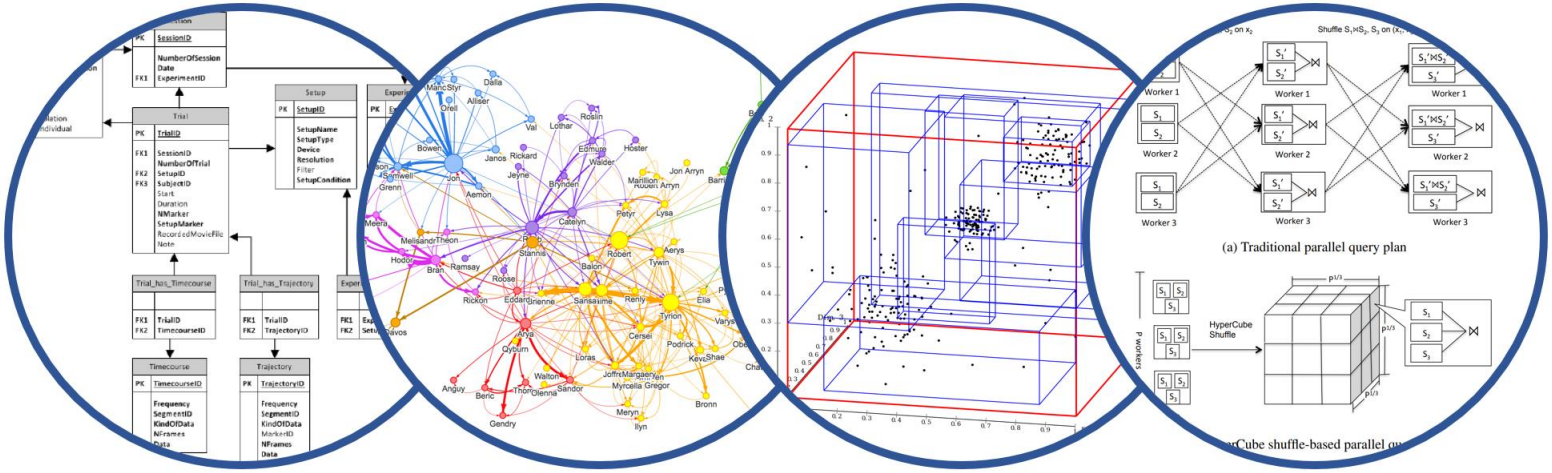


Introduction to Data Management

Design Theory

Paul G. Allen School of Computer Science and Engineering
University of Washington, Seattle

Announcements

- HW3 due on Wednesday
- Midterm on Friday, 10/25 in class
  - Closed books, no cheat sheet (you won’t need it)
  - Some practice midterms on the course website

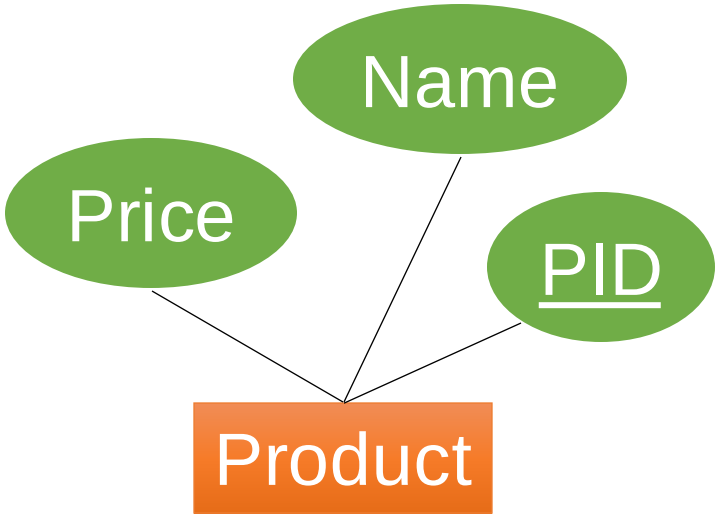
Recap: Entity Sets

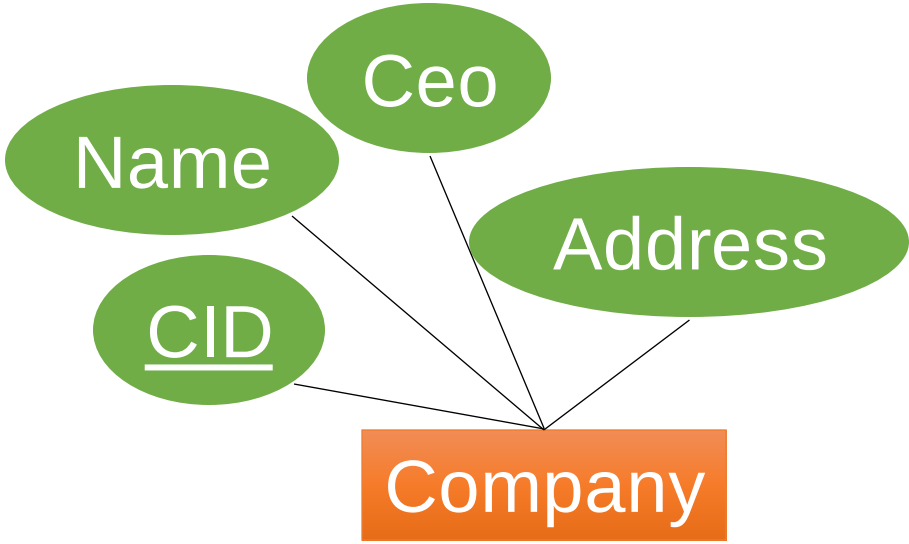
Product

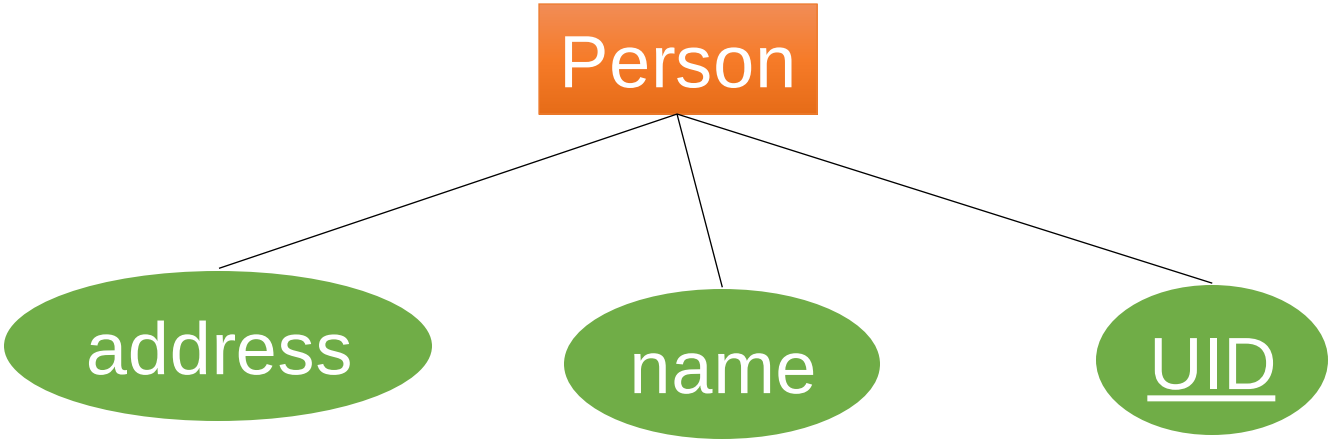
Company

Person

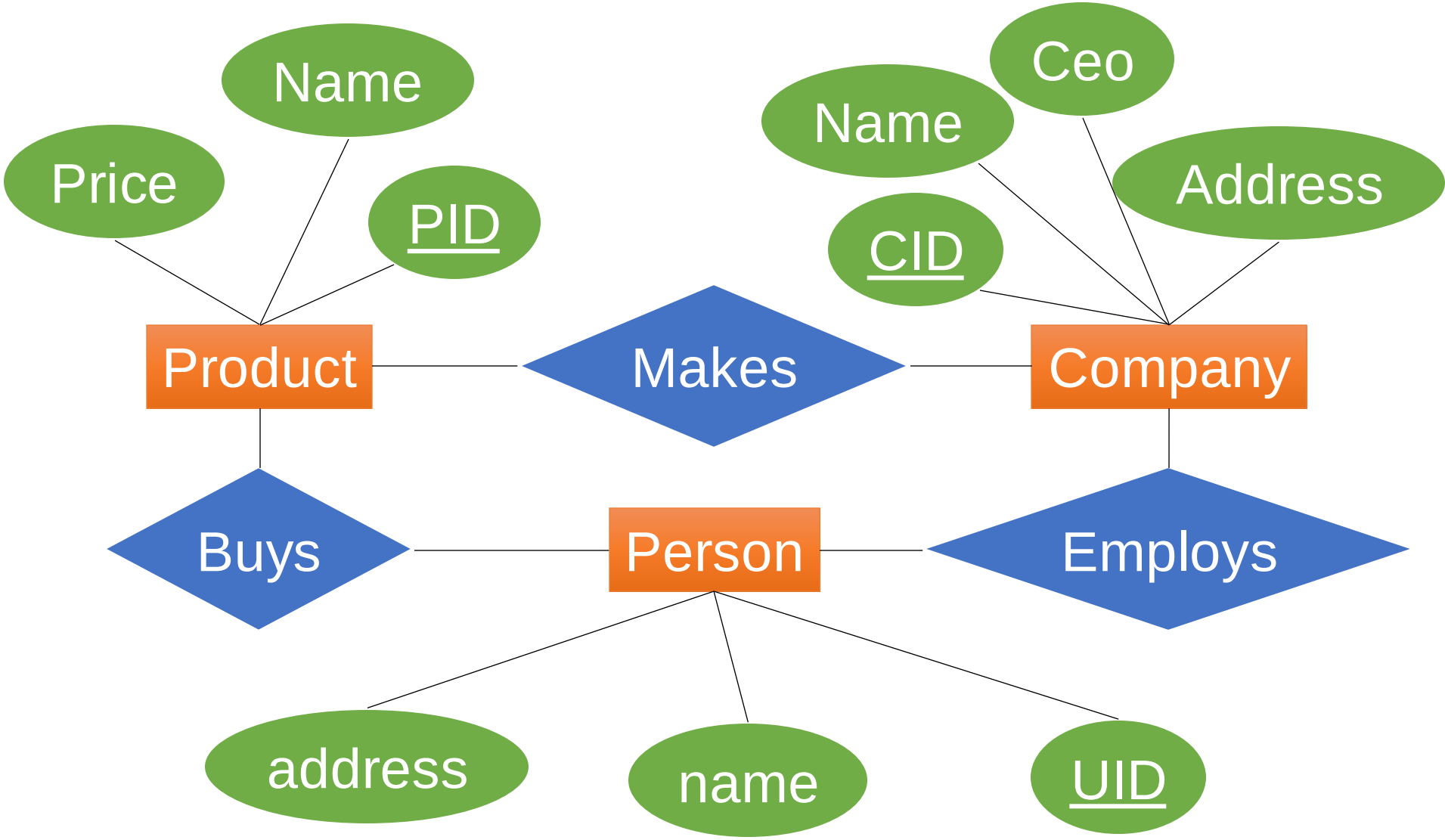
Recap: Attributes



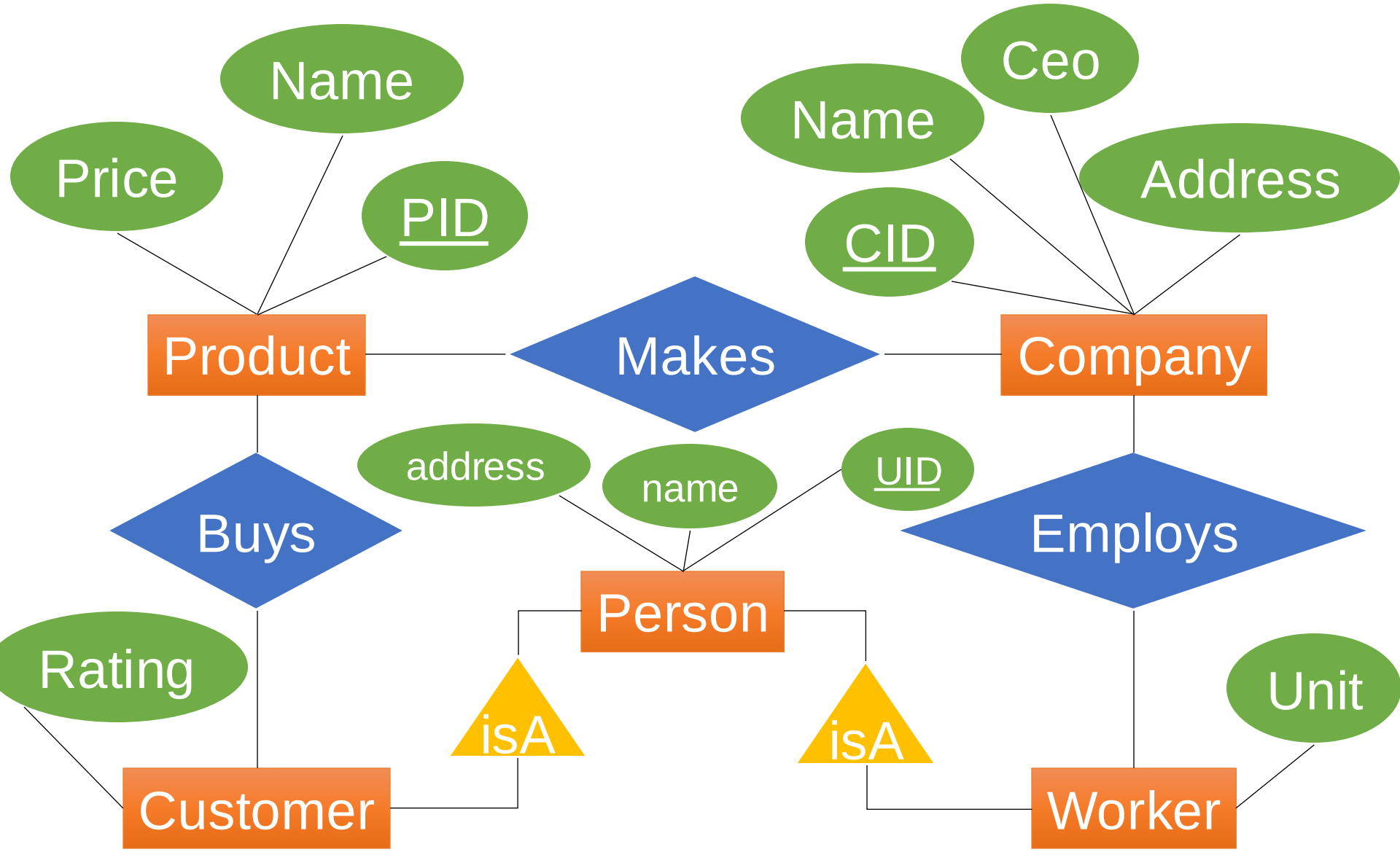




Recap: Relationships



Recap: Inheritance



Agenda for Today

- Details of ER Diagrams
- Convert an ER diagram to SQL

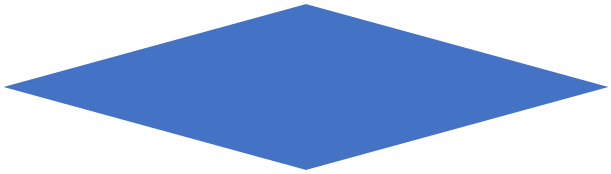
ER Diagrams: Building Blocks

- These are all the components we will learn about

Entity set

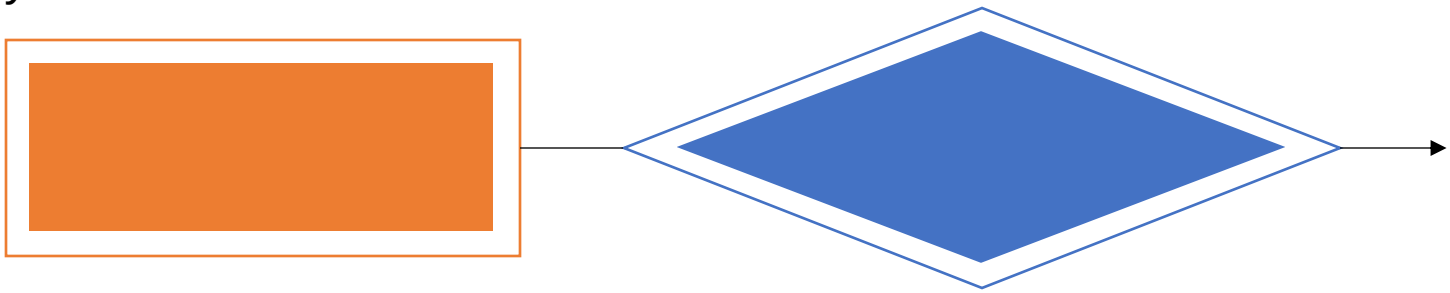
Attribute

Relationship



Subclass

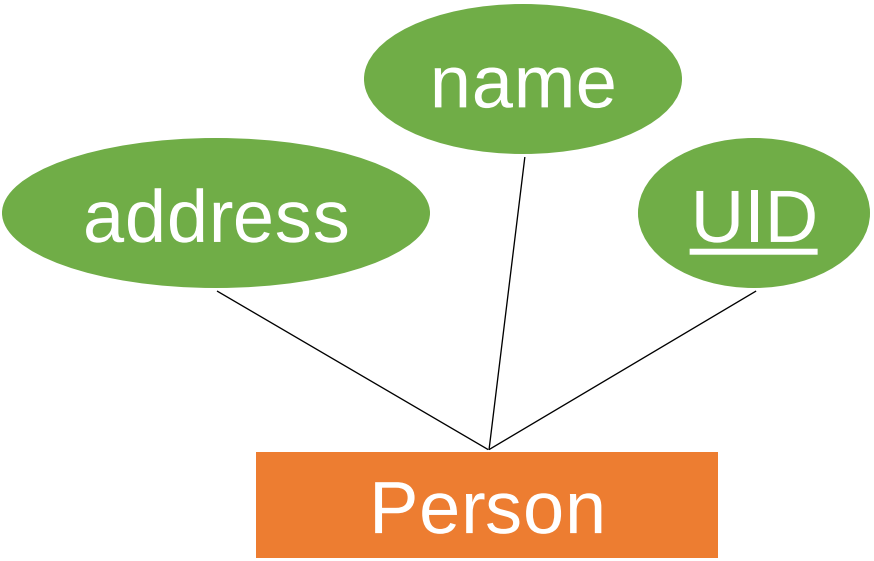
Weak Entity



Entity Sets

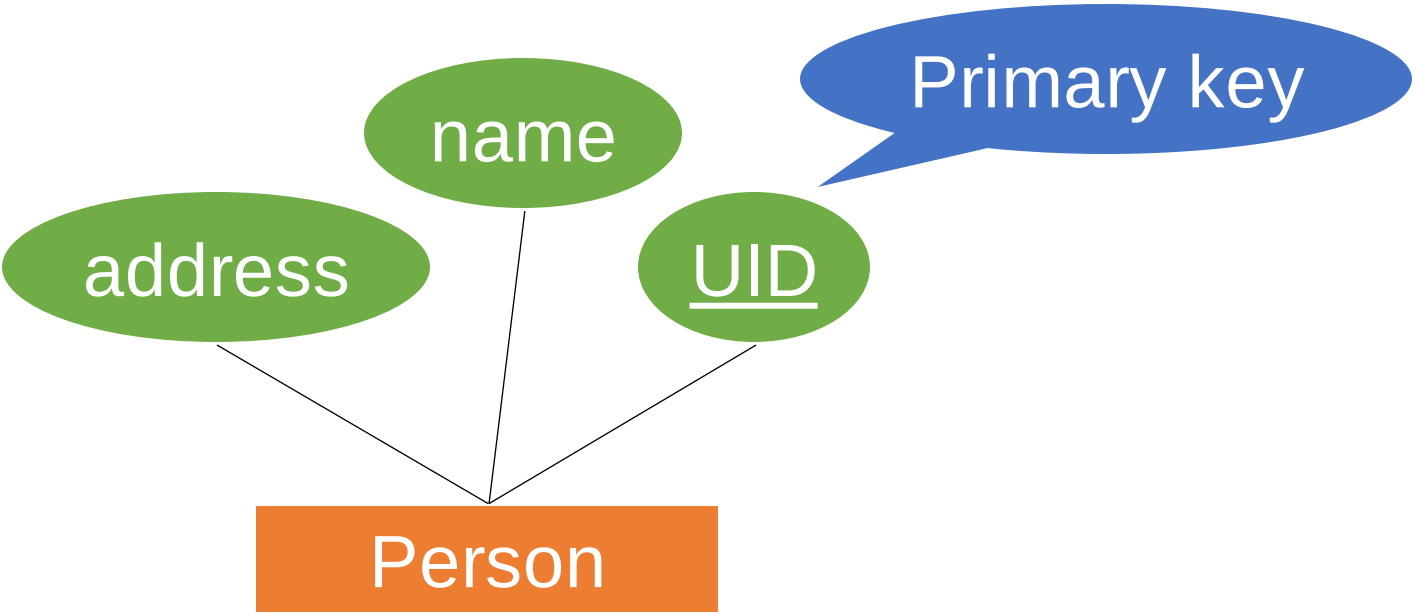
Entity Set

- **Entity set** is the same as a **class**
- An **entity** is the same as an **object**
- An **attribute** is the same as a **field** of a class



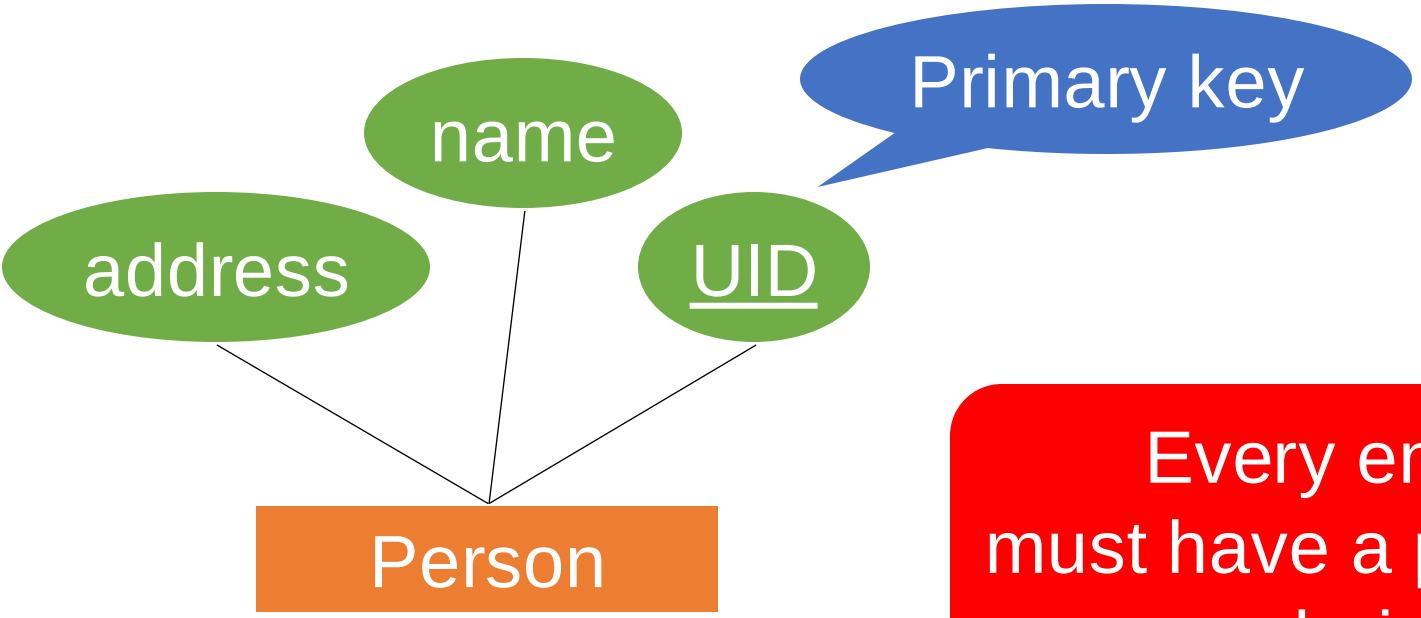
Entity Set

- **Entity set** is the same as a **class**
- An **entity** is the same as an **object**
- An **attribute** is the same as a **field** of a class



Entity Set

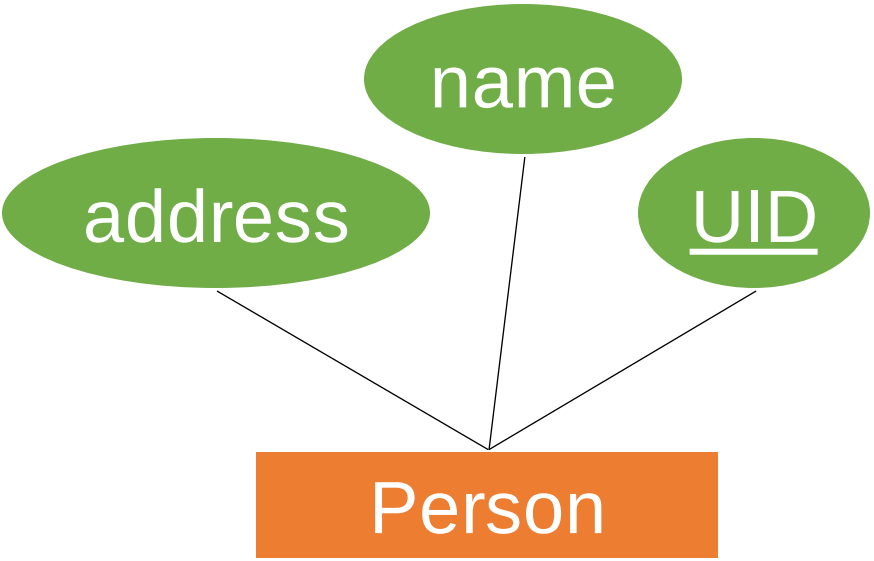
- **Entity set** is the same as a **class**
- An **entity** is the same as an **object**
- An **attribute** is the same as a **field** of a class



Every entity set must have a primary key, or a derived one.

Entity Set to SQL

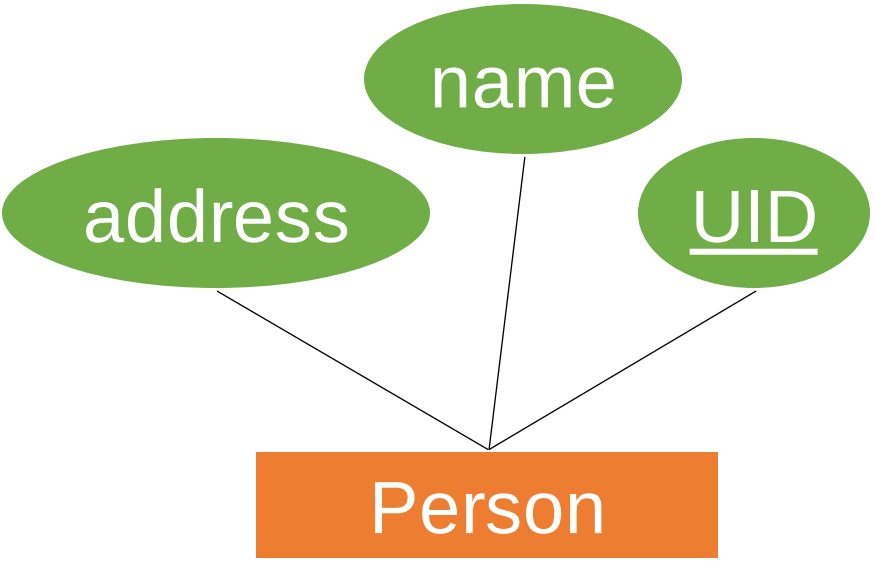
- **Entity set** is the same as a **class**
- An **entity** is the same as an **object**
- An **attribute** is the same as a **field** of a class



How do we represent in SQL?

Entity Set to SQL

- **Entity set** is the same as a **class**
- An **entity** is the same as an **object**
- An **attribute** is the same as a **field** of a class



How do we represent in SQL?

```
CREATE TABLE
  Person (
    UID INT PRIMARY KEY,
    name TEXT,
    address TEXT);
```

Relationships

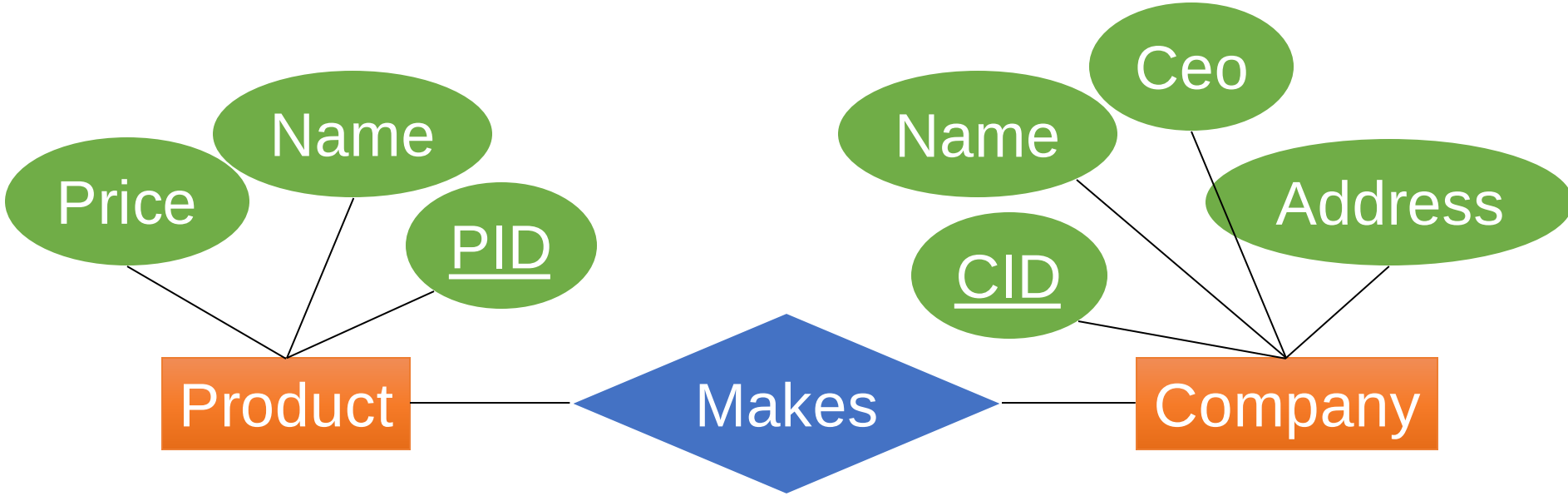
Relationships

- A **relationship** relates entities from two entity sets

A subset of the cross product: $R \subseteq A \times B$

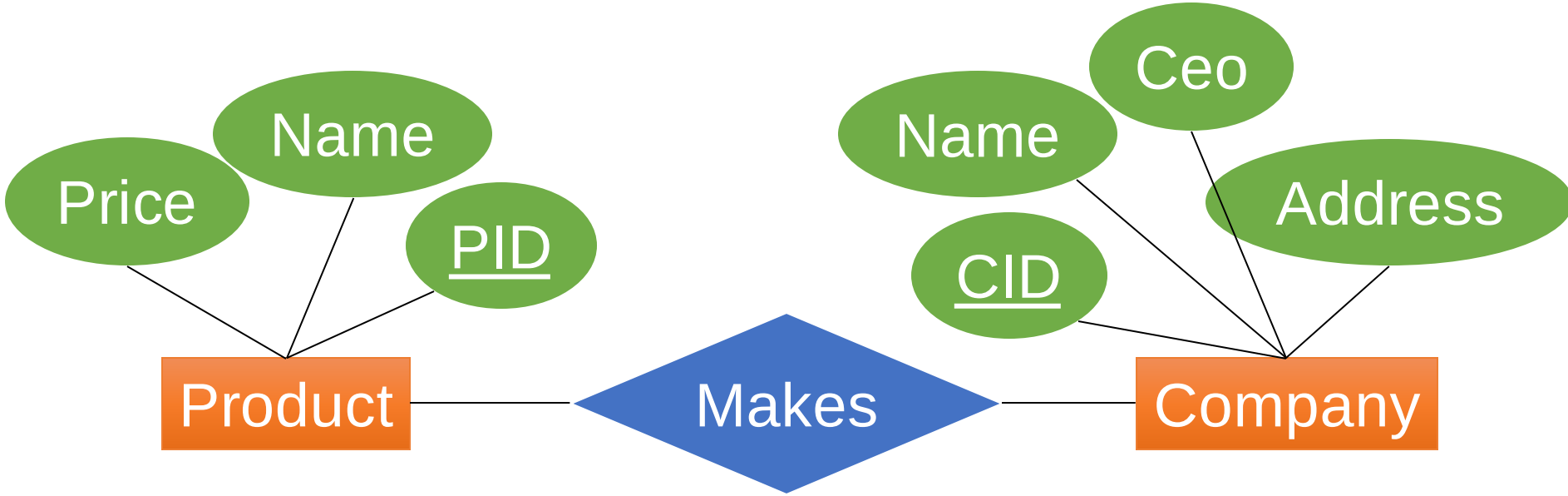
Relationships

- A **relationship** relates entities from two entity sets



Relationships

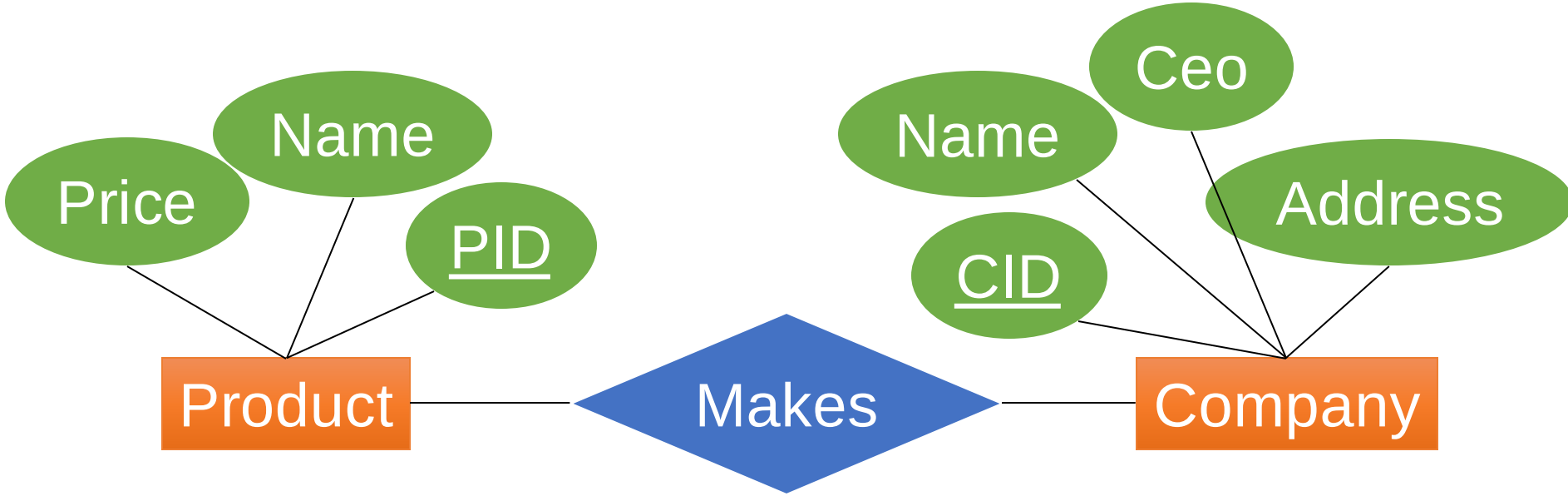
- A **relationship** relates entities from two entity sets



How do
we represent
in SQL?

Relationships

- A **relationship** relates entities from two entity sets

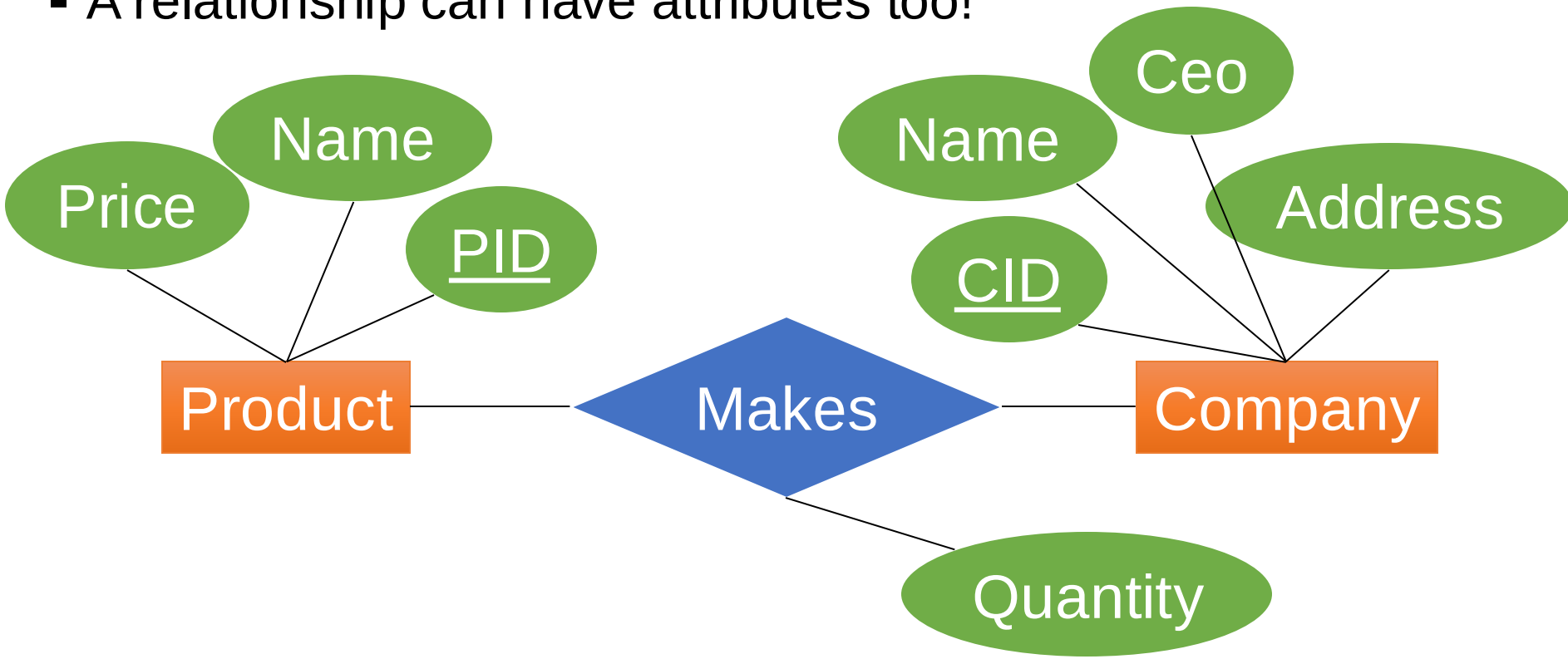


How do we represent in SQL?

```
CREATE TABLE
  Makes (
    PID INT References Product,
    CID INT References Company);
```

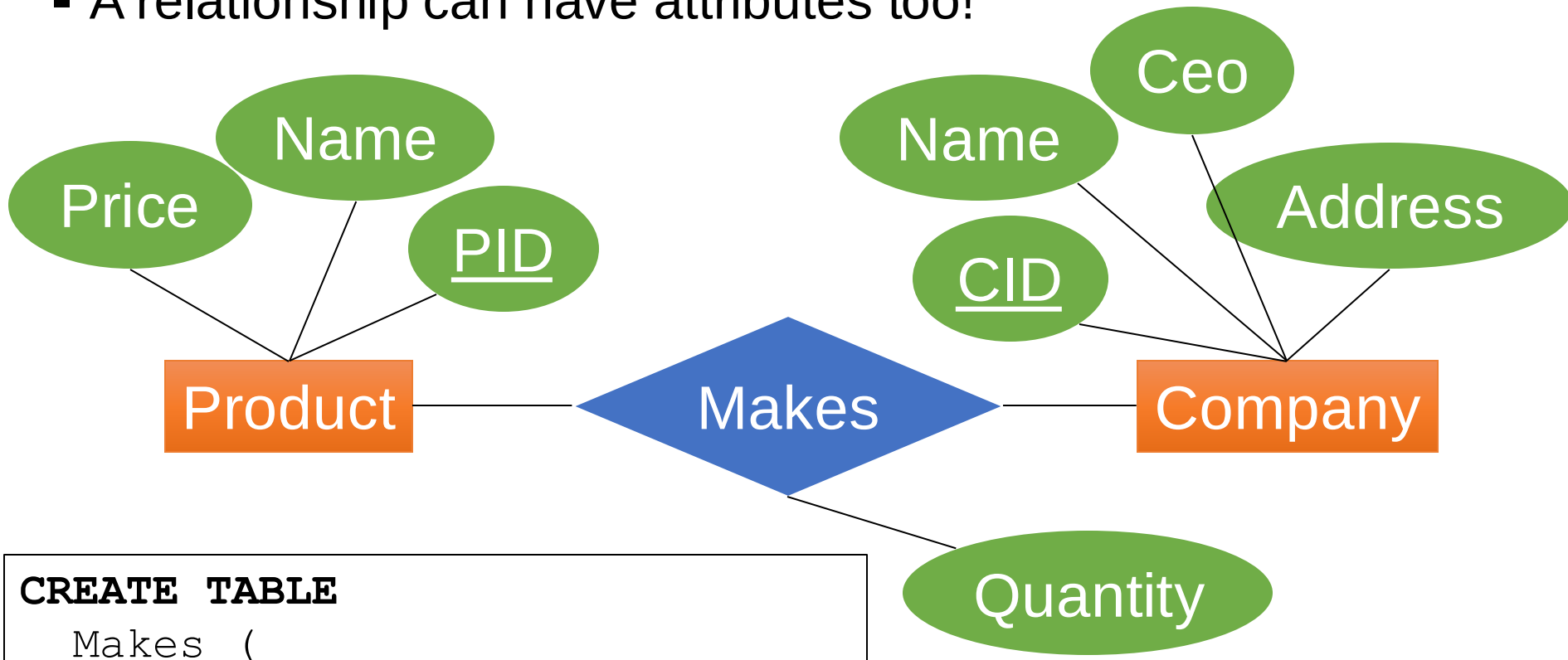
Relationships

- A **relationship** relates entities from two entity sets
- A relationship can have attributes too!



Relationships

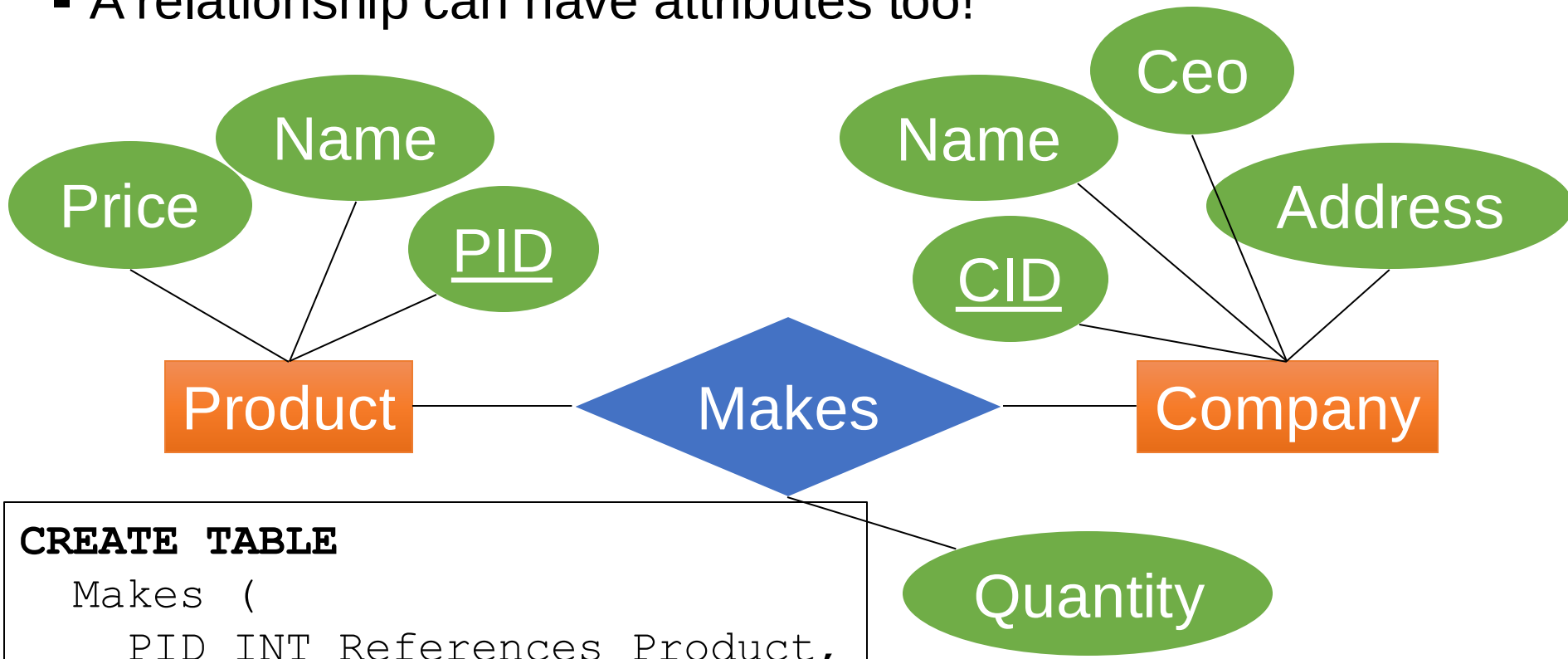
- A **relationship** relates entities from two entity sets
- A relationship can have attributes too!




```
CREATE TABLE
  Makes (
    PID INT References Product,
    CID INT References Company,
    Quantity Int);
```

Relationships

- A **relationship** relates entities from two entity sets
- A relationship can have attributes too!




```
CREATE TABLE
  Makes (
    PID INT References Product,
    CID INT References Company,
    Quantity Int,
    Primary Key (UID, CID));
```

Key in a relationships consists of entities only

Relationship Multiplicity

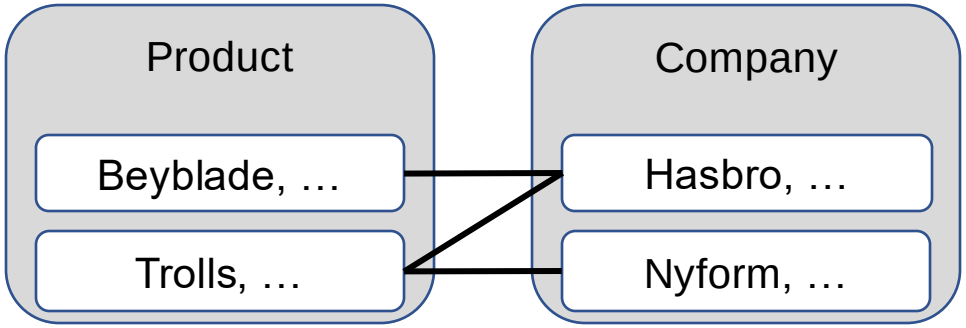
- One-to-one 
- Many-to-one 
- Many-to-many 

Relationship Multiplicity

- One-to-one
- Many-to-one
- **Many-to-many**

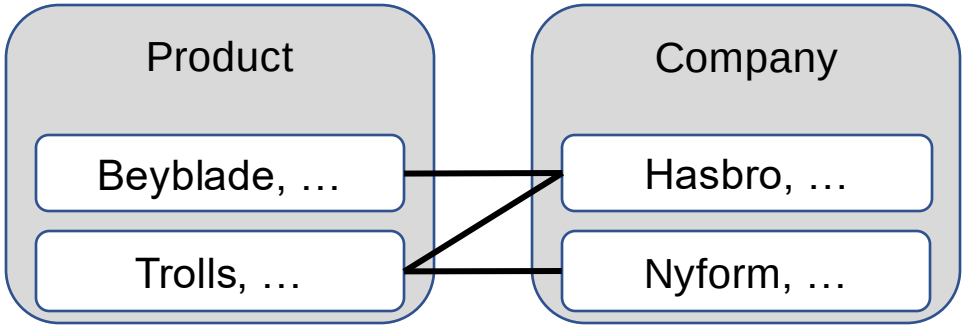
Relationship Multiplicity

- One-to-one
- Many-to-one
- **Many-to-many**



Relationship Multiplicity

- One-to-one
- Many-to-one
- **Many-to-many**



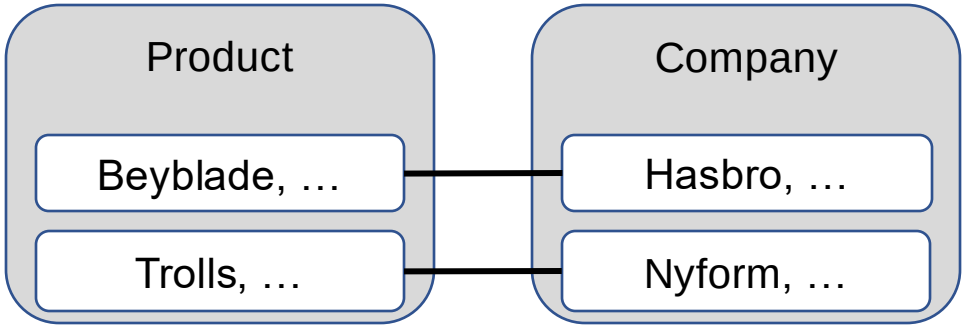

```
CREATE TABLE Product (
  PID int PRIMARY KEY, ...);
CREATE TABLE Company (
  CID int PRIMARY KEY, ...);
CREATE TABLE Makes (
  PID int REFERENCES Product,
  CID int REFERENCES Company);
```

Relationship Multiplicity

- **One-to-one**
- Many-to-one
- Many-to-many

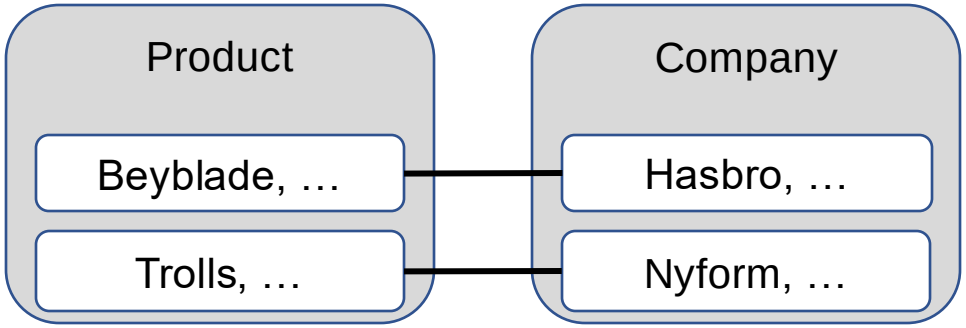
Relationship Multiplicity

- **One-to-one**
- Many-to-one
- Many-to-many



Relationship Multiplicity

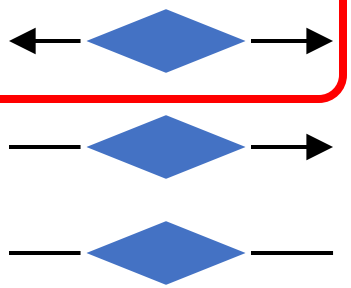
- **One-to-one**
- Many-to-one
- Many-to-many



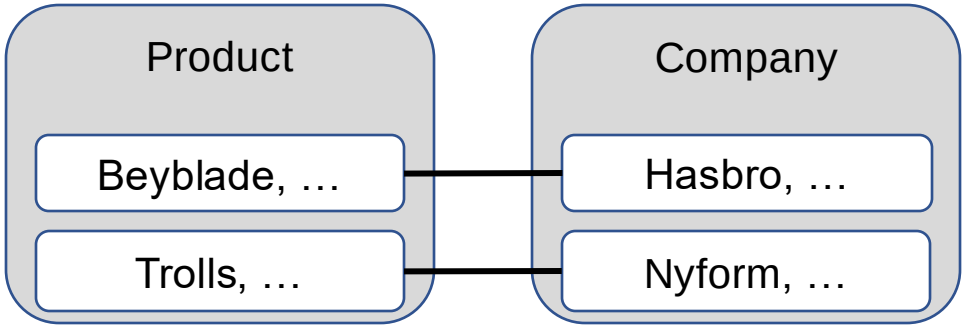

```
CREATE TABLE Product (
  PID int PRIMARY KEY, ...);
CREATE TABLE Company (
  CID int PRIMARY KEY, ...);
CREATE TABLE Makes (
  PID int UNIQUE
          REFERENCES Product,
  CID int UNIQUE
          REFERENCES Company);
```

Relationship Multiplicity

- **One-to-one**
- Many-to-one
- Many-to-many



We will revise this shortly



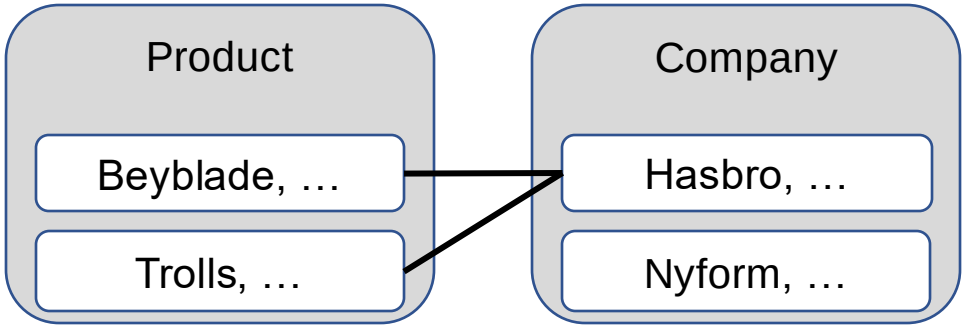

```
CREATE TABLE Product (
  PID int PRIMARY KEY, ...);
CREATE TABLE Company (
  CID int PRIMARY KEY, ...);
CREATE TABLE Makes (
  PID int UNIQUE
          REFERENCES Product,
  CID int UNIQUE
          REFERENCES Company);
```

Relationship Multiplicity

- One-to-one
- **Many-to-one**
- Many-to-many

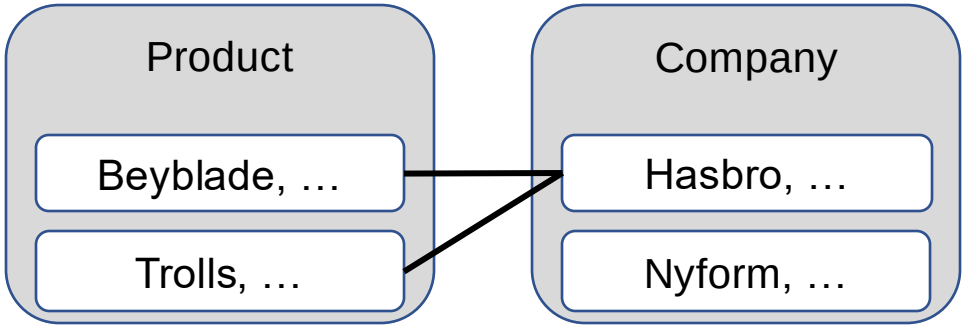
Relationship Multiplicity

- One-to-one
- **Many-to-one**
- Many-to-many



Relationship Multiplicity

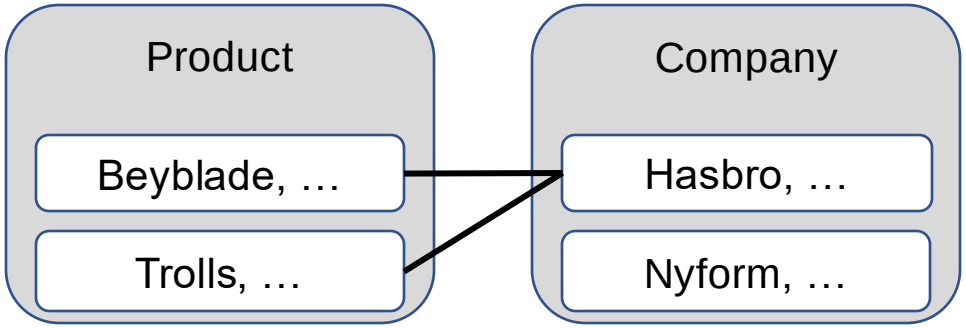
- One-to-one
- **Many-to-one**
- Many-to-many




```
CREATE TABLE Product (
  PID int PRIMARY KEY, ...);
CREATE TABLE Company (
  CID int PRIMARY KEY, ...);
CREATE TABLE Makes (
  PID int UNIQUE
          REFERENCES Product,
  CID int REFERENCES Company);
```

Relationship Multiplicity

- One-to-one
- **Many-to-one**
- Many-to-many

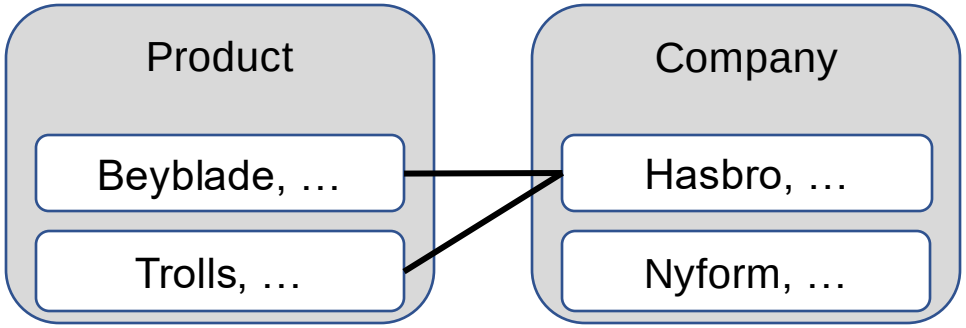



```
CREATE TABLE Product (
  PID int PRIMARY KEY, ...);
CREATE TABLE Company (
  CID int PRIMARY KEY, ...);
CREATE TABLE Makes (
  PID int PRIMARY KEY
          REFERENCES Product,
  CID int REFERENCES Company);
```

Better

Relationship Multiplicity

- One-to-one
- **Many-to-one**
- Many-to-many

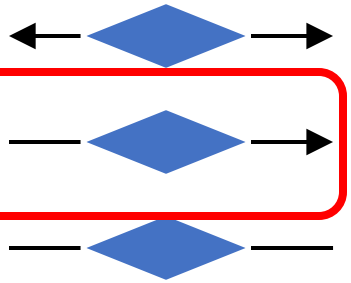


Do we need the Makes table?

```
CREATE TABLE Product (
  PID int PRIMARY KEY, ...);
CREATE TABLE Company (
  CID int PRIMARY KEY, ...);
CREATE TABLE Makes (
  PID int PRIMARY KEY
          REFERENCES Product,
  CID int REFERENCES Company);
```

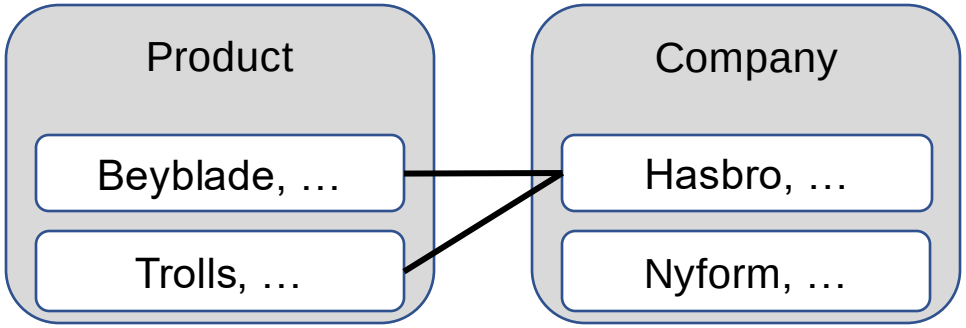
Relationship Multiplicity

- One-to-one
- **Many-to-one**
- Many-to-many



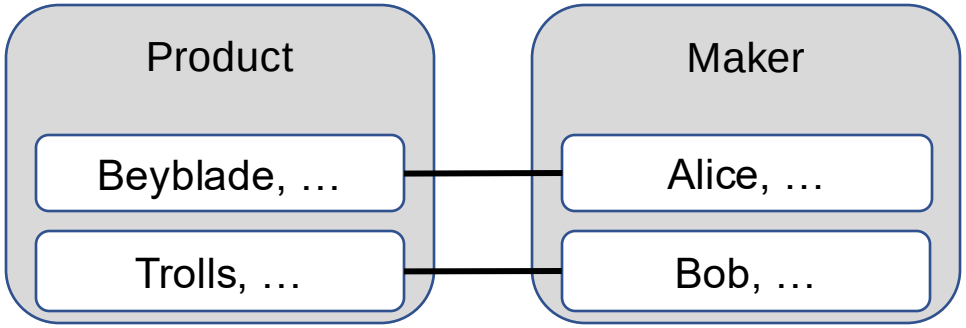
We don't need separate table!

```
CREATE TABLE Product (
  PID int PRIMARY KEY,
  CID int REFERENCES Company,
  ...);
CREATE TABLE Company (
  CID int PRIMARY KEY, ...);
```



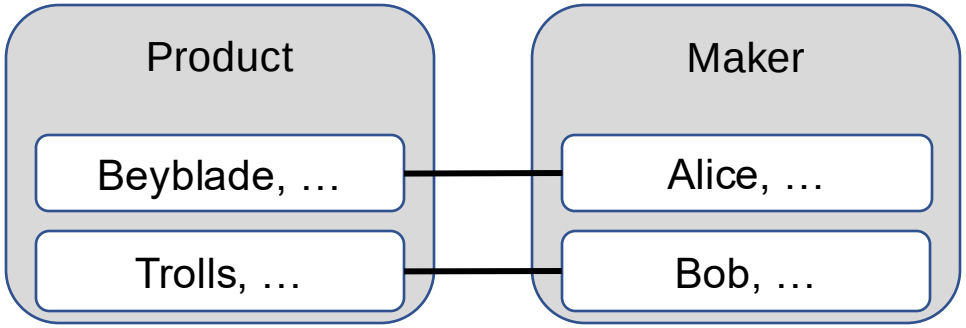
Relationship Multiplicity

- **One-to-one**
- Many-to-one
- Many-to-many



Relationship Multiplicity

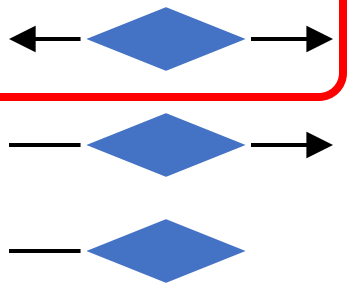
- **One-to-one**
- Many-to-one
- Many-to-many



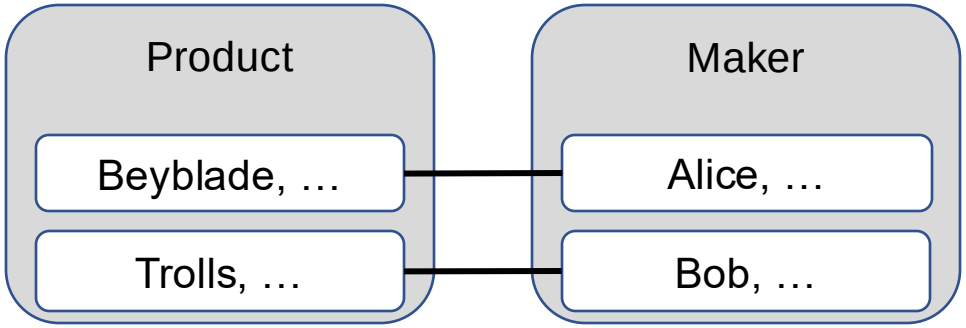

```
CREATE TABLE Product (
  PID int PRIMARY KEY,
  MID Int Reference Maker,
  ...);
CREATE TABLE Maker (
  MID int PRIMARY KEY,
  PID int Reference Product,
  ...);
```

Relationship Multiplicity

- **One-to-one**
- Many-to-one
- Many-to-many



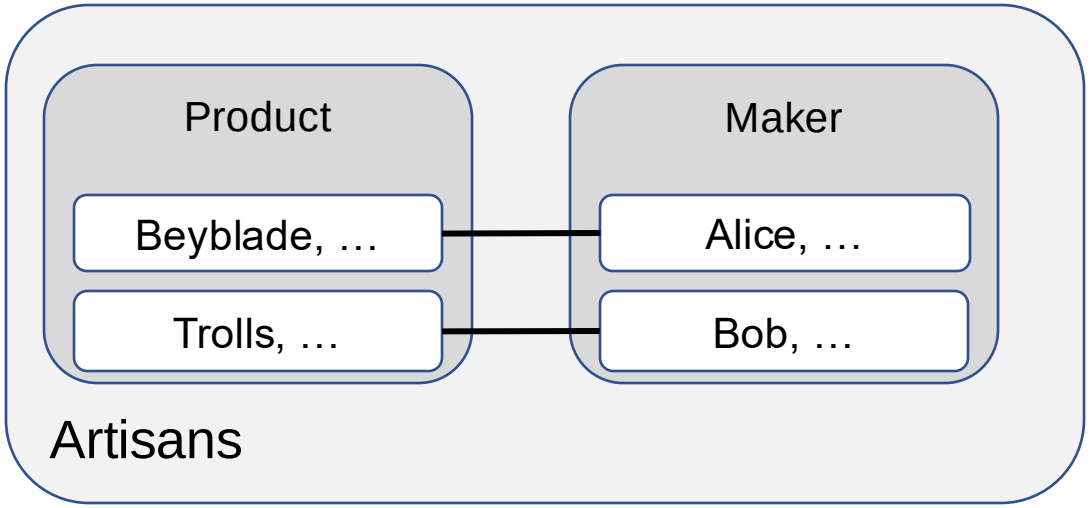
This is one option…




```
CREATE TABLE Product (
  PID int PRIMARY KEY,
  MID Int Reference Maker,
  ...);
CREATE TABLE Maker (
  MID int PRIMARY KEY,
  PID int Reference Product,
  ...);
```

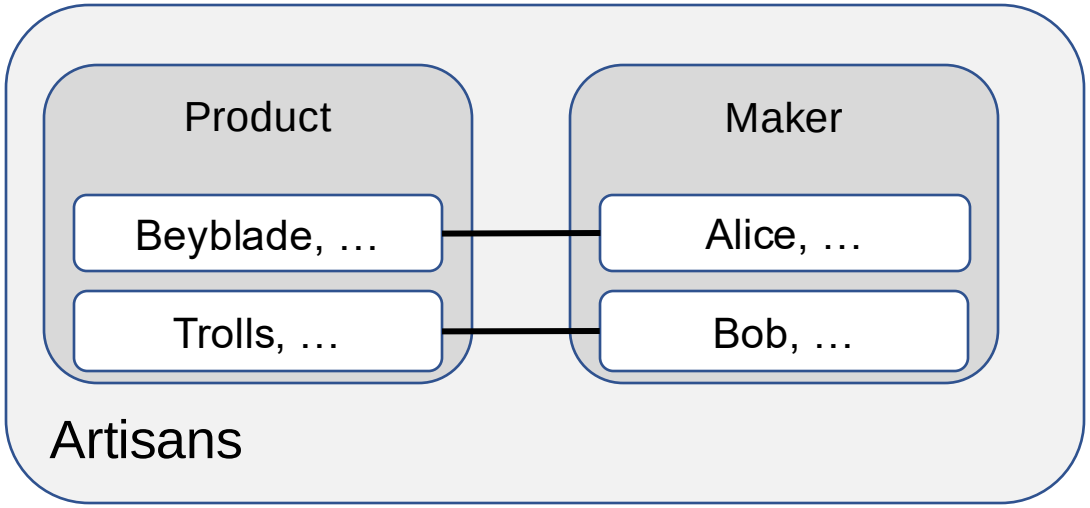
Relationship Multiplicity

- **One-to-one**
- Many-to-one
- Many-to-many



Relationship Multiplicity

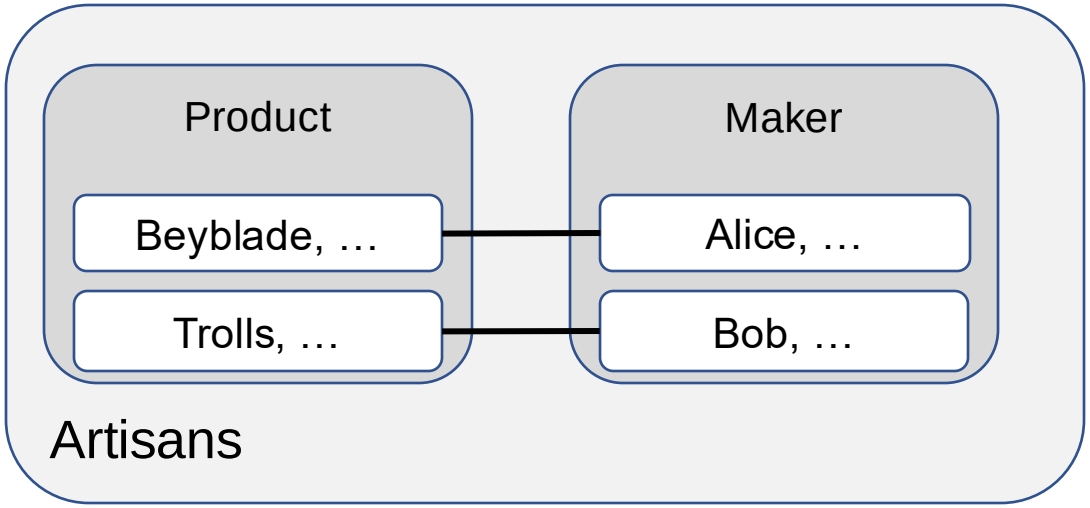
- **One-to-one**
- Many-to-one
- Many-to-many




```
CREATE TABLE Artisans (
  AID int PRIMARY KEY,
  ProductName text,
  MakerName text,
  ...);
```

Relationship Multiplicity

- **One-to-one**
- Many-to-one
- Many-to-many

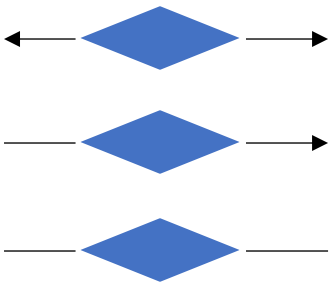



```
CREATE TABLE Artisans (
  AID int PRIMARY KEY,
  ProductName text,
  MakerName text,
  ...);
```

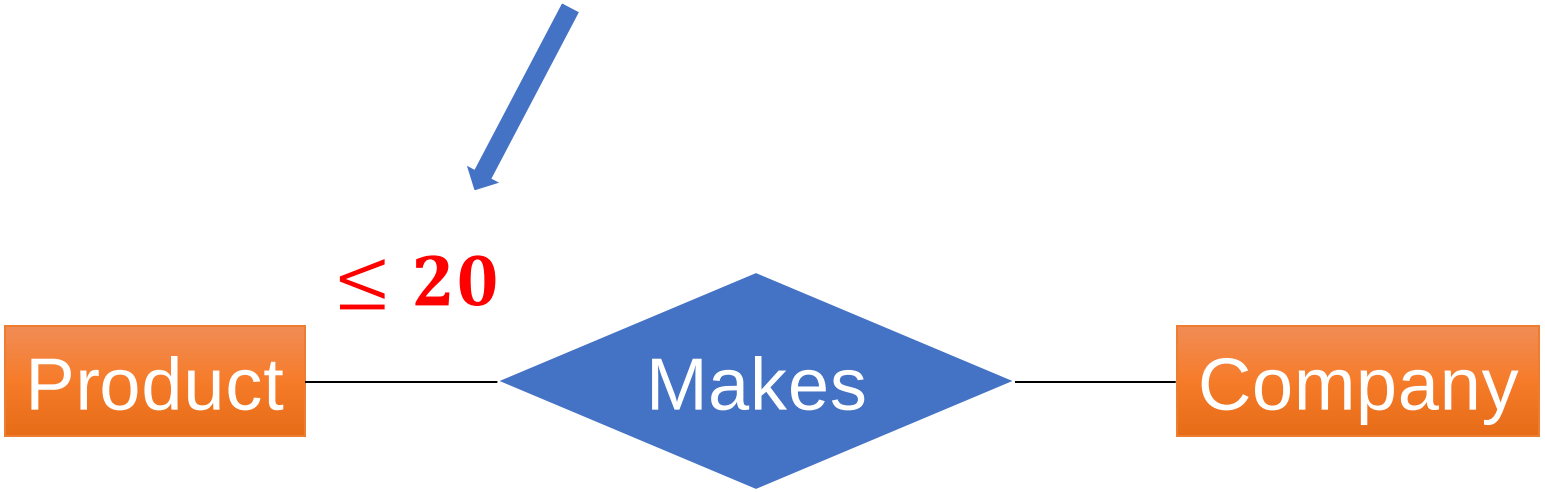
…and this is the second option

Multiplicity Constraints

- One-to-one
- Many-to-one
- Many-to-many



- Each company manufactures at most 20 products
- OK in ER, but most SQL systems don't support

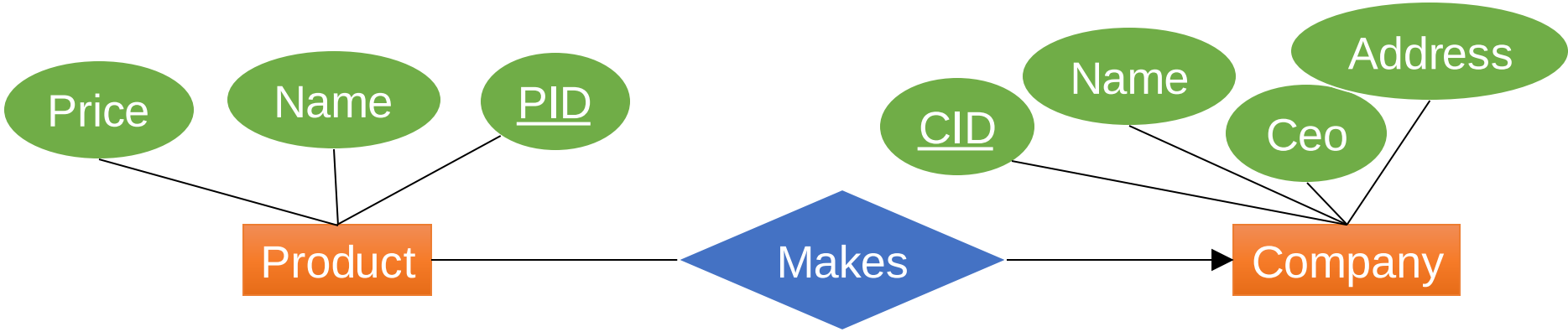


Referential Integrity Constraints

(a complicated name for something very simple)

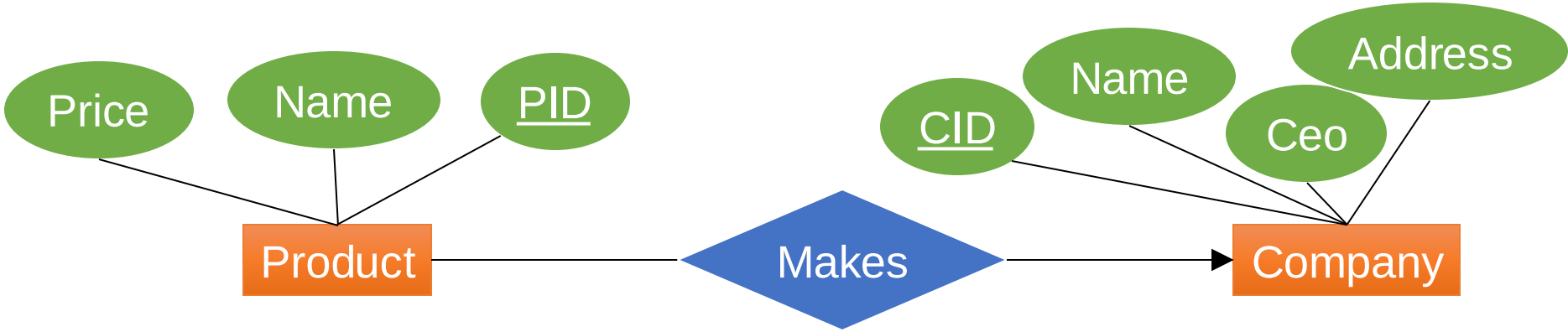
Referential Integrity Constraint

- Regular arrow: at most one
- Rounded arrow: exactly one



Referential Integrity Constraint

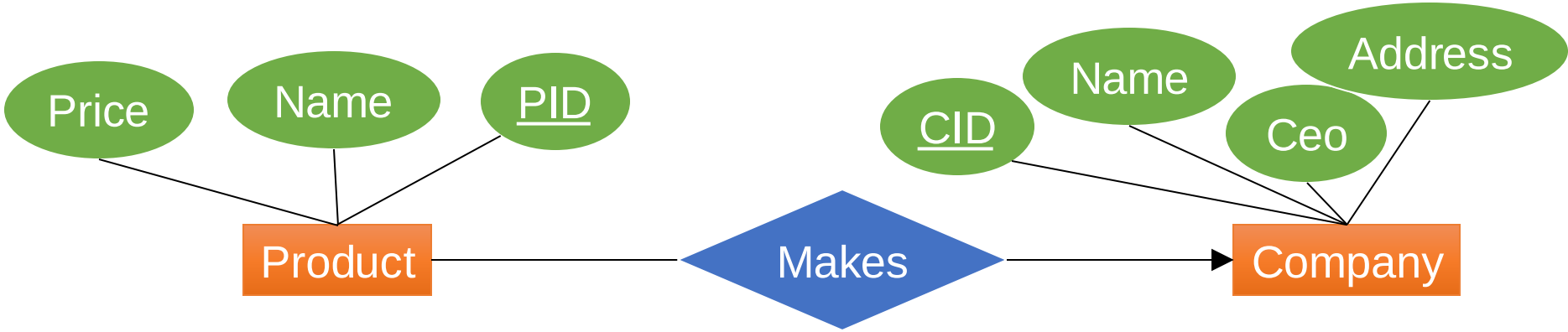
- **Regular arrow**: at most one
- Rounded arrow: exactly one



Referential Integrity Constraint

- **Regular arrow**: at most one
- Rounded arrow: exactly one

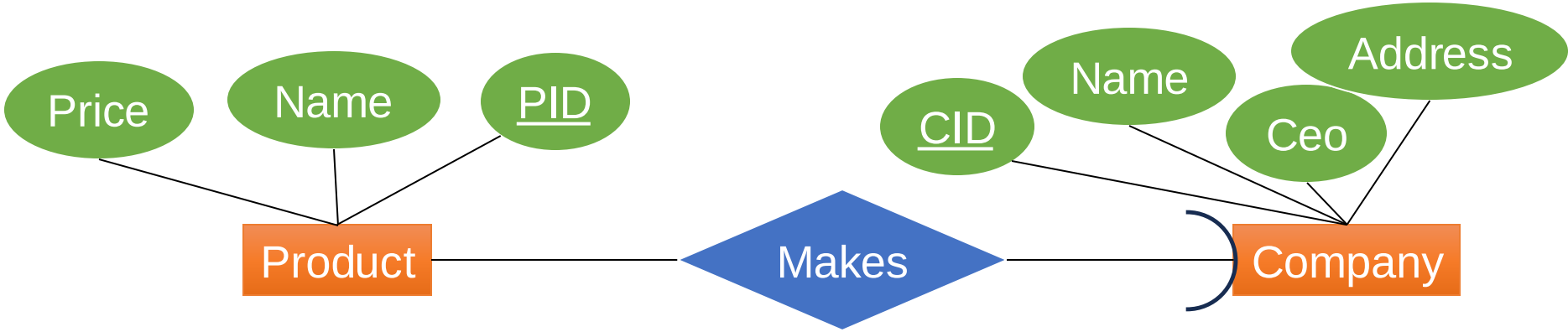
```
CREATE TABLE Product (
  PID int PRIMARY KEY,
  CID int REFERENCES Company,
  ...);
CREATE TABLE Company (
  CID int PRIMARY KEY, ...);
```



Referential Integrity Constraint

- Regular arrow: at most one
- **Rounded arrow**: exactly one

```
CREATE TABLE Product (
  PID int PRIMARY KEY,
  CID int REFERENCES Company
      NOT NULL,
  ...);
CREATE TABLE Company (
  CID int PRIMARY KEY, ...);
```

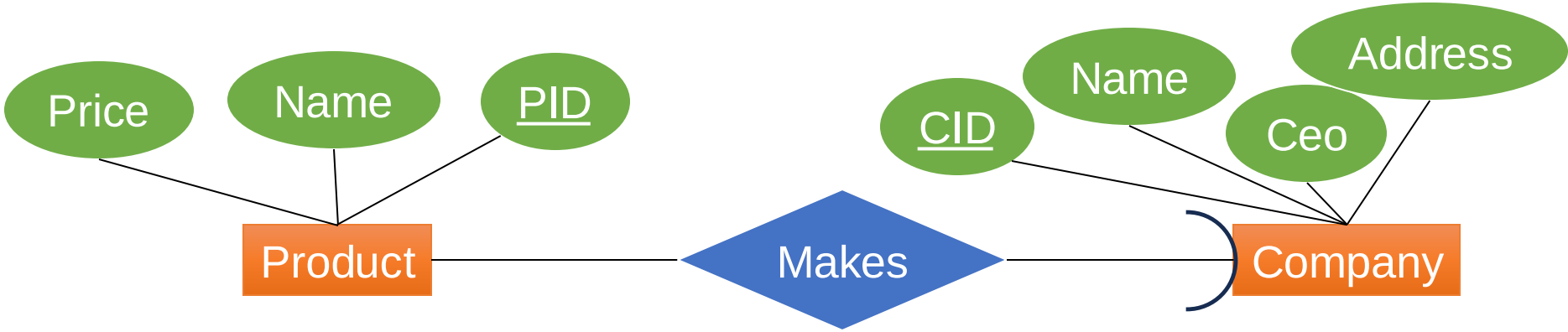


Referential Integrity Constraint

- Regular arrow: at most one
- **Rounded arrow**: exactly one

This is called a
"referential integrity constraint"

```
CREATE TABLE Product (
  PID int PRIMARY KEY,
  CID int REFERENCES Company
      NOT NULL,
  ...);
CREATE TABLE Company (
  CID int PRIMARY KEY, ...);
```



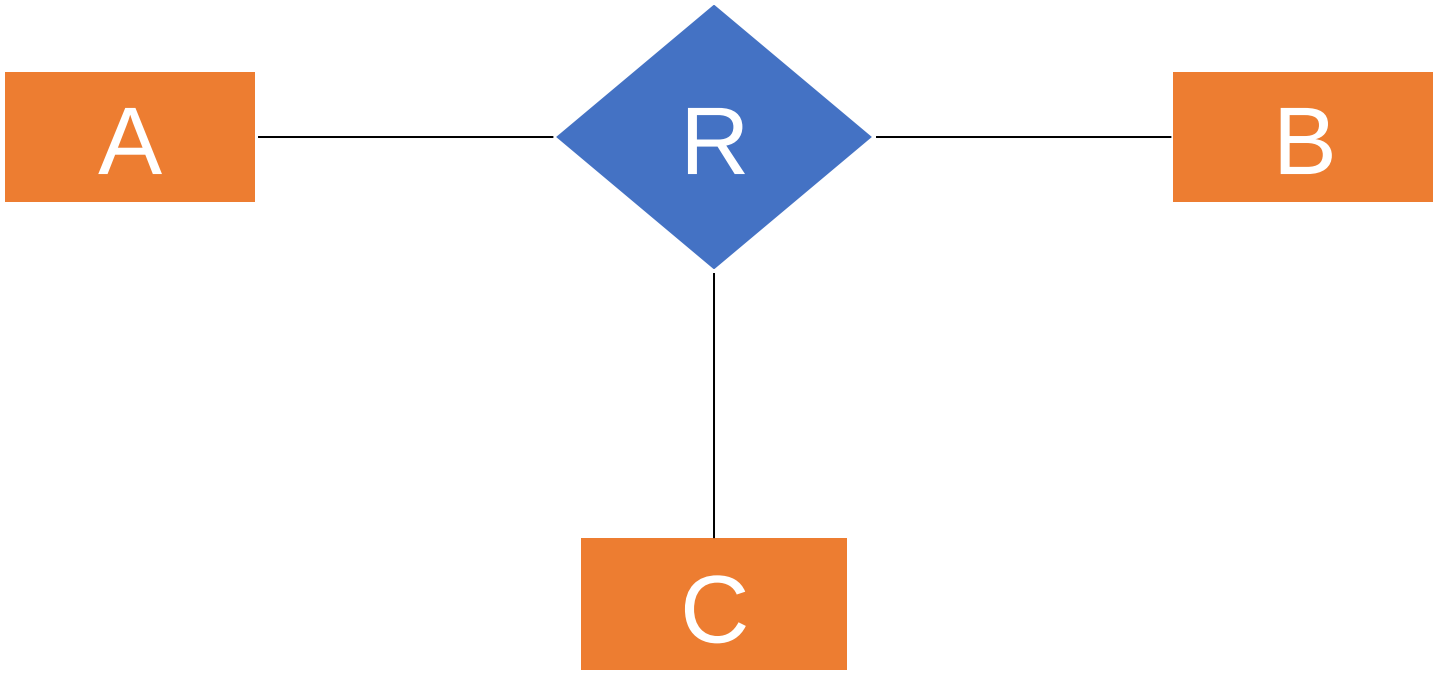
Multi-way Relationships

Multi-Way Relationships

- So far we saw **binary relationships**: they connect two entity sets

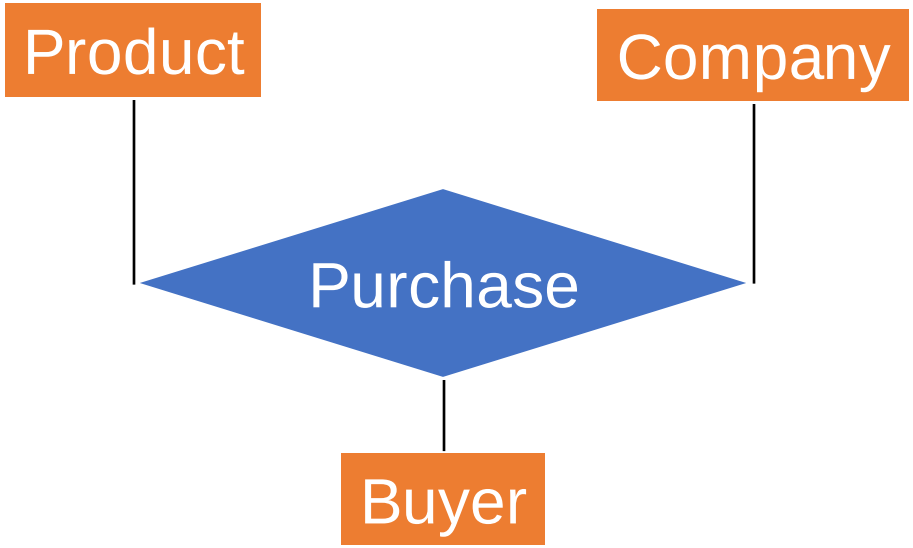
- Also possible: **multi-way relationships**: they connect three or more entity sets

Multi-Way Relationships

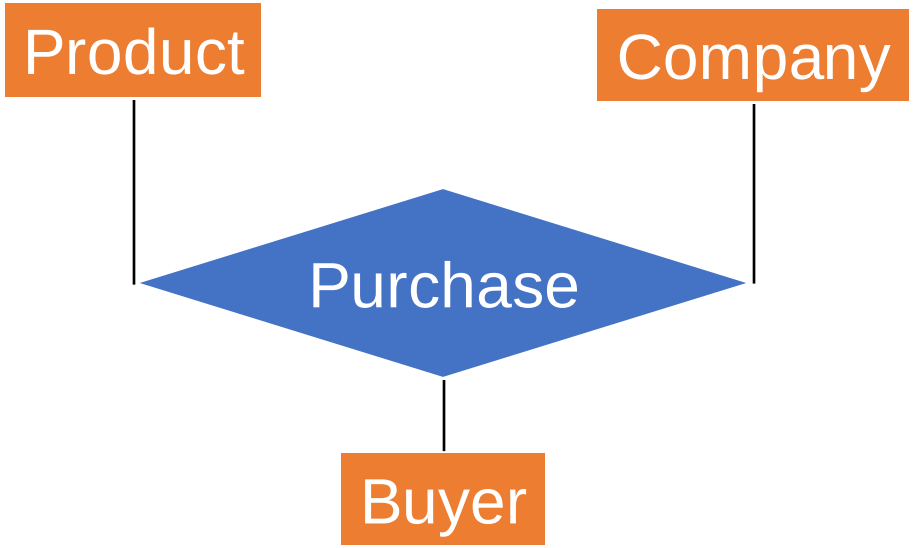


R is a subset of the cross product: $R \subseteq A \times B \times C$

Multi-Way Relationships

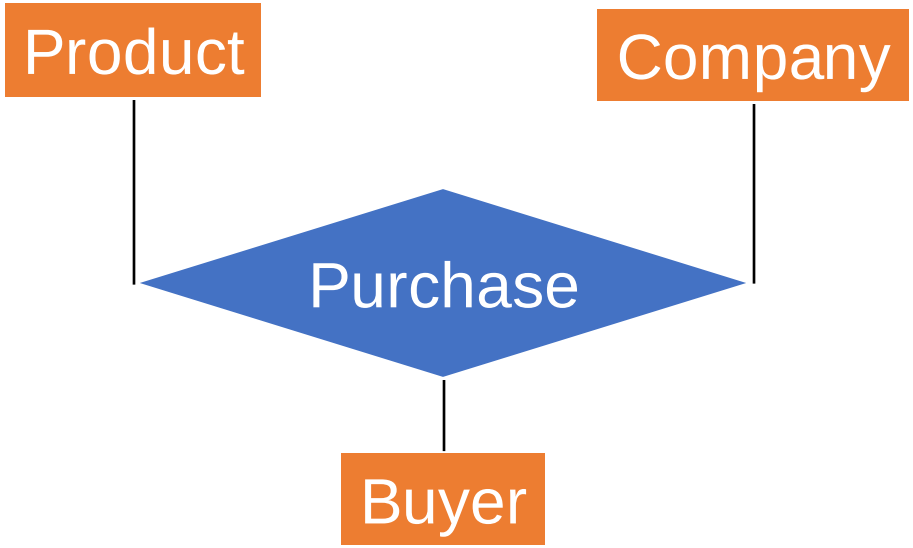


Multi-Way Relationships



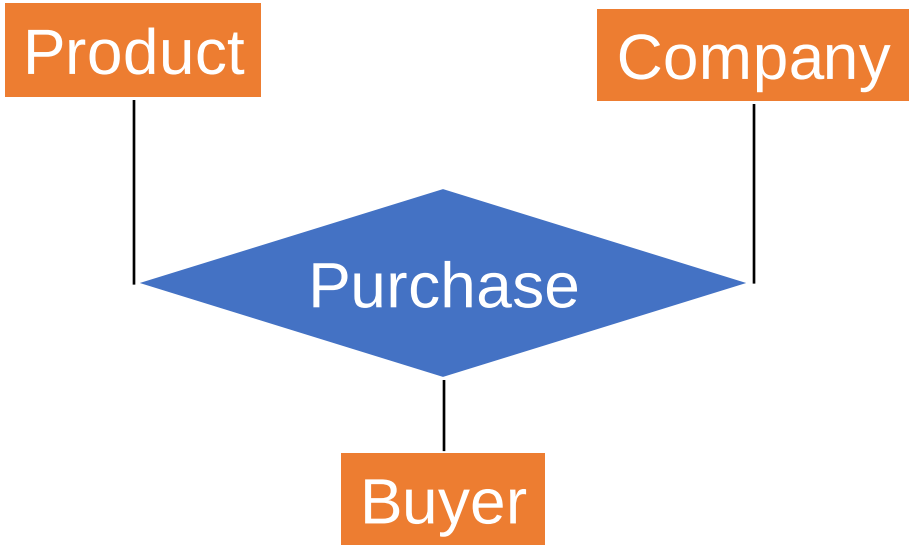

```
CREATE TABLE Product (
    PID INT PRIMARY KEY,...);
CREATE TABLE Company (
    CID INT PRIMARY KEY,...);
CREATE TABLE Buyer (
    BID INT PRIMARY KEY,...);
```

Multi-Way Relationships




```
CREATE TABLE Product (
    PID INT PRIMARY KEY,...);
CREATE TABLE Company (
    CID INT PRIMARY KEY,...);
CREATE TABLE Buyer (
    BID INT PRIMARY KEY,...);

CREATE TABLE Purchase (
    PID INT REFERENCES Product,
    CID INT REFERENCES Company,
    BID INT REFERENCES Buyer,
    ...);
```

Multi-Way Relationships

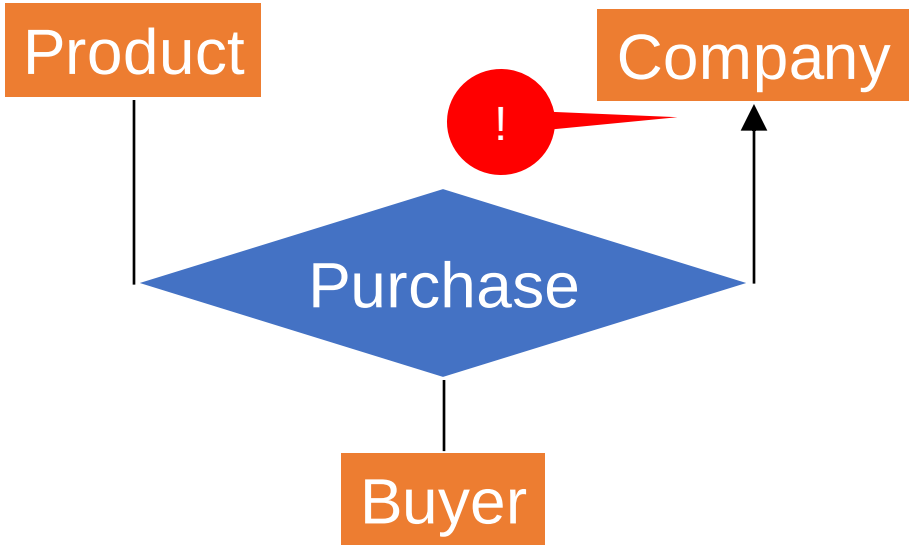



```
CREATE TABLE Product (
    PID INT PRIMARY KEY,...);
CREATE TABLE Company (
    CID INT PRIMARY KEY,...);
CREATE TABLE Buyer (
    BID INT PRIMARY KEY,...);

CREATE TABLE Purchase (
    PID INT REFERENCES Product,
    CID INT REFERENCES Company,
    BID INT REFERENCES Buyer,
    ...);
```

Purchase

PID	CID	BID
0035 (soap)	345 (Dial)	555 (Alice)
0035 (soap)	345 (Dial)	666 (Bob)
0041 (lotion)	123 (Nivea)	555 (Alice)
…		

Multi-Way Relationships

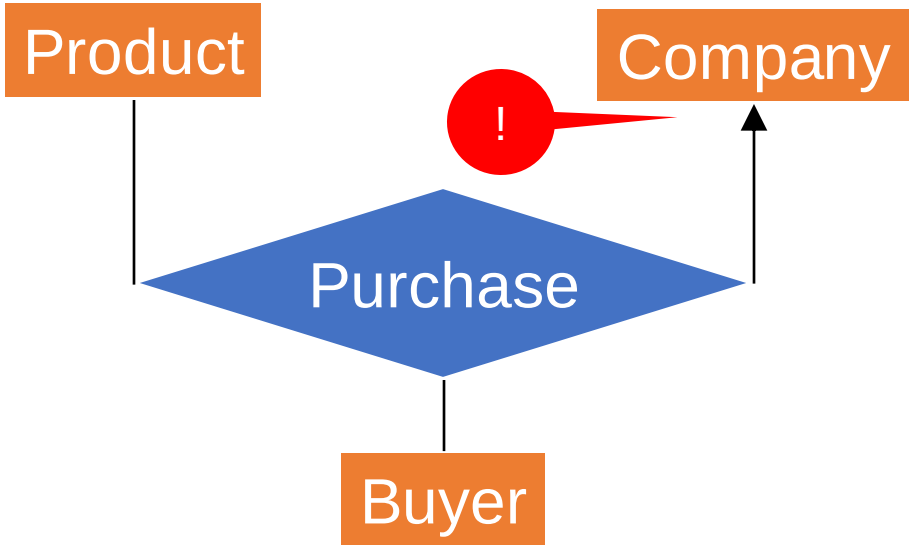



```
CREATE TABLE Product (
    PID INT PRIMARY KEY,...);
CREATE TABLE Company (
    CID INT PRIMARY KEY,...);
CREATE TABLE Buyer (
    BID INT PRIMARY KEY,...);

CREATE TABLE Purchase (
    PID INT REFERENCES Product,
    CID INT REFERENCES Company,
    BID INT REFERENCES Buyer,
    ...);
```

Purchase

PID	CID	BID
0035 (soap)	345 (Dial)	555 (Alice)
0035 (soap)	345 (Dial)	666 (Bob)
0041 (lotion)	123 (Nivea)	555 (Alice)
…		

Multi-Way Relationships



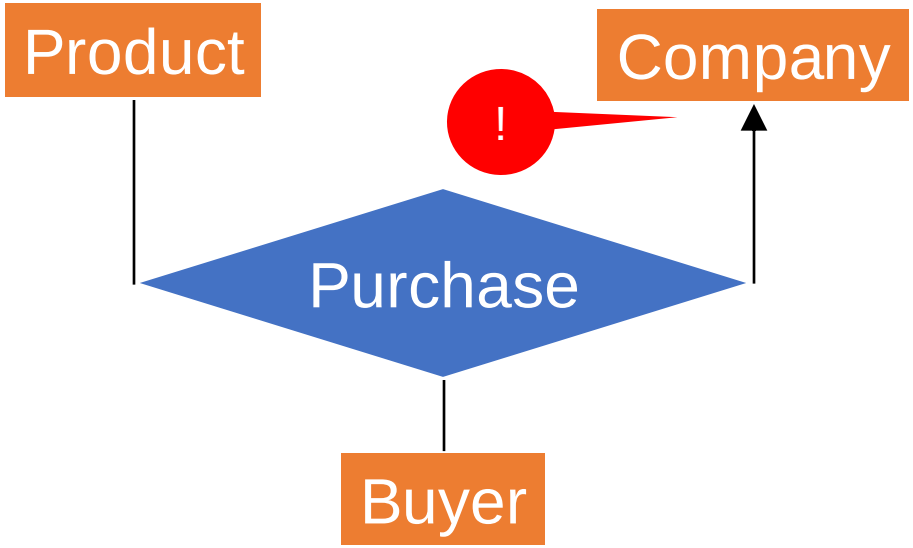

```
CREATE TABLE Product (
    PID INT PRIMARY KEY,...);
CREATE TABLE Company (
    CID INT PRIMARY KEY,...);
CREATE TABLE Buyer (
    BID INT PRIMARY KEY,...);

CREATE TABLE Purchase (
    PID INT REFERENCES Product,
    CID INT REFERENCES Company,
    BID INT REFERENCES Buyer,
    ...);
```

Purchase

PID	CID	BID
0035 (soap)	345 (Dial)	555 (Alice)
0035 (soap)	345 (Dial)	666 (Bob)
0041 (lotion)	123 (Nivea)	555 (Alice)
…		

Arrow means:
a buyer always buys a product from the same company

Multi-Way Relationships



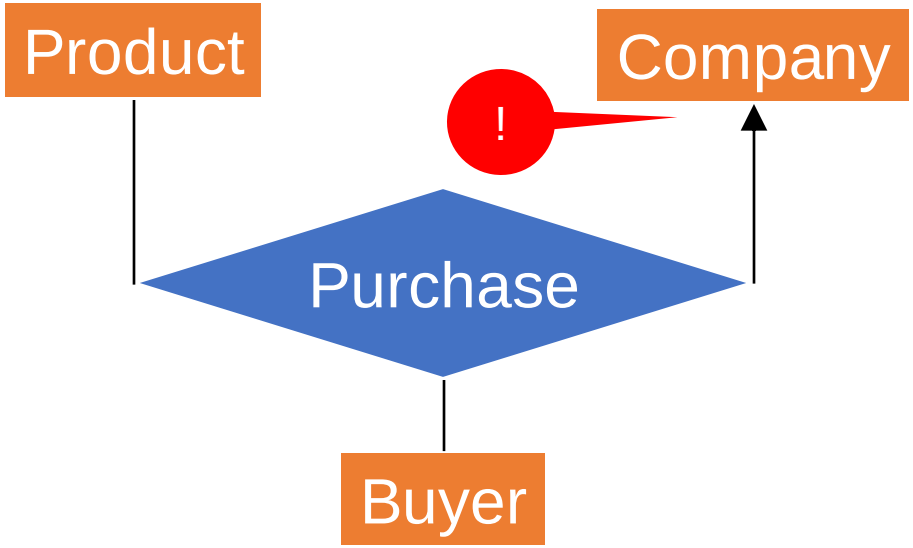

```
CREATE TABLE Product (
    PID INT PRIMARY KEY,...);
CREATE TABLE Company (
    CID INT PRIMARY KEY,...);
CREATE TABLE Buyer (
    BID INT PRIMARY KEY,...);

CREATE TABLE Purchase (
    PID INT REFERENCES Product,
    CID INT REFERENCES Company,
    BID INT REFERENCES Buyer,
    PRIMARY KEY (BID, PID),
    ...);
```

Purchase

PID	CID	BID
0035 (soap)	345 (Dial)	555 (Alice)
0035 (soap)	345 (Dial)	666 (Bob)
0041 (lotion)	123 (Nivea)	555 (Alice)
…		

Arrow means:
a buyer always buys a product from the same company

Multi-Way Relationships




```
CREATE TABLE Product (
    PID INT PRIMARY KEY,...);
CREATE TABLE Company (
    CID INT PRIMARY KEY,...);
CREATE TABLE Buyer (
    BID INT PRIMARY KEY,...);

CREATE TABLE Purchase (
    PID INT REFERENCES Product,
    CID INT REFERENCES Company,
    BID INT REFERENCES Buyer,
    PRIMARY KEY (BID, PID),
    ...);
```

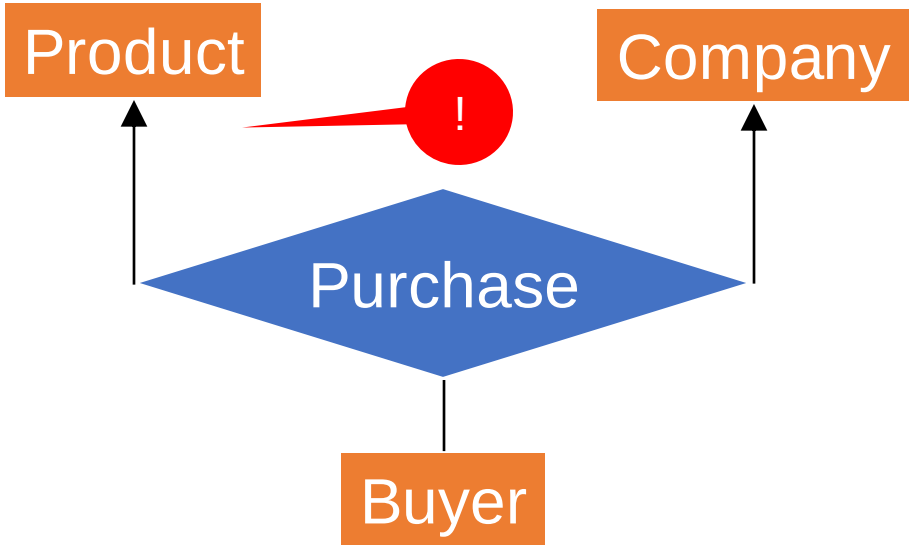
Purchase

PID	CID	BID
0035 (soap)	345 (Dial)	555 (Alice)
0035 (soap)	345 (Dial)	666 (Bob)
0041 (lotion)	123 (Nivea)	555 (Alice)
0035 (soap)	**456 (Dove)**	**555 (Alice)**

Arrow means:
a buyer always buys a product from the same company

Not allowed

Multi-Way Relationships



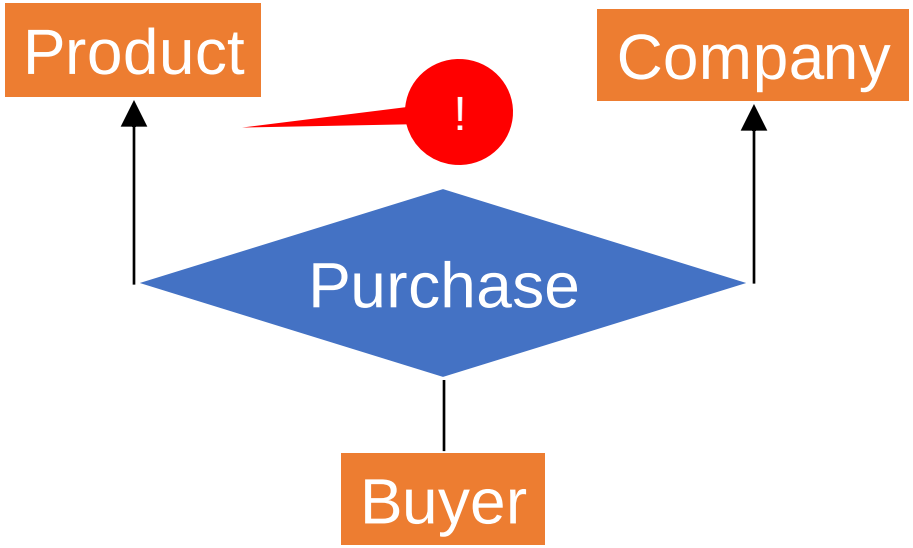

```
CREATE TABLE Product (
    PID INT PRIMARY KEY,...);
CREATE TABLE Company (
    CID INT PRIMARY KEY,...);
CREATE TABLE Buyer (
    BID INT PRIMARY KEY,...);

CREATE TABLE Purchase (
    PID INT REFERENCES Product,
    CID INT REFERENCES Company,
    BID INT REFERENCES Buyer,
    PRIMARY KEY (BID, PID),
    ...);
```

Purchase

PID	CID	BID
0035 (soap)	345 (Dial)	555 (Alice)
0035 (soap)	345 (Dial)	666 (Bob)
0041 (lotion)	123 (Nivea)	555 (Alice)
…		

What do **two arrows** this mean?

Multi-Way Relationships



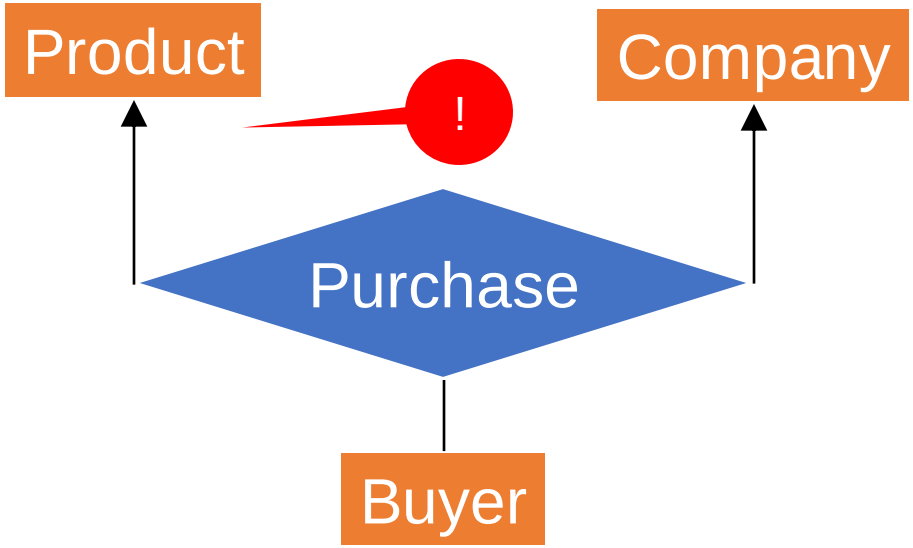

```
CREATE TABLE Product (
    PID INT PRIMARY KEY,...);
CREATE TABLE Company (
    CID INT PRIMARY KEY,...);
CREATE TABLE Buyer (
    BID INT PRIMARY KEY,...);

CREATE TABLE Purchase (
    PID INT REFERENCES Product,
    CID INT REFERENCES Company,
    BID INT REFERENCES Buyer,
    PRIMARY KEY (BID, PID),
    ...);
```

Purchase

PID	CID	BID
0035 (soap)	345 (Dial)	555 (Alice)
0035 (soap)	345 (Dial)	666 (Bob)
0041 (lotion)	123 (Nivea)	555 (Alice)
…		

What do **two arrows** this mean?
We read each arrow separately:

Multi-Way Relationships



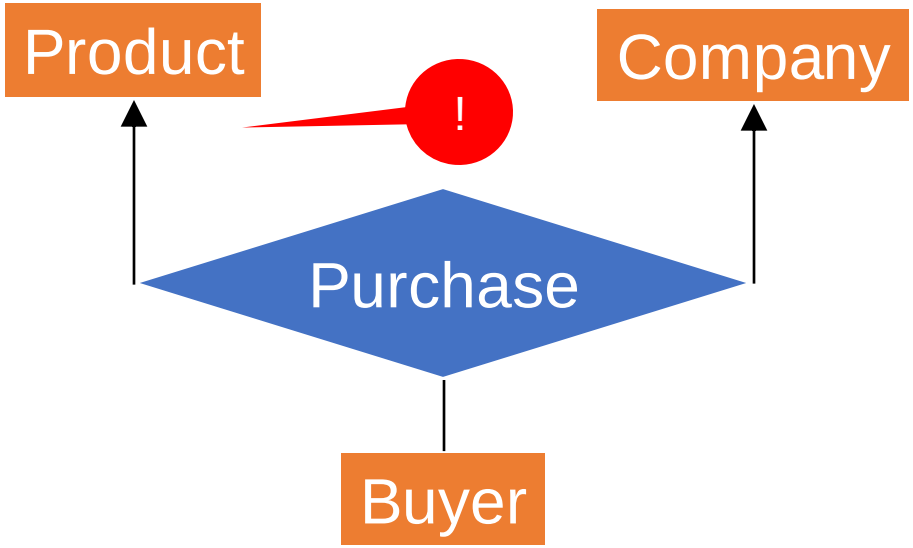

```
CREATE TABLE Product (
    PID INT PRIMARY KEY,...);
CREATE TABLE Company (
    CID INT PRIMARY KEY,...);
CREATE TABLE Buyer (
    BID INT PRIMARY KEY,...);

CREATE TABLE Purchase (
    PID INT REFERENCES Product,
    CID INT REFERENCES Company,
    BID INT REFERENCES Buyer,
    UNIQUE (BID, PID),
    UNIQUE (BID, CID),
    ...);
```

Purchase

PID	CID	BID
0035 (soap)	345 (Dial)	555 (Alice)
0035 (soap)	345 (Dial)	666 (Bob)
0041 (lotion)	123 (Nivea)	555 (Alice)
…		

What do **two arrows** this mean?
We read each arrow separately:

Multi-Way Relationships



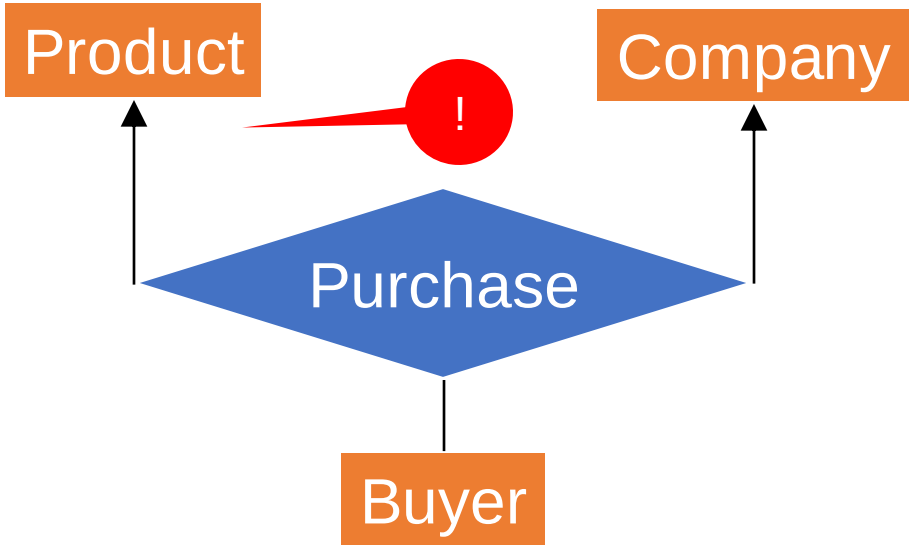

```
CREATE TABLE Product (
    PID INT PRIMARY KEY,...);
CREATE TABLE Company (
    CID INT PRIMARY KEY,...);
CREATE TABLE Buyer (
    BID INT PRIMARY KEY,...);

CREATE TABLE Purchase (
    PID INT REFERENCES Product,
    CID INT REFERENCES Company,
    BID INT REFERENCES Buyer,
    UNIQUE (BID, PID),
    UNIQUE (BID, CID),
    ...);
```

Purchase

PID	CID	BID
0035 (soap)	345 (Dial)	555 (Alice)
0035 (soap)	345 (Dial)	666 (Bob)
0041 (lotion)	123 (Nivea)	555 (Alice)
…		

What do **two arrows** this mean?
We read each arrow separately:
…
and every buyer buys at most one product from each company

Multi-Way Relationships




```
CREATE TABLE Product (
    PID INT PRIMARY KEY,...);
CREATE TABLE Company (
    CID INT PRIMARY KEY,...);
CREATE TABLE Buyer (
    BID INT PRIMARY KEY,...);

CREATE TABLE Purchase (
    PID INT REFERENCES Product,
    CID INT REFERENCES Company,
    BID INT REFERENCES Buyer,
    UNIQUE (BID, PID),
    UNIQUE (BID, CID),
    ...);
```

Purchase

PID	CID	BID
0035 (soap)	345 (Dial)	555 (Alice)
0035 (soap)	345 (Dial)	666 (Bob)
0041 (lotion)	123 (Nivea)	555 (Alice)
06 (soft soap)	**345 (Dial)**	**555 (Alice)**

What do **two arrows** this mean?
We read each arrow separately:
…
and every buyer buys at most one product from each company

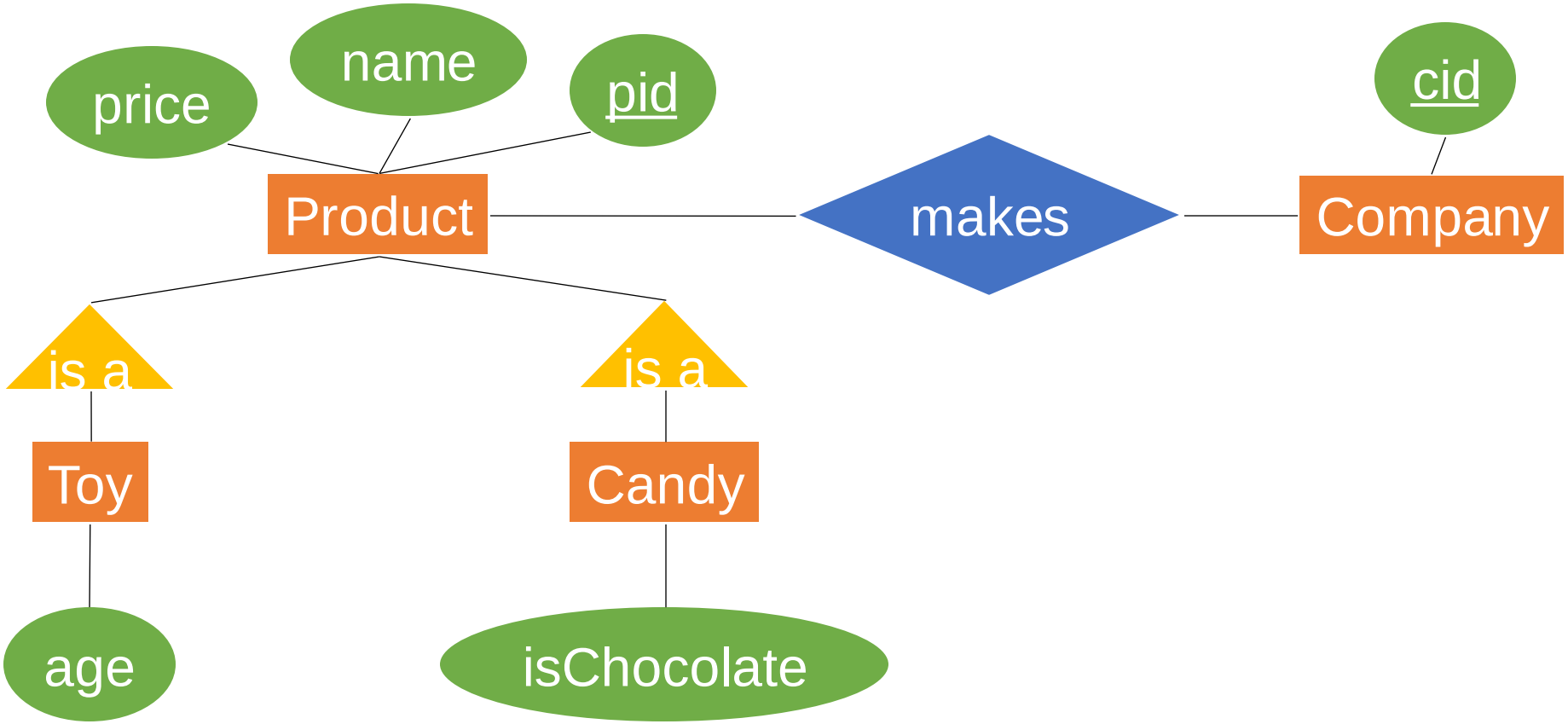
Summary of Relationships

- Multiplicity constraints:
  - Many-many: separate table
  - Many-one: no separate table
  - Multiplicity constraints: only in ER
- Referential integrity: foreign key NOT NULL
- Multi-way relationships: foreign key to each

Subclasses

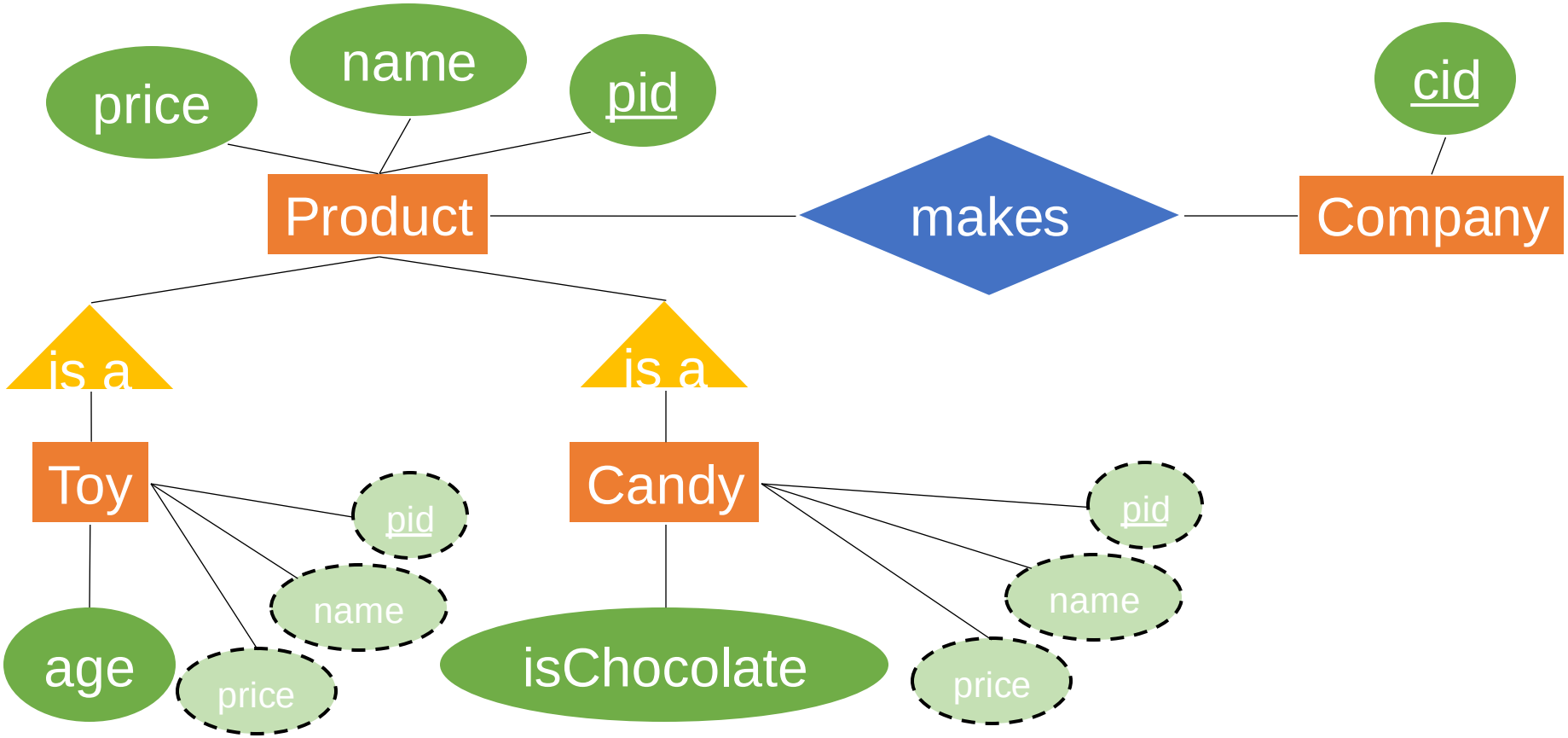
Subclassing

- Entity set may be a **subclass** of another entity set



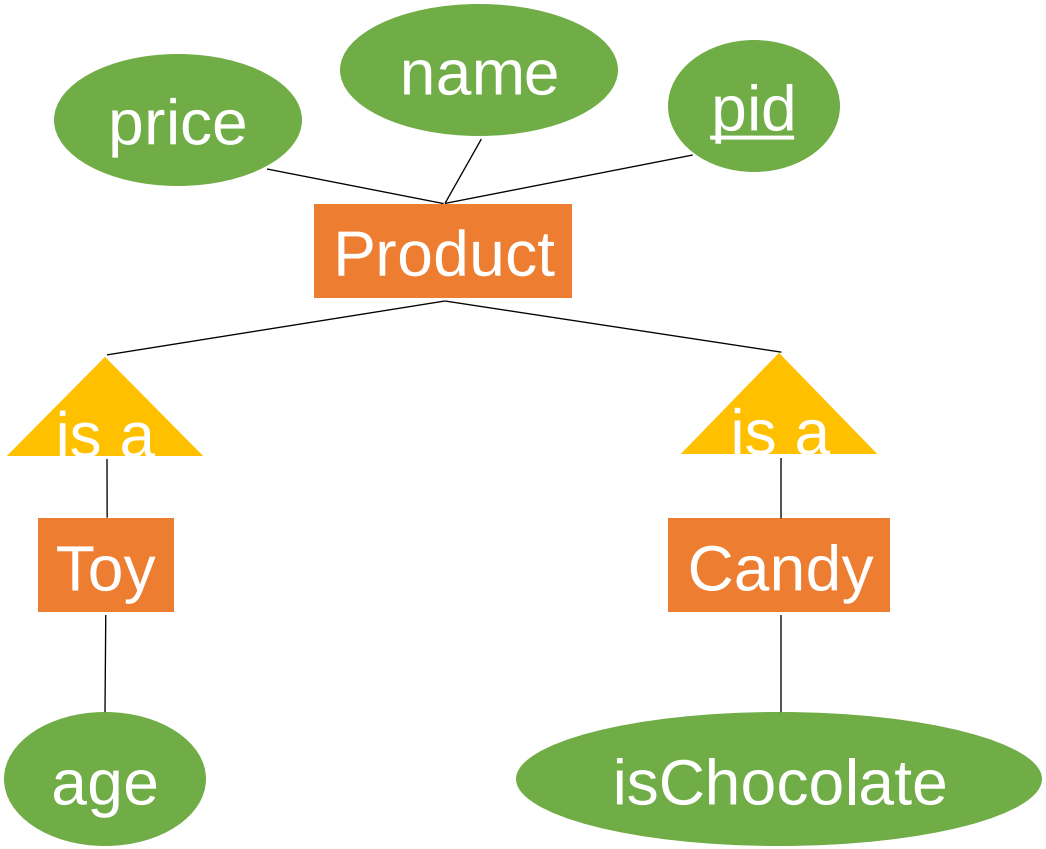
Subclassing

- Entity set may be a subclass of another entity set
- **Inherits** attributes of superclass



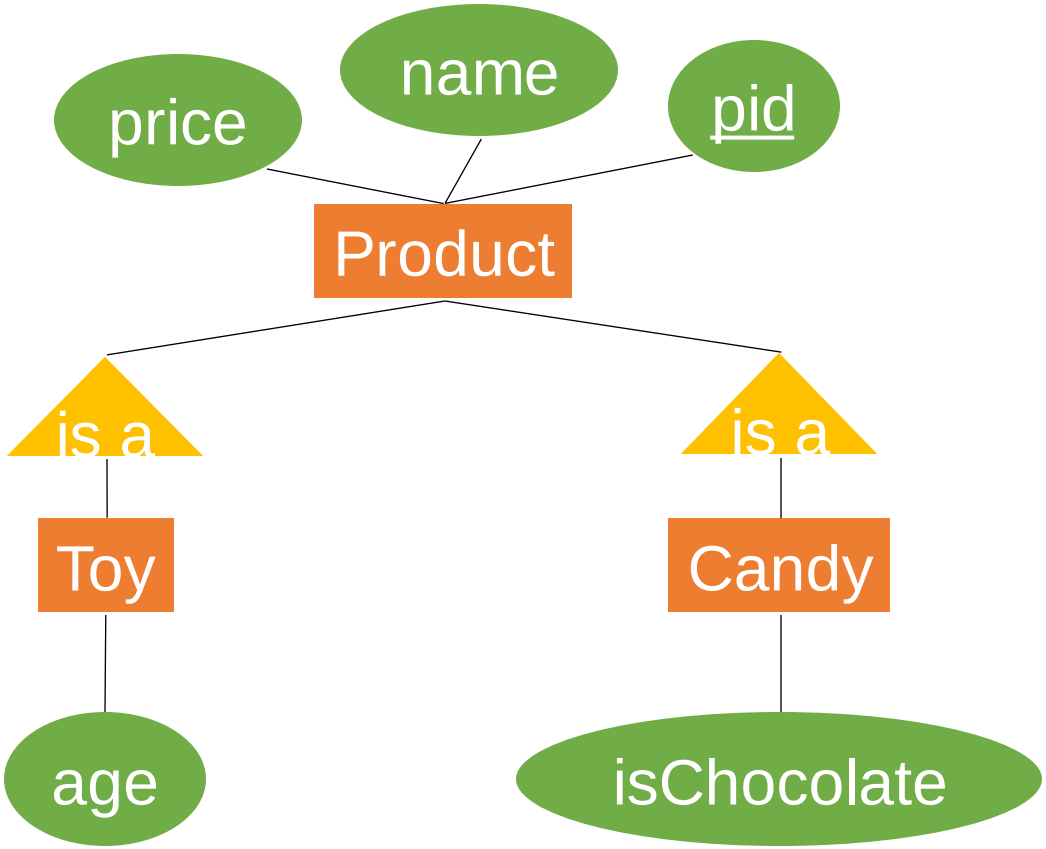
Representing Subclasses in SQL

- Each entity set becomes a relation



Representing Subclasses in SQL

- Each entity set becomes a relation

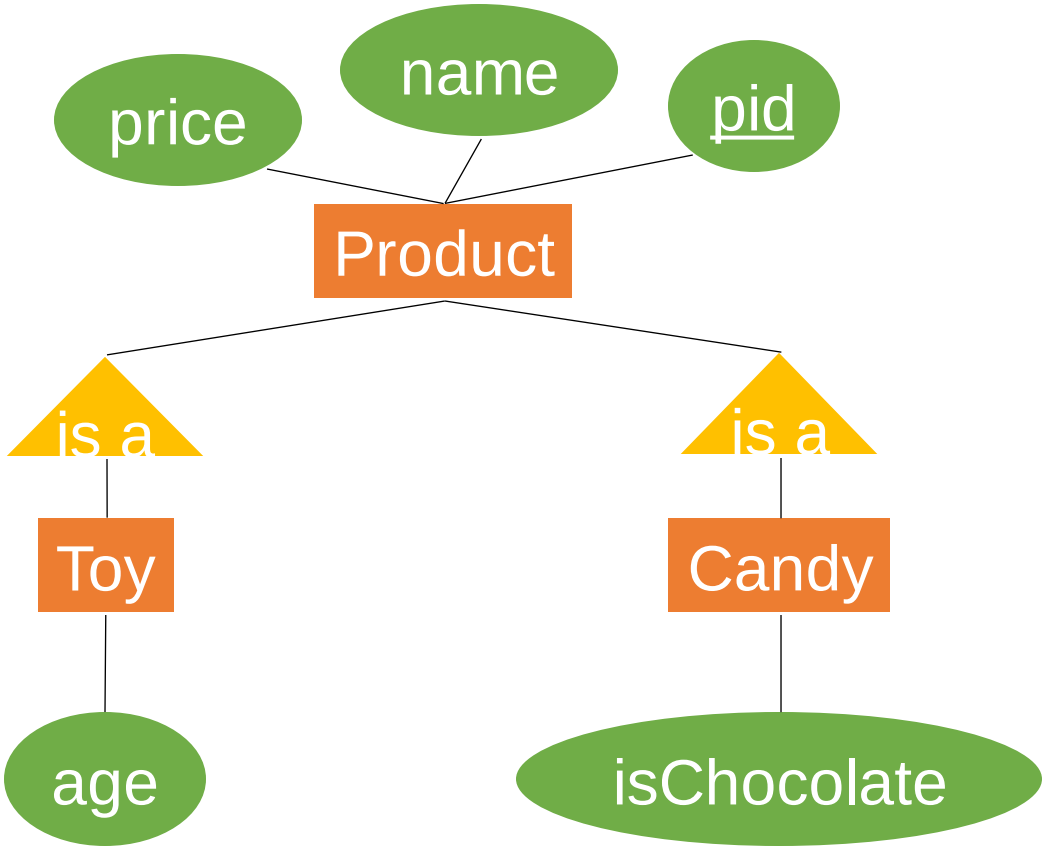


Product

pid	name	price
012	Lego	99
123	M&M	5
234	Computer	2999
345	Ball	15
456	Skittles	3
567	M&M toy	49

Representing Subclasses in SQL

- Each entity set becomes a relation



Product

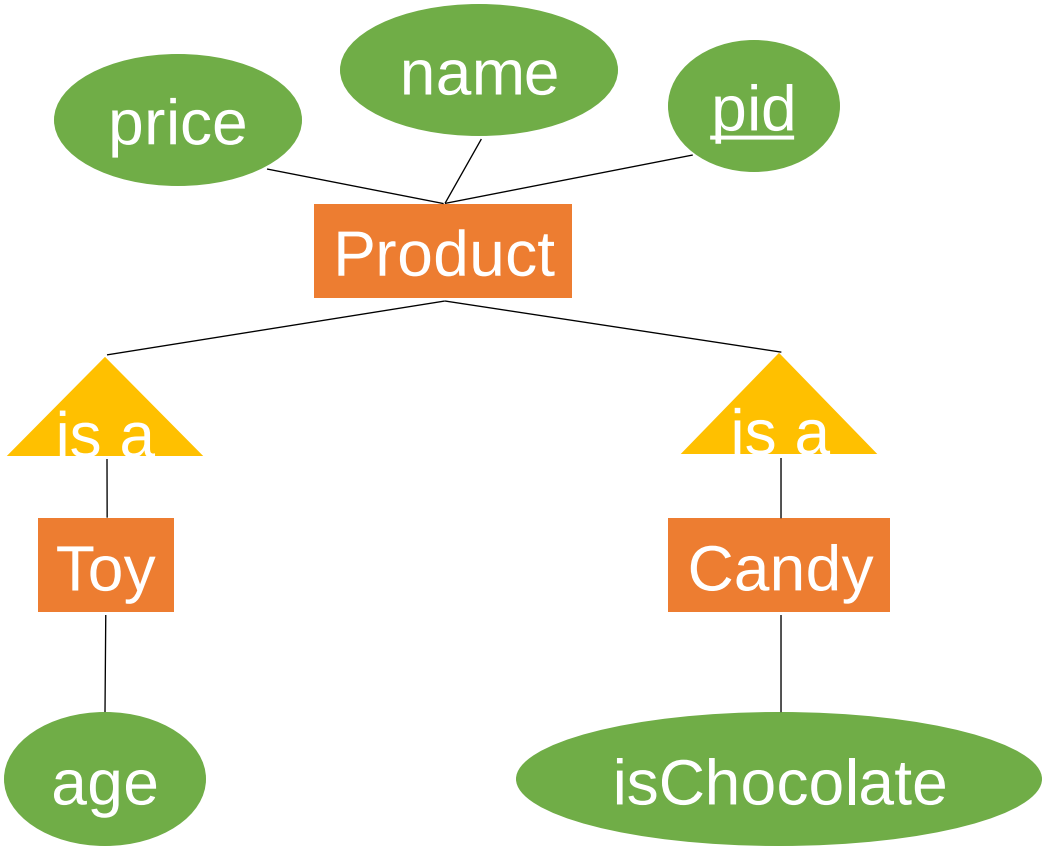
pid	name	price
012	Lego	99
123	M&M	5
234	Computer	2999
345	Ball	15
456	Skittles	3
567	M&M toy	49

Toy

pid	age
012	8
345	2
567	3

Representing Subclasses in SQL

- Each entity set becomes a relation



Product

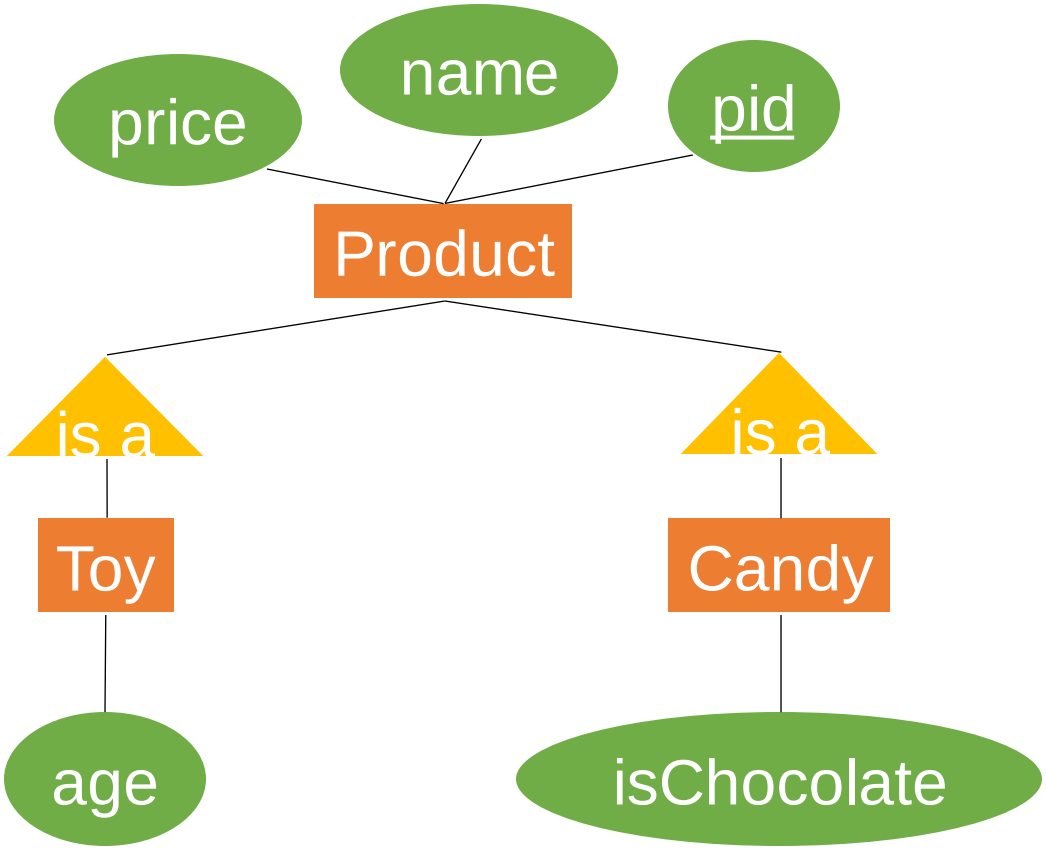
pid	name	price
012	Lego	99
123	M&M	5
234	Computer	2999
345	Ball	15
456	Skittles	3
567	M&M toy	49

Toy

pid	age
012	8
345	2
567	3

Representing Subclasses in SQL

- Each entity set becomes a relation



Product

pid	name	price
012	Lego	99
123	M&M	5
234	Computer	2999
345	Ball	15
456	Skittles	3
567	M&M toy	49

Toy

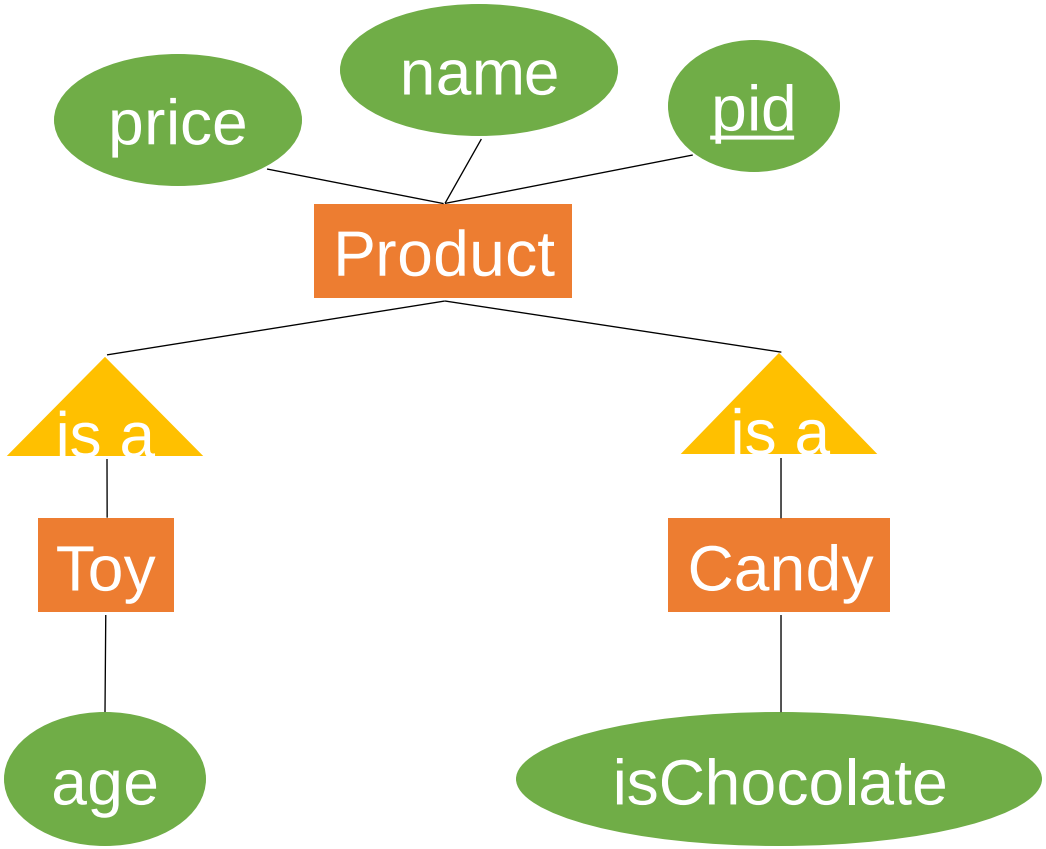
pid	age
012	8
345	2
567	3

Candy

pid	isChoc
123	yes
456	no
567	no

Representing Subclasses in SQL

- Each entity set becomes a relation



Product

pid	name	price
012	Lego	99
123	M&M	5
234	Computer	2999
345	Ball	15
456	Skittles	3
567	M&M toy	49

Toy

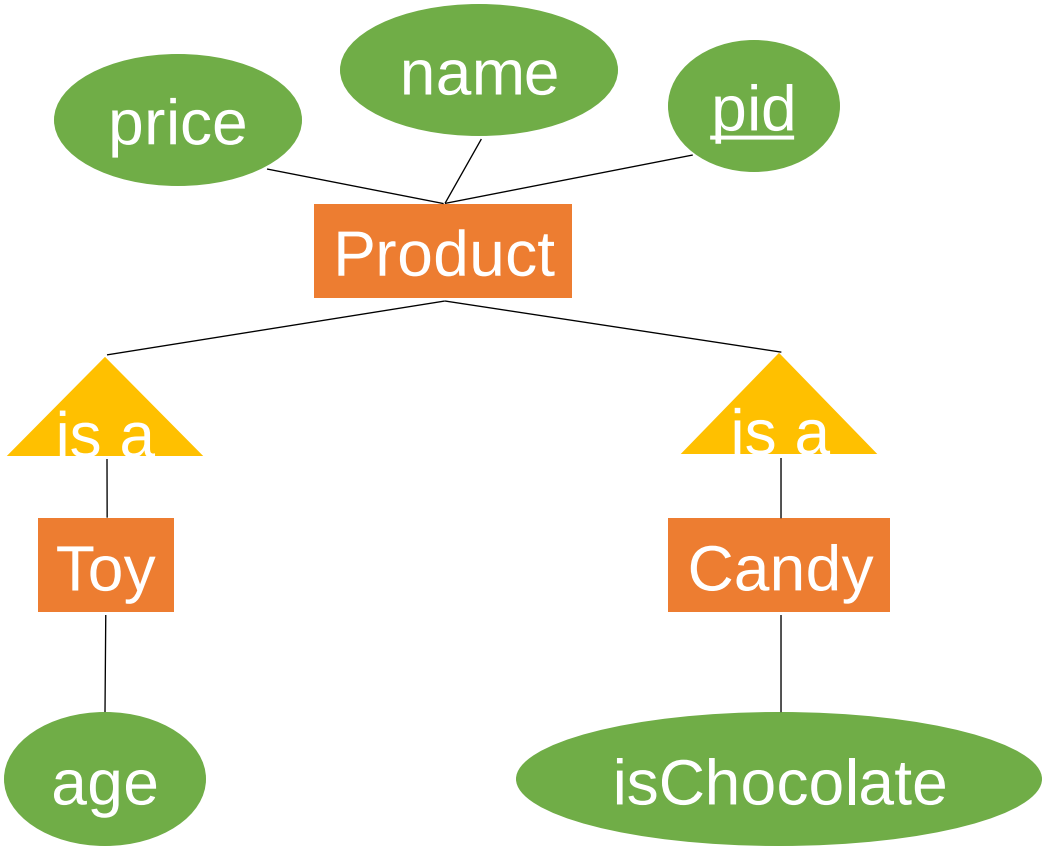
pid	age
012	8
345	2
567	3

Candy

pid	isChoc
123	yes
456	no
567	no

Representing Subclasses in SQL

- Each entity set becomes a relation



Product

pid	name	price
012	Lego	99
123	M&M	5
234	Computer	2999
345	Ball	15
456	Skittles	3
567	M&M toy	49

Toy

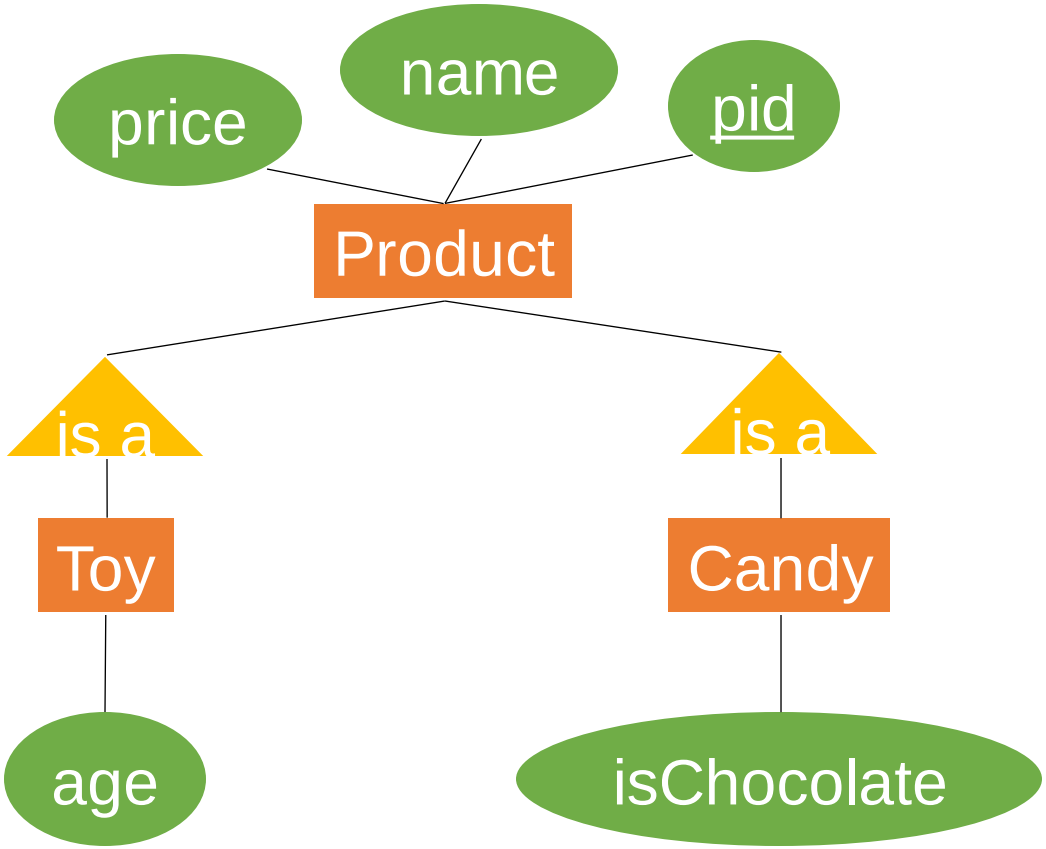
pid	age
012	8
345	2
567	3

Candy

pid	isChoc
123	yes
456	no
567	no

Representing Subclasses in SQL

- Each entity set becomes a relation



Product

pid	name	price
012	Lego	99
123	M&M	5
234	Computer	2999
345	Ball	15
456	Skittles	3
567	M&M toy	49

Toy

pid	age
012	8
345	2
567	3

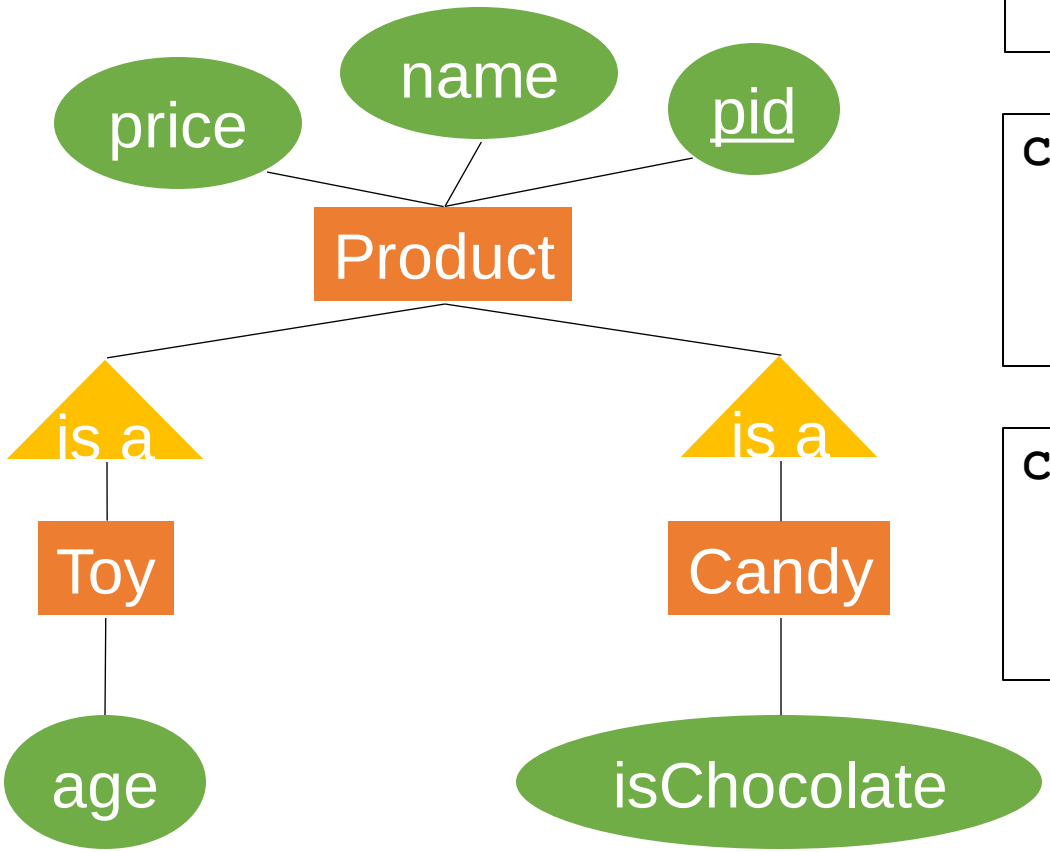
Candy

pid	isChoc
123	yes
456	no
567	no

567 MM& toy is both Toy and Candy!

Representing Subclasses in SQL

- Each entity set becomes a relation




```
CREATE TABLE Product (
  pid INT PRIMARY KEY,
  name TEXT,
  price FLOAT);
```

```
CREATE TABLE Toy (
  pid INT PRIMARY KEY
      REFERENCES Product,
  age INT);
```

```
CREATE TABLE Candy (
  pid INT PRIMARY KEY
      REFERENCES Product,
  isChocolate INT);
```

Discussion: Subclassing

- Entity set may be a subclass of another entity set
  - Inherits <u>all</u> the attributes of the superclass

- Some DBMSs support inheritance
  - However, we will simply represent inheritance using foreign keys and joins with the subclass and superclass

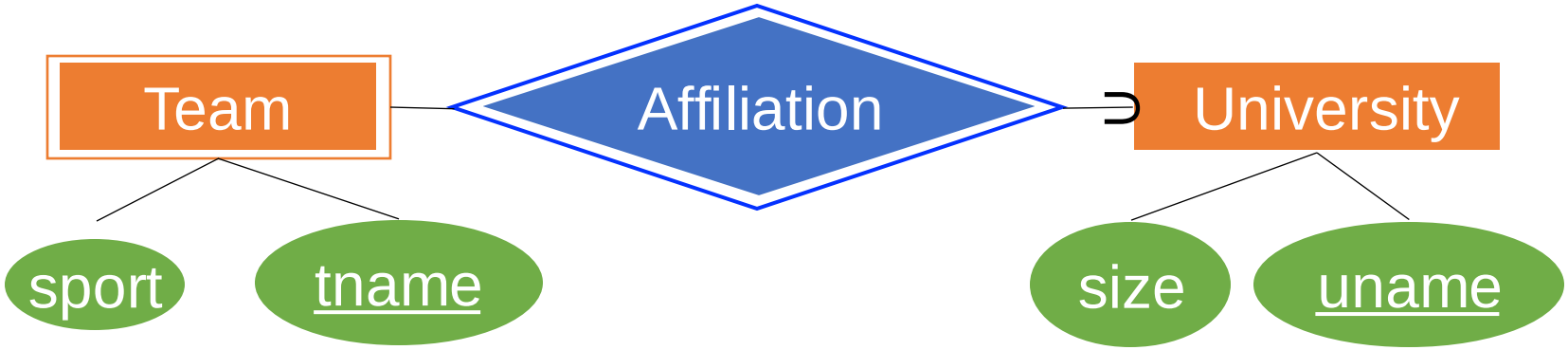
Weak Entity Sets

Weak Entity Set

- **Weak entity set:** key includes key from another entity set

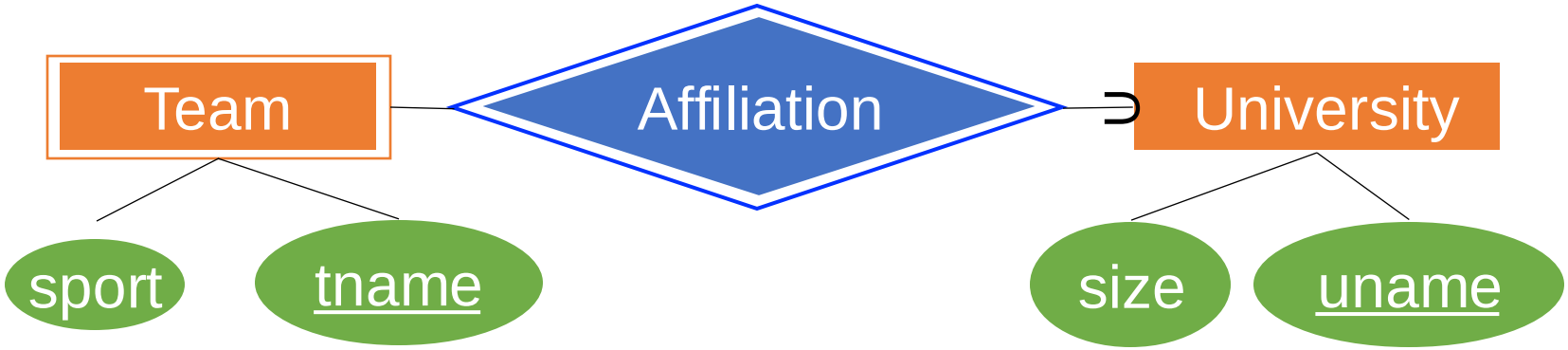
Weak Entity Set

- **Weak entity set:** key includes key from another entity set



Weak Entity Set

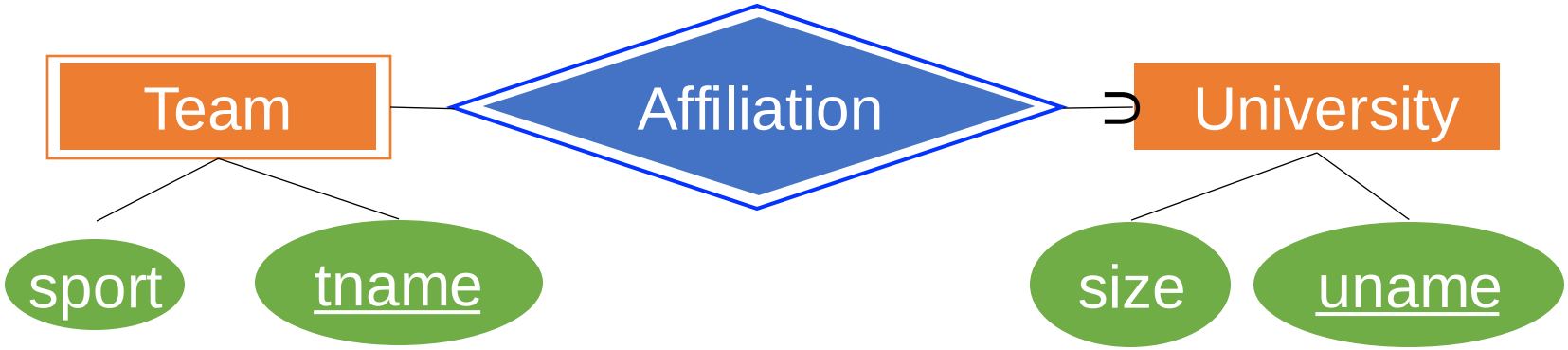
- **Weak entity set:** key includes key from another entity set



- The key of Team is (tname, uname) together
  - tname is not enough e.g. “Huskies” could be UCONN or UW

Weak Entity Set

- **Weak entity set:** key includes key from another entity set

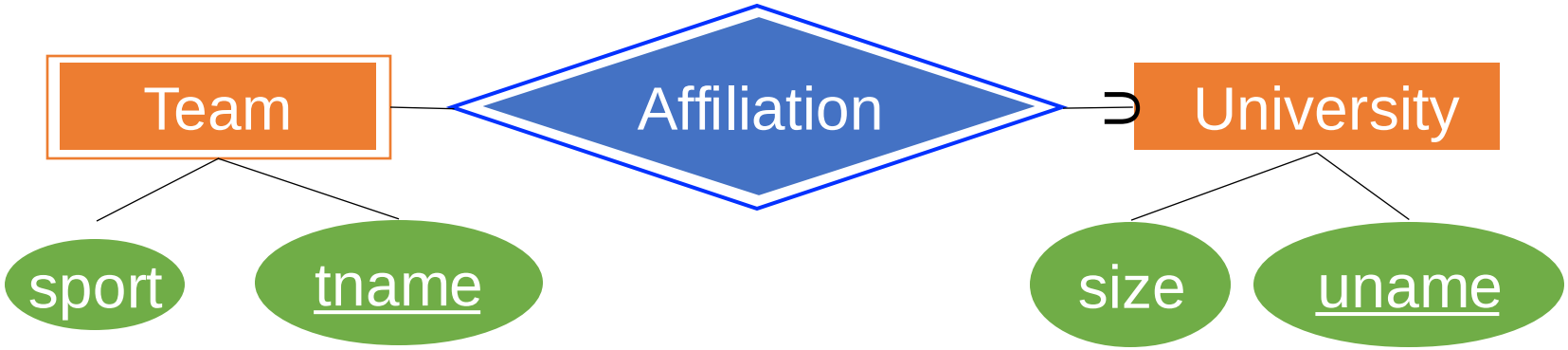


- The key of Team is (tname, uname) together
  - tname is not enough e.g. “Huskies” could be UCONN or UW

- The weak entity set and its relationship to the other (entity set's) key are both depicted with double-outlines

Weak Entity Set

- **Weak entity set:** key includes key from another entity set




```
CREATE TABLE University (
  uname TEXT PRIMARY KEY,
  size INT);
```

```
CREATE TABLE Team (
  uname TEXT REFERENCES University,
  tname TEXT,
  sport TEXT,
  PRIMARY KEY (uname, tname));
```

Discussion

What you should know:

- Design simple ER diagrams
- Understand:
  relationships, inheritance, weak entity sets
- Convert (correctly!) ER diagrams to SQL