CSE 344

MARCH 2ND – E/R DIAGRAMS

ADMINISTRIVIA

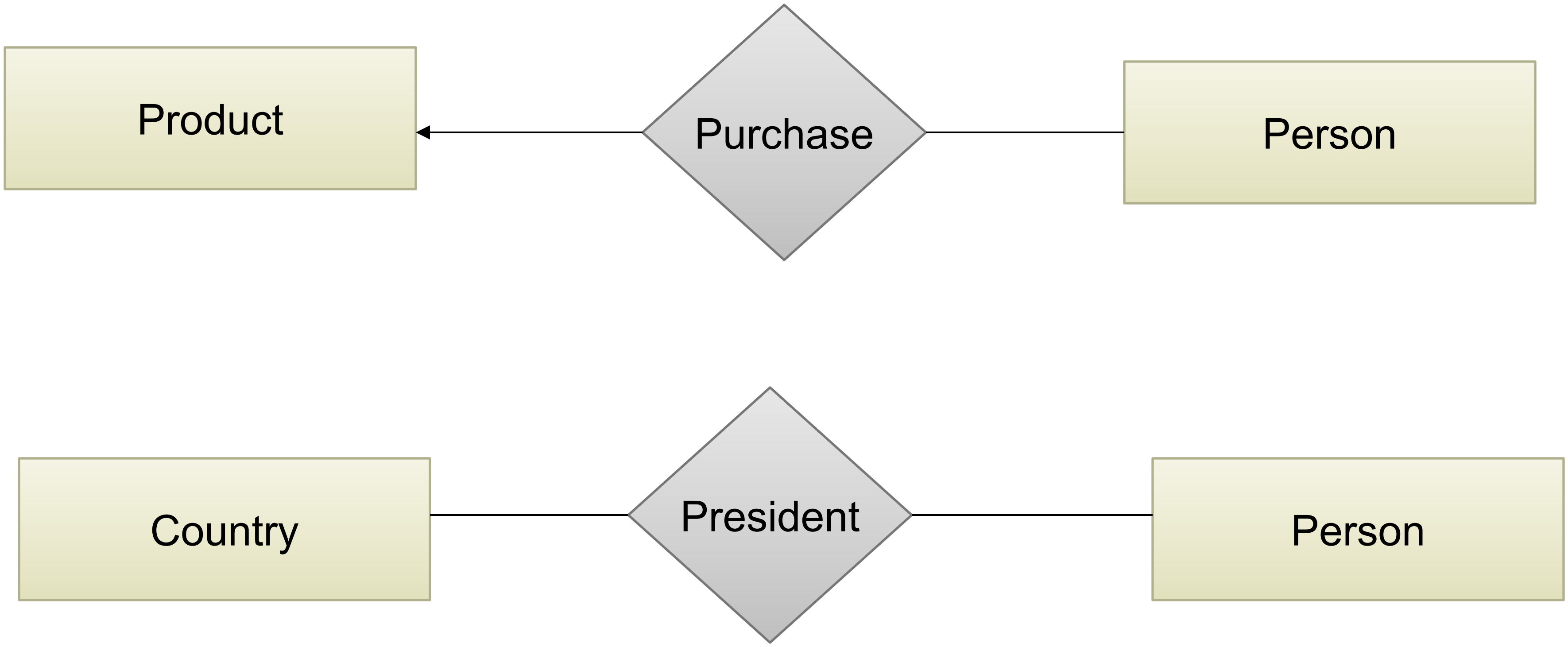
- **All HWs Out**
  - For HW8, if you need additional Azure credit, send me an email
  - Transactions, starting today
  - Only one tag for HW8!

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?)

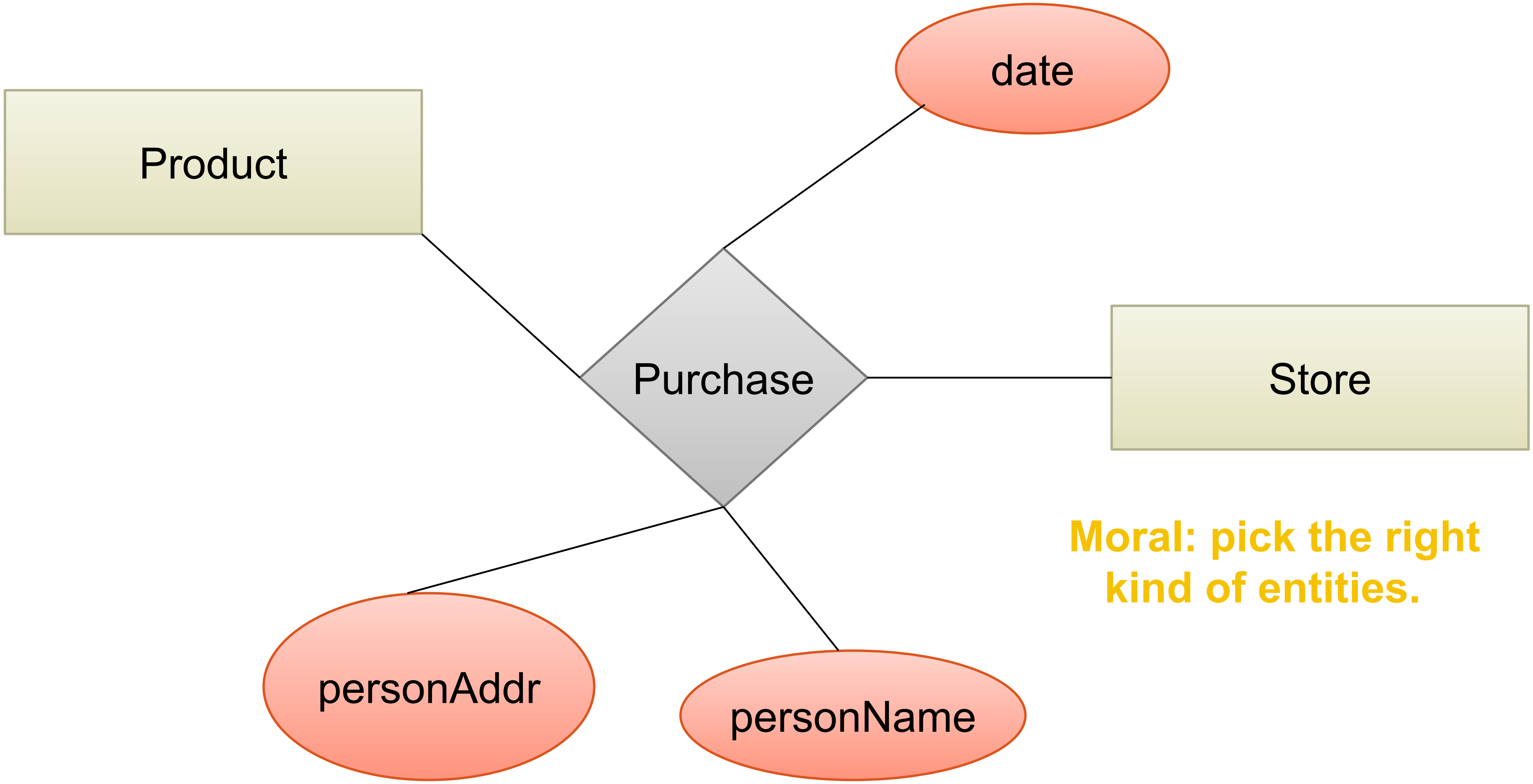
3. DESIGN PRINCIPLES

What's wrong?



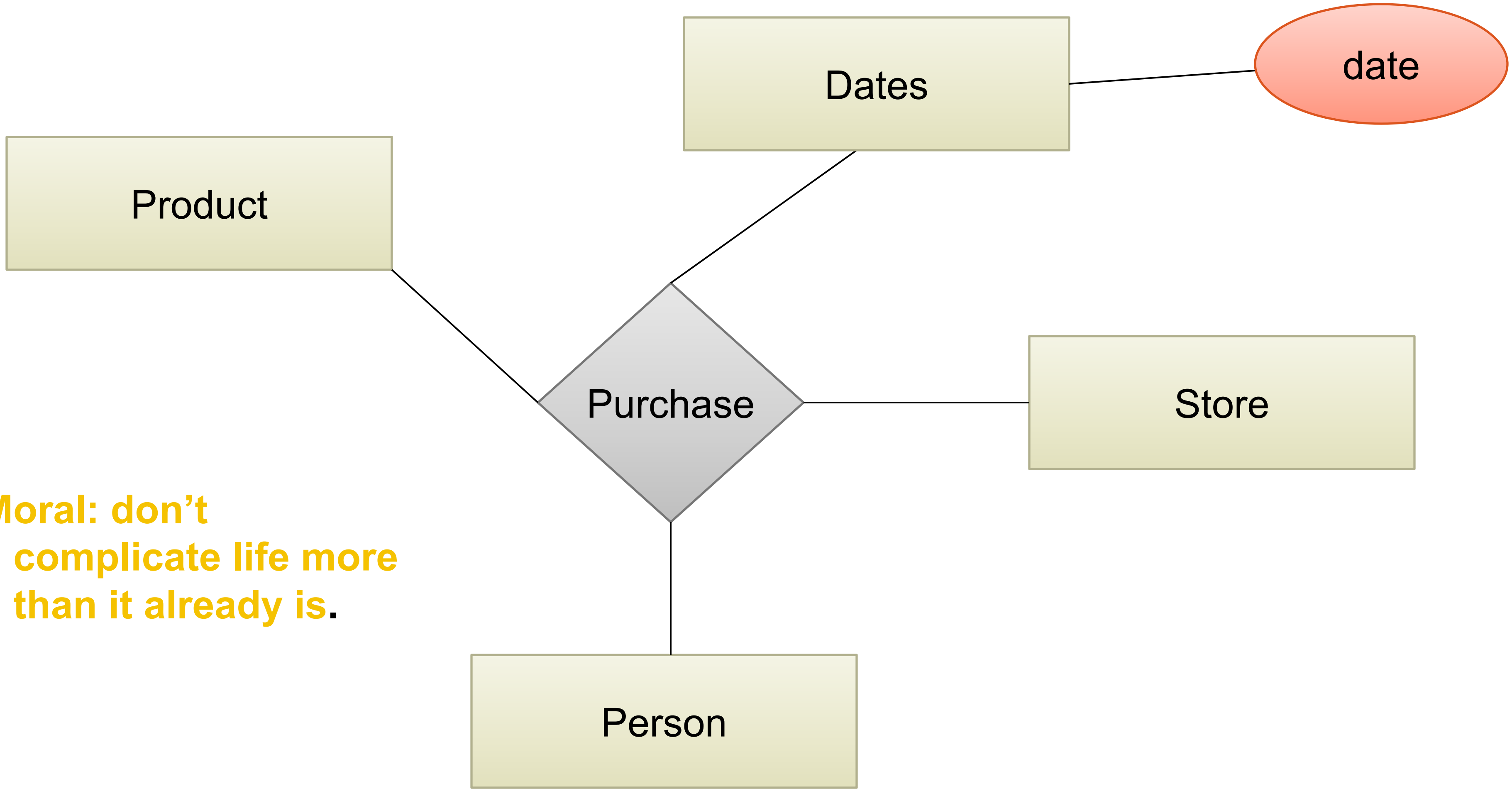
Moral: Be faithful to the specifications of the application!

DESIGN PRINCIPLES: WHAT'S WRONG?



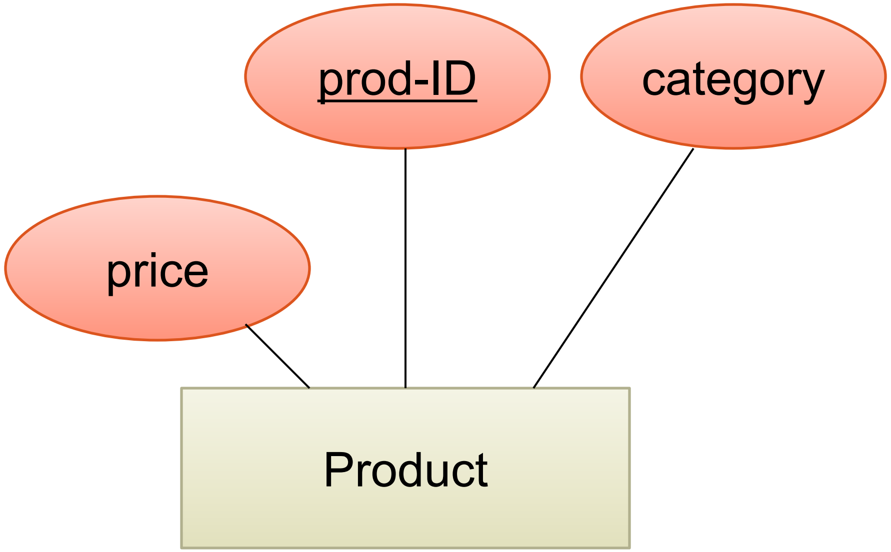
Moral: pick the right kind of entities.

DESIGN PRINCIPLES: WHAT'S WRONG?



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

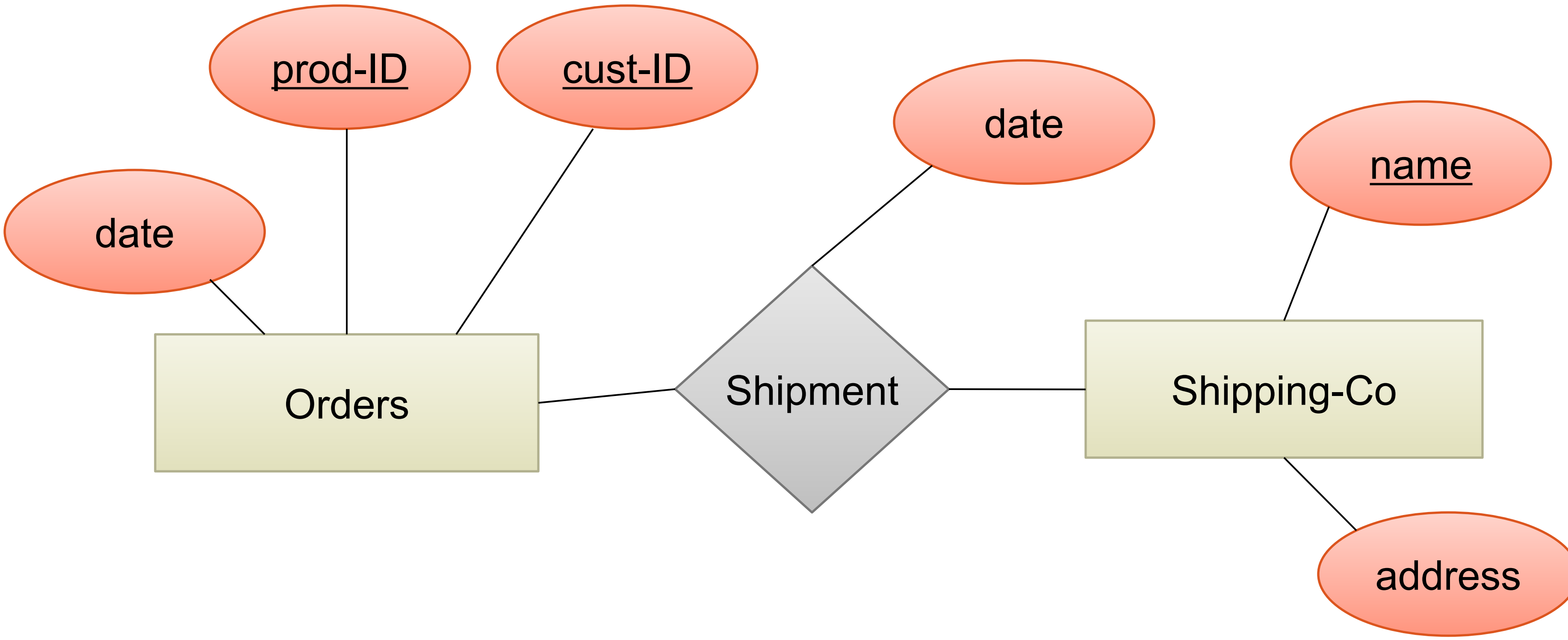
ENTITY SET TO RELATION



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

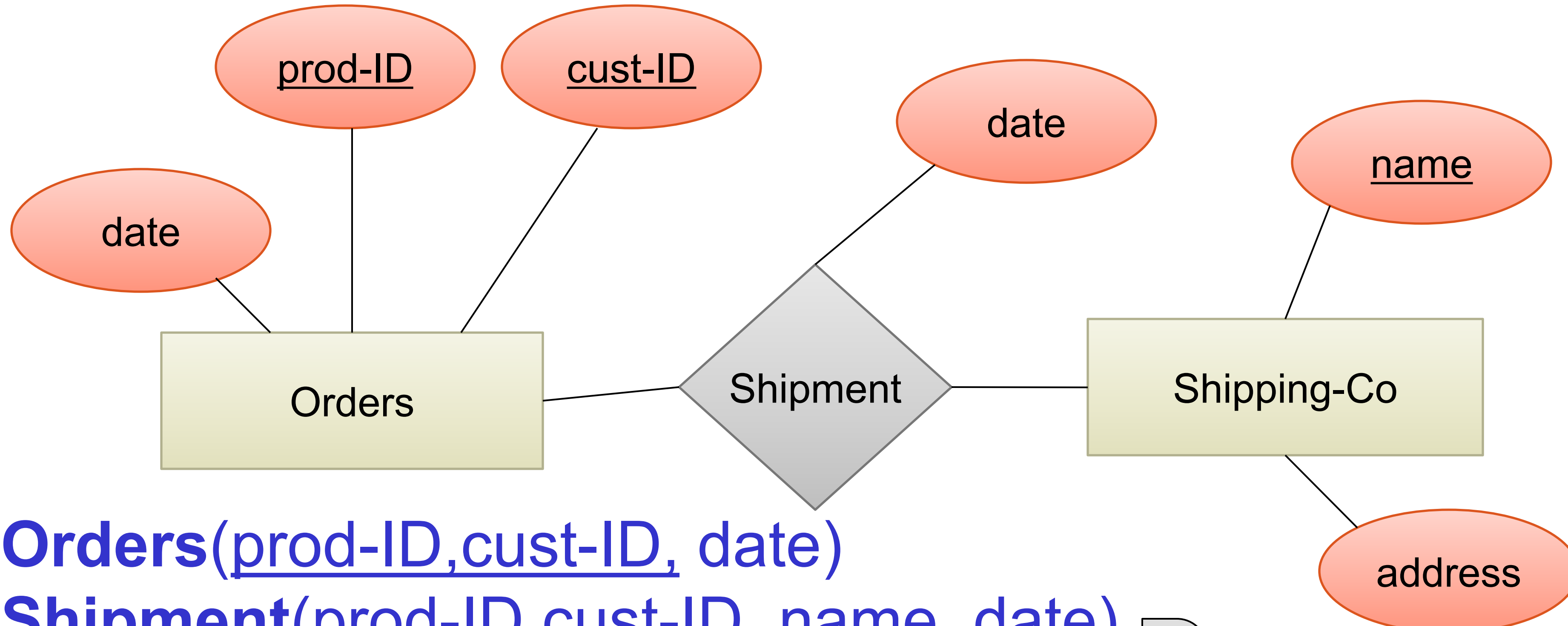
<u>prod-ID</u>	category	price
Gizmo55	Camera	99.99
Pokemn19	Toy	29.99

N-N RELATIONSHIPS TO RELATIONS



Represent this in relations

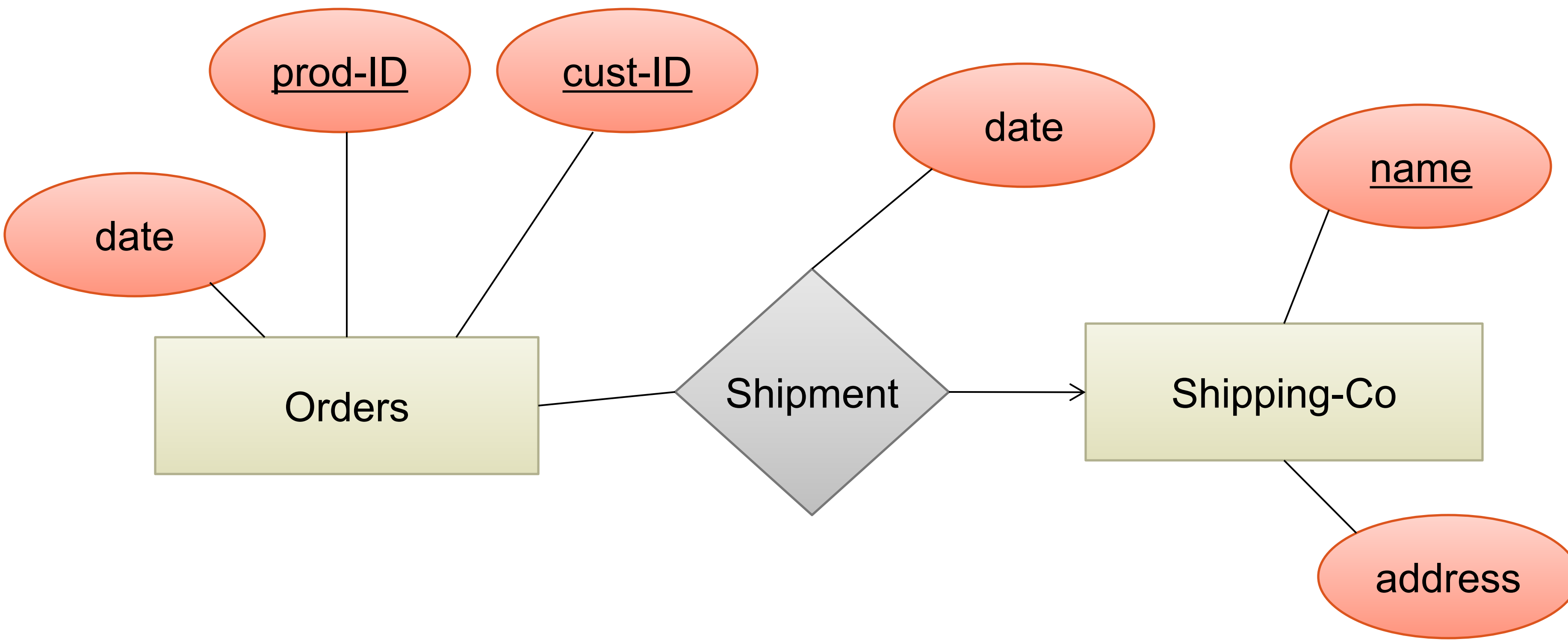
N-N RELATIONSHIPS TO RELATIONS



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

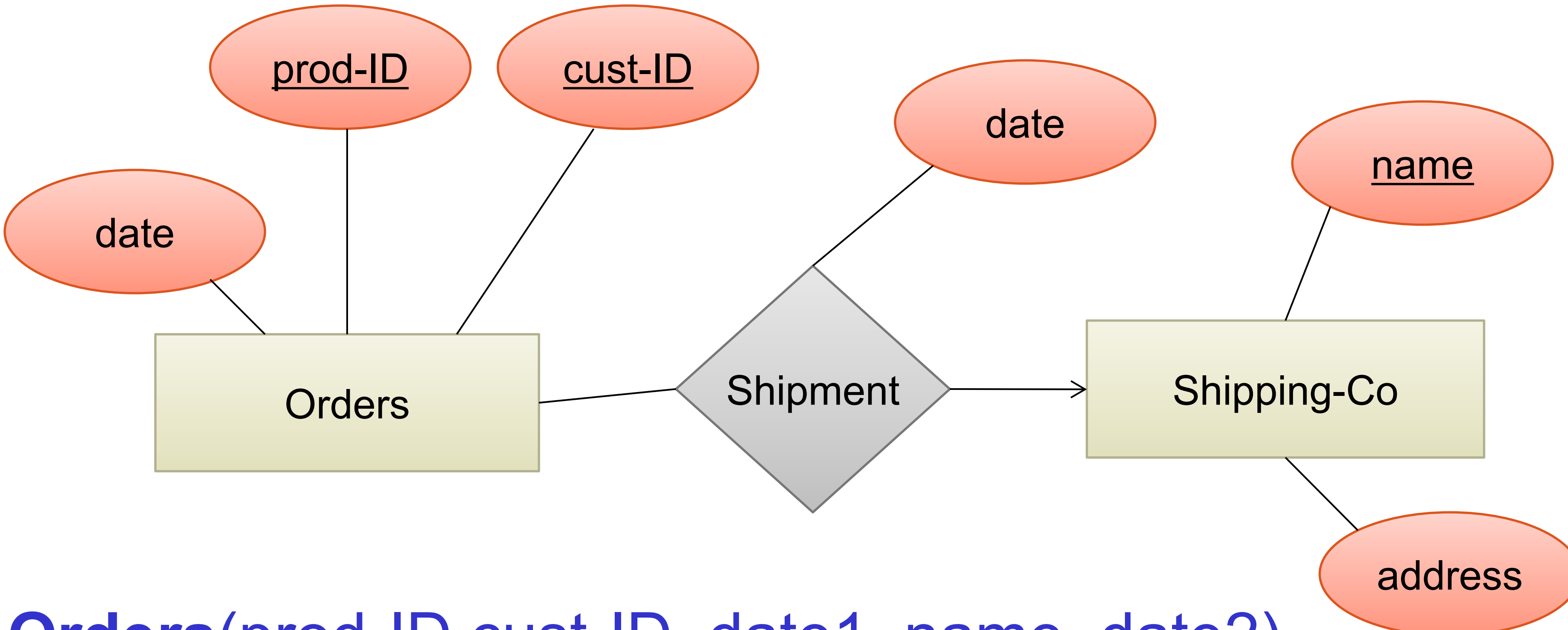
<u>prod-ID</u>	<u>cust-ID</u>	<u>name</u>	date
Gizmo55	Joe12	UPS	4/10/2011
Gizmo55	Joe12	FEDEX	4/9/2011

N-1 RELATIONSHIPS TO RELATIONS



Represent this in relations

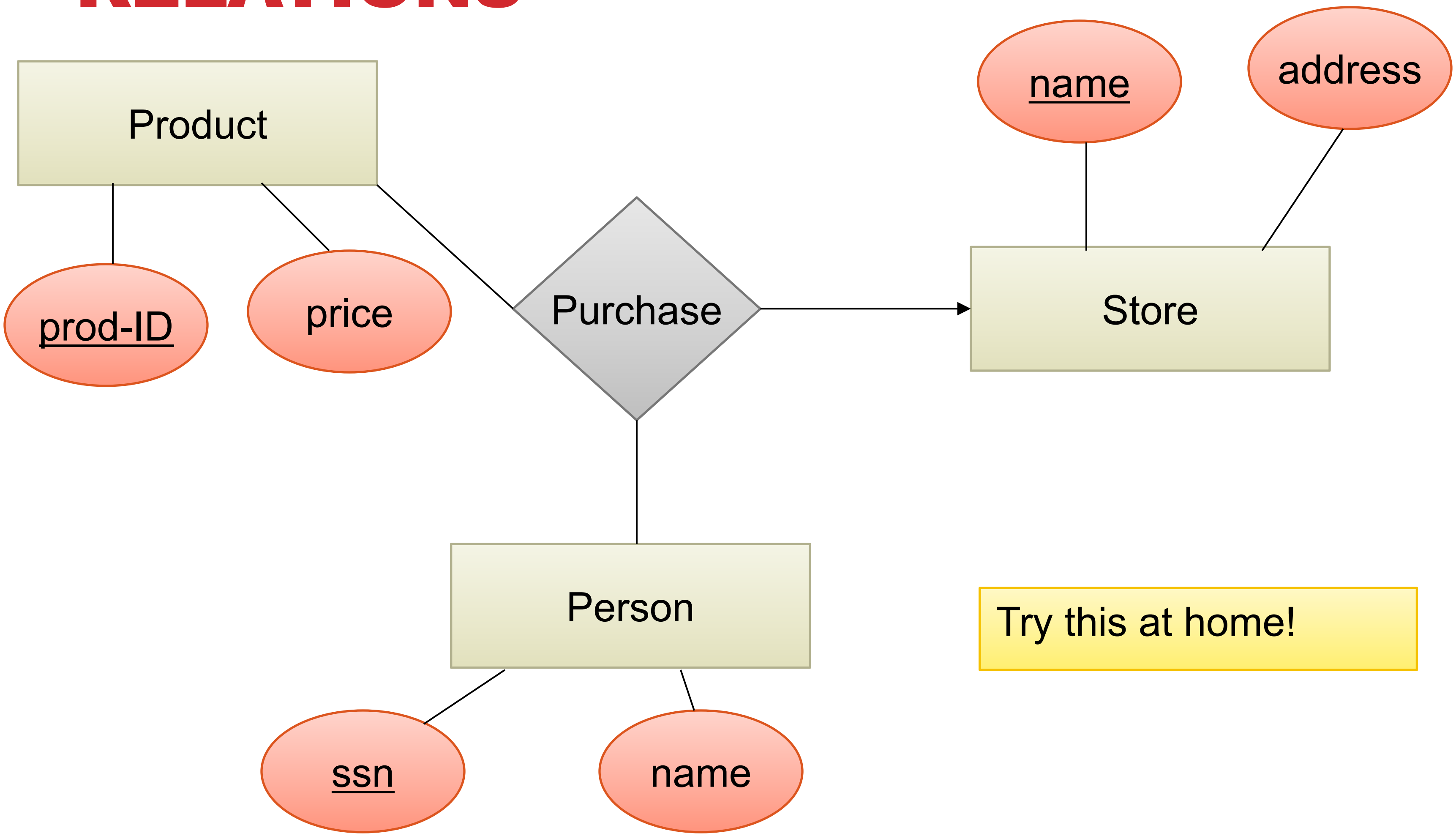
N-1 RELATIONSHIPS TO RELATIONS



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

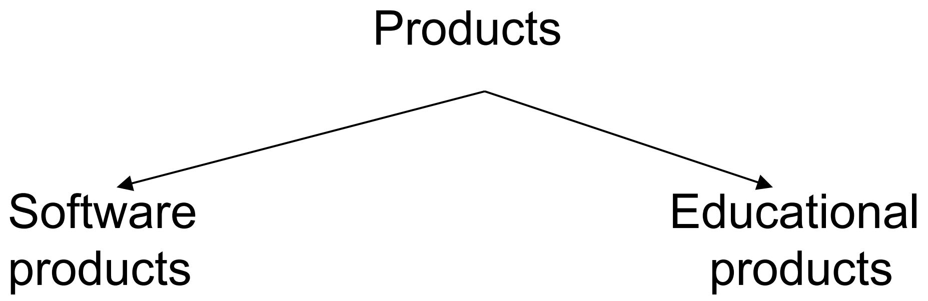


Try this at home!

Purchase(prod-ID, ssn, name)

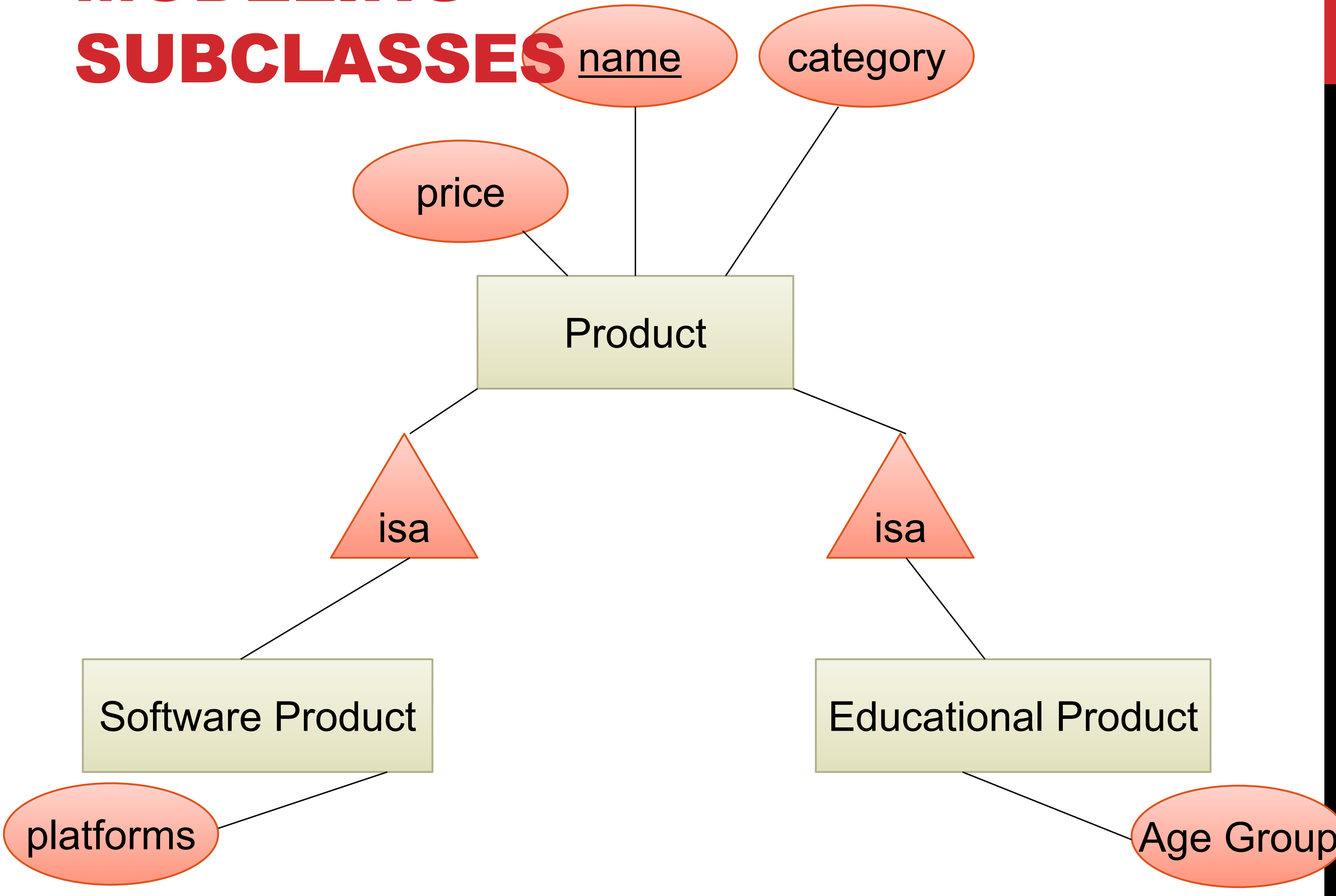
MODELING SUBCLASSES

Some objects in a class may be special
- define a new class
- better: define a *subclass*

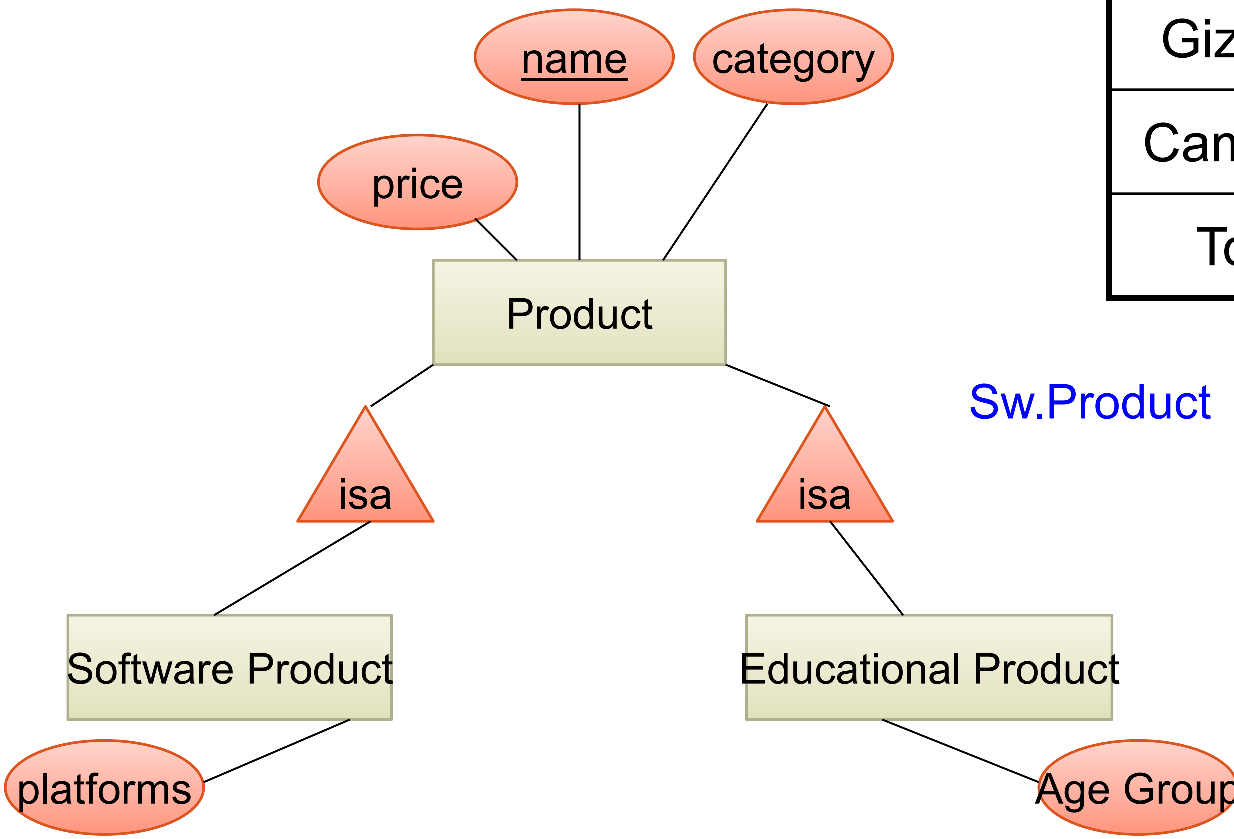


So --- we define subclasses in E/R

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

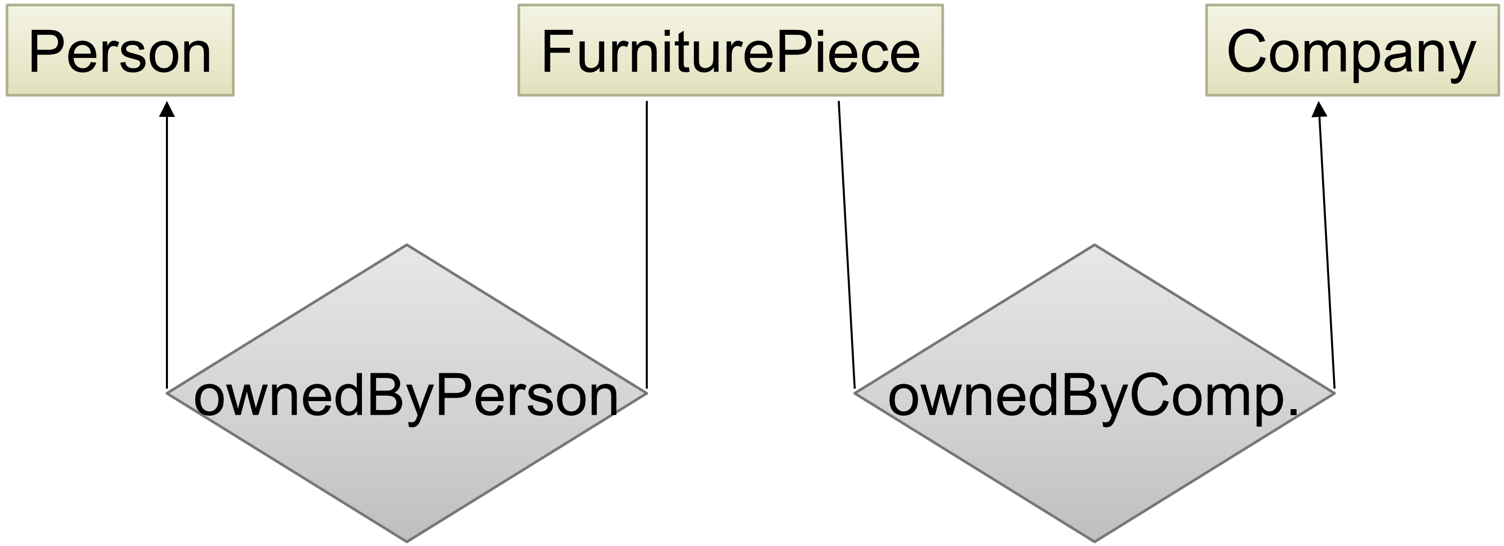
MODELING UNION TYPES WITH SUBCLASSES

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

MODELING UNION TYPES WITH SUBCLASSES

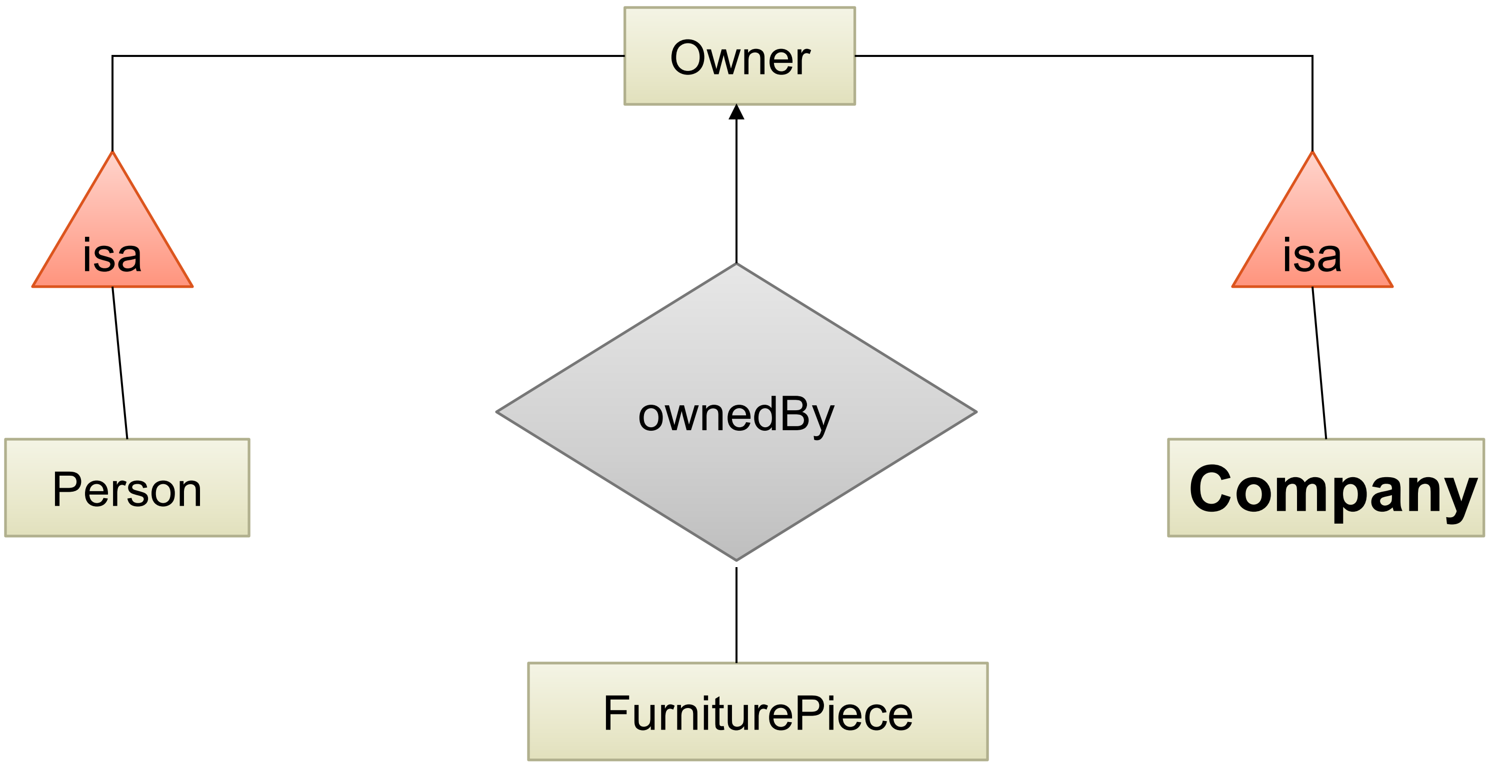
Say: each piece of furniture is owned either by a person or by a company

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



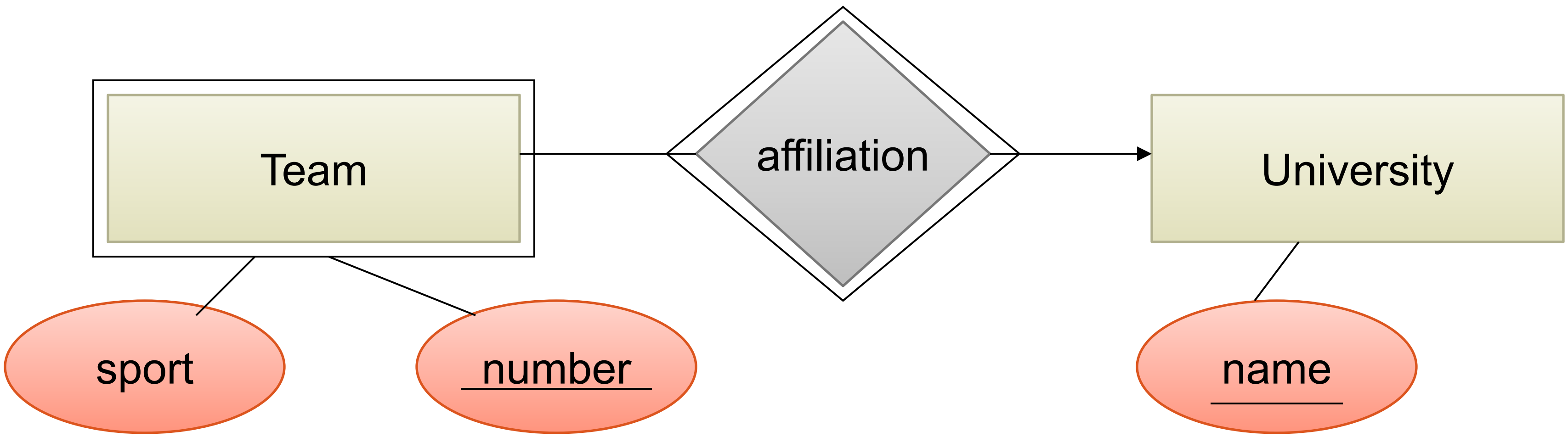
MODELING UNION TYPES WITH SUBCLASSES

Solution 2: better, more laborious



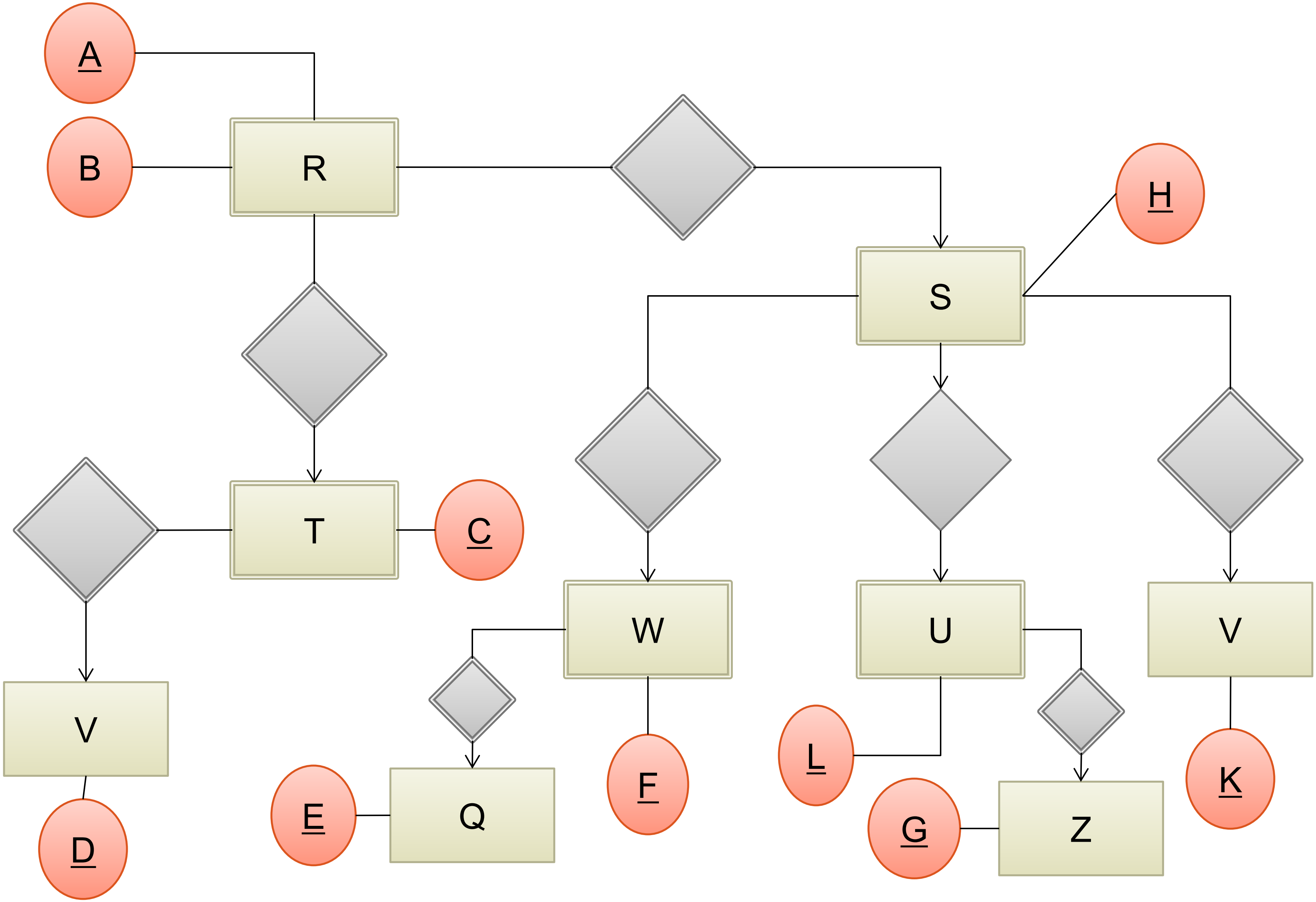
WEAK ENTITY SETS

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



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

WHAT ARE THE KEYS OF R ?



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 father.

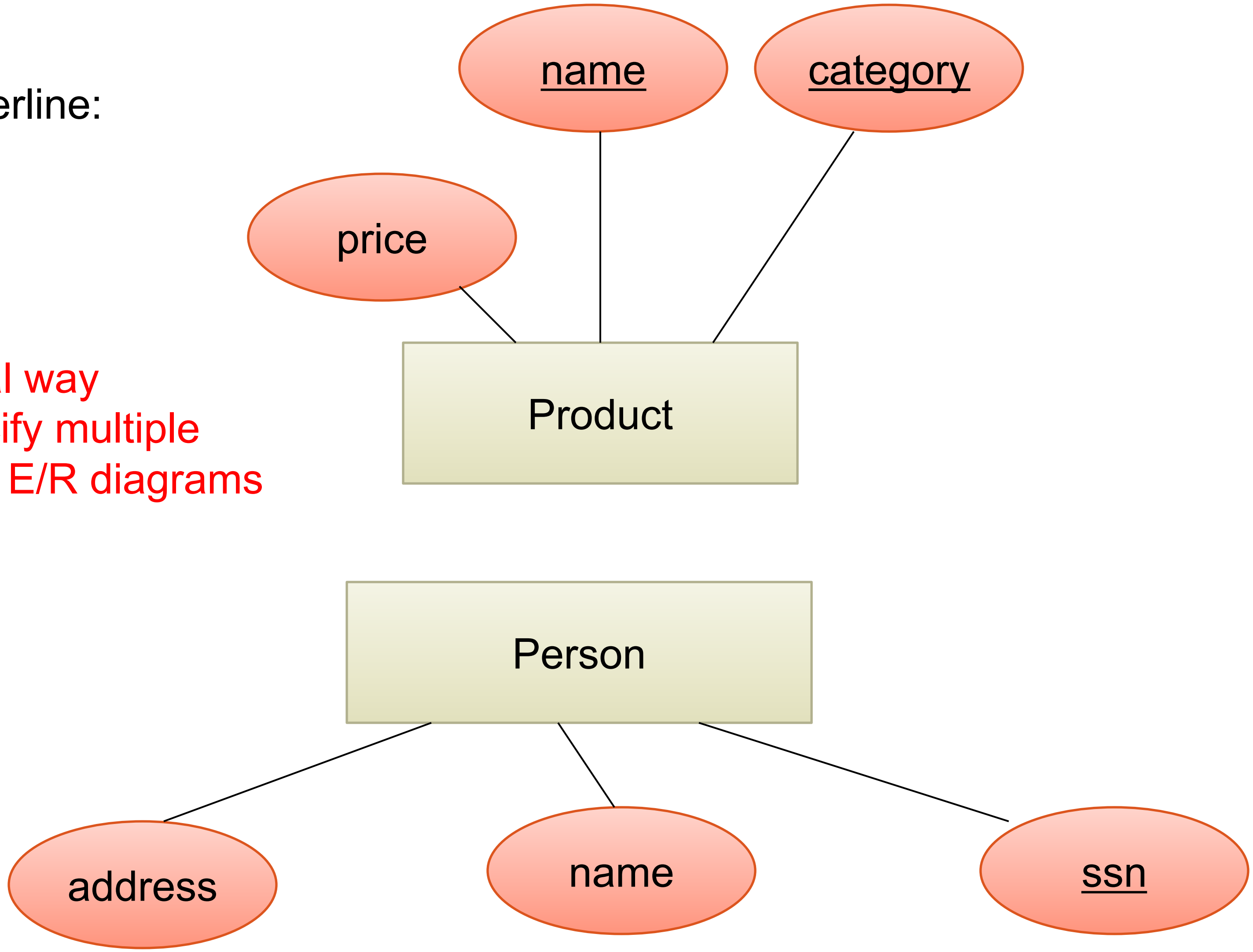
Referential integrity constraints: if you work for a company, it must exist in the database.

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

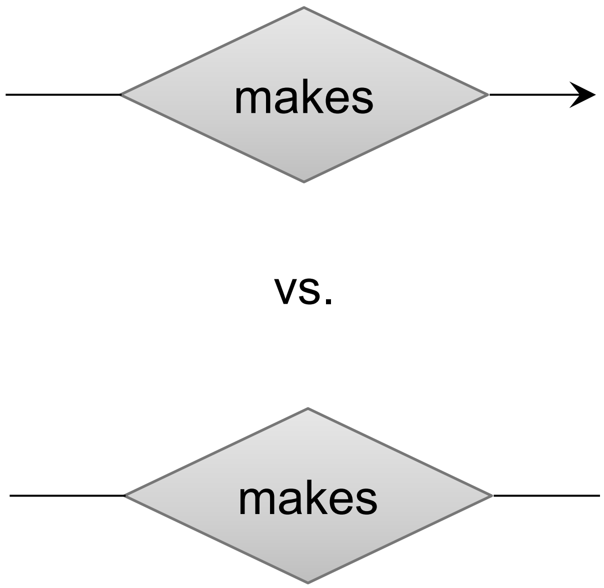
KEYS IN E/R DIAGRAMS

Underline:

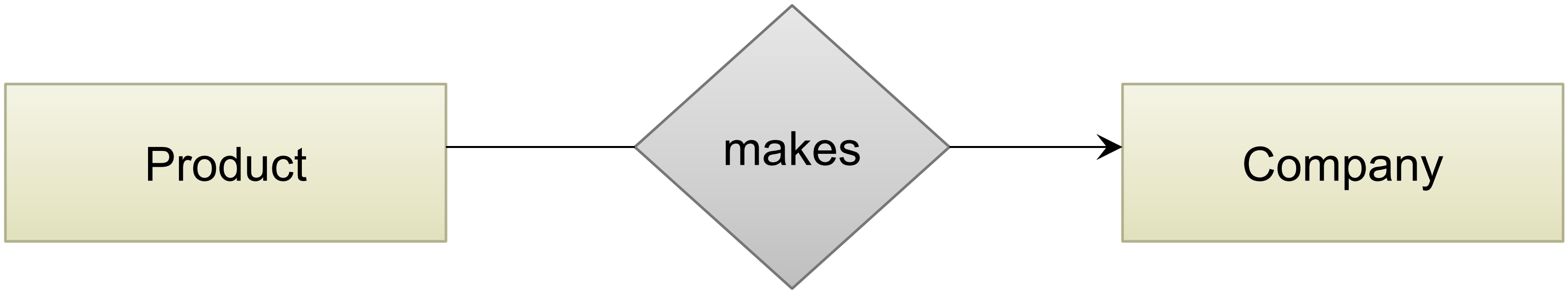
No formal way
to specify multiple
keys in E/R diagrams



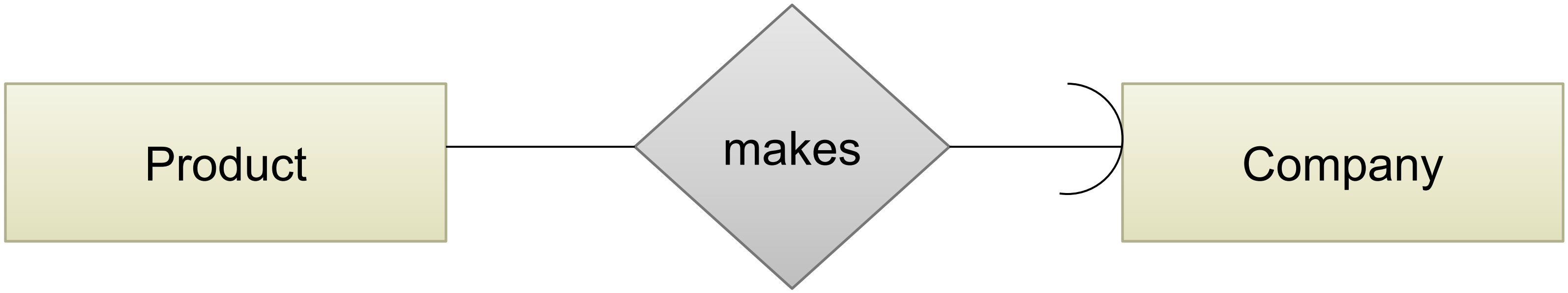
SINGLE VALUE CONSTRAINTS



REFERENTIAL INTEGRITY CONSTRAINTS

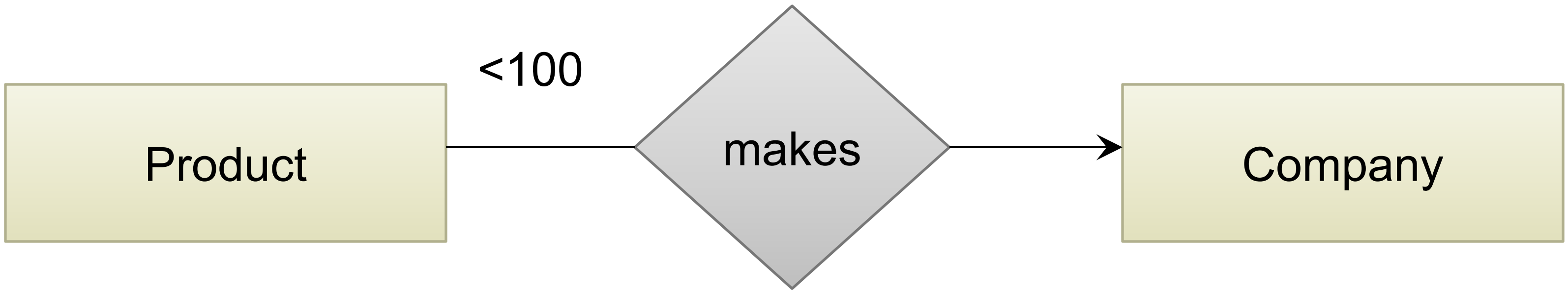


Each product made by at most one company.
Some products made by no company



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

simplest

Attribute-level constraints

Tuple-level constraints

Global constraints: assertions

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(<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 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))
```

CONSTRAINTS ON ATTRIBUTES AND TUPLES

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

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

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

Unit 7: Transactions

- Locking and schedules
- Writing DB applications

TRANSACTIONS

We use database transactions everyday

- Bank $$$ transfers
- Online shopping
- Signing up for classes

For this class, a transaction is a series of DB queries

- Read / Write / Update / Delete / Insert
- Unit of work issued by a user that is independent from others

CHALLENGES

Want to execute many apps concurrently

- All these apps read and write data to the same DB

Simple solution: only serve one app at a time

- What's the problem?

Want: multiple operations to be executed *atomically* over the same DBMS

WHAT CAN GO WRONG?

Manager: balance budgets among projects

- Remove $10k from project A
- Add $7k to project B
- Add $3k to project C

CEO: check company's total balance

- `SELECT SUM(money) FROM budget;`

This is called a dirty / inconsistent read
aka a WRITE-READ conflict

WHAT CAN GO WRONG?

App 1:

```
SELECT inventory FROM products WHERE pid = 1
```

App 2:

```
UPDATE products SET inventory = 0 WHERE pid = 1
```

App 1:

```
SELECT inventory * price FROM products
WHERE pid = 1
```

This is known as an unrepeatable read
aka READ-WRITE conflict

WHAT CAN GO WRONG?

Account 1 = $100
Account 2 = $100
Total = $200

- App 1:
  - Set Account 1 = $200
  - Set Account 2 = $0
- App 2:
  - Set Account 2 = $200
  - Set Account 1 = $0
- At the end:
  - Total = $200

- App 1: Set Account 1 = $200
- App 2: Set Account 2 = $200
- App 1: Set Account 2 = $0
- App 2: Set Account 1 = $0
- At the end:
  - Total = $0

This is called the lost update aka WRITE-WRITE conflict

WHAT CAN GO WRONG?

Paying for Tuition (Underwater Basket Weaving)

- Fill up form with your mailing address
- Put in debit card number (because you don't trust the gov't)
- Click submit
- Screen shows money deducted from your account
- [Your browser crashes]

Lesson:

Changes to the database should be ALL or NOTHING

TRANSACTIONS

Collection of statements that are executed atomically (logically speaking)

```
BEGIN TRANSACTION
  [SQL statements]
COMMIT    or    ROLLBACK (=ABORT)
```

```
[single SQL statement]
```

If BEGIN… missing, then TXN consists of a single instruction

KNOW YOUR TRANSACTIONS: ACID

Atomic

- State shows either all the effects of txn, or none of them

Consistent

- Txn moves from a DBMS state where integrity holds, to another where integrity holds
  - remember integrity constraints?

Isolated

- Effect of txns is the same as txns running one after another (i.e., looks like batch mode)

Durable

- Once a txn has committed, its effects remain in the database

ATOMIC

Definition: A transaction is ATOMIC if all its updates must happen or not at all.

Example: move $100 from A to B

- `UPDATE accounts SET bal = bal – 100 WHERE acct = A;`
- `UPDATE accounts SET bal = bal + 100 WHERE acct = B;`

- ```
  BEGIN TRANSACTION;
  UPDATE accounts SET bal = bal – 100 WHERE acct
  = A;
  UPDATE accounts SET bal = bal + 100 WHERE acct
  = B;
  COMMIT;
  ```
  ```

ISOLATED

- **Definition:**
    - An execution ensures that transactions are isolated, if the effect of each transaction is as if it were the only transaction running on the system.

CONSISTENT

Recall: integrity constraints govern how values in tables are related to each other

- Can be enforced by the DBMS, or ensured by the app

How consistency is achieved by the app:

- App programmer ensures that txns only takes a consistent DB state to another consistent state
- DB makes sure that txns are executed atomically

Can defer checking the validity of constraints until the end of a transaction

DURABLE

A transaction is durable if its effects continue to exist after the transaction and even after the program has terminated

How?

- By writing to disk!
- More in 444

ROLLBACK TRANSACTIONS

If the app gets to a state where it cannot complete the transaction successfully, execute ROLLBACK

The DB returns to the state prior to the transaction

What are examples of such program states?

ACID

Atomic

Consistent

Isolated

Durable

Again: by default each statement is its own txn

- Unless auto-commit is off then each statement starts a new txn