CSE 344

AUGUST 1ST

ENTITIES

EXAMS

- **Will hand back after class**
- **Quartiles**
  - 1 0 – 67
  - 2 68 – 74
  - 3 74 – 82
  - 4 82 – 100
  - (no one actually got 0 or 100)

ADMINISTRIVIA

- **HW6 due Wednesday**
  - Spark SQL interface much easier to use
    - definitely the best choice when doing real work
  - BUT important to understand how this works
    - SQL query becomes a sequence of calls to other APIs
    - multiple calls get grouped into a single job
      - only need real splits where “reshuffles” occur
  - RDDs are much faster than MapReduce
    - no disk means the bottleneck is network cost
    - (if we had more time, we’d analyze this more...
      only one operation actually has any network cost)

DATABASE DESIGN

What it is:

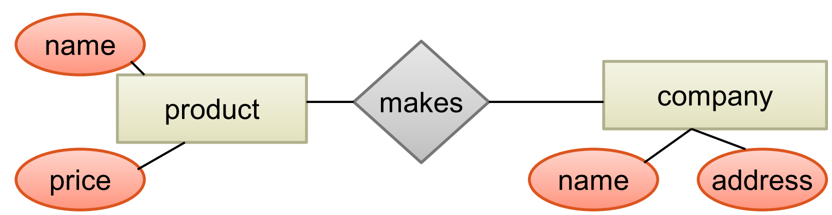
Starting from scratch, design the database schema: relation, attributes, keys, foreign keys, constraints etc

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 PROCESS

Conceptual Model:



Relational Model:

Tables + constraints

And also functional dep.

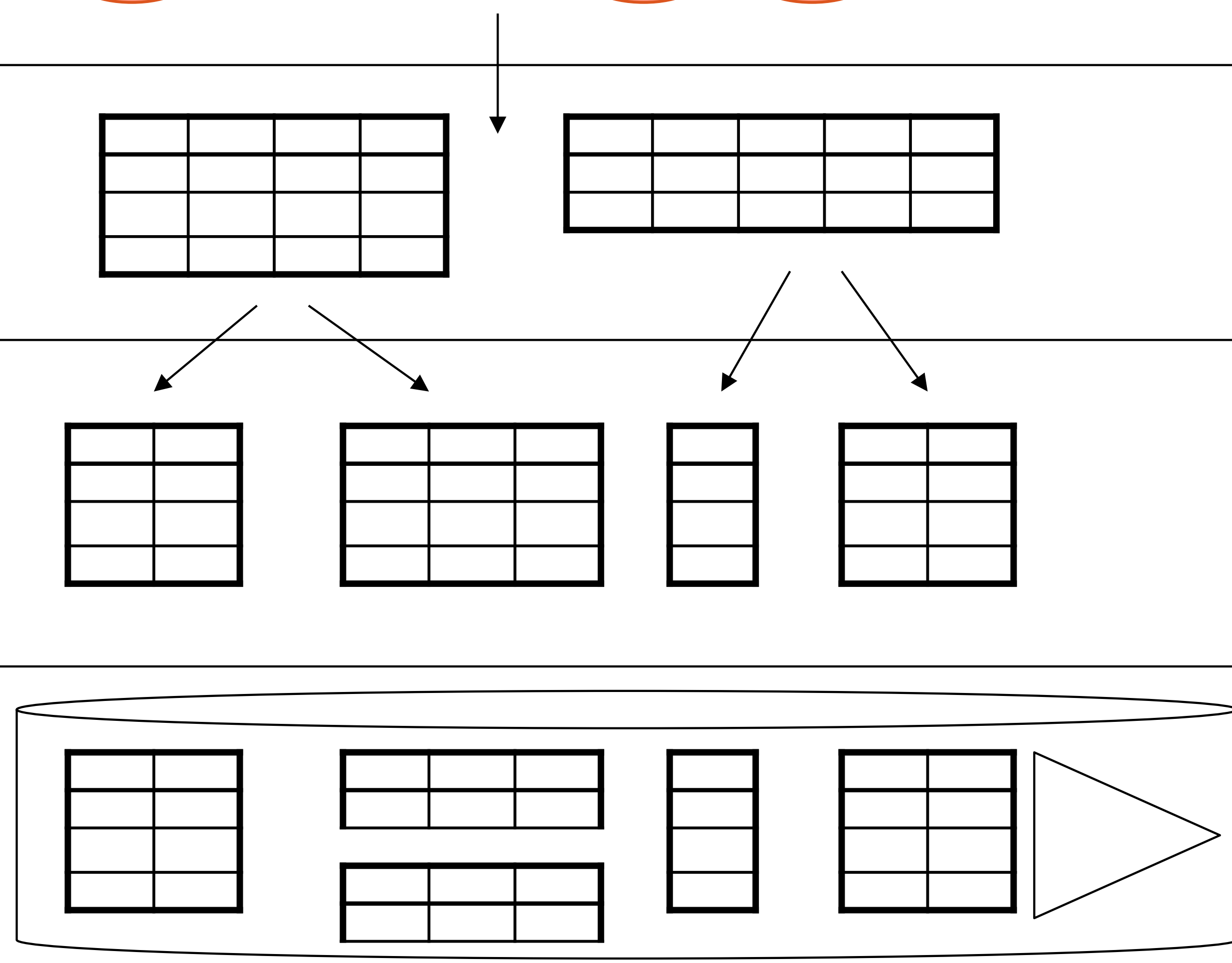
Normalization:

Eliminates anomalies

Conceptual Schema

Physical storage details

Physical Schema

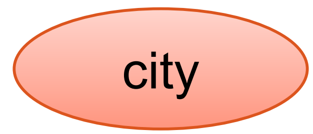


ENTITY / RELATIONSHIP DIAGRAMS

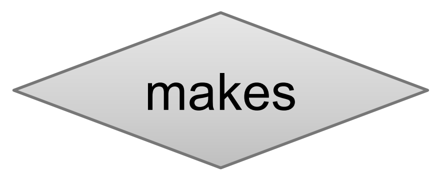
Entity set = a class

- An entity = an object

Attribute



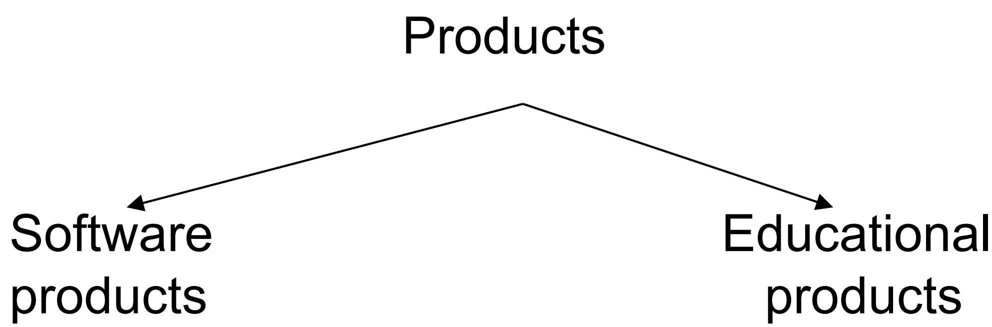
Relationship



MODELING SUBCLASSES

Some objects in a class may be special

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

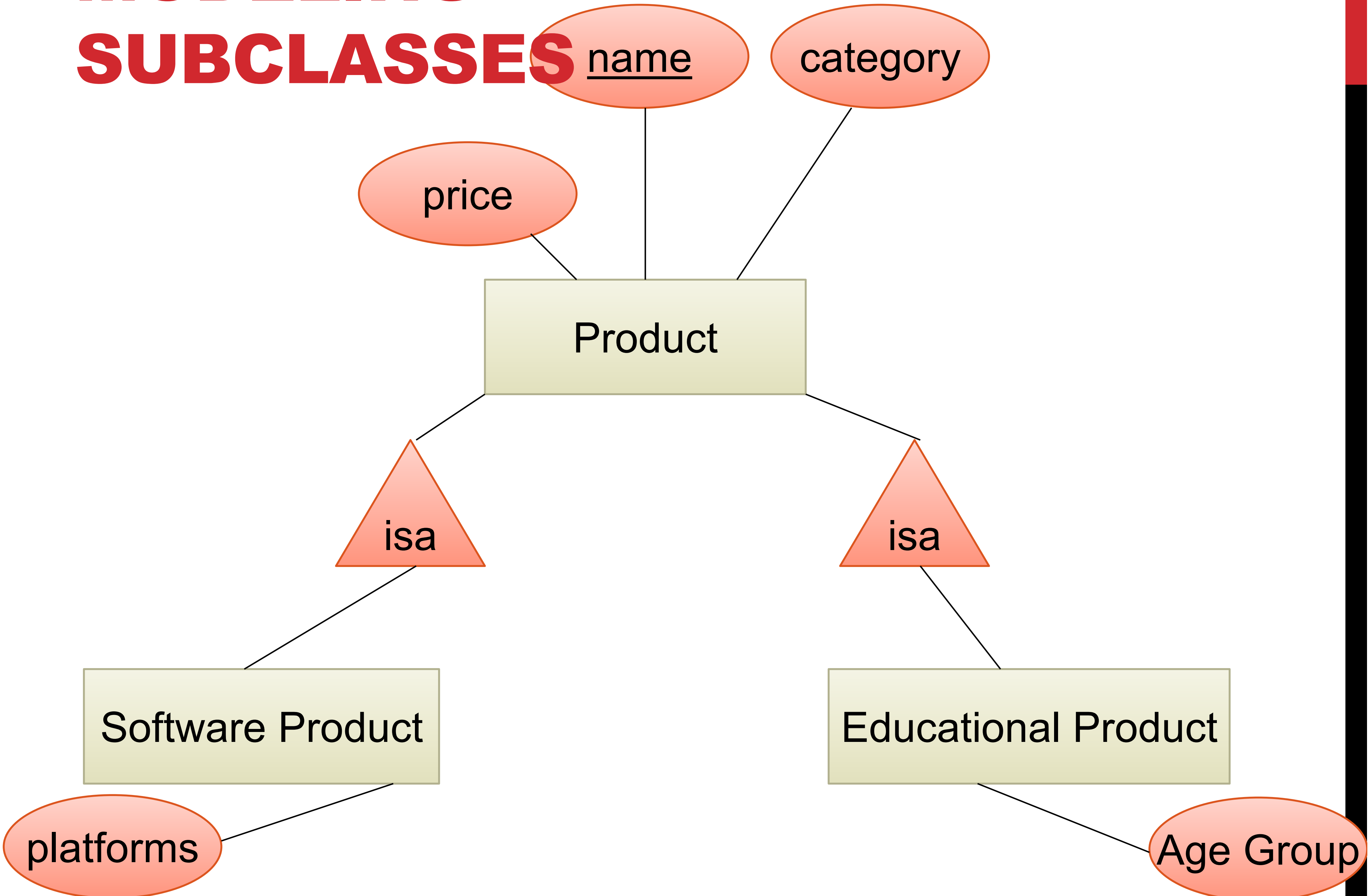


So --- we define subclasses in E/R

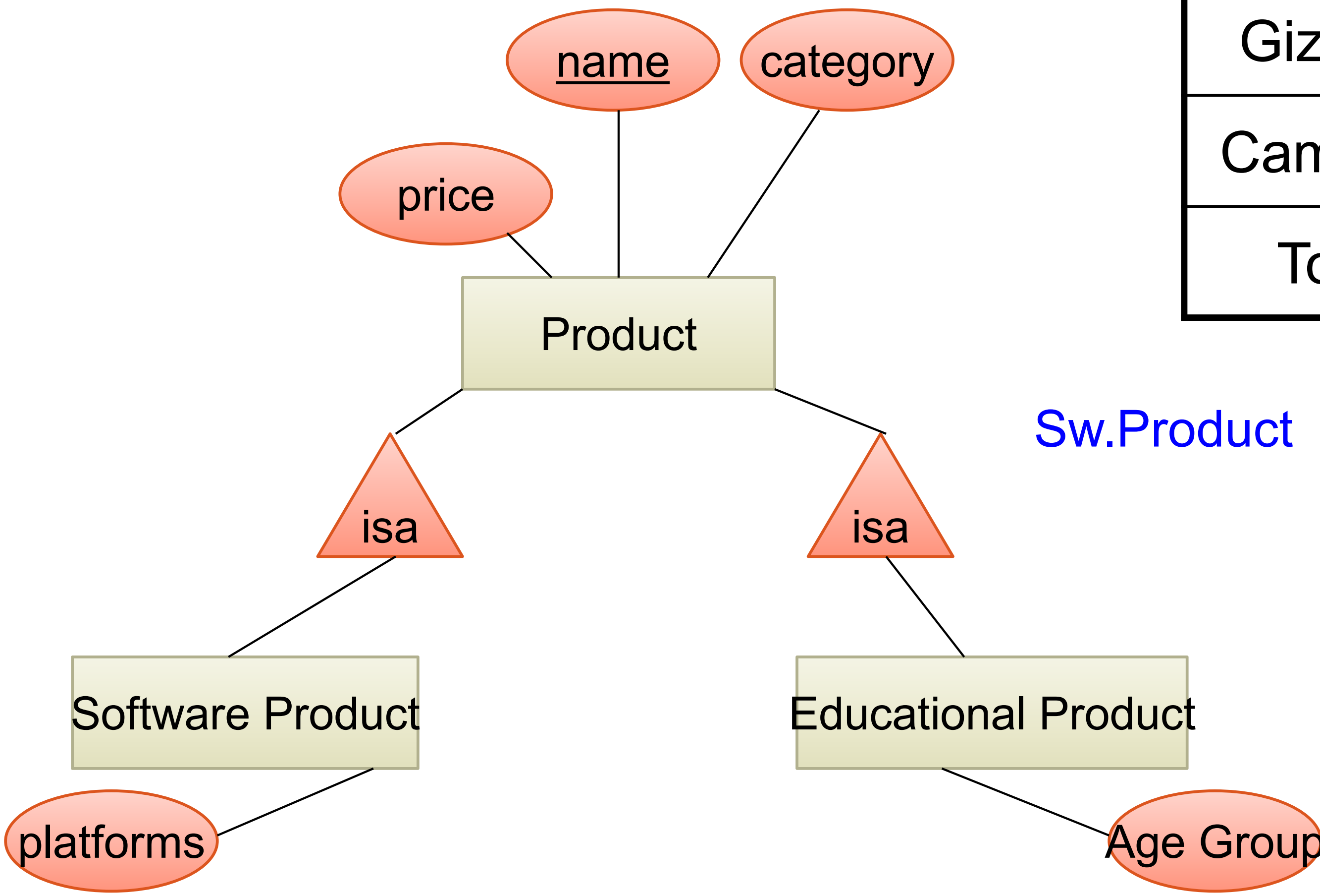
REVIEW

- **Last time, we covered**
  - keys
  - one-to-one, many-to-one, and many-to-many relationships
  - multi-way relationships
  - design principles
    - include all the important entities
    - don’t include the unnecessary ones (e.g. Dates)
    - (lack of) constraints should match reality
  - mapping to tables
    - entity sets become tables
    - many-to-many relationships become tables
    - *-to-one relationships become simple FK references

MODELING SUBCLASSES



MODELING SUBCLASSES



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...

Is this representation subclassing in Java sense?

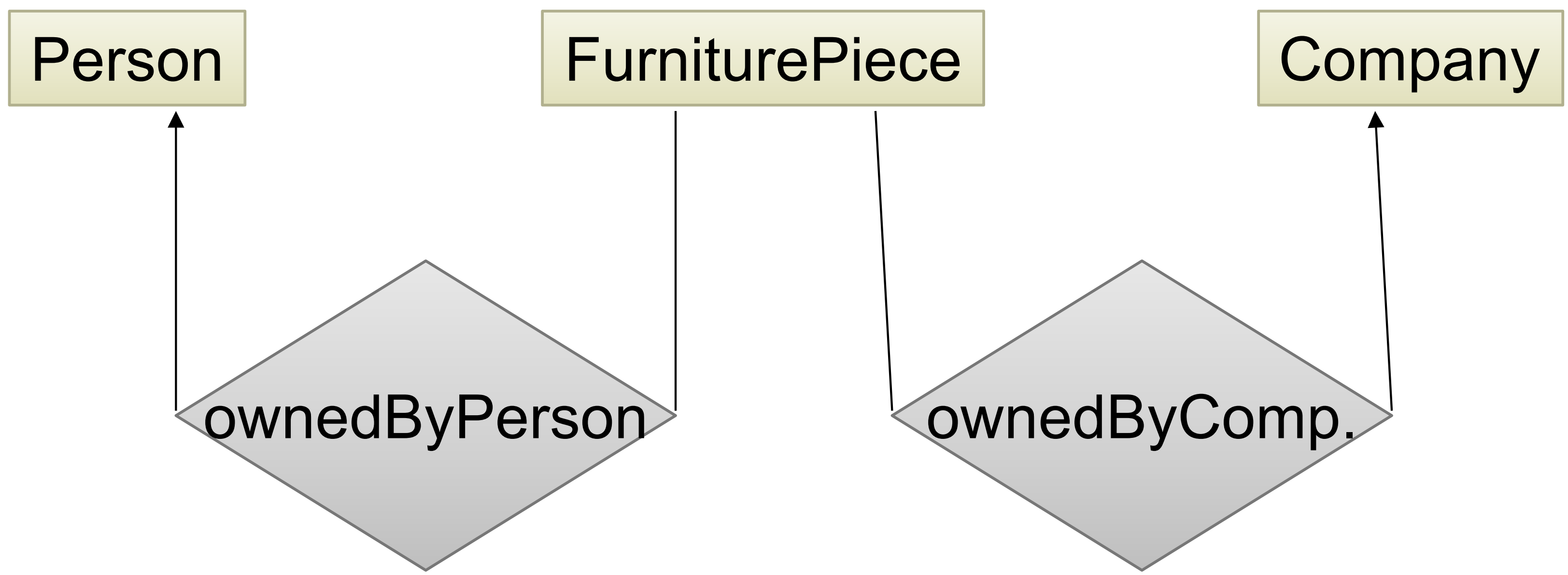
MODELING UNION TYPES WITH SUBCLASSES

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

MODELING UNION TYPES WITH SUBCLASSES

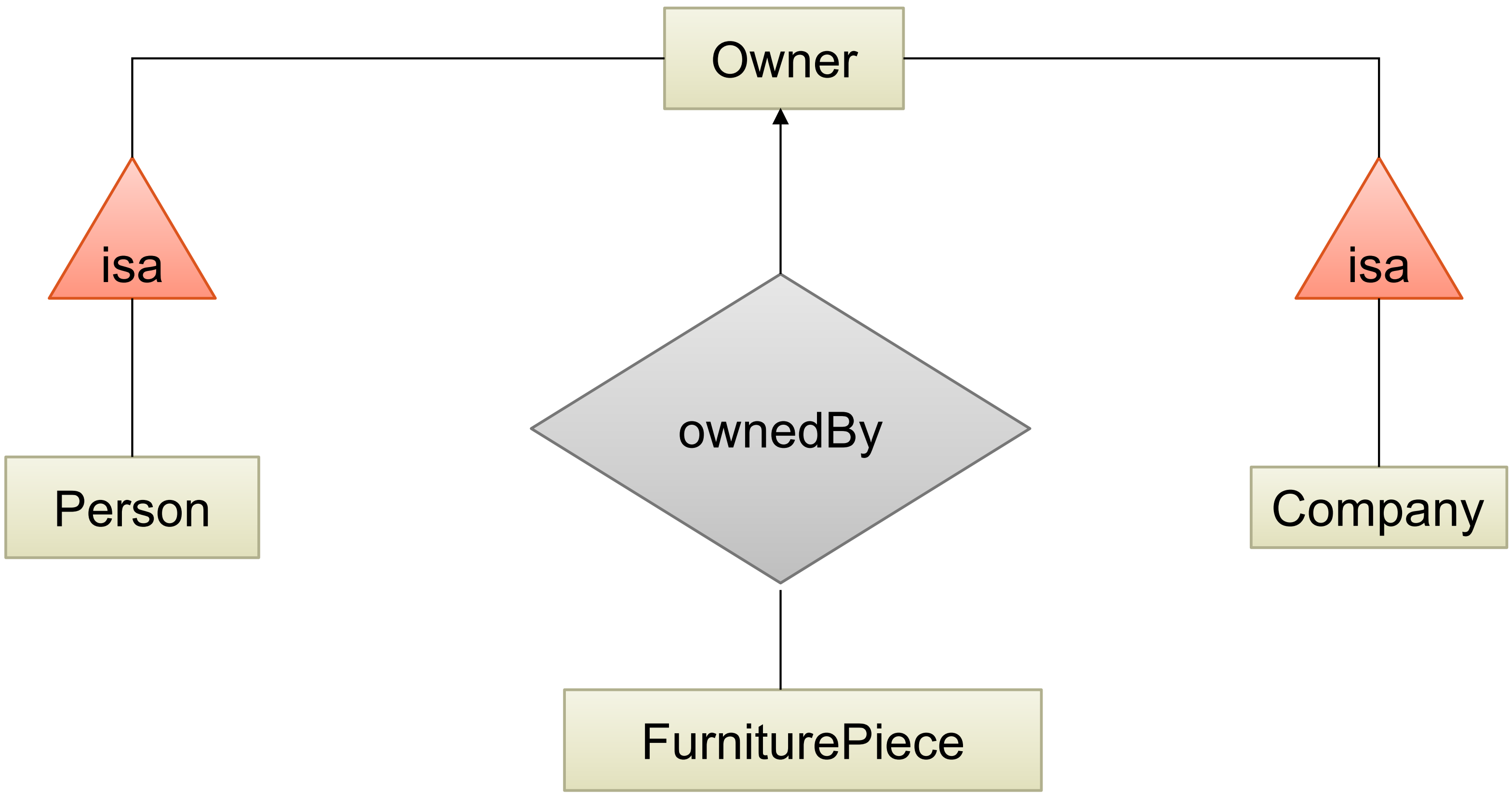
Say: each piece of furniture is owned either by a person or by a company

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



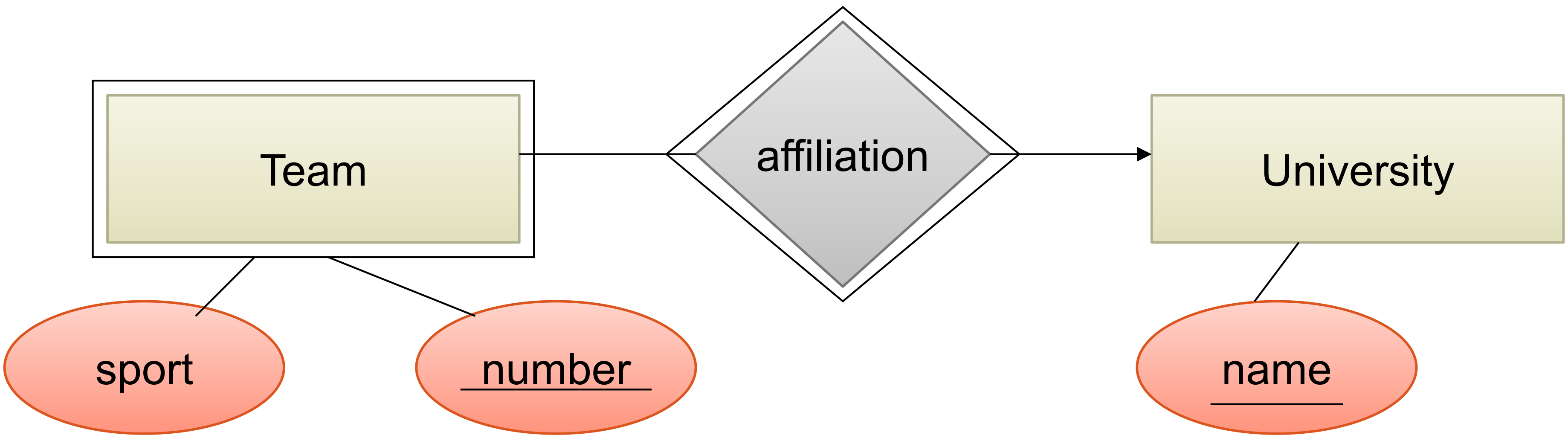
MODELING UNION TYPES WITH SUBCLASSES

Solution 2: better, more laborious



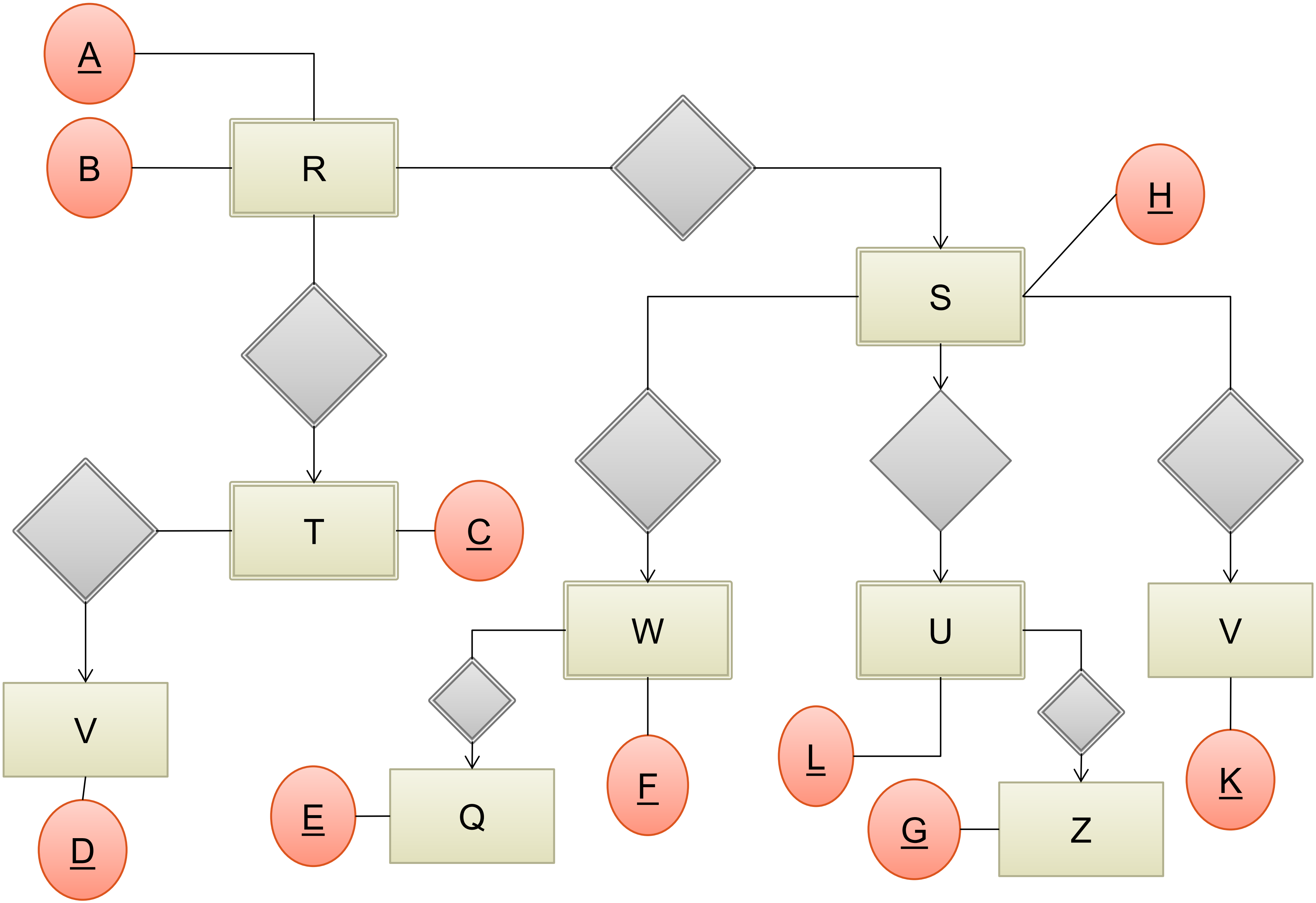
WEAK ENTITY SETS

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 ?



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.

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: a person can have only one biological father.

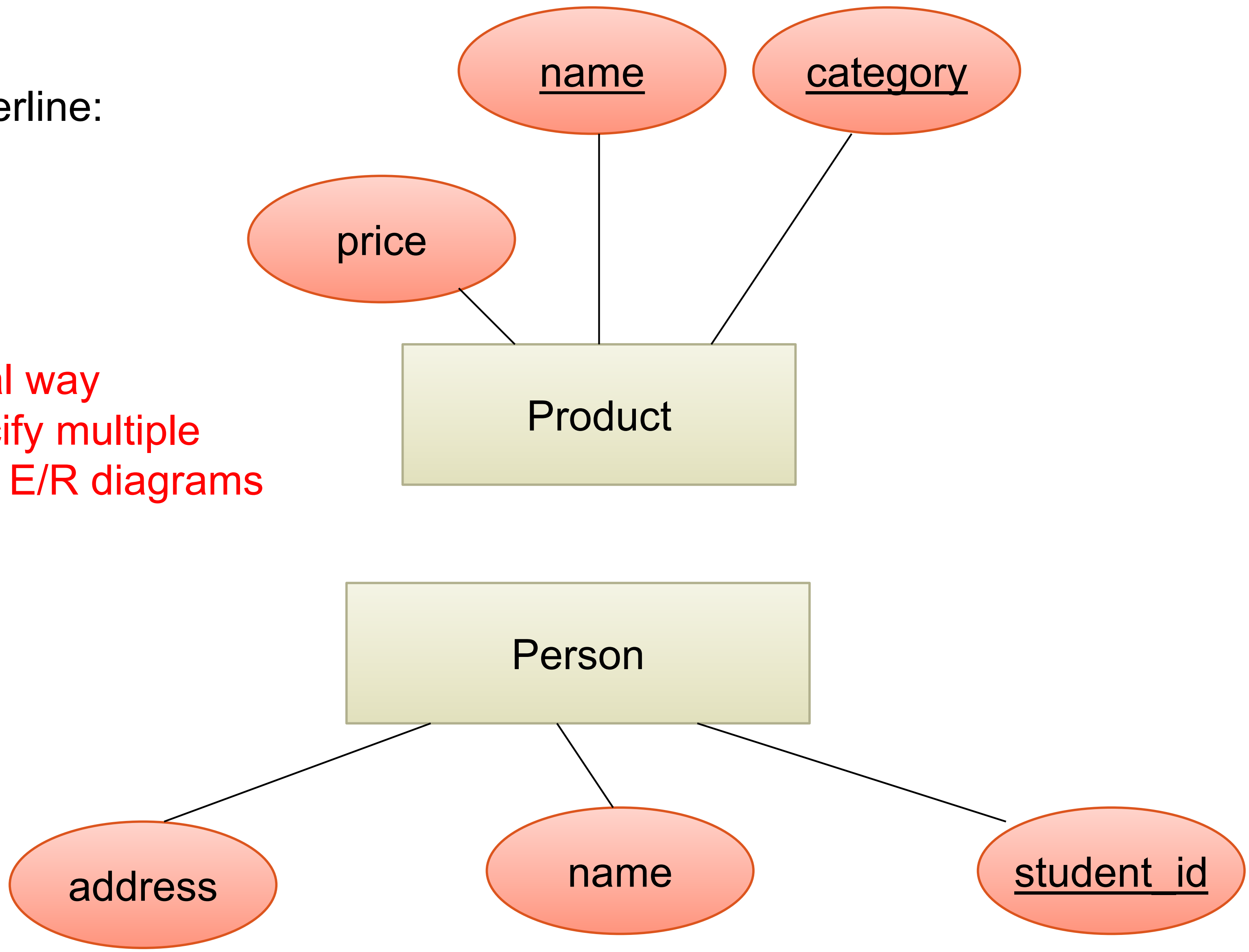
Referential integrity constraints: if you work for a company, it must exist in the database.

Other constraints: peoples' ages are between 0 and 120

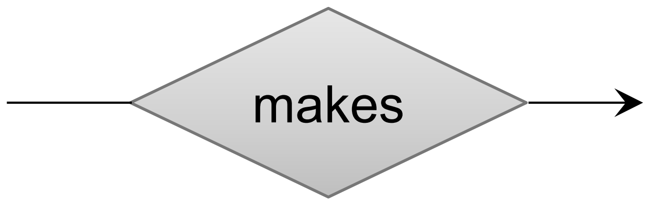
KEYS IN E/R DIAGRAMS

Underline:

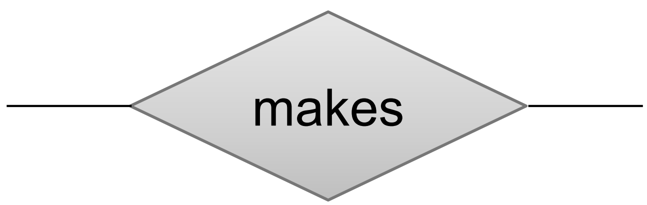
No formal way to specify multiple keys in E/R diagrams



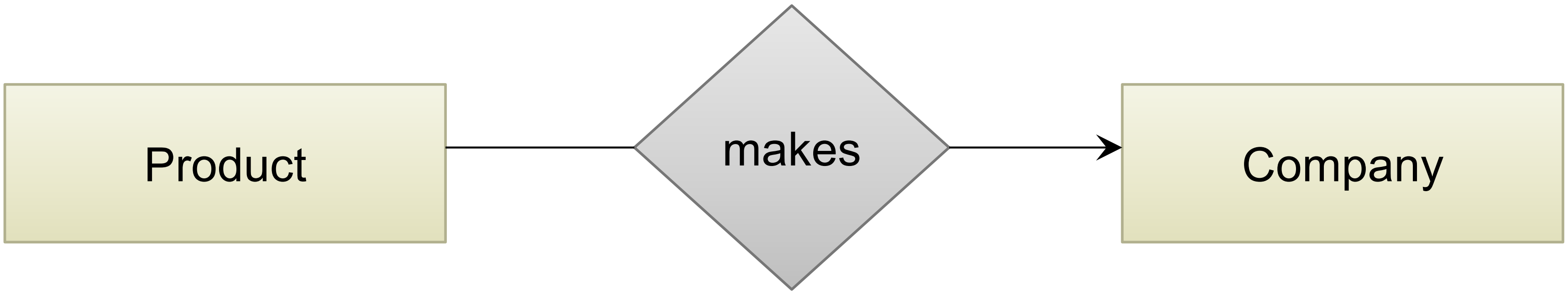
SINGLE VALUE CONSTRAINTS



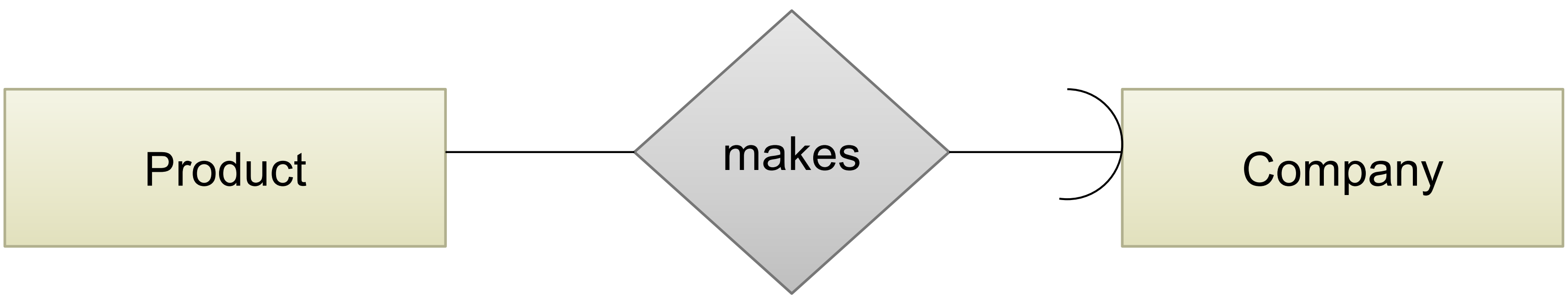
vs.



REFERENTIAL INTEGRITY CONSTRAINTS

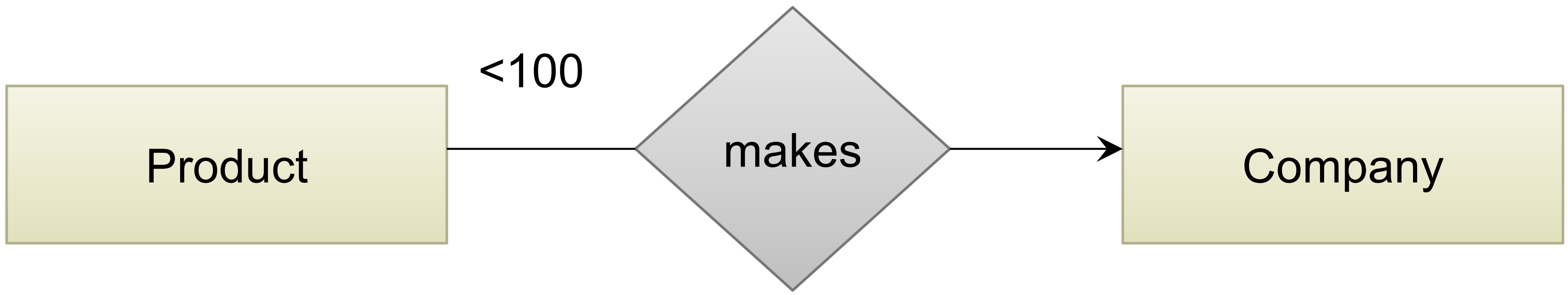


Each product made by at most one company.
Some products made by no company



Each product made by *exactly* one company.

OTHER CONSTRAINTS



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

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(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

- probably want to add NOT NULL (already implied by PRIMARY KEY)

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 has three policies for maintaining referential integrity:

NO ACTION reject violating modifications (default)

CASCADE after delete/update do delete/update

SET NULL set foreign-key field to NULL

SET DEFAULT set foreign-key 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	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 R (
        A int NOT NULL,
        B int CHECK (B > 50 and B < 100),
        C varchar(20),
      D int,
        CHECK (C >= 'd' or D > 0))
```

Attribute constraints are only checked when that attribute changes
Tuple constraint is checked when **any** attribute changes.

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, they provide triggers

NORMALIZATION

RELATIONAL SCHEMA DESIGN

Name	<u>StudentID</u>	<u>PhoneNumber</u>	City
Fred	123456789	206-555-1234	Seattle
Fred	123456789	206-555-6543	Seattle
Joe	987654321	908-555-2121	Westfield

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

Primary key is thus (StudentID, PhoneNumber)

There are problems with this schema...

RELATIONAL SCHEMA DESIGN

Name	StudentID	PhoneNumber	City
Fred	123456789	206-555-1234	Seattle
Fred	123456789	206-555-6543	Seattle
Joe	987654321	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:

Recover as **natural join** of the two tables below

Name	StudentID	PhoneNumber	City
Fred	123456789	206-555-1234	Seattle
Fred	123456789	206-555-6543	Seattle
Joe	987654321	908-555-2121	Westfield

Name	StudentID	City
Fred	123456789	Seattle
Joe	987654321	Westfield

StudentID	PhoneNumber
123456789	206-555-1234
123456789	206-555-6543
987654321	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** $B_1 .. B_m$

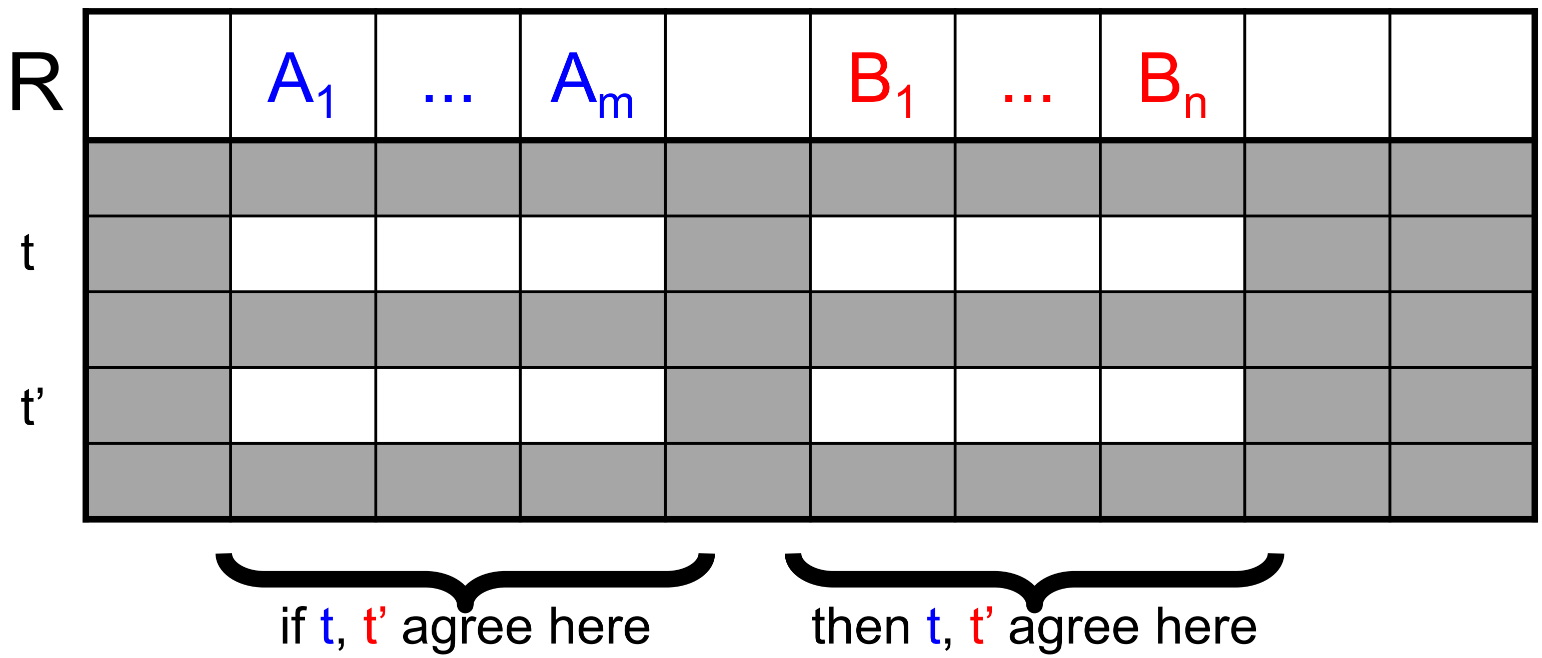
$A_1, A_2, \ldots, A_n \rightarrow B_1, B_2, \ldots, B_m$

FUNCTIONAL DEPENDENCIES (FDS)

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

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

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



FUNCTIONAL DEPENDENCIES (FDS)

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

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

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

Logically equivalent:

$\neg \exists$ t, t' $\in$ R,

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

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

name	category	color	department	price
Gizmo	Gadget	Green	Toys	49
Tweaker	Gadget	Green	Toys	99

Do all the FDs hold on this instance?

EXAMPLE

name → color
category → department
color, category → price

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

What about this one ?

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	StudentID	PhoneNumber	City
Fred	123456789	206-555-1234	Seattle
Fred	123456789	206-555-6543	Seattle
Joe	987654321	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?