The Relational Model

CS 564- Fall 2015

How To Build a DB Application

- Pick an application
- Figure out what to model (ER model)
  - Output: ER diagram
- Transform the ER diagram to a relational schema
- Refine the relational schema (normalization)
- Now ready to implement the schema and load the data!

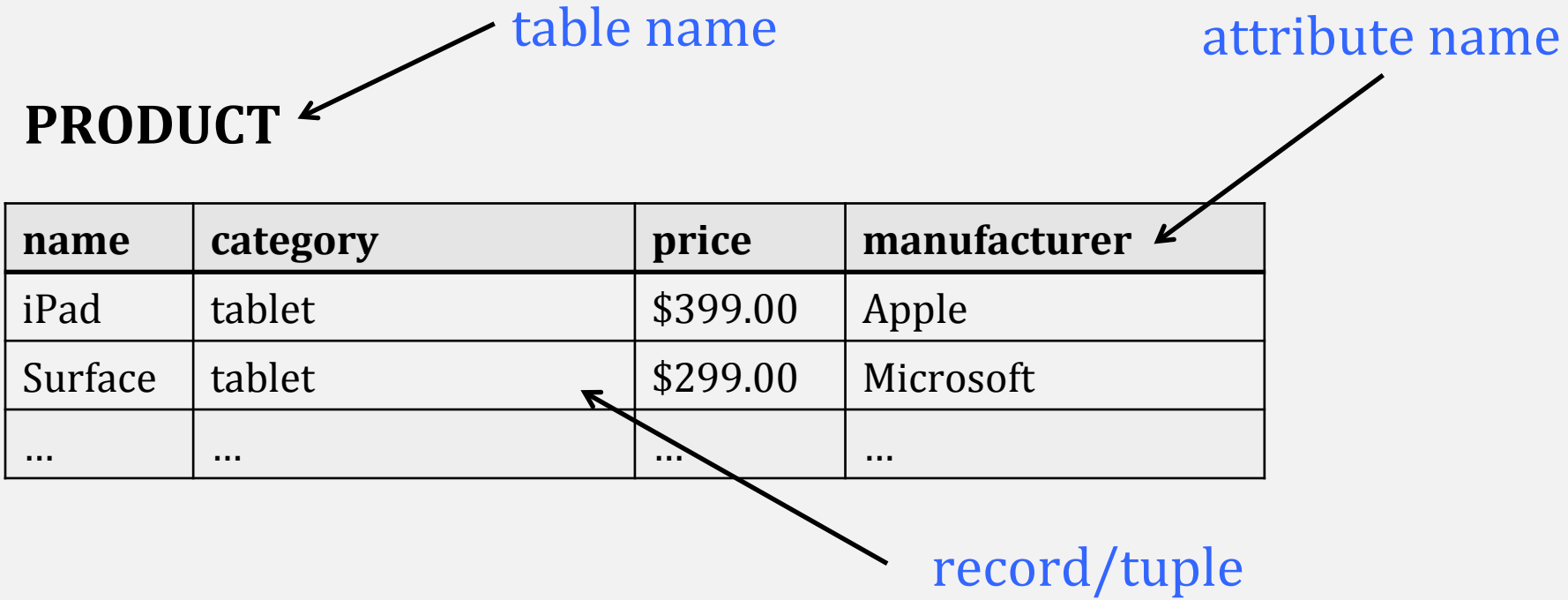
ER Model vs Relational Model

- **ER model**
  - many concepts: entities, relations, attributes, etc
  - well-suited for capturing the app requirements
  - not well-suited for computer implementation
- **Relational model**
  - has just a single concept: relation
  - world is represented with a collection of tables
  - well-suited for efficient manipulations on computers

Relational Model: Basics

RELATION

The data is stored in tables (relations)



DOMAINS

- Each attribute has an atomic type called domain
- A domain specifies the set of values allowed
- Examples:
  - integer
  - string
  - real

PRODUCT(name: *string*,
category: *string*,
price: *real*,
manufacturer: *string*)

SCHEMA

The schema of a relation:

- relation name + attribute names
- **Product**(name, price, category, manufacturer)
- In practice we add the domain for each attribute

The schema of a database

- a collection of relation schemas

INSTANCE

The instance of a relation:

- a set of tuples or records

The instance of a database

- a collection of relation instances

EXAMPLE

PRODUCT(name: *string*,
category: *string*,
price: *real*,
manufacturer: *string*)

schema

instance

name	category	price	manufacturer
iPad	tablet	$399.00	Apple
Surface	tablet	$299.00	Microsoft
…	…	…	…

SCHEMA VS INSTANCE

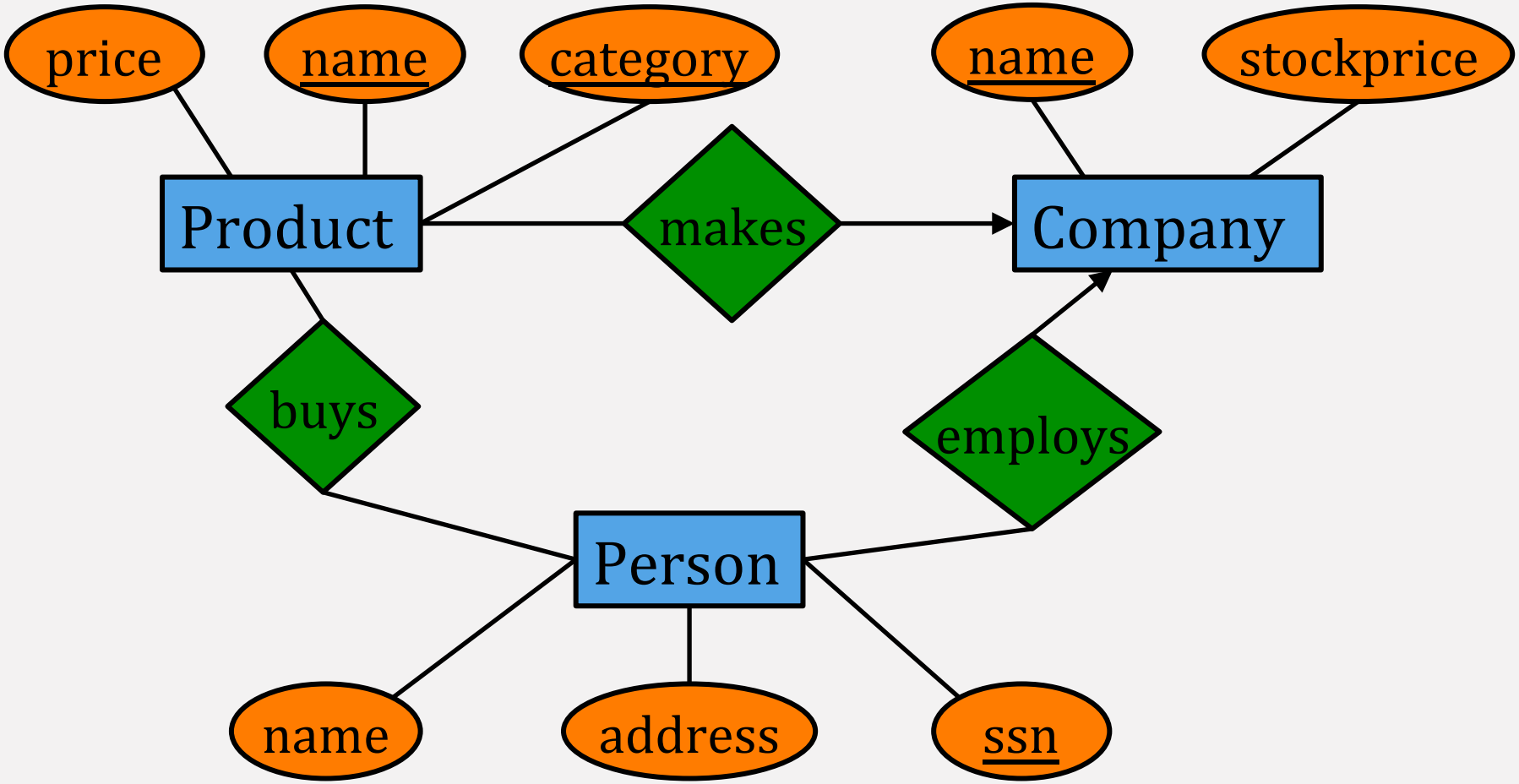
- Analogy with programming languages:
  - schema = type
  - instance = value
- Important distinction:
  - schema = stable over long periods of time
  - instance = changes constantly, as data is inserted/updated/deleted

ER to Relational Model

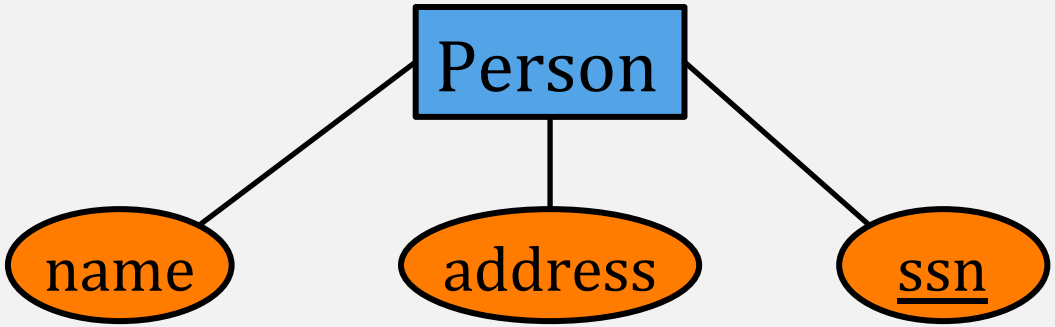
TRANSLATION

- Basic cases:
  - entity set **E** -- > relation with attributes of **E**
  - relationship **R** -- > relation with attributes being keys of related entity sets + attributes of **R**
- Special cases:
  - combining two relations
  - weak entity sets
  - is-a relationships

RUNNING EXAMPLE



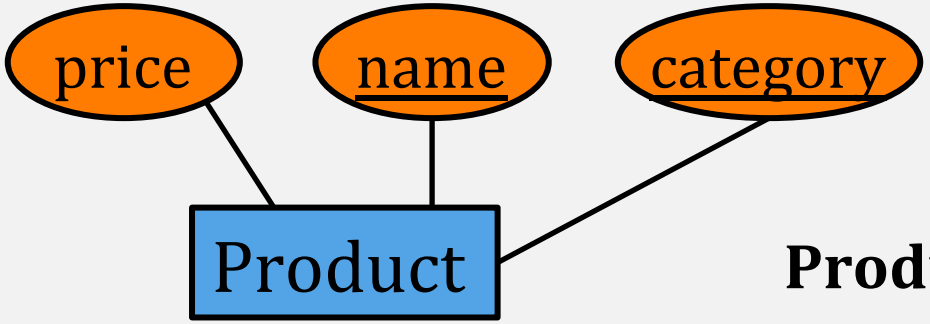
ENTITY SET TO RELATION



Person(ssn, name, address)

```
CREATE TABLE Person (ssn CHAR(11) PRIMARY KEY,
                     name CHAR(40),
                     address CHAR(50))
```

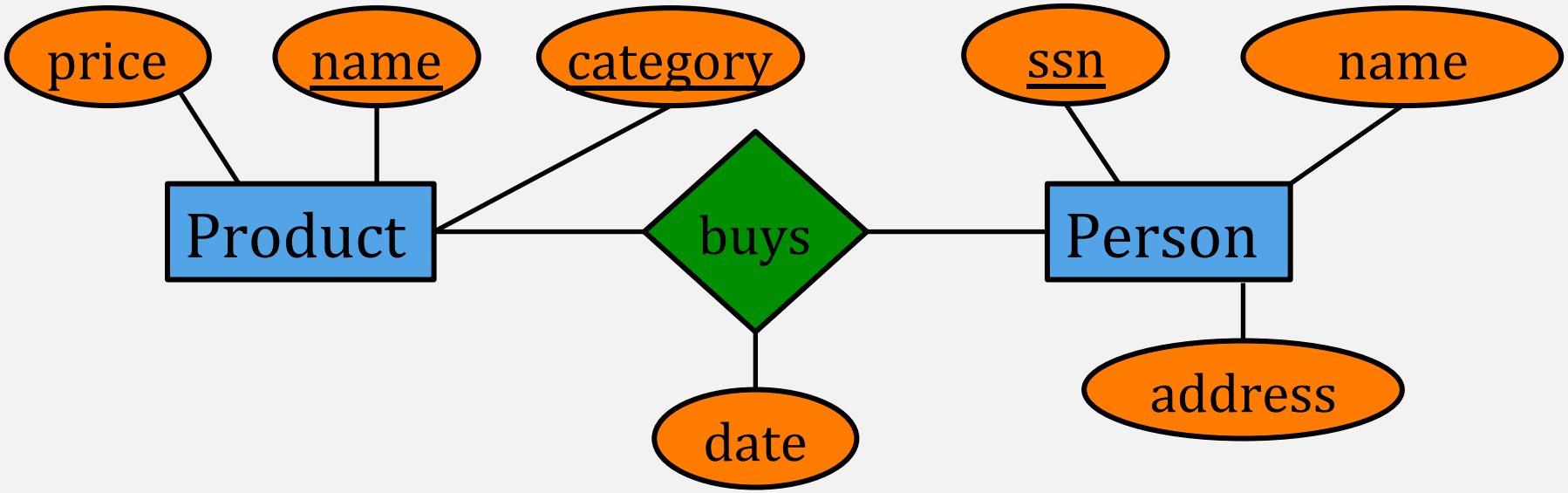
ENTITY SET TO RELATION



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

```
CREATE TABLE Product (name CHAR(40),
                      category CHAR(20),
                      price REAL,
                      PRIMARY KEY(name, category))
```

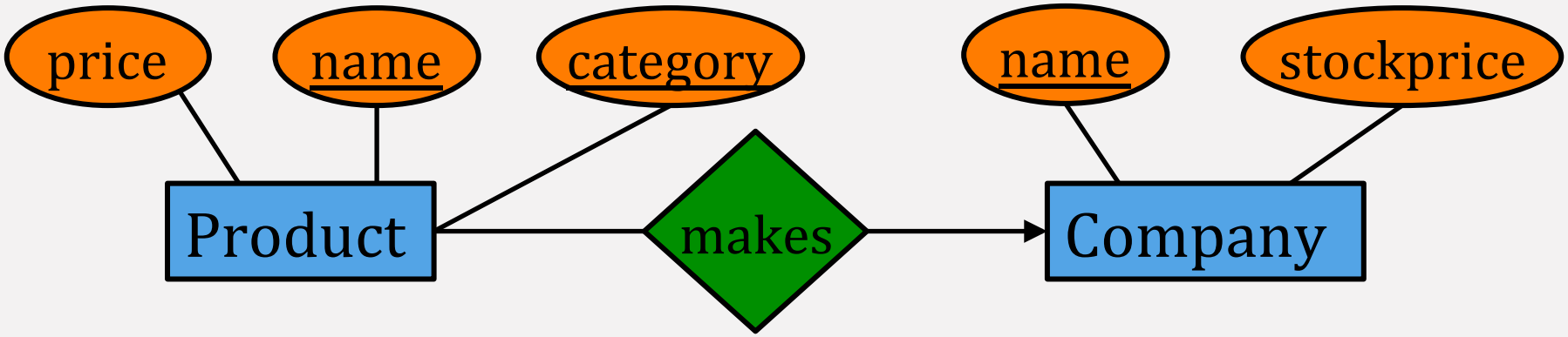
Relationship to Relation



Product(name, category, price)
Person(ssn, name, address)

Buys(prodname, prodcategory, ssn, date)

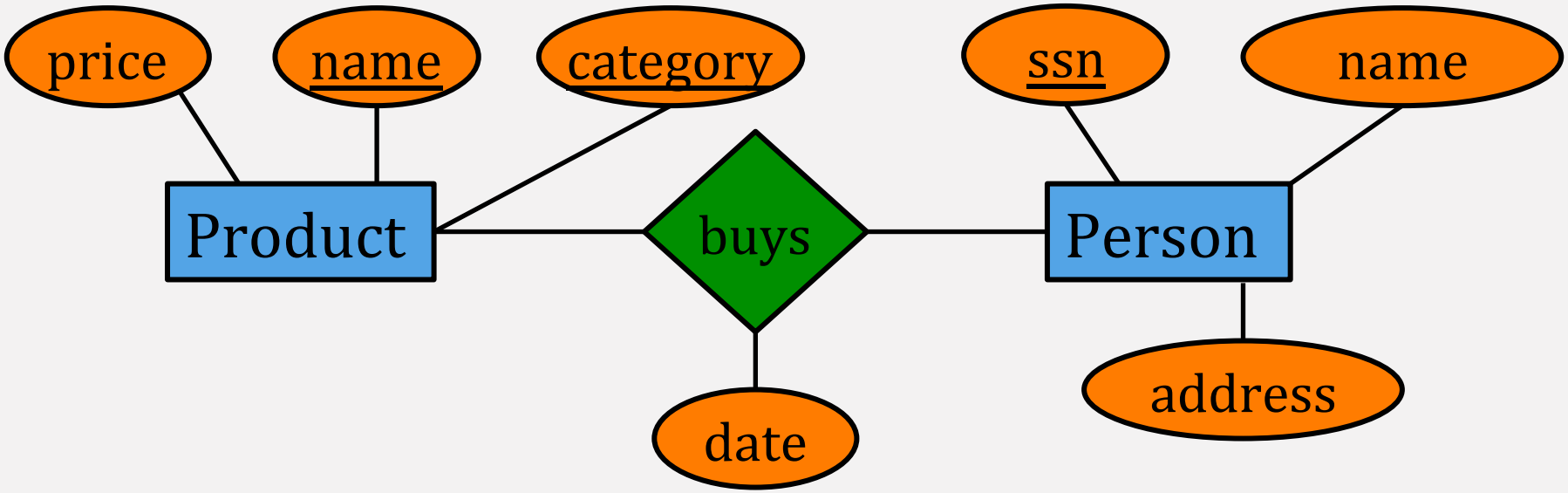
MANY-ONE RELATIONSHIPS



No need for a **Makes** relation; instead modify **Product**:

Product(name, category, price, company_name)
Company(name, stockprice)

MANY-MANY RELATIONS



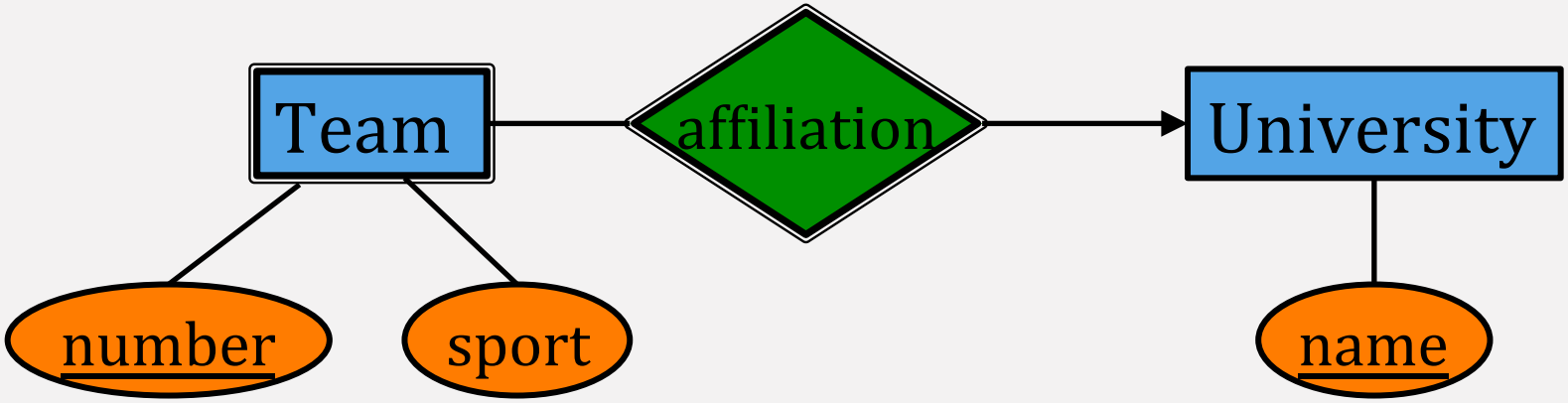
Product(<u>name, category</u>, price, ssn)

What is wrong here?

RELATIONSHIP TO RELATION: SQL

```
CREATE TABLE Buys
   (prodname CHAR(40),
    prodcategory CHAR(20),
    ssn CHAR(11),
    date DATE,
    PRIMARY KEY(prodname,prodcategory,ssn)
    FOREIGN KEY (ssn)
        REFERENCES Person,
    FOREIGN KEY (prodname, prodcategory)
        REFERENCES Product(name, category))
```

WEAK ENTITY SETS



Team(<u>number, affiliated-university</u>, sport)

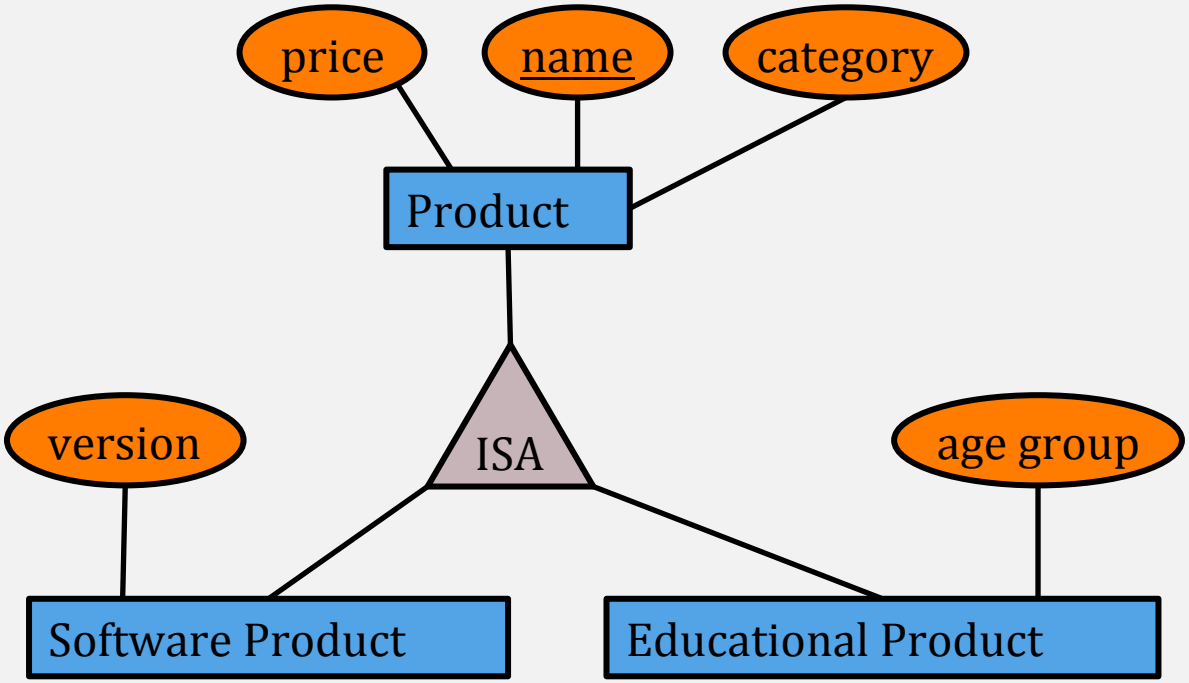
- **Affiliation** does not need a separate relation!
- Attribute '**name**' needed as part of the key

Weak Entity Sets

- The relation for a weak entity set must include:
  - attributes for its complete key (including those in other entity sets)
  - its own, non-key attributes

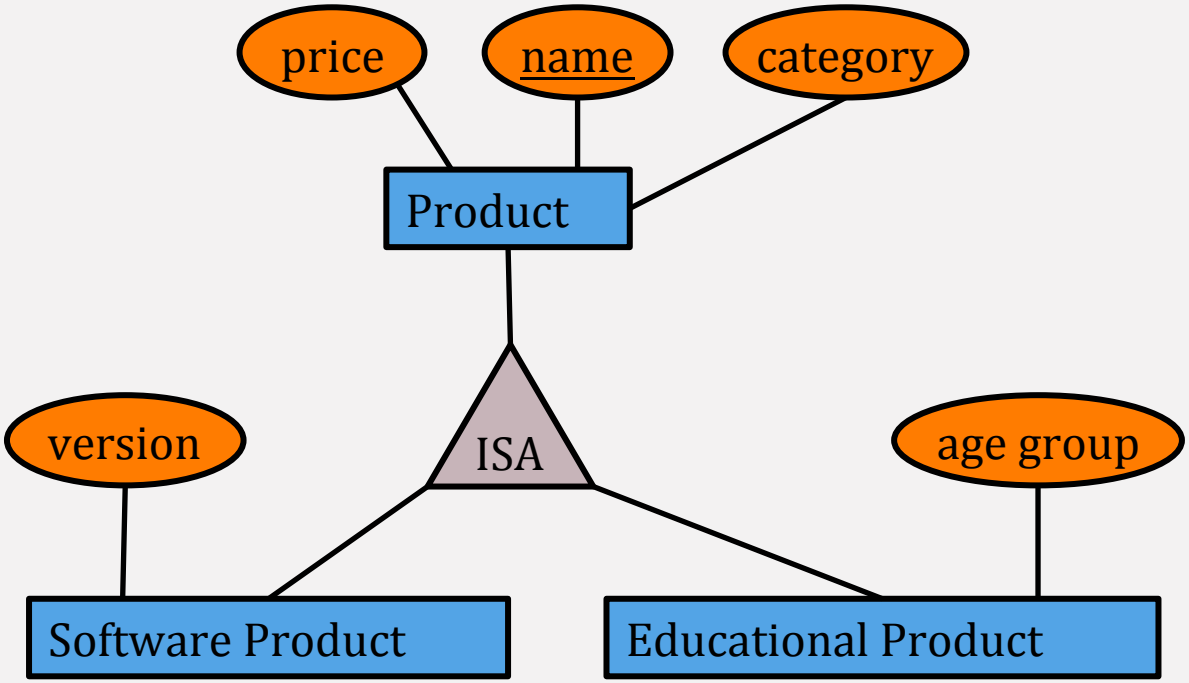
- A supporting (double-diamond) relationship is redundant and produces no relation

SUBCLASSES: OPTION 1



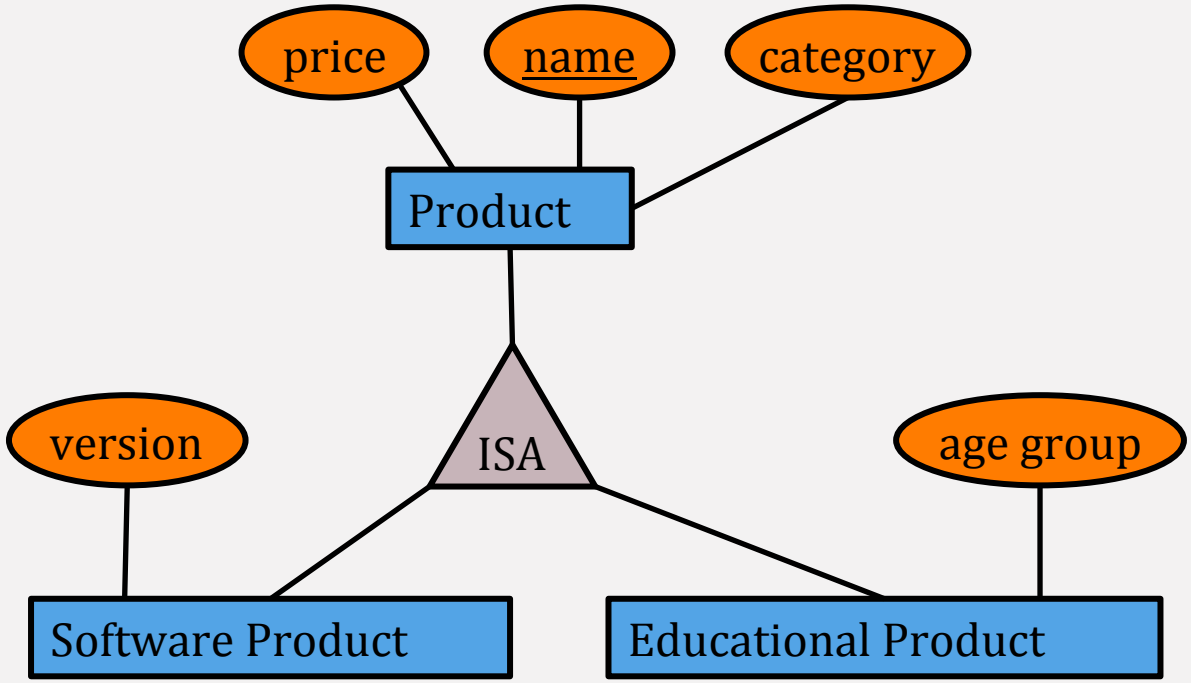
- **Product**(name, category, price)
- **SoftwareProduct**(name, category, price, version)
- **EducationalProduct**(name, category, price, age-group)

SUBCLASSES: OPTION 2



- **Product**(name, category, price)
- **SoftwareProduct**(name, version)
- **EducationalProduct**(name, age-group)

SUBCLASSES: OPTION 3



- **Product**(name, category, price, version, age-group)
- Use NULL to denote that the attribute makes no sense for a specific tuple

SUBCLASSES RECAP

Three approaches:

1. create a relation for each class with all its attributes
2. create one relation for each subclass with only the key attribute(s) and attributes attached to it
3. create one relation; entities have null in attributes that do not belong to them

RECAP

Relational Model

- relation, attributes
- schema vs instance

ER to Relational

- entity set, relationship → relation
- primary keys, foreign keys
- special cases: weak entity sets, subclasses