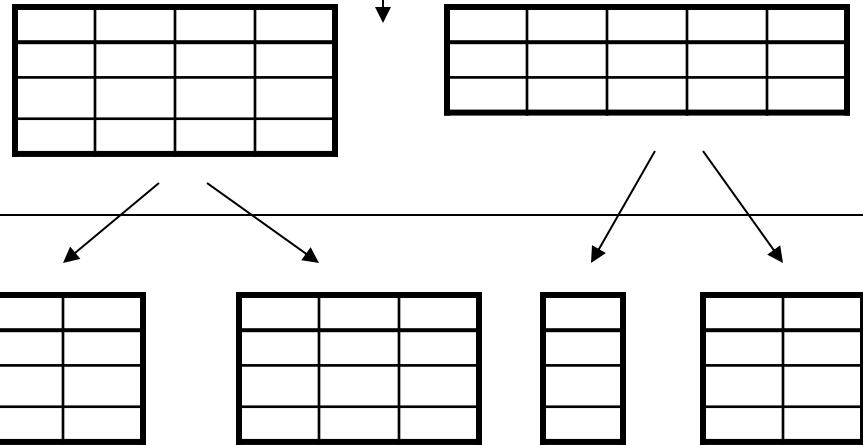
- Consider issues such as:
 - What entities to model
 - How entities are related
 - What constraints exist in the domain
- Several formalisms exists
 - We discuss E/R diagrams
 - UML, model-driven architecture
- Reading: Sec. 4.1-4.6

Database Design Process

Conceptual Model:



Relational Model:
Tables + constraints
And also functional dep.

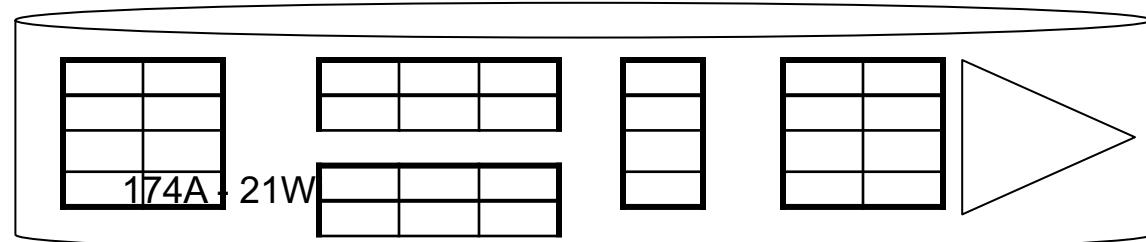


Normalization:
Eliminates anomalies

Conceptual Schema

Physical storage details

Physical Schema



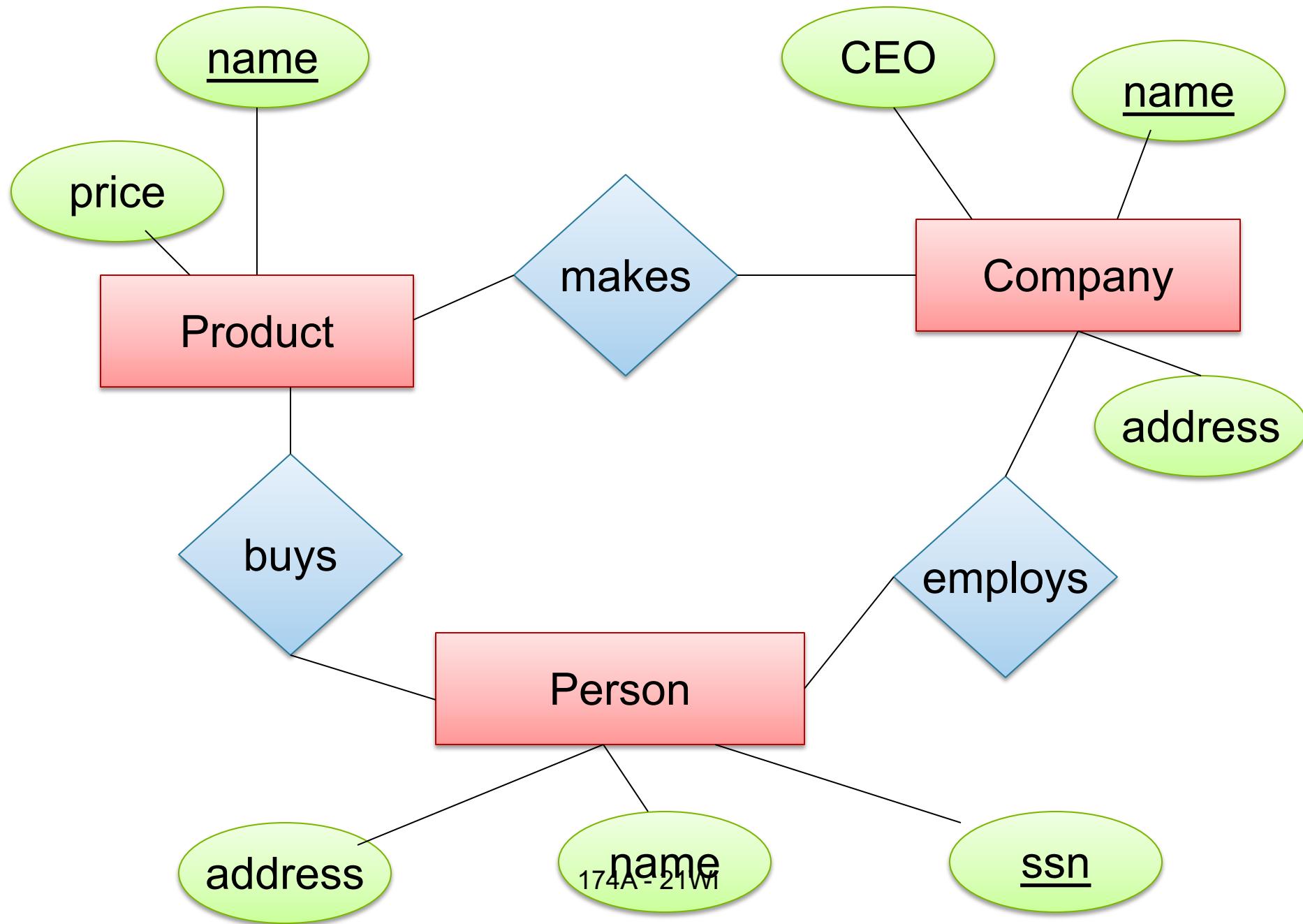
Entity / Relationship Diagrams

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

Product

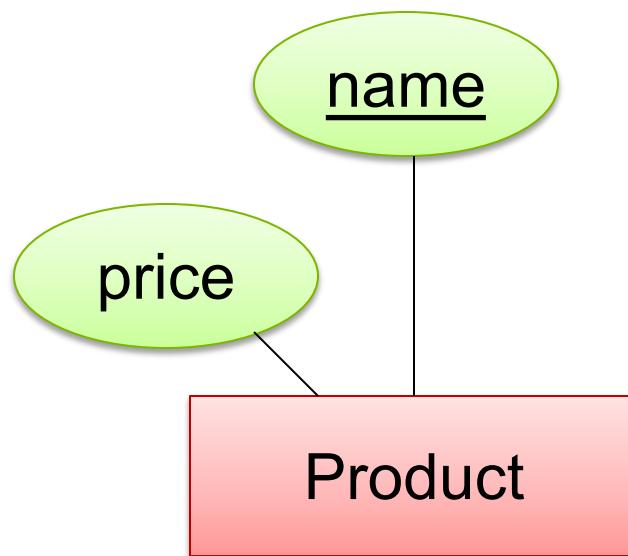
city

makes



Keys in E/R Diagrams

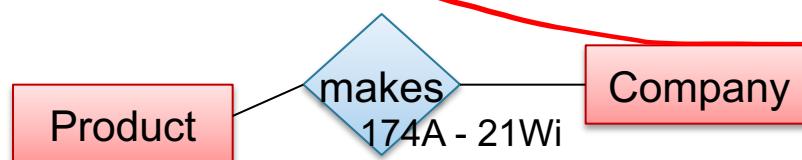
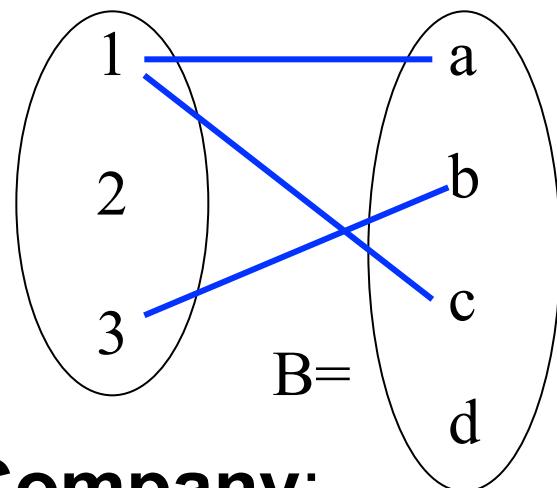
- Every entity set must have a key



What is a Relation ?

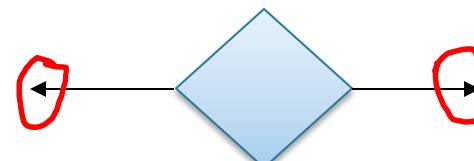
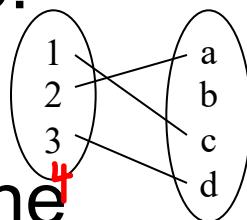
- A mathematical definition:
 - if A, B are sets, then a relation R is a subset of $A \times B$
- $A=\{1,2,3\}, \quad B=\{a,b,c,d\},$
$$A \times B = \{(1,a), (1,b), (1,c), (1,d), (2,a), (2,b), (2,c), (2,d), (3,a), (3,b), (3,c), (3,d)\}$$

$$R = \{(1,a), (1,c), (3,b)\}$$
- **makes** is a subset of **Product × Company**:

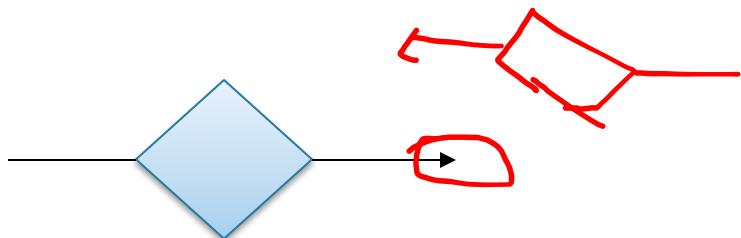
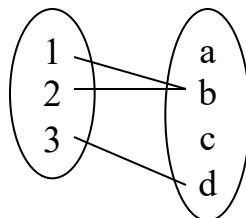


Multiplicity of E/R Relations

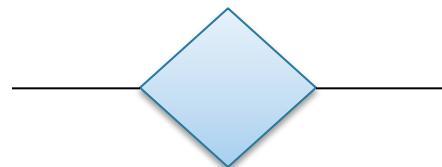
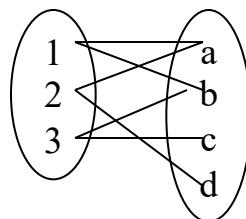
- one-one:

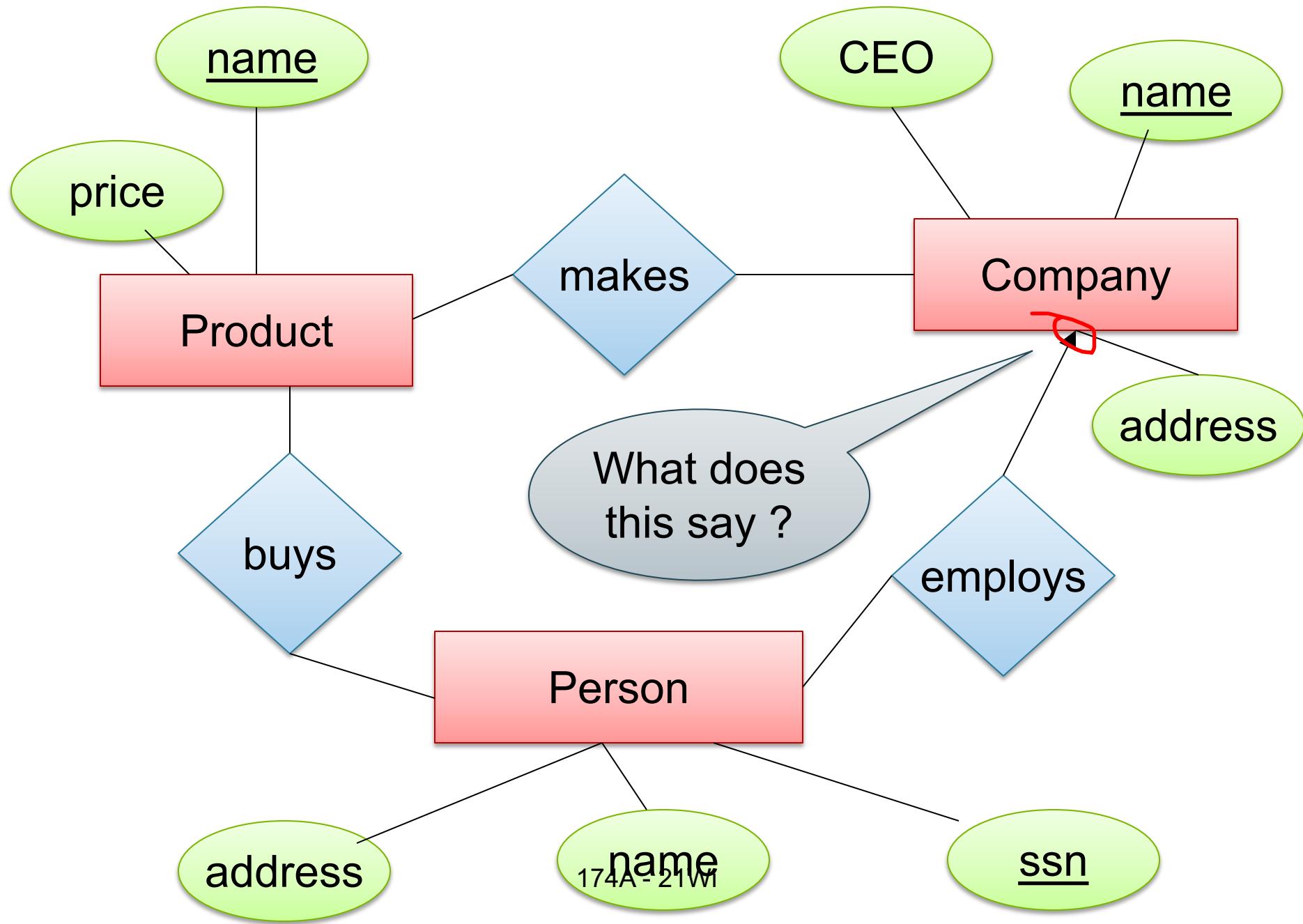


- many-one

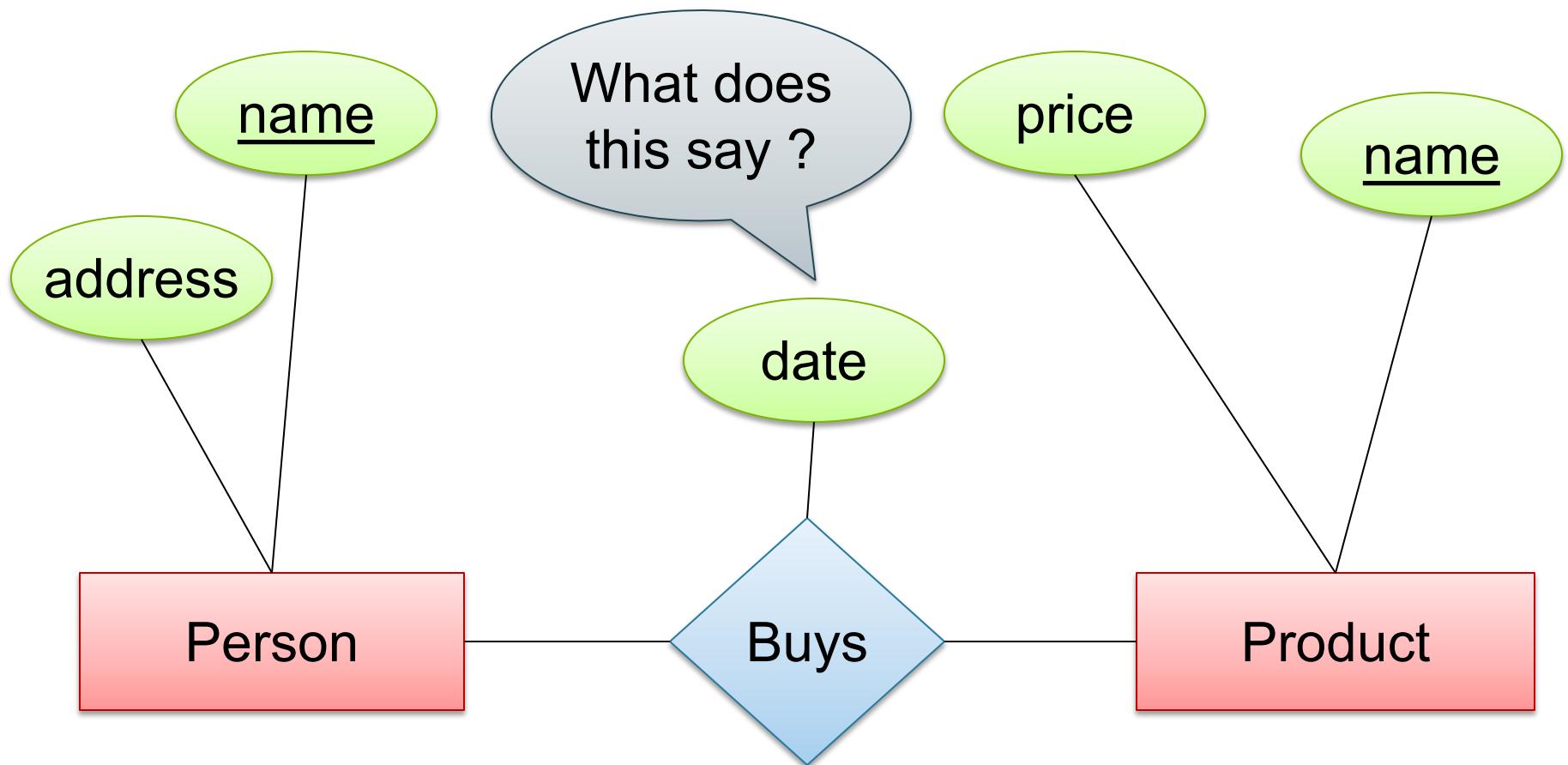


- many-many



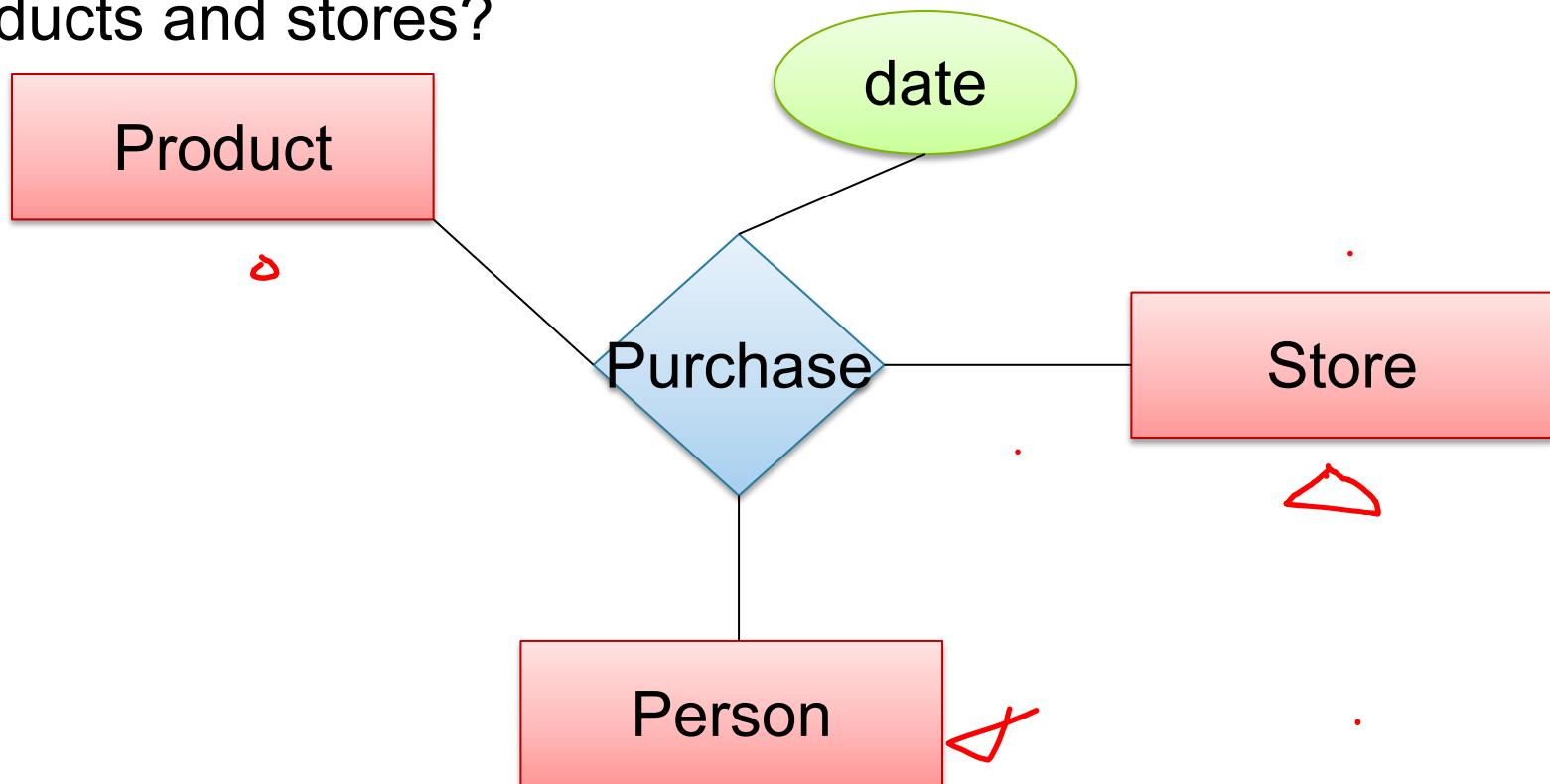


Attributes on Relationships



Multi-way Relationships

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

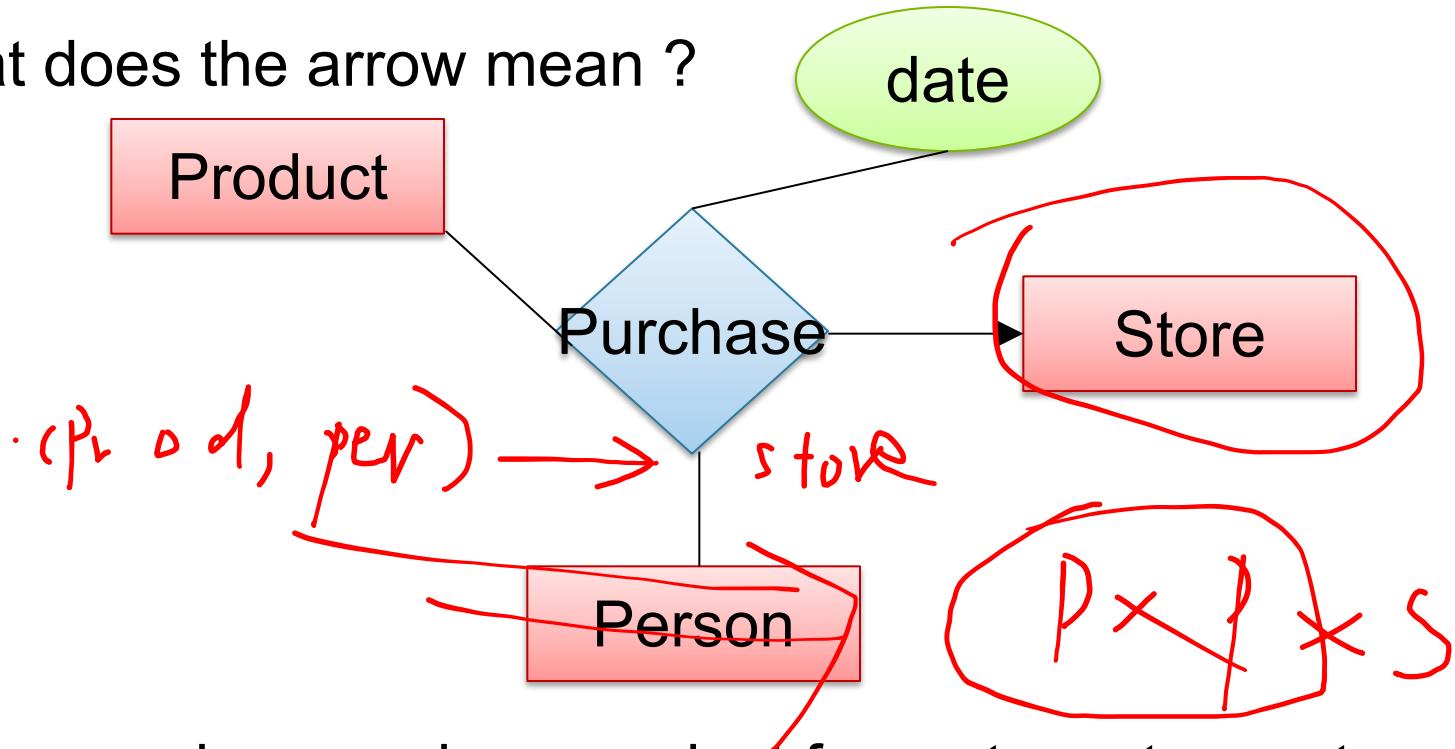


Can still model as a mathematical set (How?)

As a set of triples $\subseteq \text{Product} \times^2 \text{Person} \times \text{Store}$

Arrows in Multiway Relationships

Q: What does the arrow mean ?

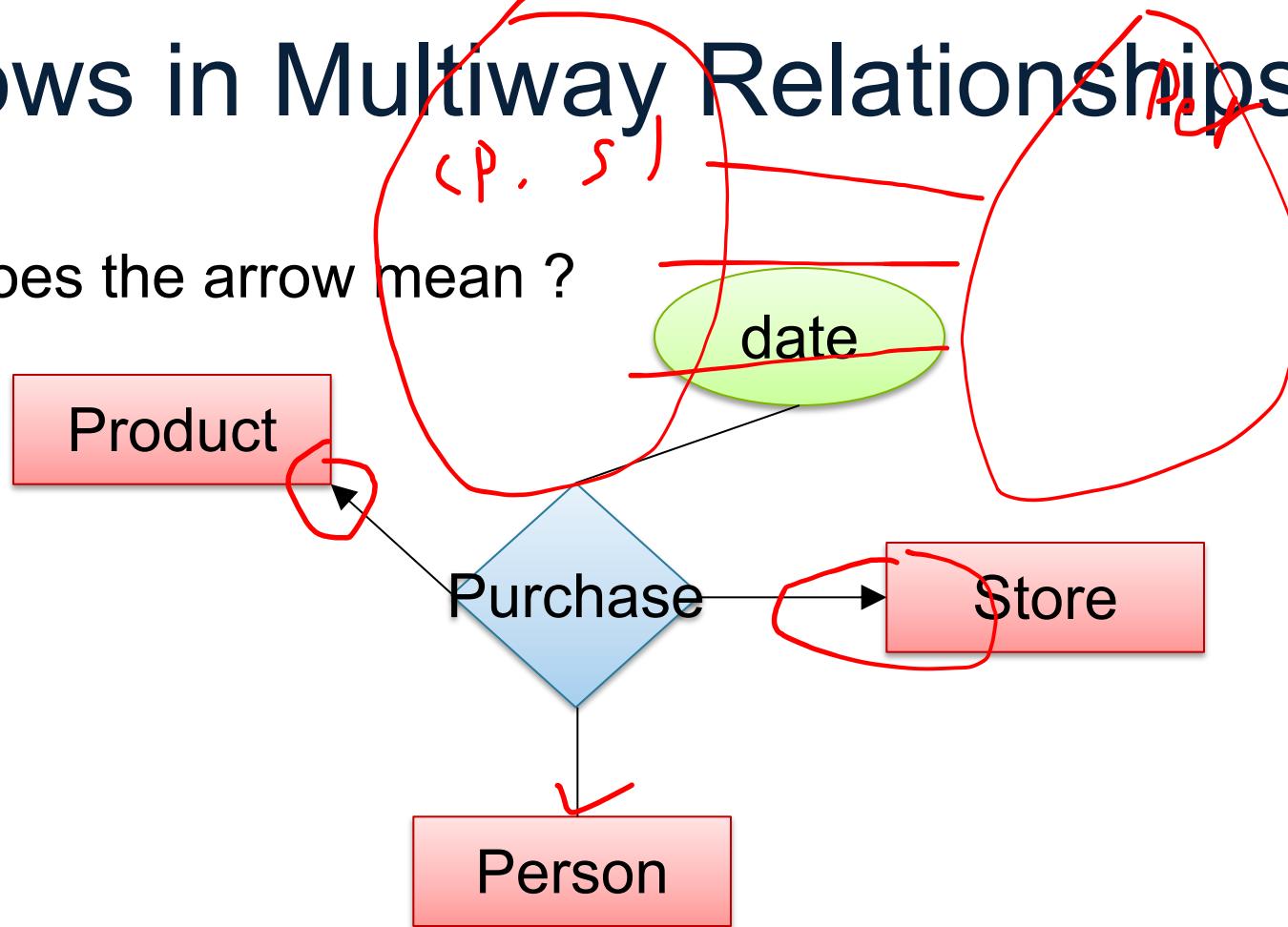


A: Any person buys a given product from at most one store

[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]

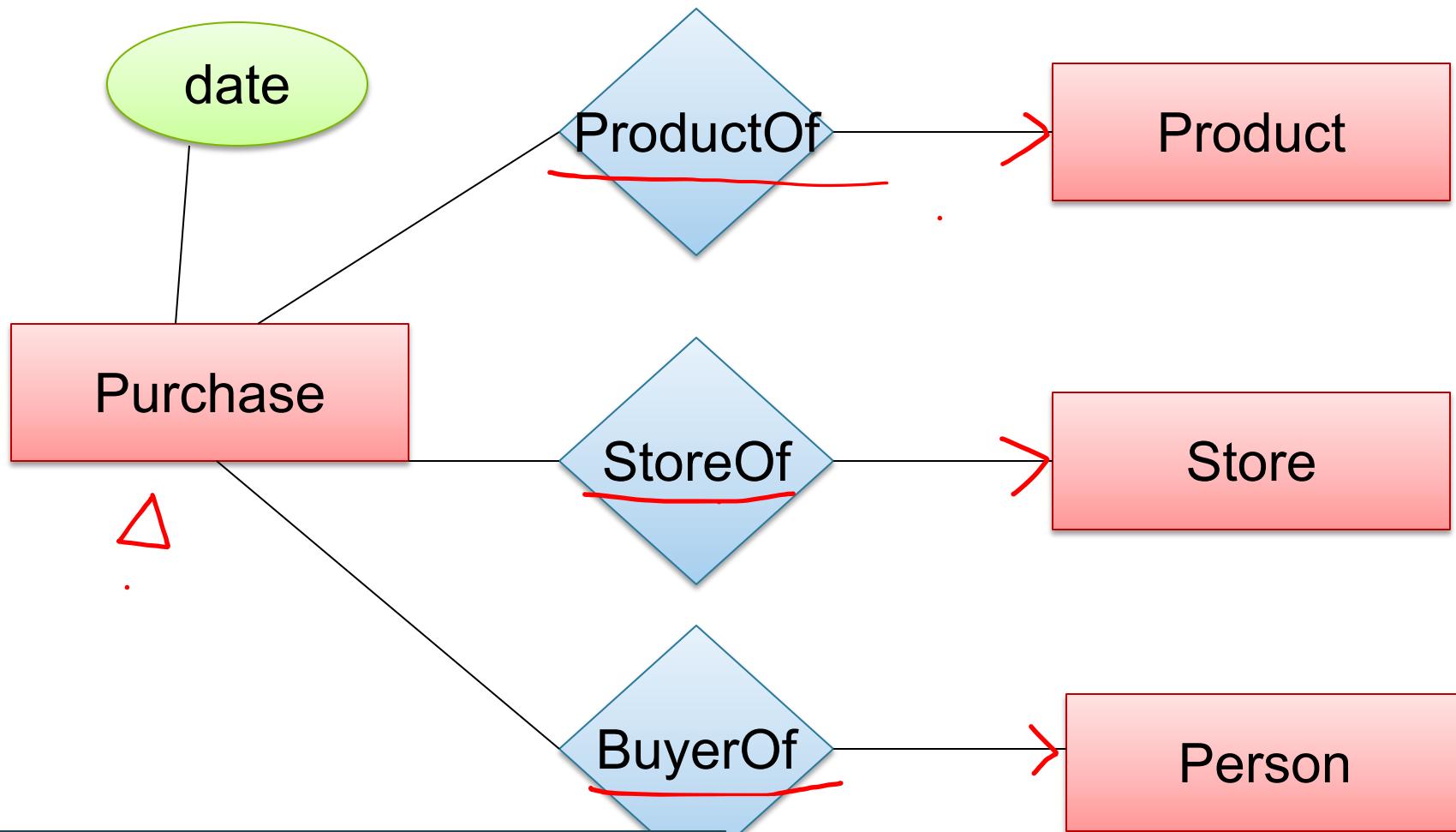
Arrows in Multiway Relationships

Q: What does the arrow mean ?



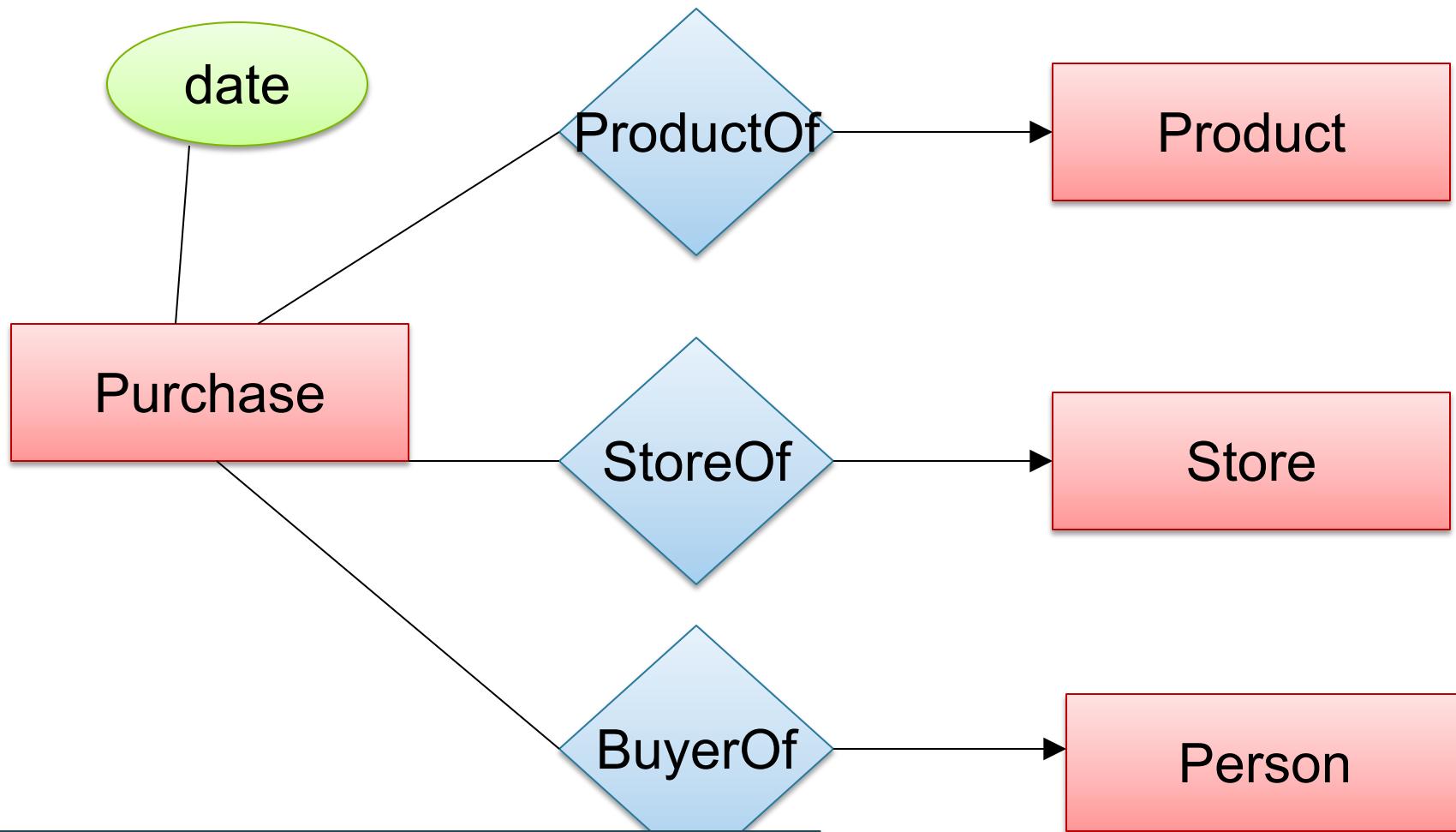
A: Any person buys a given product from at most one store
AND every store sells to every person at most one product

Converting Multi-way Relationships to Binary



Arrows go in which direction? ^{174A - 21Wi}

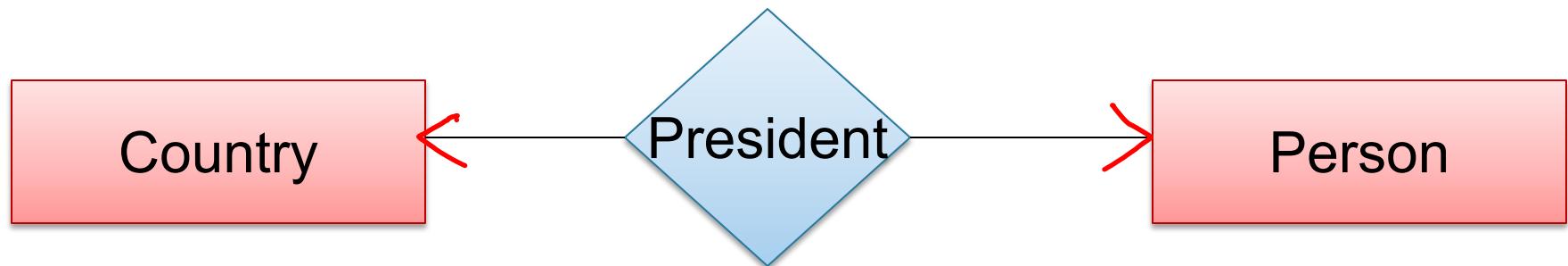
Converting Multi-way Relationships to Binary



174A - 21Wi
Make sure you understand why!

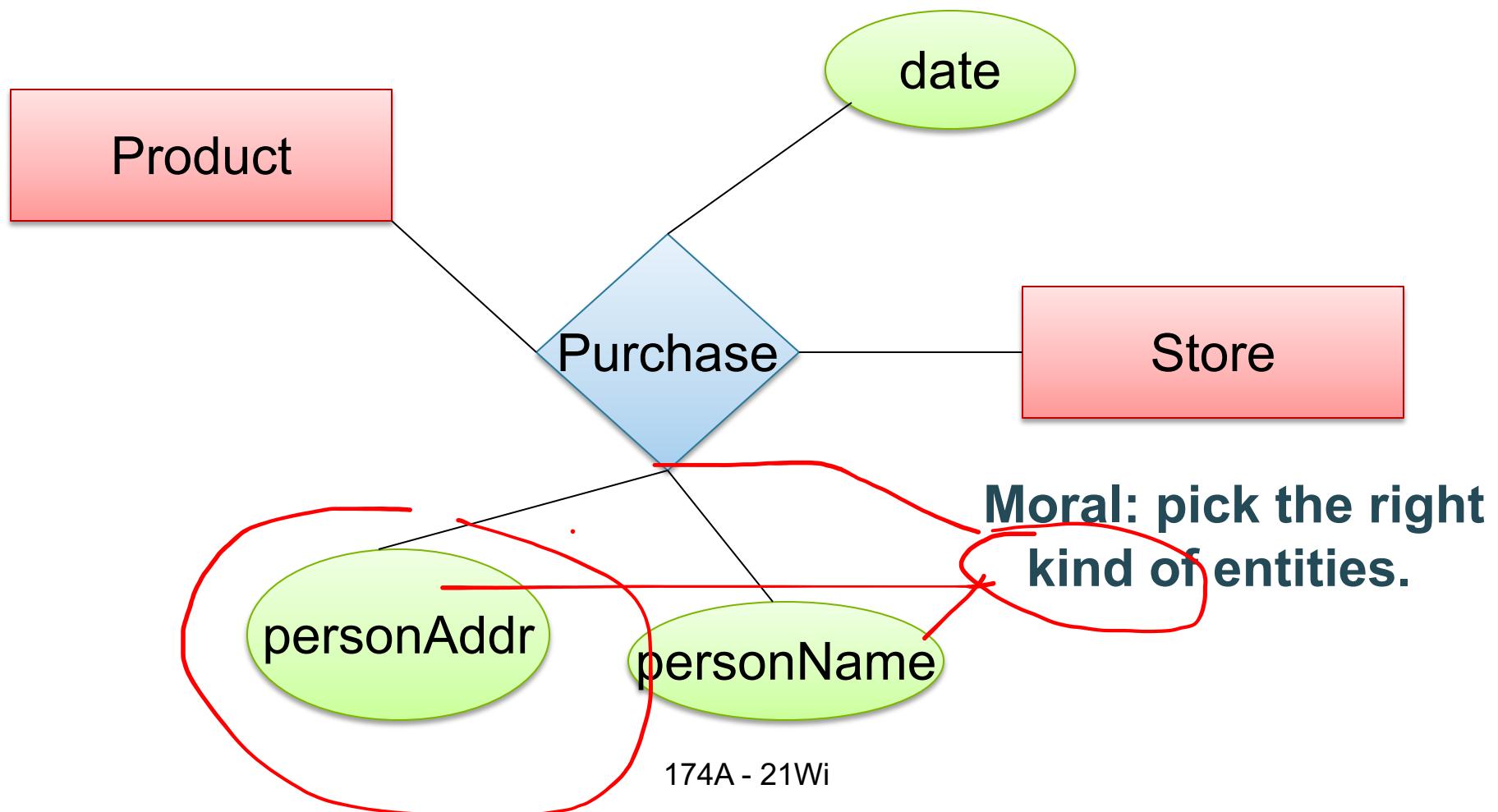
3. Design Principles

What's wrong?

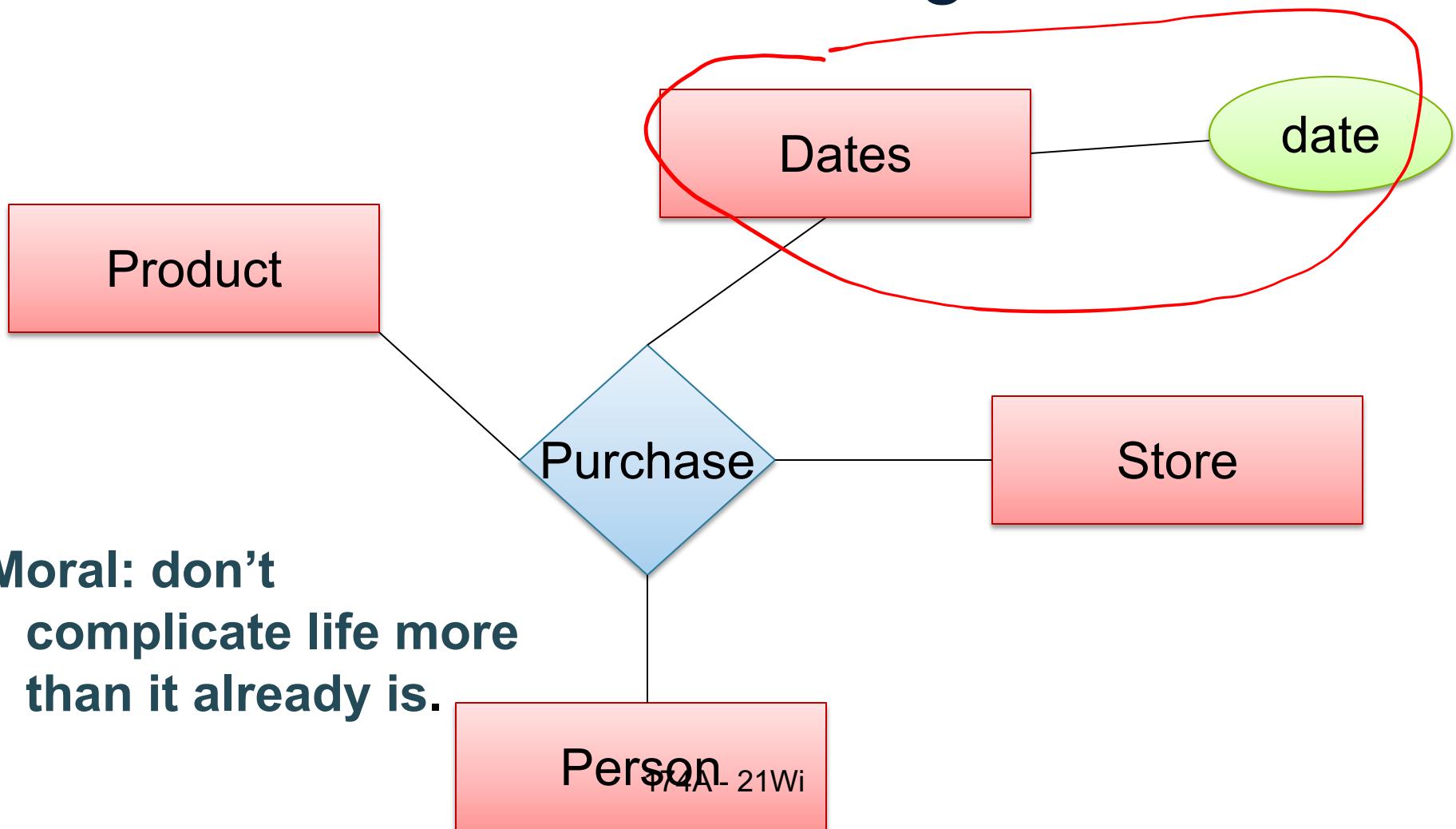


Moral: Be faithful to the specifications of the application!

Design Principles: What's Wrong?



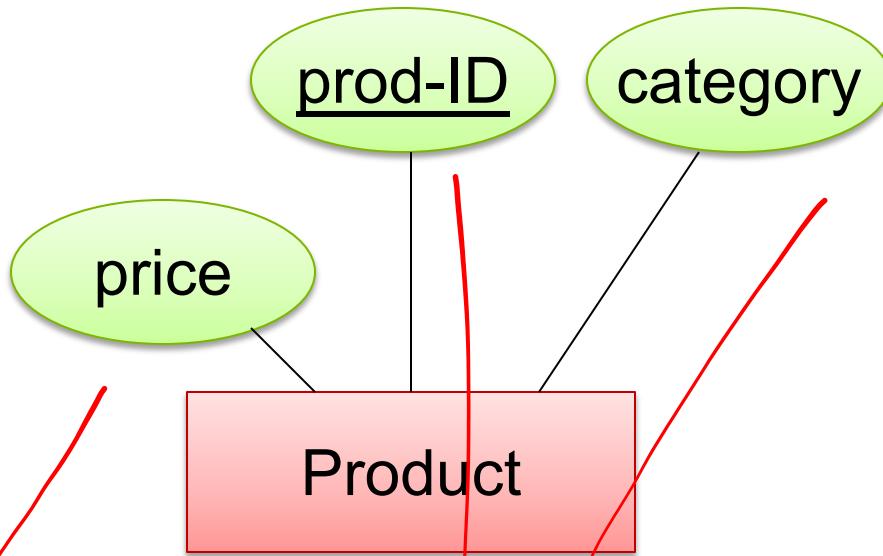
Design Principles: What's Wrong?



From E/R Diagrams to Relational Schema

- Entity set → relation
- Relationship → relation

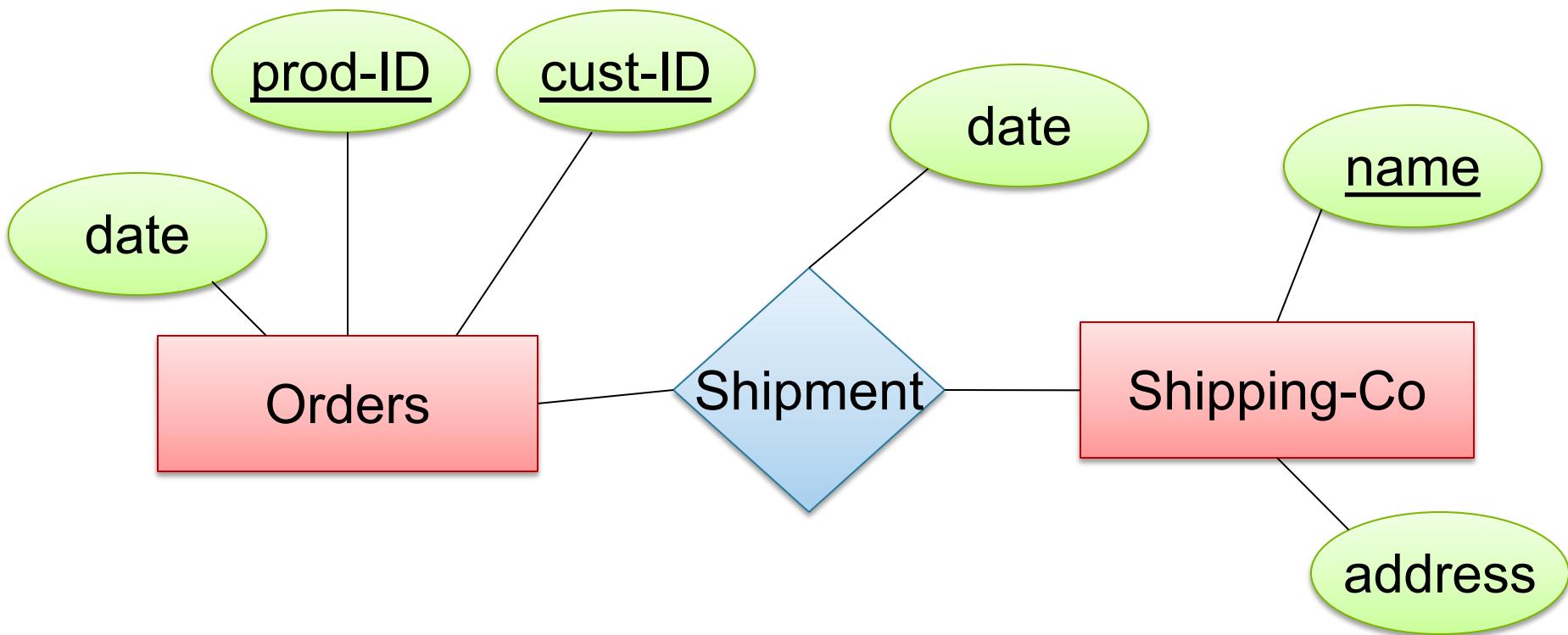
Entity Set to Relation



Product(prod-ID, category, price)

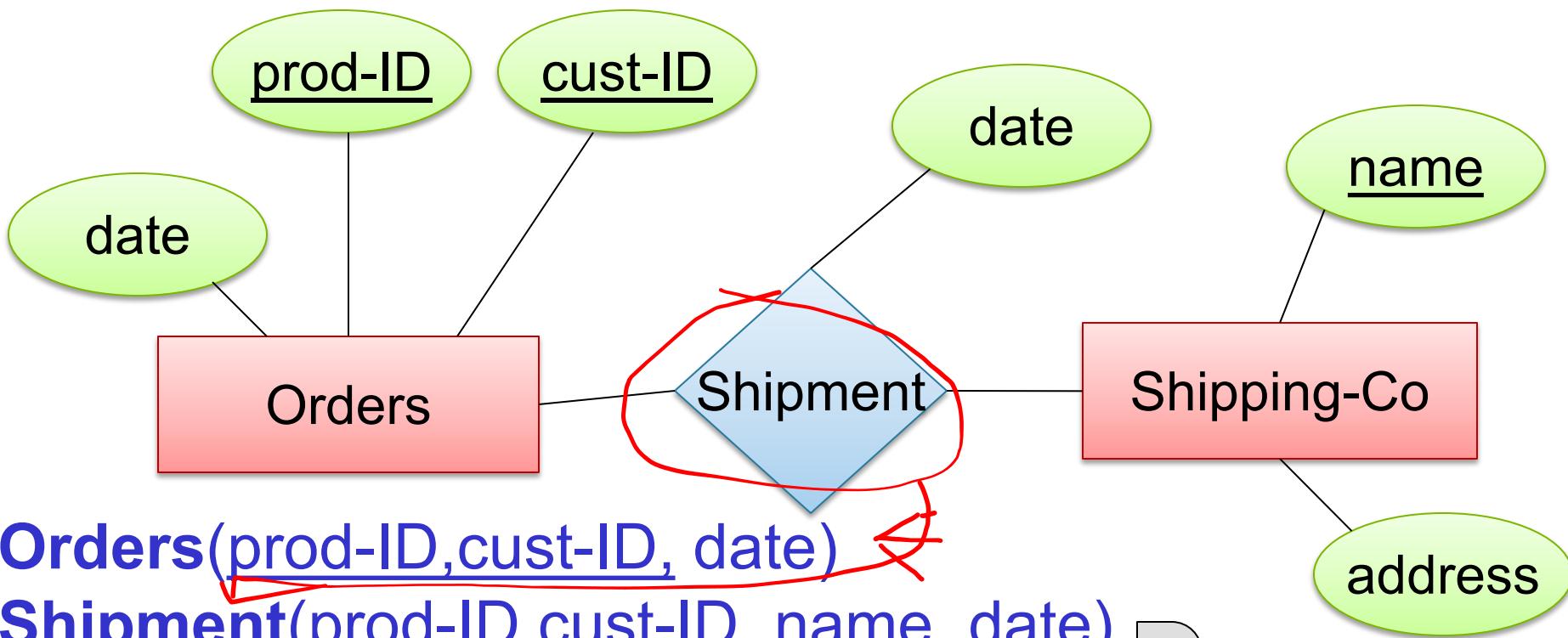
<u>prod-ID</u>	category	price
Gizmo55	Camera	99.99
Pokemn19	Toy 174A - 21Wi	29.99

N-N Relationships to Relations



Represent this in relations

N-N Relationships to Relations



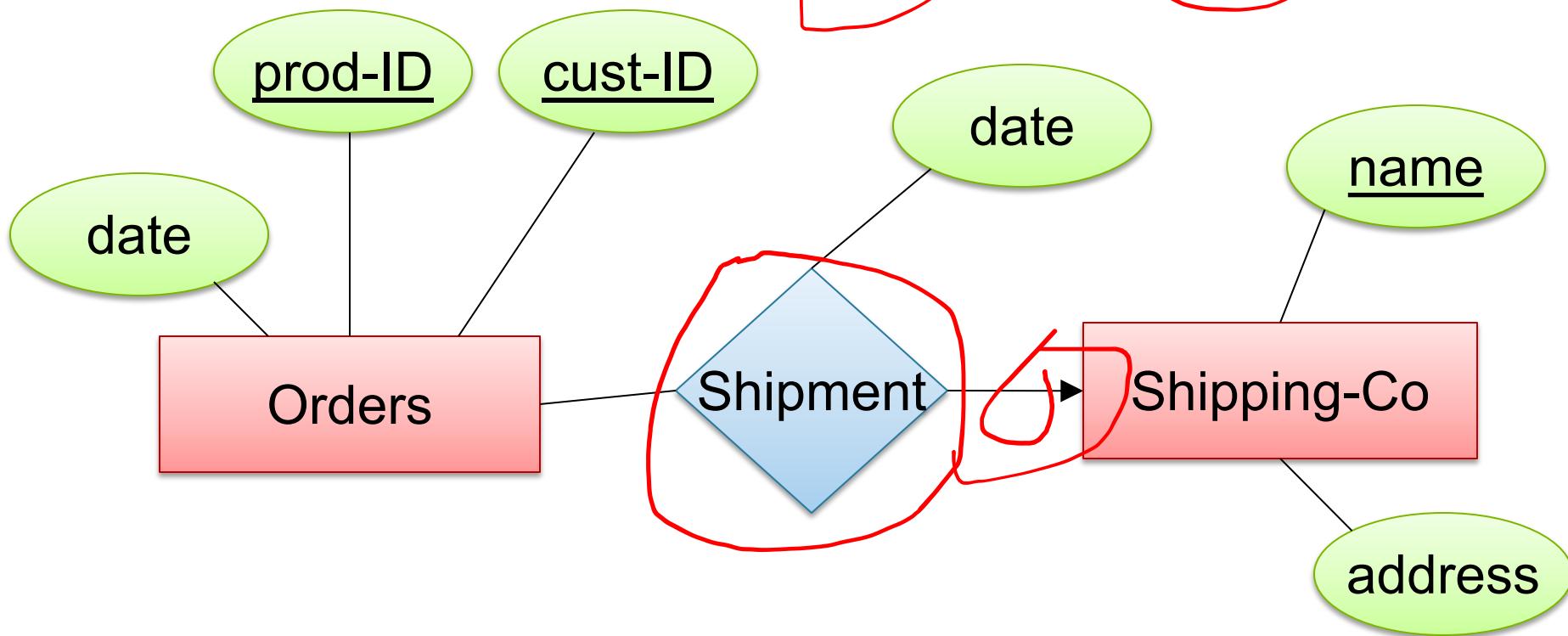
Orders(prod-ID,cust-ID, date)

Shipment(prod-ID,cust-ID, name, date)

Shipping-Co(name, address)

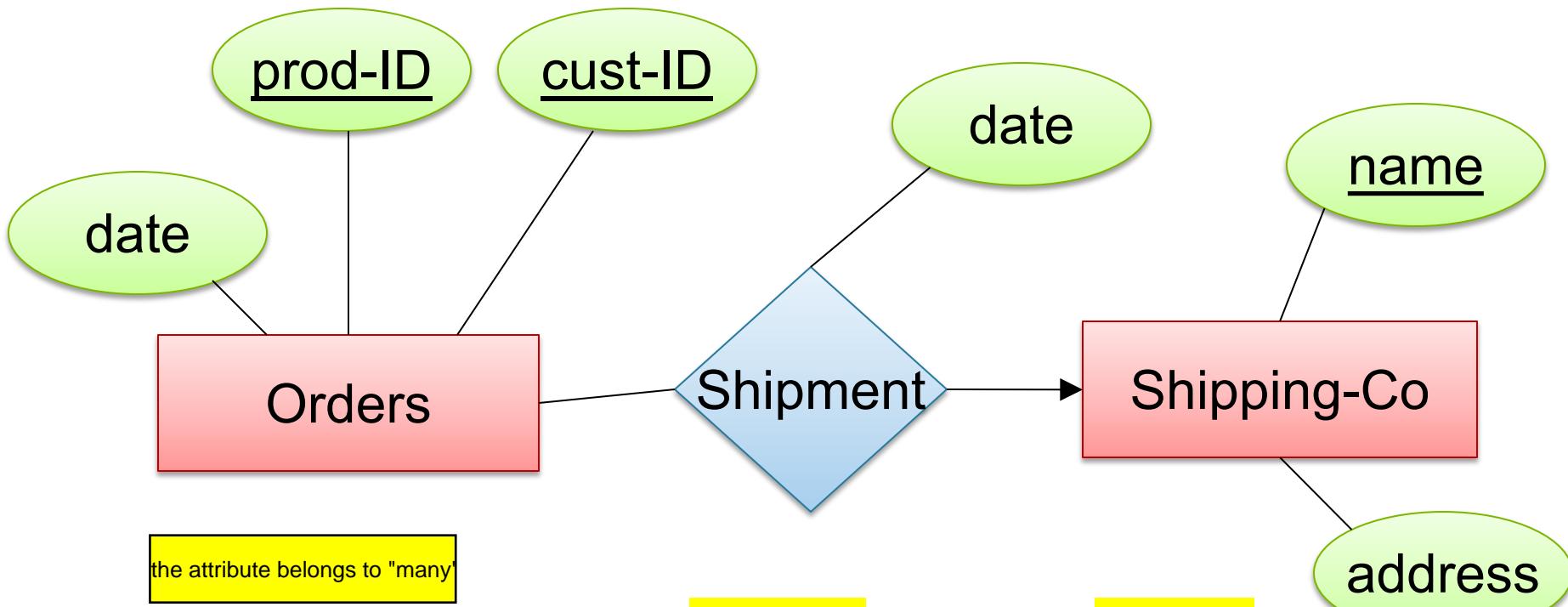
prod-ID	cust-ID	name	date
Gizmo55	Joe12	UPS	4/10/2011
Gizmo55	Joe12	FEDEX	4/9/2011

N-1 Relationships to Relations



Represent this in relations

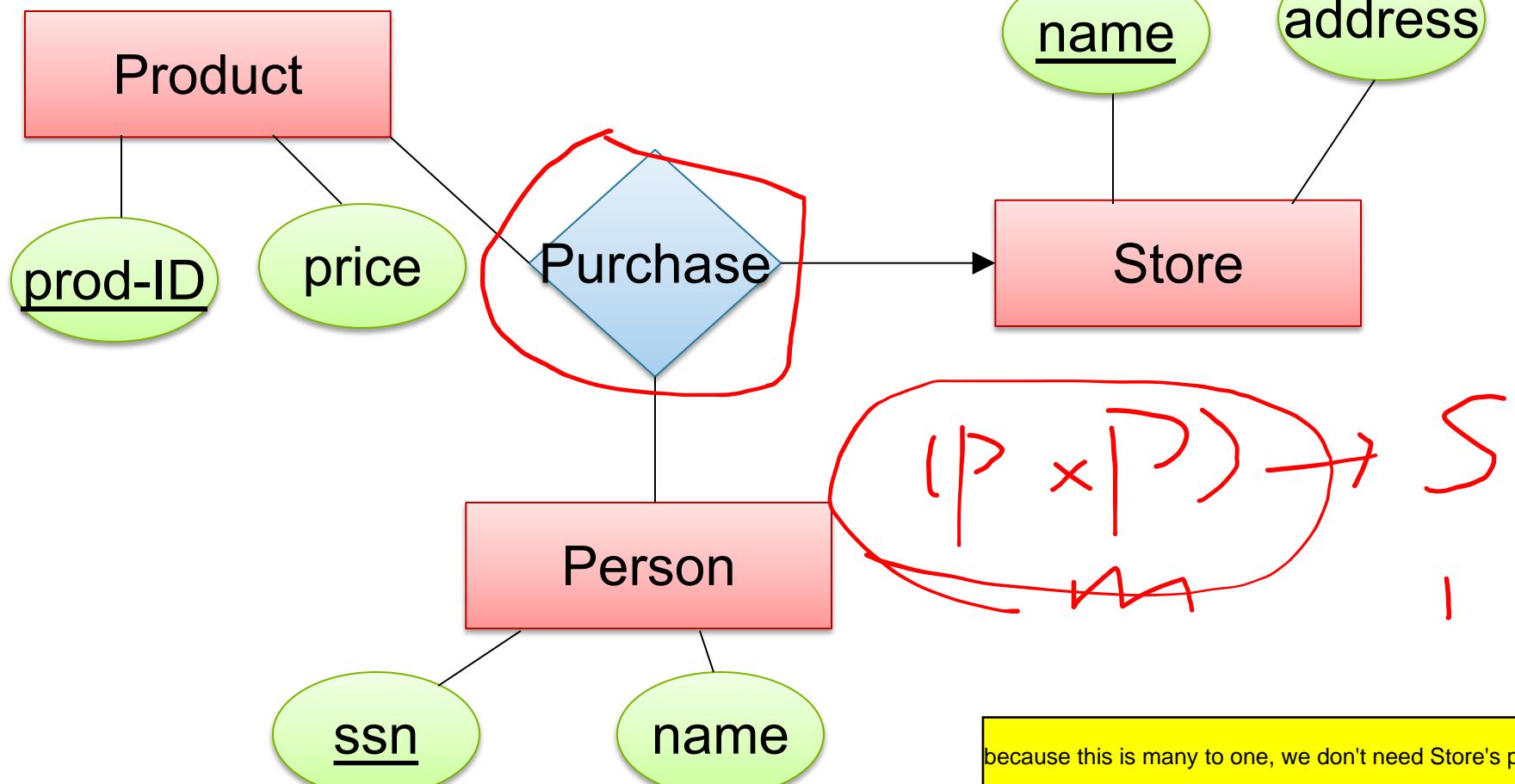
N-1 Relationships to Relations



Orders(prod-ID,cust-ID, date1, name, date2)
Shipping-Co(name, address)

Remember: no separate relations for many-one relationship

Multi-way Relationships to Relations

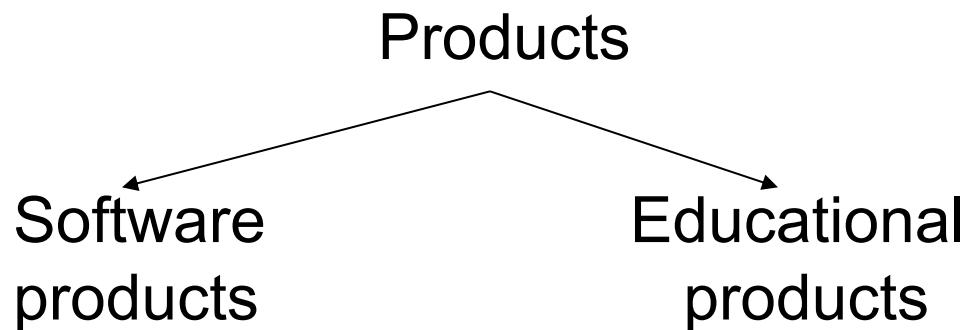


Purchase(prod-ID, ssn, name)

Modeling Subclasses

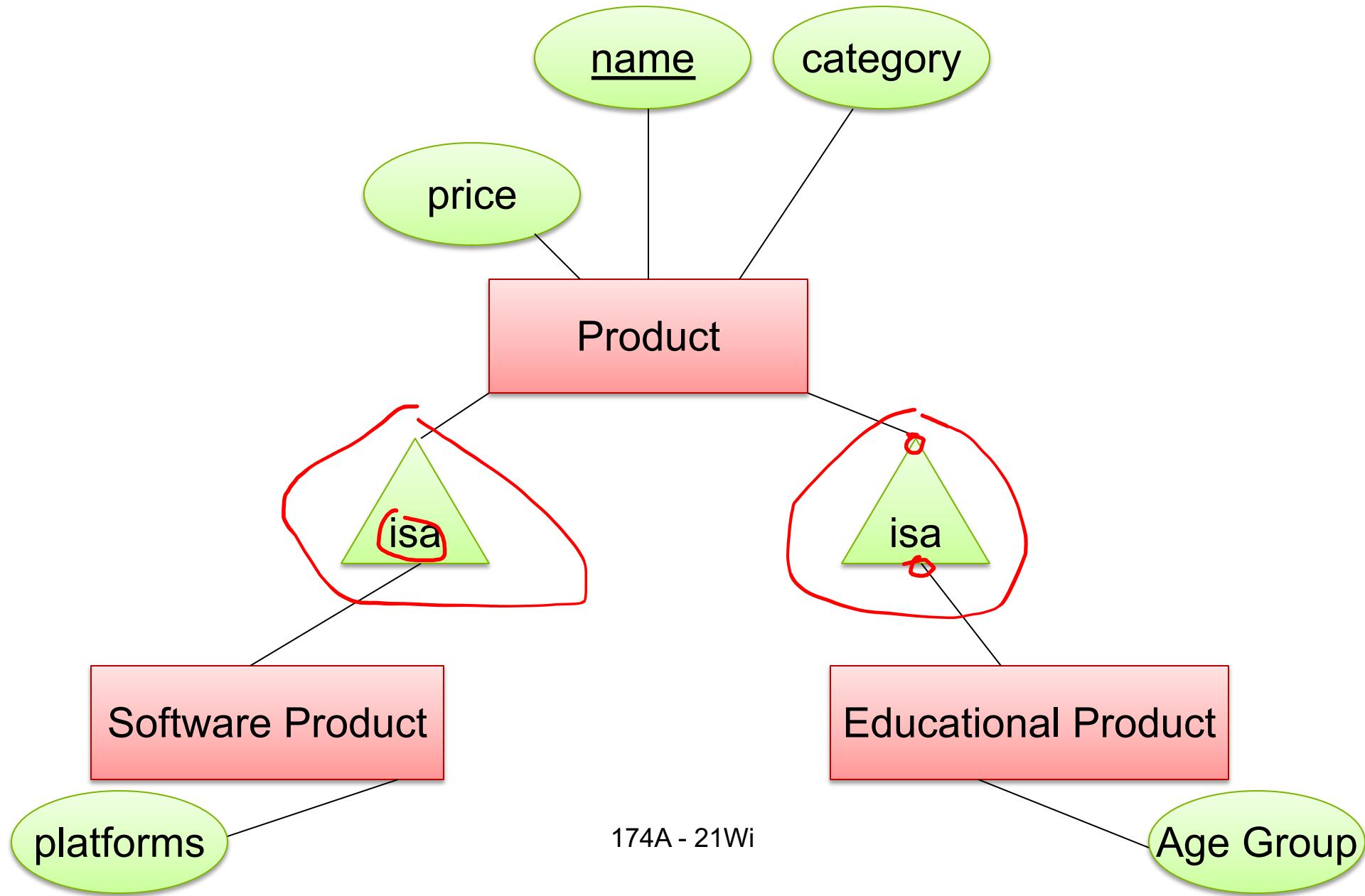
Some objects in a class may be special

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

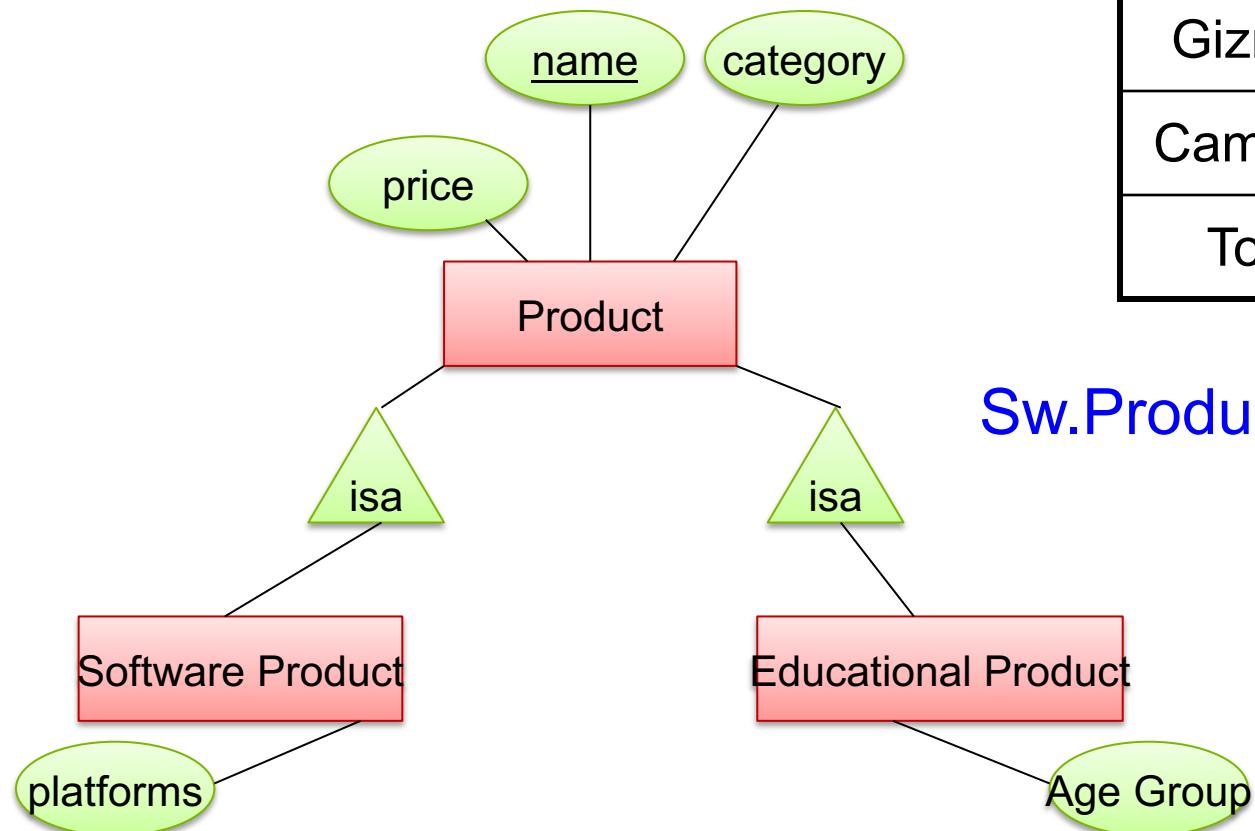


So --- we define subclasses in E/R

Subclasses



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

FurniturePiece

Person

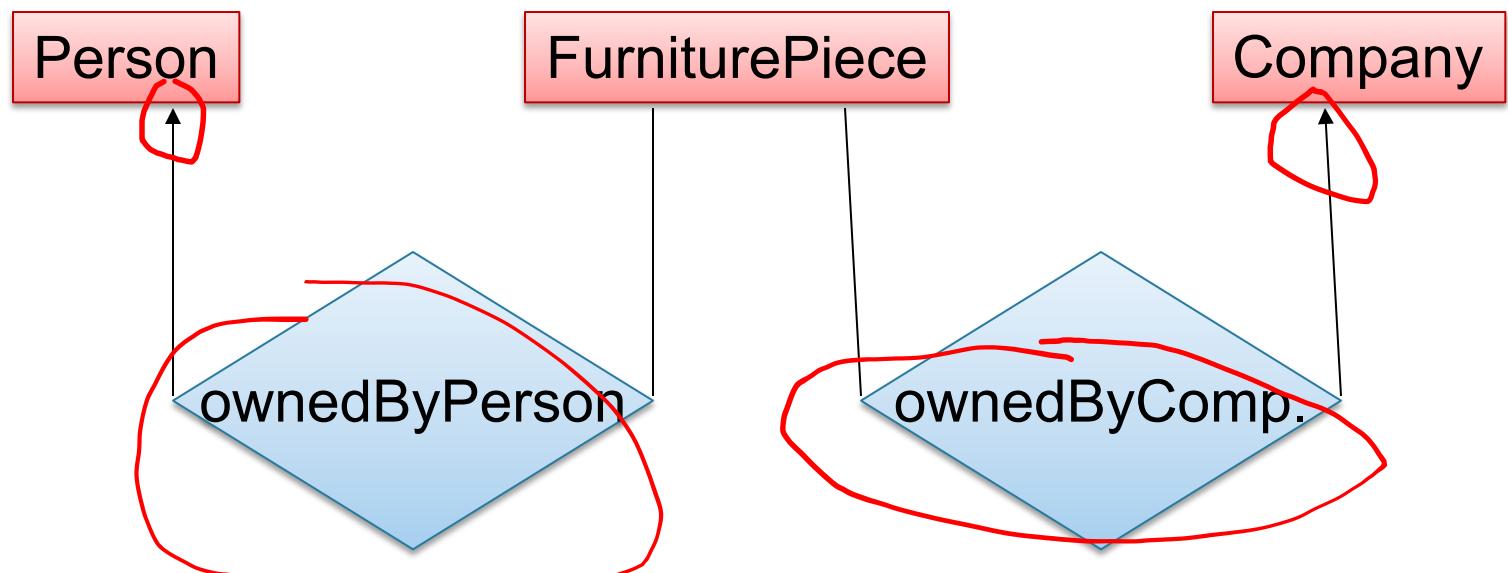
Company

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

Modeling Union Types with Subclasses

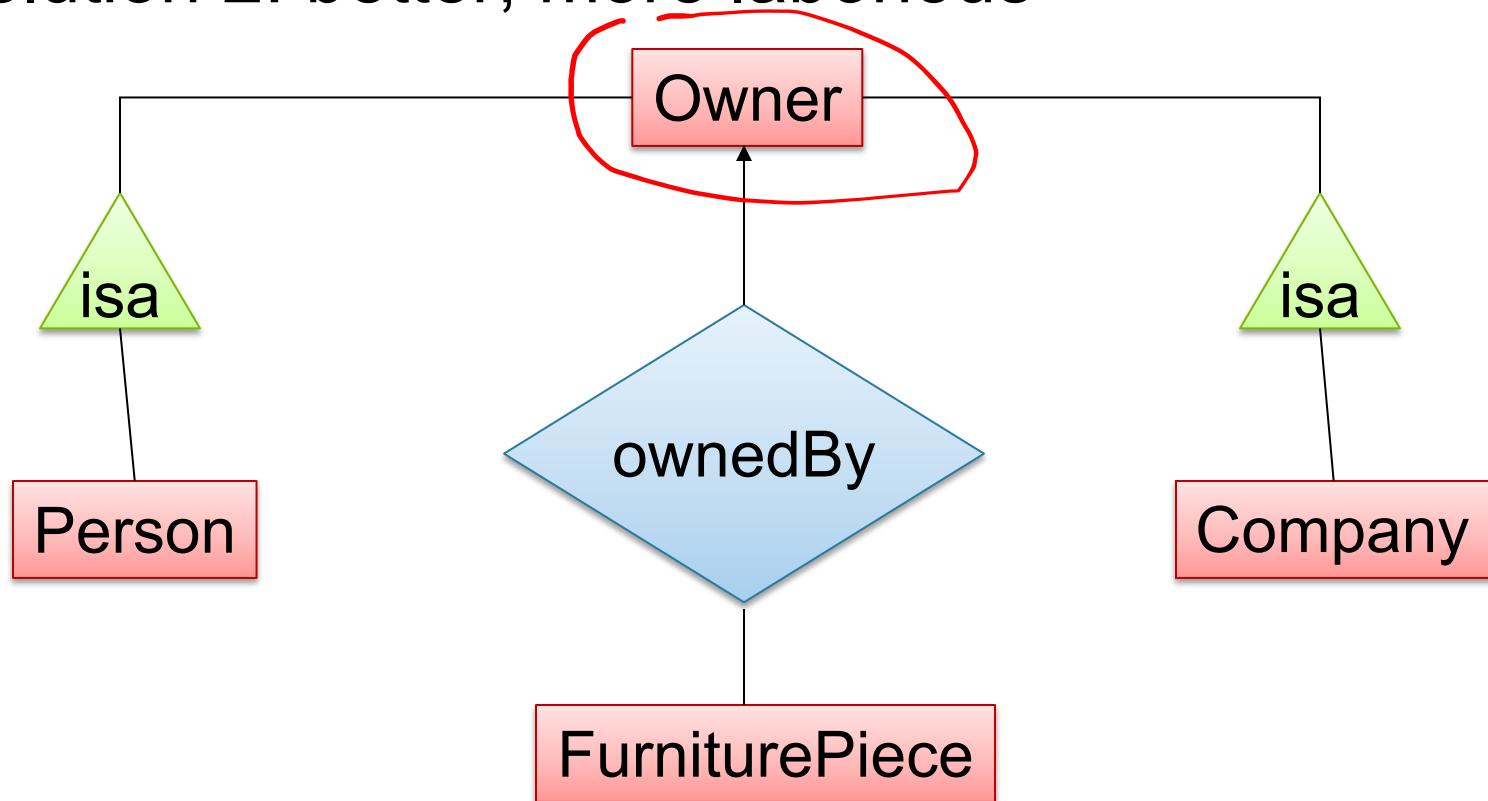
Say: each piece of furniture is owned either by a person or by a company

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



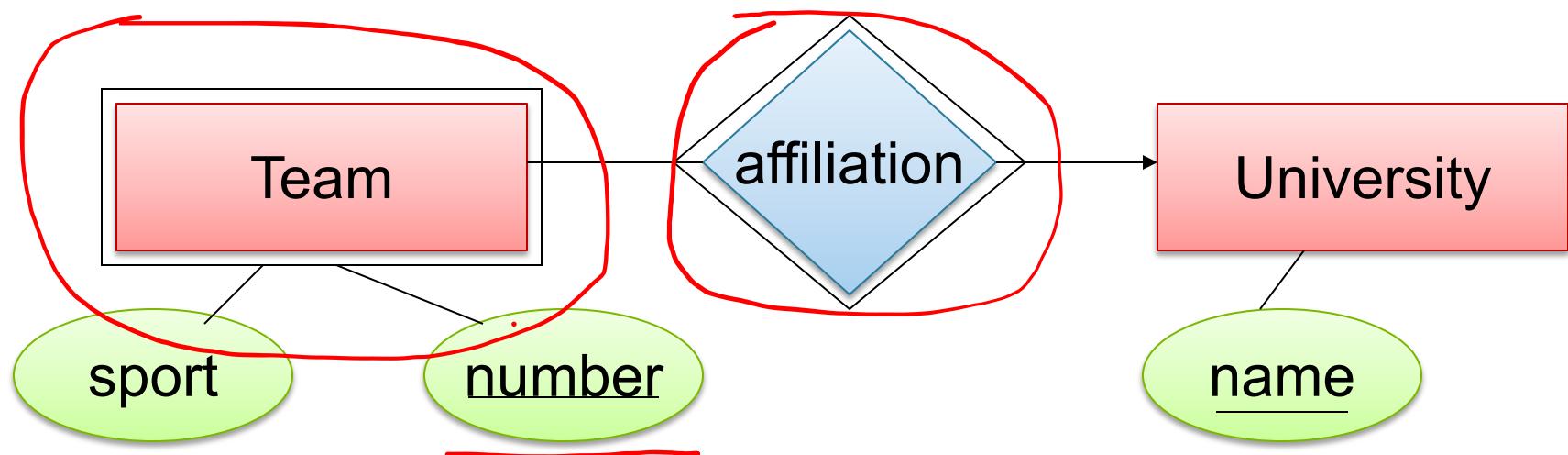
Modeling Union Types with Subclasses

Solution 2: better, more laborious



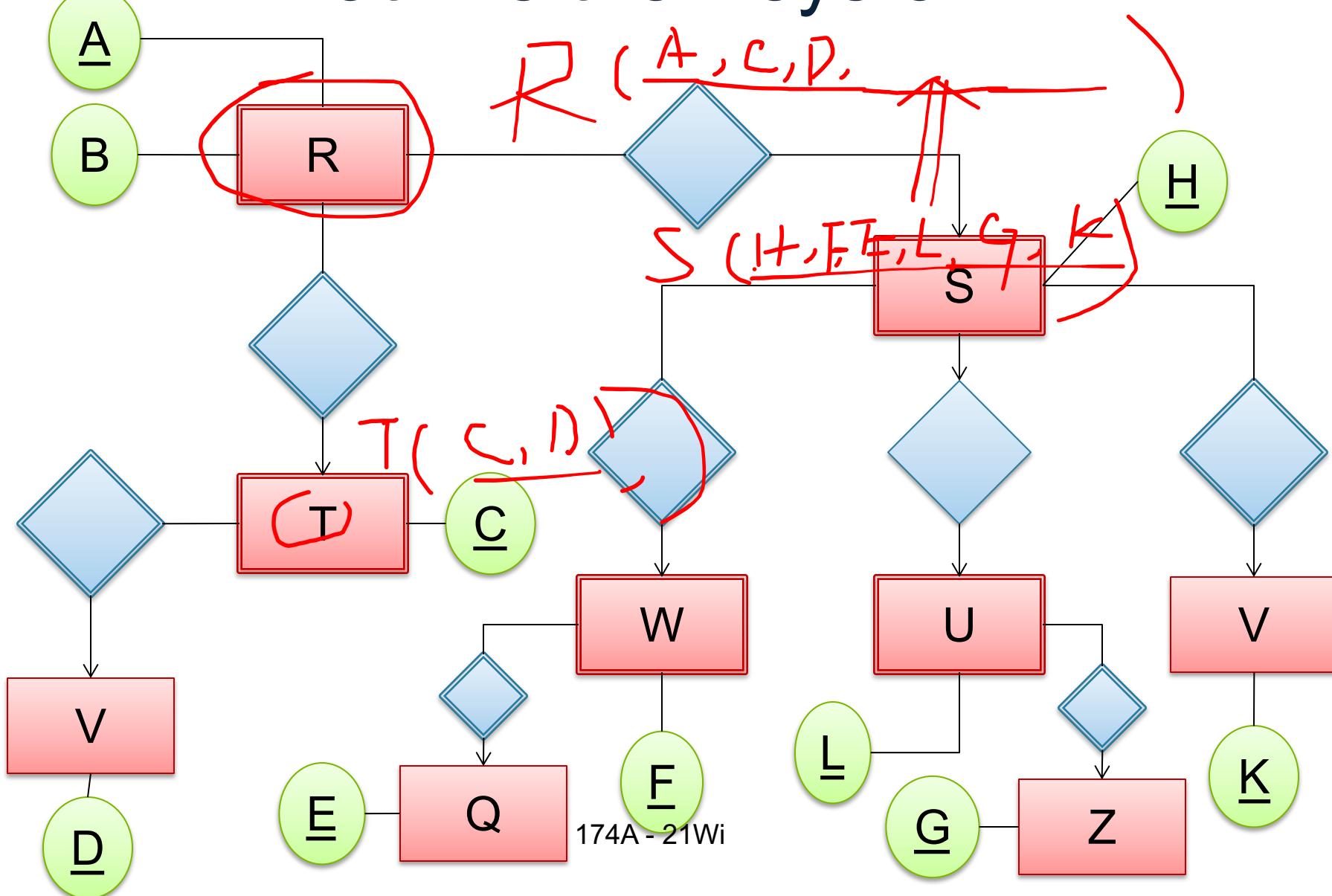
Weak Entity Sets

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 ?



Fundamentals of Database Systems

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

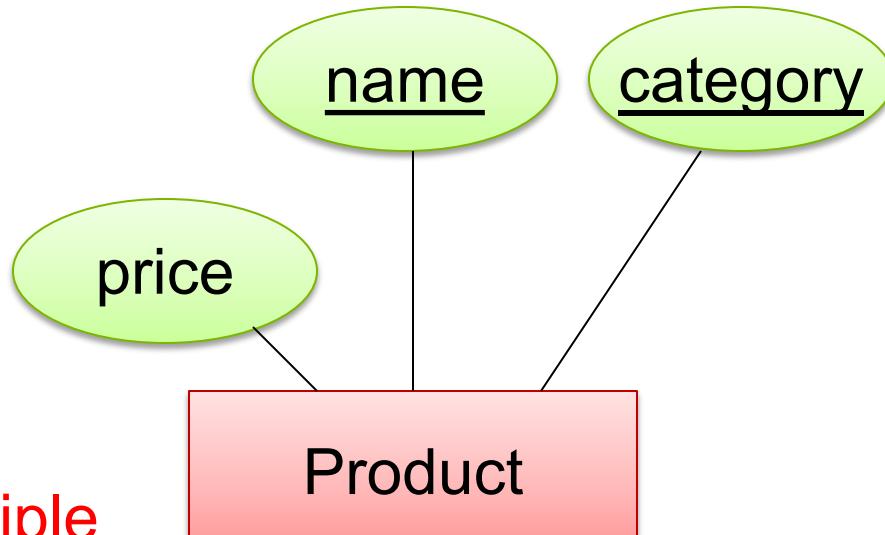
How? The DBMS checks and enforces IC during updates

Constraints in E/R Diagrams

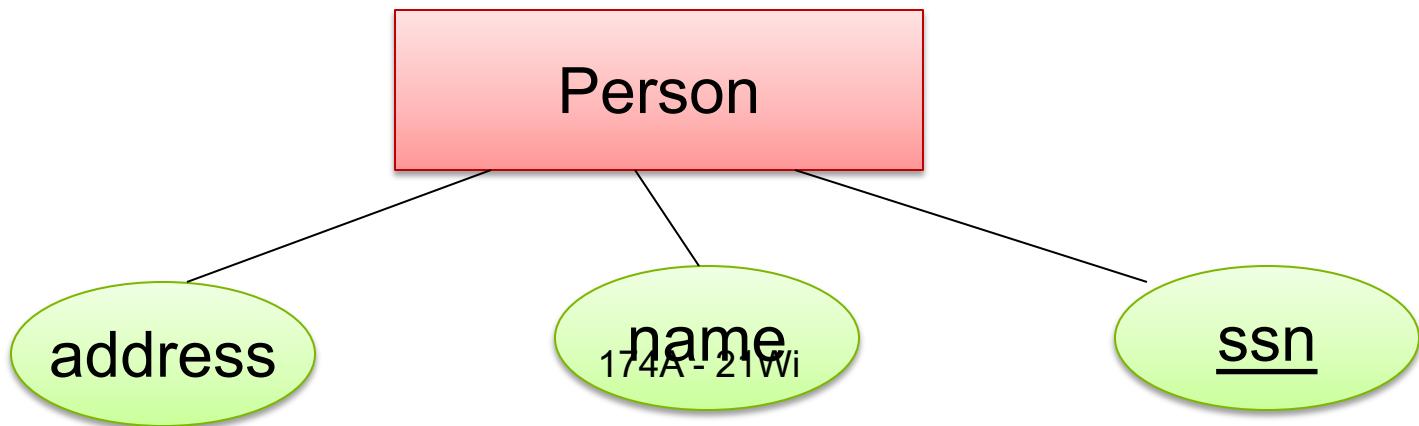
- Keys
- Single-value constraints
- Referential integrity constraints
- General constraints

Keys in E/R Diagrams

Underline:



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



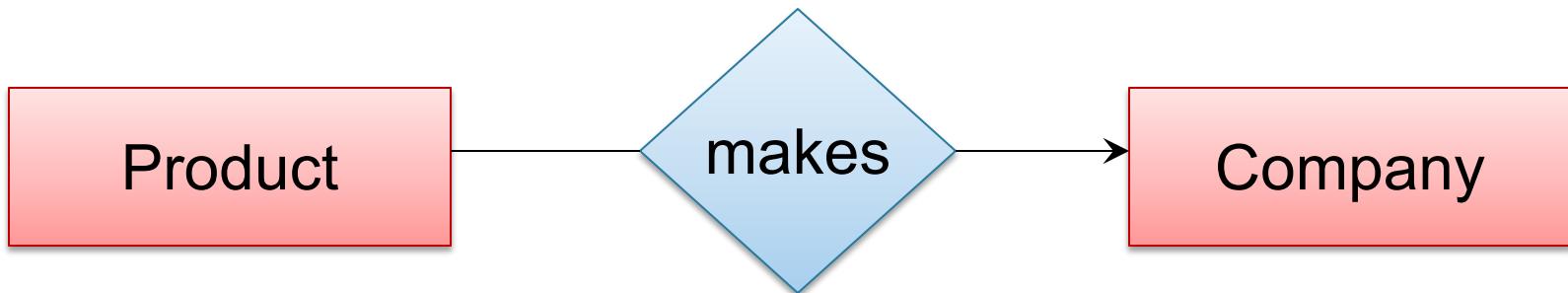
Single Value Constraints



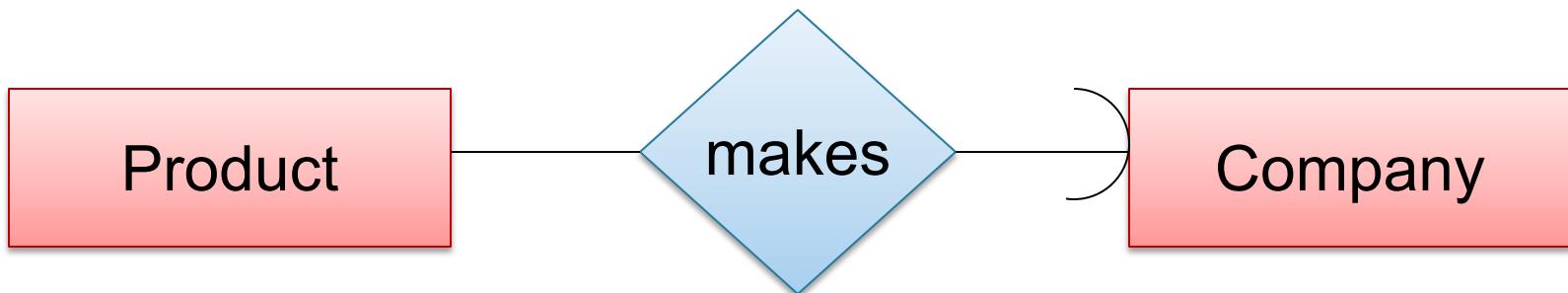
vs.



Referential Integrity Constraints

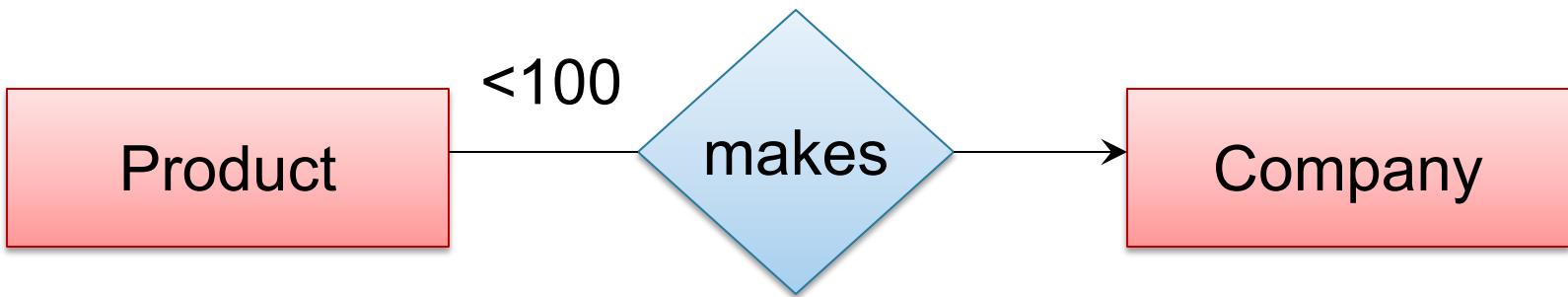


Each product made by at most one company.
Some products made by no company



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(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(name, category, 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

prodName is a **foreign key** to Product(name)
name must be a **key** in Product

May write
just Product
if name is PK

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

The diagram illustrates two tables: **Product** and **Purchase**. An arrow points from the **Product** table to the **Purchase** table, indicating a relationship or update flow between them.

Name	Category
Gizmo	gadget
Camera	Photo
OneClick	Photo

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...)

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	Gizmo
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 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) )
```

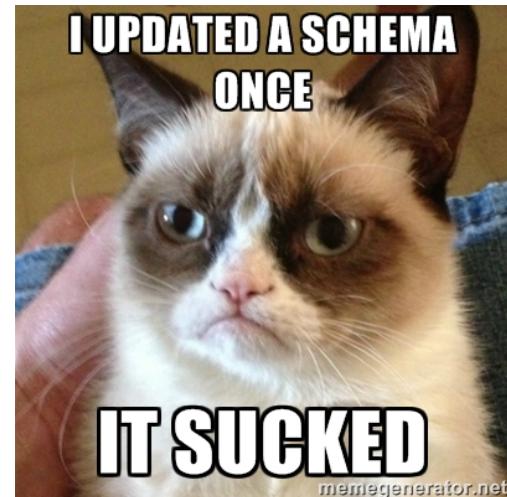
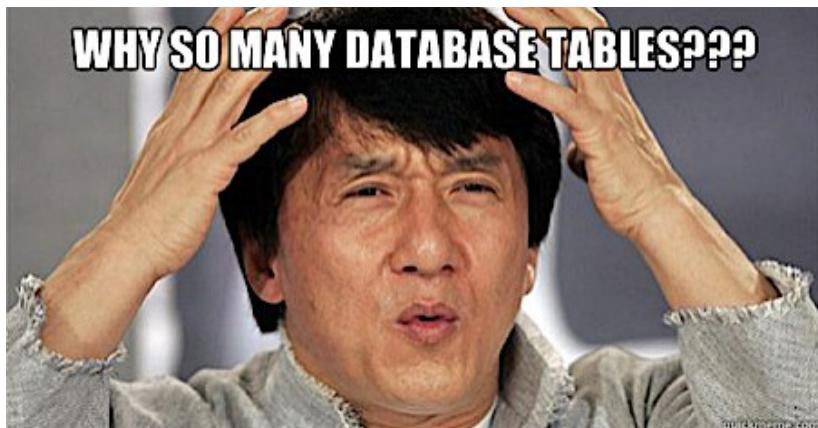
But most DBMSs do not implement assertions
Because it is hard to support them efficiently
Instead, they provide triggers

Introduction to Data Management

CSE 344

Design Theory and BCNF

What makes good schemas?



Relational Schema Design

Name	<u>SSN</u>	<u>PhoneNumber</u>	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	<u>SSN</u>	<u>PhoneNumber</u>	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, \dots, A_n

then they must also agree on the attributes

B_1, B_2, \dots, B_m

Formally:

$A_1 \dots A_n$ **determines** $B_1 \dots B_m$

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

Functional Dependencies (FDs)

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

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

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

R	A_1	\dots	A_m	B_1	\dots	B_n	
t							
t'							

if t, t' agree here then $t[A_1 \dots A_m] = t'[A_1 \dots A_m]$ and $t[B_1 \dots B_n] = t'[B_1 \dots B_n]$

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?

174A - 21Wi

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	<u>SSN</u>	<u>PhoneNumber</u>	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:

$\text{name} \rightarrow \text{color}$

$\text{category} \rightarrow \text{department}$

$\text{color, category} \rightarrow \text{price}$

Then this FD also holds:

$\text{name, category} \rightarrow \text{price}$

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, \dots, A_n

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

Example:

1. name \rightarrow color
2. category \rightarrow department
3. color, category \rightarrow price

Closures:

$$\text{name}^+ = \{\text{name}, \text{color}\}$$

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

$$\text{color}^+ = \{\text{color}\}$$

Closure Algorithm

$X = \{A_1, \dots, A_n\}$.

Repeat until X doesn't change **do:**

if $B_1, \dots, B_n \rightarrow C$ is a FD **and**
 B_1, \dots, B_n are all in X
then add C to X .

Example:

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

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

Hence: $\text{name, category} \rightarrow \text{color, department, price}$

Why do we care?

FD = Functional Dependency.

- 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 → C
A, D → E
B → D
A, F → B

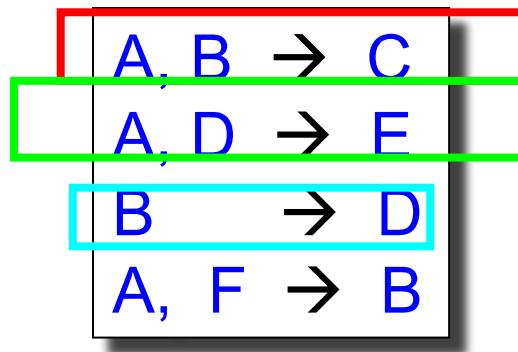
Compute $\{A, B\}^+$ $X = \{A, B,$ }

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

Example

In class:

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



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

1. A,B->C
2. B->D
3. A

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

1. A,F->B
2. B->D
3. A,D->E

Example

In class:

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

A, B → C
A, D → E
B → D
A, F → 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 → C
A, D → E
B → D
A, F → B

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

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

Practice at Home

Find all FD's implied by:

$$\begin{array}{l} A, B \rightarrow C \\ A, D \rightarrow B \\ B \rightarrow D \end{array}$$

Practice at Home

Find all FD's implied by:

$$\begin{array}{l} A, B \rightarrow C \\ A, D \rightarrow B \\ B \rightarrow D \end{array}$$

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

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

$$\begin{aligned} AB^+ &= ABCD, \quad AC^+ = AC, \quad AD^+ = ABCD, \\ &\quad BC^+ = BCD, \quad BD^+ = BD, \quad CD^+ = CD \end{aligned}$$

$$ABC^+ = ABD^+ = ACD^+ = ABCD \text{ (no need to compute— why ?)}$$

$$BCD^+ = BCD, \quad ABCD^+ = ABCD$$

Practice at Home

Find all FD's implied by:

$$\begin{array}{l} A, B \rightarrow C \\ A, D \rightarrow B \\ B \rightarrow D \end{array}$$

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

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

$$\begin{aligned} AB^+ &= ABCD, \quad AC^+ = AC, \quad AD^+ = ABCD, \\ &\quad BC^+ = BCD, \quad BD^+ = BD, \quad CD^+ = CD \end{aligned}$$

$$ABC^+ = ABD^+ = ACD^+ = ABCD \text{ (no need to compute— why ?)}$$

$$BCD^+ = BCD, \quad ABCD^+ = ABCD$$

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

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

Keys

- A **superkey** is a set of attributes A_1, \dots, A_n s.t. for any other attribute B , we have $A_1, \dots, A_n \rightarrow B$
- 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 $X^+ = [\text{all attributes}]$, 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 → price
category → 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 → B
B → C
C → A

Key or Keys ?

We can we have more than one key!

What are the keys here ?

A → B
B → C
C → A

AB → C
BC → A

Key or Keys ?

We can we have more than one key!

What are the keys here ?

A → B
B → C
C → A

A → BC
B → AC

Eliminating Anomalies

Main idea:

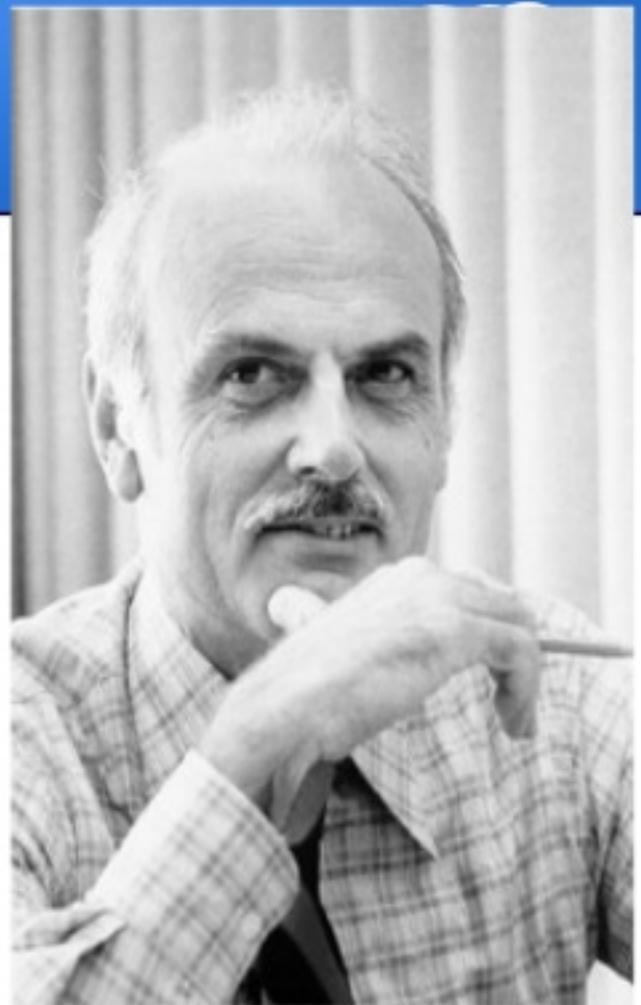
- $X \rightarrow A$ is OK if X is a (super)key
- $X \rightarrow A$ is not OK otherwise
 - Need to decompose the table, but how?

Boyce-Codd Normal Form

Dr. Raymond F. Boyce

Edgar Frank “Ted” Codd

"A Relational Model of Data for
Large Shared Data Banks"



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.

Definition. A relation R is in BCNF if:

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

BCNF Decomposition Algorithm

Normalize(R)

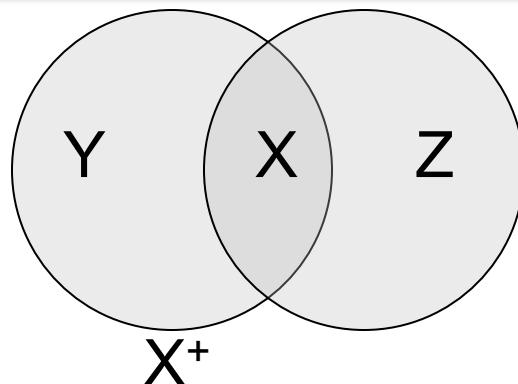
find X s.t.: $X \neq X^+$ and $X^+ \neq [all\ attributes]$

if (not found) then “ R is in BCNF”

let $Y = X^+ - X$; $Z = [all\ attributes] - X^+$

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

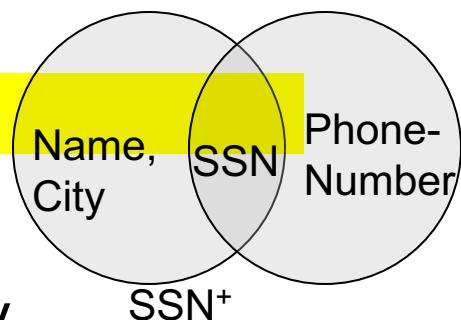
$SSN \rightarrow Name, City$

The only key is: { $SSN, PhoneNumber$ }

Hence $SSN \rightarrow Name, City$ is a “bad” dependency

In other words:

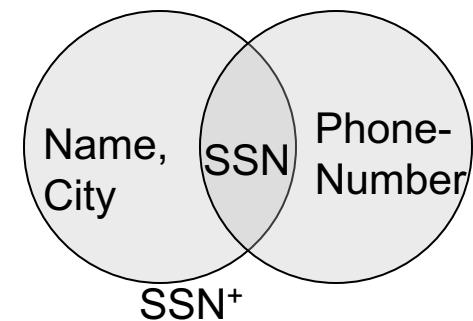
$SSN^+ = SSN, Name, City$ and is neither SSN nor All Attributes



Example BCNF Decomposition

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

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



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

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)

$SSN \rightarrow name, age$

$age \rightarrow hairColor$

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

Example BCNF Decomposition

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

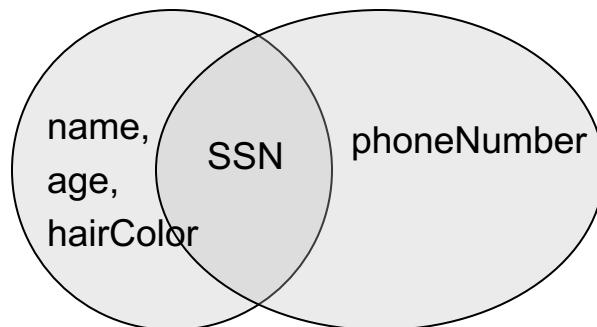
$\text{SSN} \rightarrow \text{name, age}$

$\text{age} \rightarrow \text{hairColor}$

Iteration 1: Person: $\text{SSN}^+ = \text{SSN, name, age, hairColor}$

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

Phone(SSN, phoneNumber)



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

Example BCNF Decomposition

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

$\text{SSN} \rightarrow \text{name, age}$

$\text{age} \rightarrow \text{hairColor}$

What are
the keys ?

Iteration 1: Person: $\text{SSN}^+ = \text{SSN, name, age, hairColor}$

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

Phone(SSN, phoneNumber)

Iteration 2: P: $\text{age}^+ = \text{age, hairColor}$

Decompose: People(SSN, name, age)

Hair(age, hairColor)

Phone(SSN, phoneNumber)

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

Example BCNF Decomposition

$\text{Person}(\text{name}, \text{SSN}, \text{age}, \text{hairColor}, \text{phoneNumber})$

$\text{SSN} \rightarrow \text{name}, \text{age}$

$\text{age} \rightarrow \text{hairColor}$

Note the keys!

Iteration 1: Person : $\text{SSN}^+ = \text{SSN}, \text{name}, \text{age}, \text{hairColor}$

Decompose into: $P(\underline{\text{SSN}}, \text{name}, \text{age}, \text{hairColor})$

$\text{Phone}(\underline{\text{SSN}}, \text{phoneNumber})$

Iteration 2: P : $\text{age}^+ = \text{age}, \text{hairColor}$

Decompose: $\text{People}(\underline{\text{SSN}}, \text{name}, \text{age})$

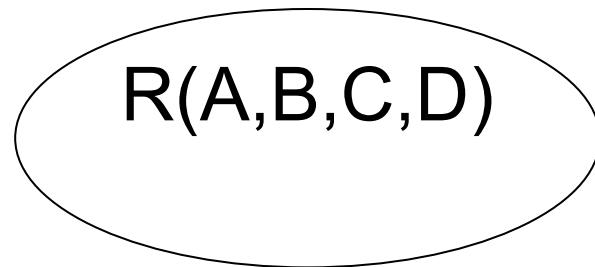
$\text{Hair}(\underline{\text{age}}, \text{hairColor})$

$\text{Phone}(\underline{\text{SSN}}, \text{phoneNumber})$

$R(A,B,C,D)$

Example: BCNF

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

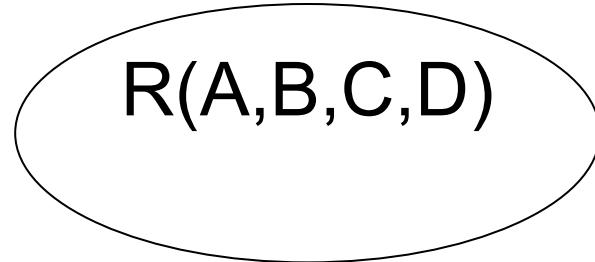


$R(A,B,C,D)$

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

Example: BCNF

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



$R(A,B,C,D)$

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

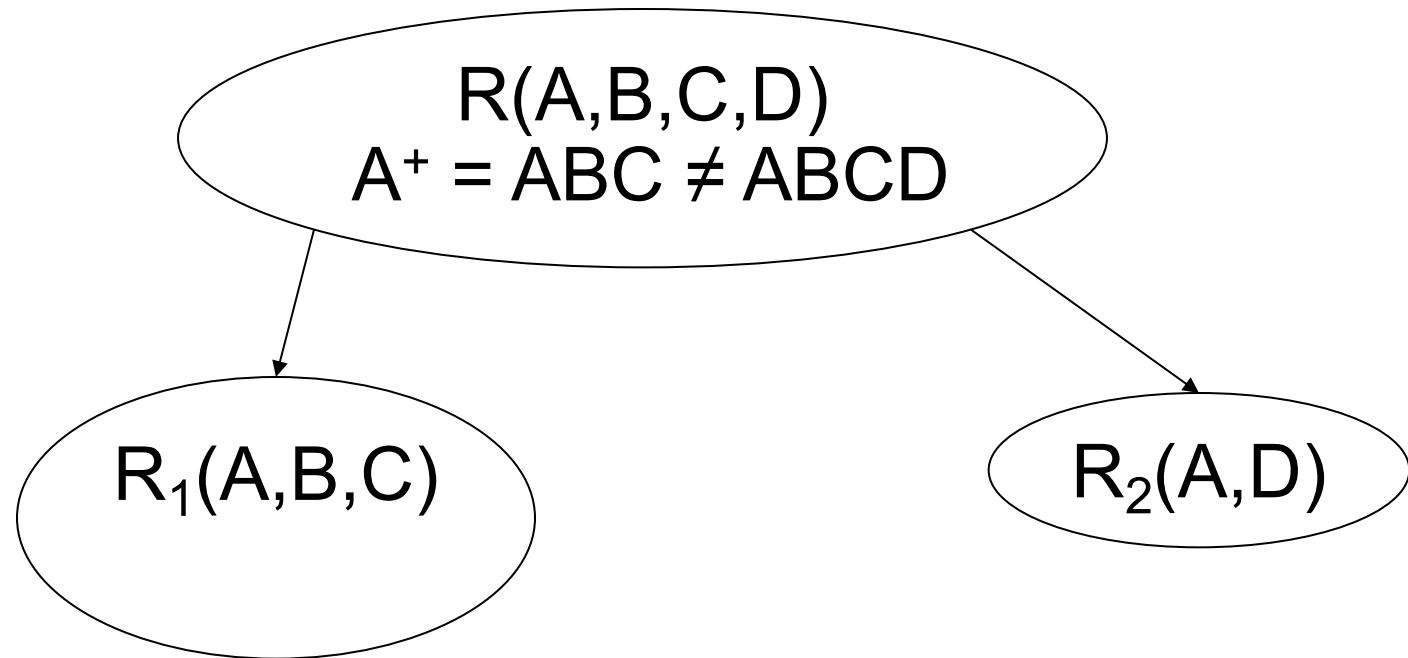
Example: BCNF

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

$R(A,B,C,D)$

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

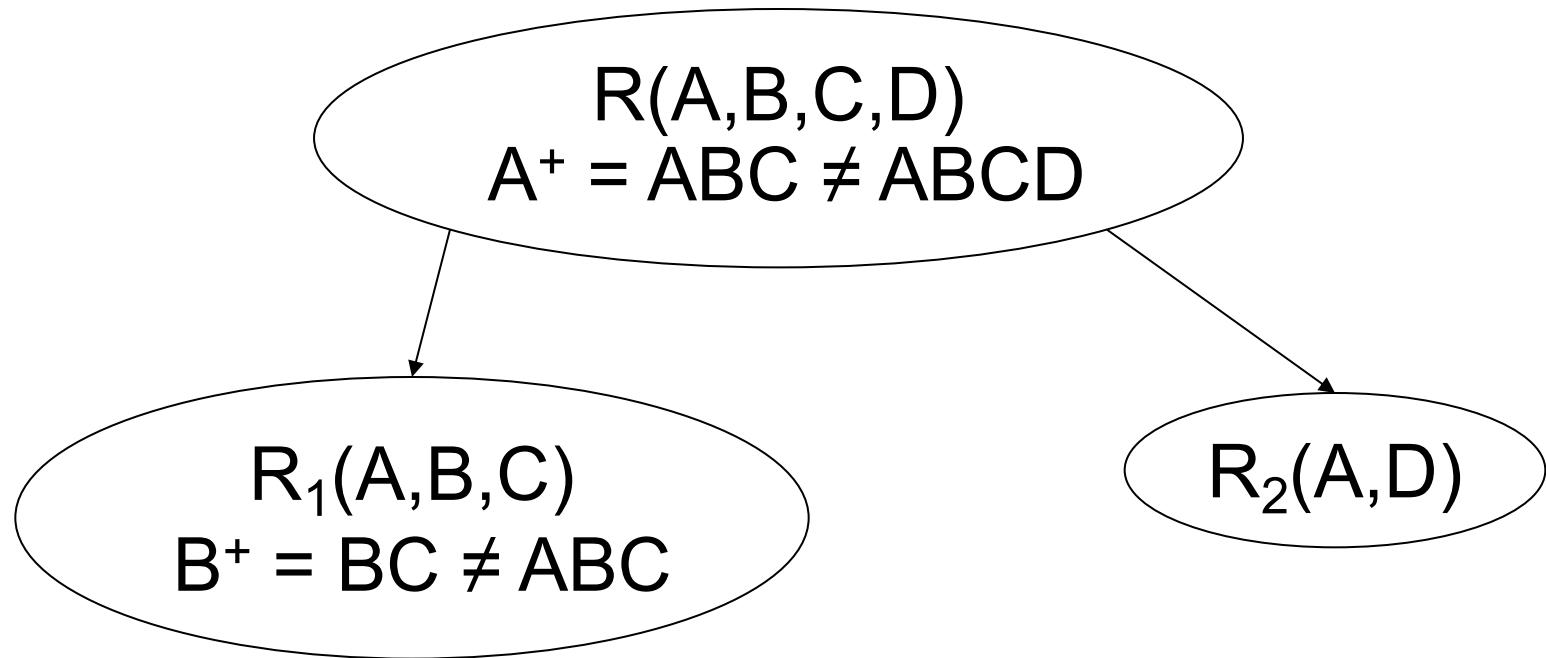
Example: BCNF



$R(A,B,C,D)$

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

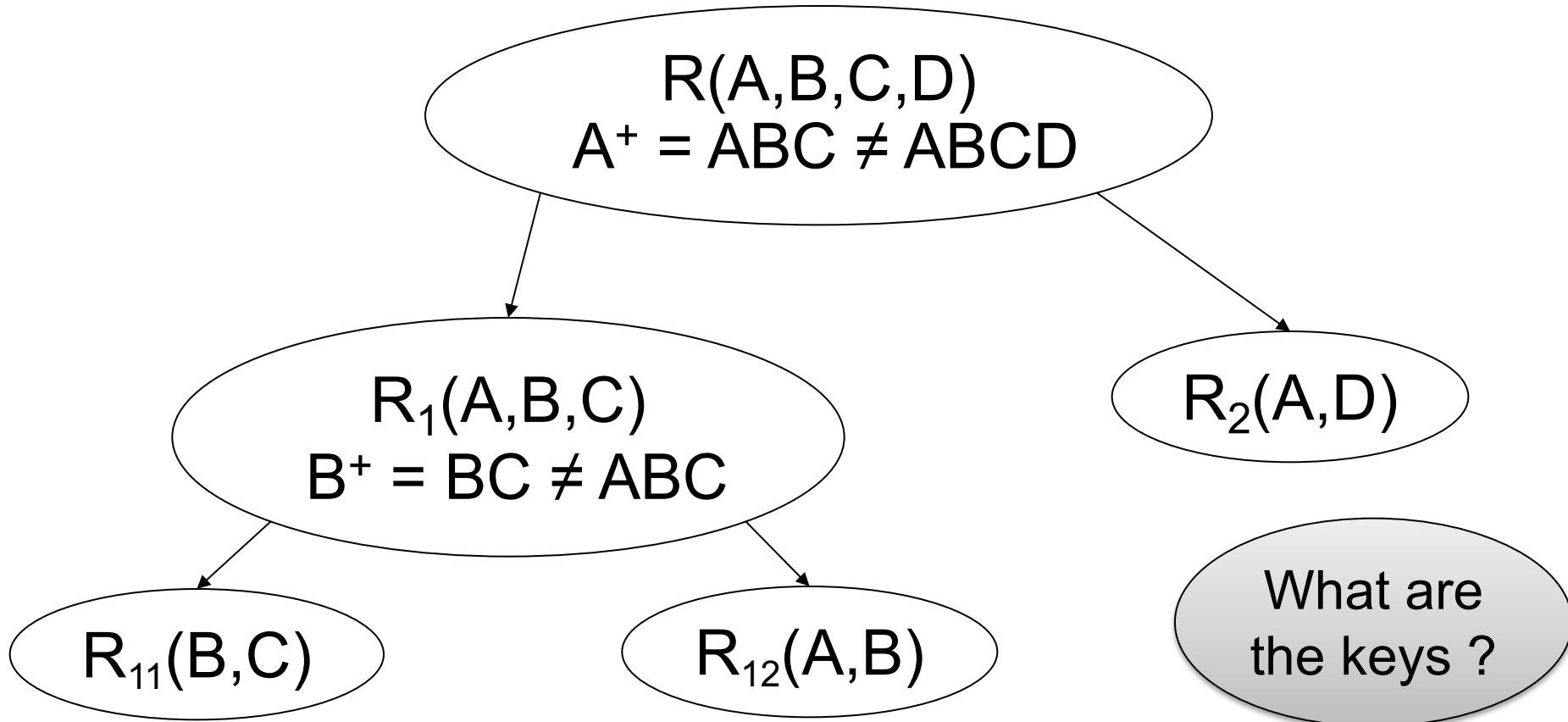
Example: BCNF



$R(A,B,C,D)$

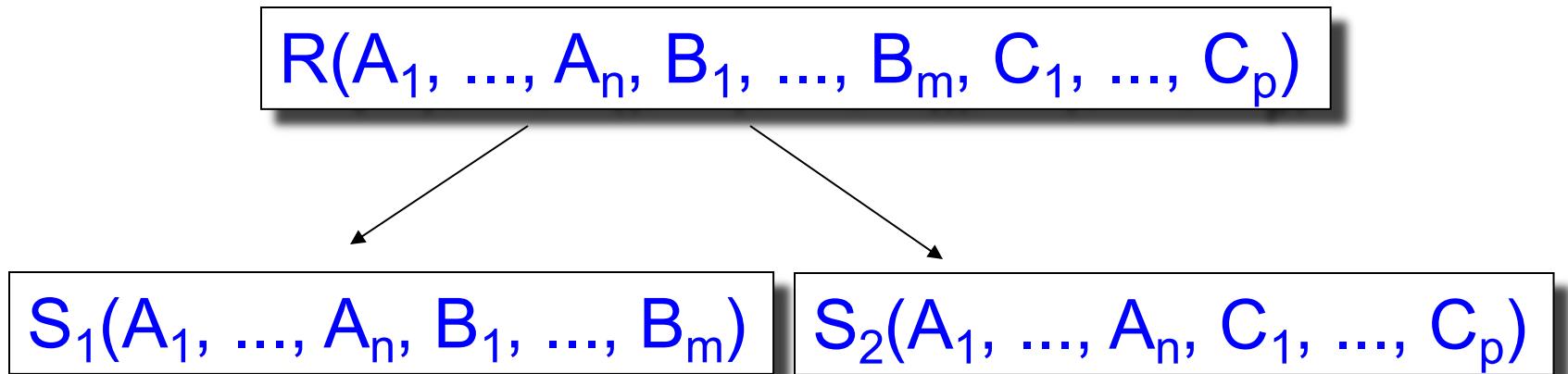
$A \rightarrow B$
 $B \rightarrow C$

Example: BCNF



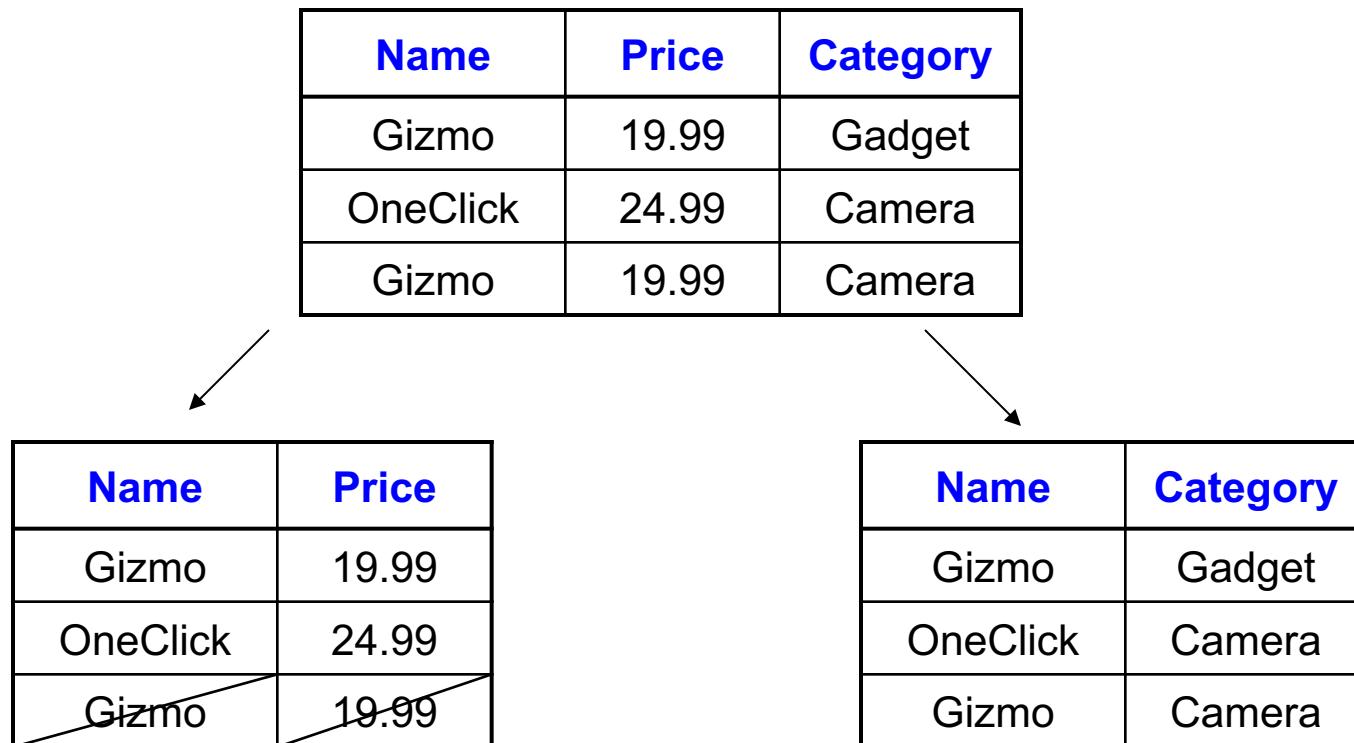
What happens if in R we first pick B^+ ? Or AB^+ ?

Decompositions in General



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

Lossless Decomposition



Lossy Decomposition

What is
lossy here?

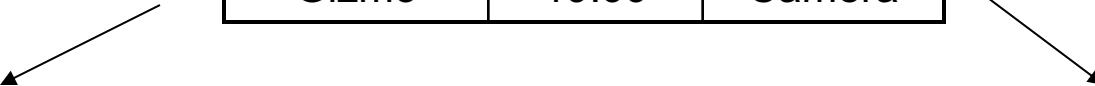
Name	Price	Category
Gizmo	19.99	Gadget
OneClick	24.99	Camera
Gizmo	19.99	Camera

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

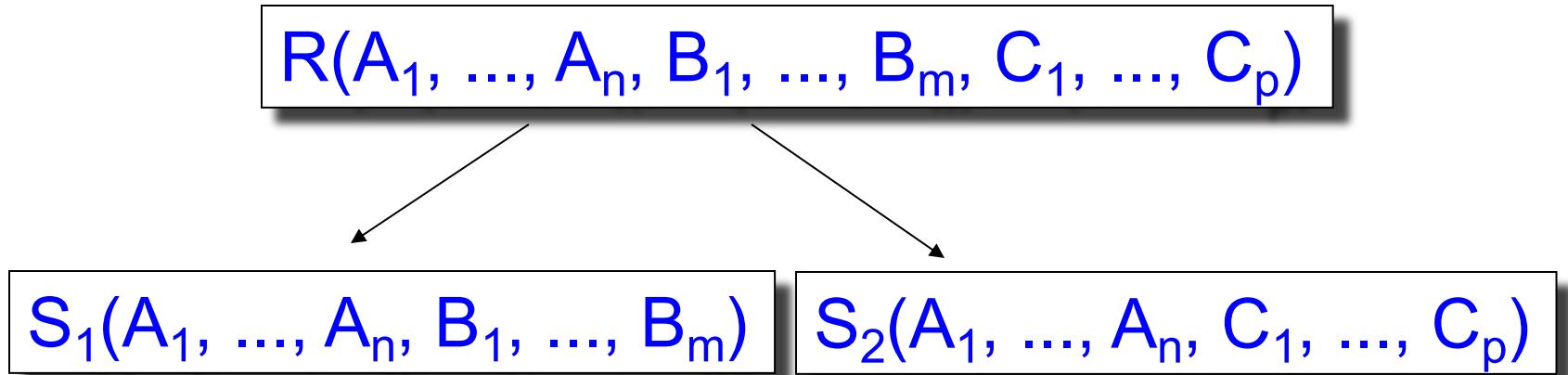
Name	Price	Category
Gizmo	19.99	Gadget
OneClick	24.99	Camera
Gizmo	19.99	Camera



Name	Category
Gizmo	Gadget
OneClick	Camera
Gizmo	Camera

Price	Category
19.99	Gadget
24.99	Camera
19.99	Camera

Decomposition in General



Let: S_1 = projection of R on $A_1, \dots, A_n, B_1, \dots, B_m$

S_2 = projection of R on $A_1, \dots, A_n, C_1, \dots, C_p$

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

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

It follows that every BCNF decomposition is lossless

Testing for Lossless Join

If we decompose R into $\Pi_{S_1}(R)$, $\Pi_{S_2}(R)$, $\Pi_{S_3}(R)$, ...
Is it true that $S_1 \bowtie S_2 \bowtie S_3 \bowtie \dots = R$?

To check “ $=$ ” we need to check “ \subseteq ” and “ \supseteq ”

$R \subseteq S_1 \bowtie S_2 \bowtie S_3 \bowtie \dots$ always holds (why?)

$R \supseteq S_1 \bowtie S_2 \bowtie S_3 \bowtie \dots$ need to check

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?

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

The Chase Test for Lossless Join

$$R(A,B,C,D) = S_1(A,D) \bowtie S_2(A,C) \bowtie S_3(B,C,D)$$

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

Lossless?

$$S_1 = \Pi_{AD}(R), S_2 = \Pi_{AC}(R), S_3 = \Pi_{BCD}(R)$$

$$R \subseteq S_1 \bowtie S_2 \bowtie S_3$$

$$\text{To check: } R \supseteq S_1 \bowtie S_2 \bowtie S_3$$

The Chase Test for Lossless Join

$$R(A,B,C,D) = S_1(A,D) \bowtie S_2(A,C) \bowtie S_3(B,C,D)$$

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

Lossless?

$$S_1 = \Pi_{AD}(R), S_2 = \Pi_{AC}(R), S_3 = \Pi_{BCD}(R)$$

$$R \subseteq S_1 \bowtie S_2 \bowtie S_3$$

To check: $R \supseteq S_1 \bowtie S_2 \bowtie S_3$

Suppose $(a,b,c,d) \in S_1 \bowtie S_2 \bowtie S_3$ Is it also in R?

The Chase Test for Lossless Join

$$R(A,B,C,D) = S_1(A,D) \bowtie S_2(A,C) \bowtie S_3(B,C,D)$$

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

Lossless?

$$S_1 = \Pi_{AD}(R), S_2 = \Pi_{AC}(R), S_3 = \Pi_{BCD}(R)$$

$$R \subseteq S_1 \bowtie S_2 \bowtie S_3$$

To check: $R \supseteq S_1 \bowtie S_2 \bowtie S_3$

Suppose $(a,b,c,d) \in S_1 \bowtie S_2 \bowtie S_3$ Is it also in R?

R must contain the following tuples:

A	B	C	D
a	b1	c1	d

Why ?

$$(a,d) \in S_1 = \Pi_{AD}(R)$$

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

Why ?

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

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

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 ?

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

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

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

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 ?

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

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

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

$A \rightarrow B$

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

The Chase Test for Lossless Join

$$R(A,B,C,D) = S_1(A,D) \bowtie S_2(A,C) \bowtie S_3(B,C,D)$$

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

Lossless?

$$S_1 = \Pi_{AD}(R), S_2 = \Pi_{AC}(R), S_3 = \Pi_{BCD}(R)$$

$$R \subseteq S_1 \bowtie S_2 \bowtie S_3$$

To check: $R \supseteq S_1 \bowtie S_2 \bowtie S_3$

Suppose $(a,b,c,d) \in S_1 \bowtie S_2 \bowtie S_3$ 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 ?

$$(a,d) \in S_1 = \Pi_{AD}(R)$$

$$(a,c) \in S_2 = \Pi_{BD}(R)$$

$$(b,c,d) \in S_3 = \Pi_{BCD}(R)$$

“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



The Chase Test for Lossless Join

$$R(A,B,C,D) = S_1(A,D) \bowtie S_2(A,C) \bowtie S_3(B,C,D)$$

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

Lossless?

$$S_1 = \Pi_{AD}(R), S_2 = \Pi_{AC}(R), S_3 = \Pi_{BCD}(R)$$

$$R \subseteq S_1 \bowtie S_2 \bowtie S_3$$

To check: $R \supseteq S_1 \bowtie S_2 \bowtie S_3$

Suppose $(a,b,c,d) \in S_1 \bowtie S_2 \bowtie S_3$ 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 ?

$$(a,d) \in S_1 = \Pi_{AD}(R)$$

$$(a,c) \in S_2 = \Pi_{BD}(R)$$

$$(b,c,d) \in S_3 = \Pi_{BCD}(R)$$

“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
a3	b	c	d

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

The Chase Test for Lossless Join

$$R(A,B,C,D) = S_1(A,D) \bowtie S_2(A,C) \bowtie S_3(B,C,D)$$

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

Lossless?

$$S_1 = \Pi_{AD}(R), S_2 = \Pi_{AC}(R), S_3 = \Pi_{BCD}(R)$$

$$R \subseteq S_1 \bowtie S_2 \bowtie S_3$$

To check: $R \supseteq S_1 \bowtie S_2 \bowtie S_3$

YES!

Suppose $(a,b,c,d) \in S_1 \bowtie S_2 \bowtie S_3$ 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 ?

$$(a,d) \in S_1 = \Pi_{AD}(R)$$

$$(a,c) \in S_2 = \Pi_{BD}(R)$$

$$(b,c,d) \in S_3 = \Pi_{BCD}(R)$$

“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
a3	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 may 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