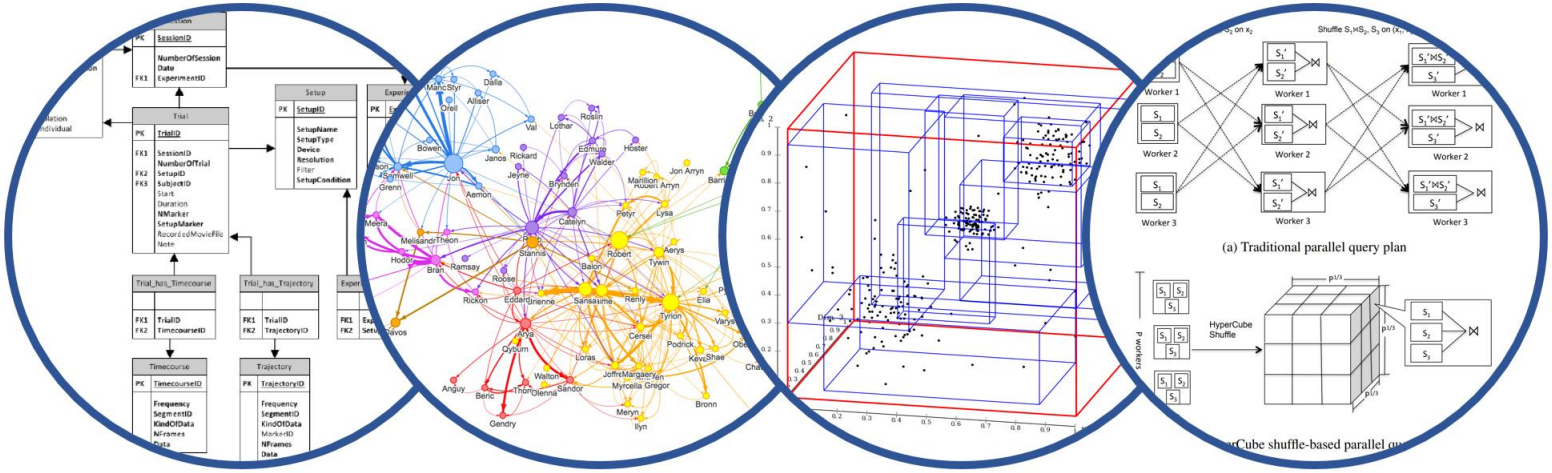


Introduction to Data Management

ER Diagrams

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

Announcements

- Quiz grades will be back early this week
- HW 4 released!
  - Topic is database design
  - Section has very helpful exercises

Recap – Relational Model

- SQL is parsed by the DBMS and translated into an RA plan that is more directly executable
- Both query types work on the assumption that you are **using relational data**
- The relational model specifies mechanics of how data <u>can</u> be organized
  - No prescription of how data <u>should</u> be organized

Goals for Today

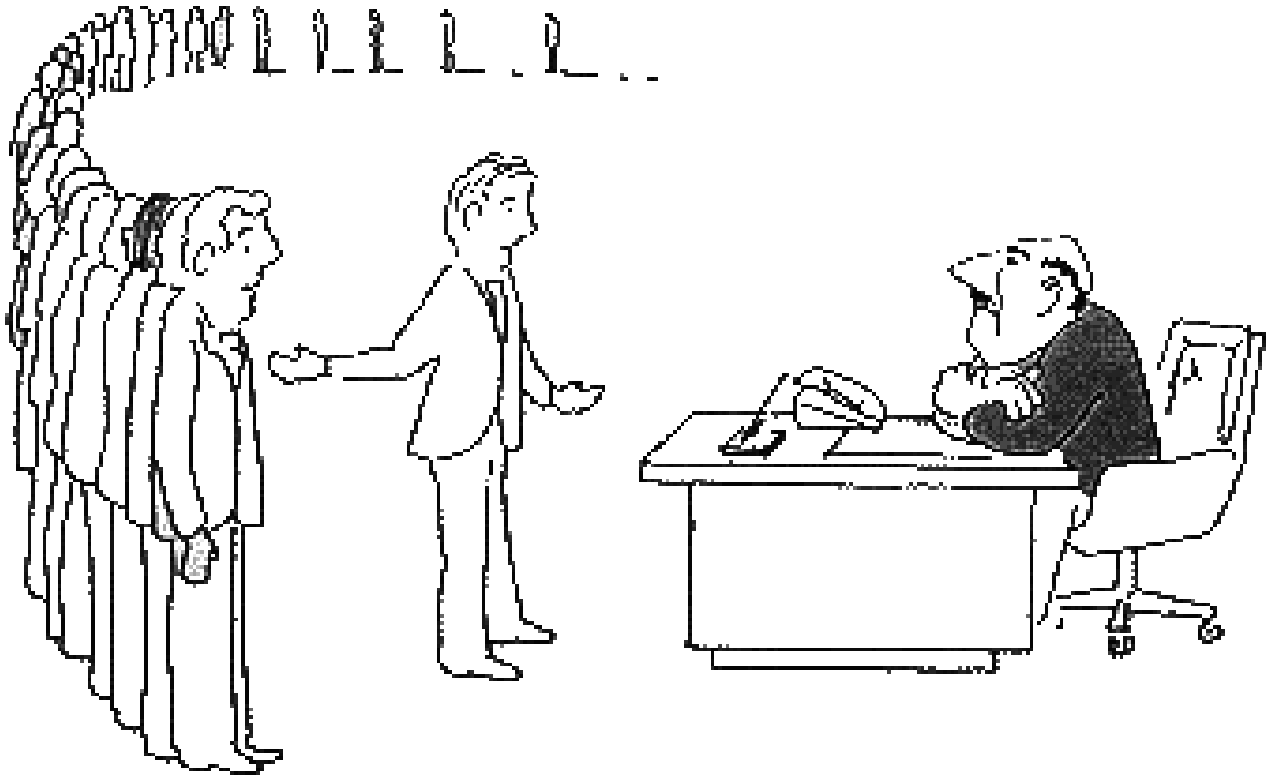
- With some application in mind, we can use an entity relationship (ER) diagram to conceptualize and communicate
- And with an ER diagram, we can use SQL to realize the model

Outline

- Introduce Database Design
- ER Diagrams
- ER-to-SQL conversion along the way
- Integrity constraints along the way

"I've got this great idea for an app…"

"I've got this great idea for an app..."



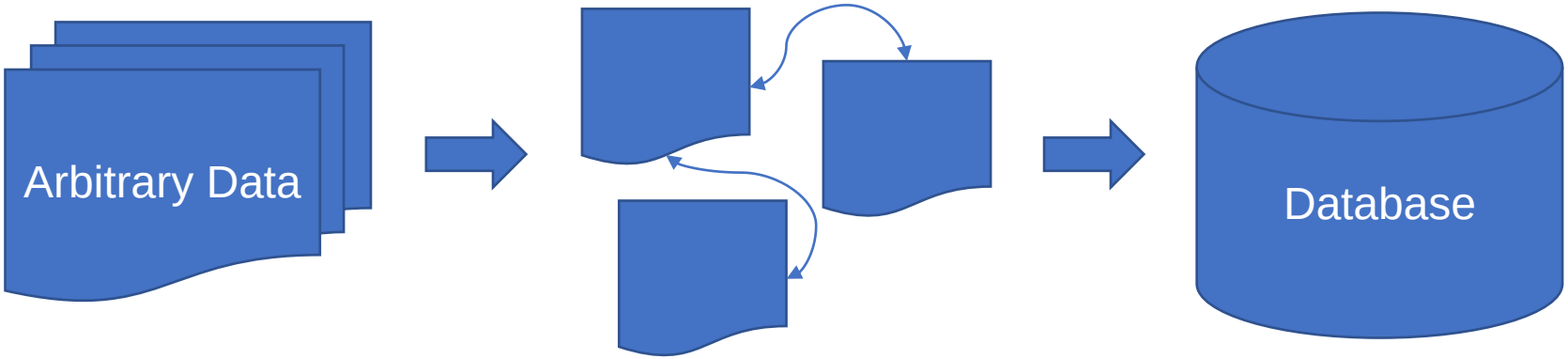
Database Design

- Communication is Key
- Other people are involved in the design process
- Non-computer scientists have to interact with the data too

Database Design

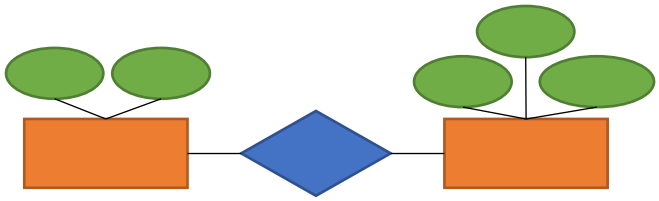
Database Design

Database Design or **Logical Design** or **Relational Schema Design** is the process of organizing data into a database model. This is done by considering what data needs to be stored and the interrelationship of the data.

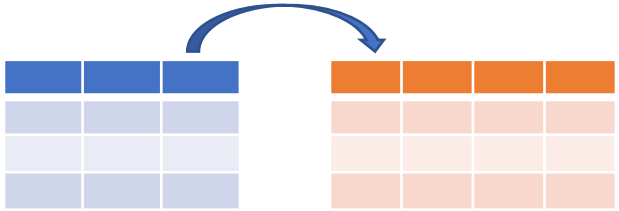


The Database Design Process

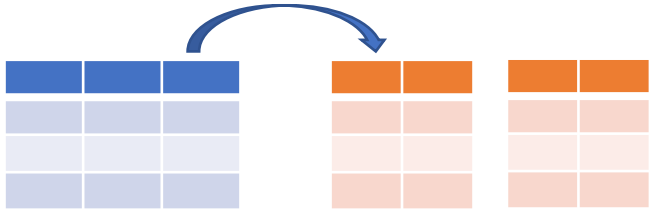
Conceptual Model



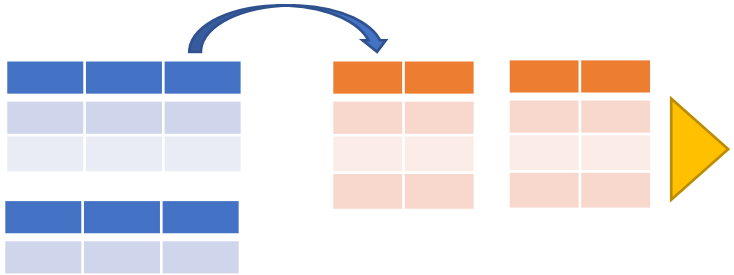
Relational Model
→ + Schema
→ + Constraints



Conceptual Schema
→ + Normalization



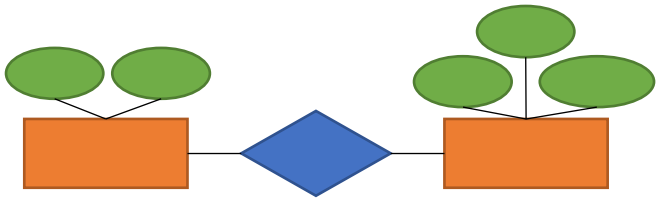
Physical Schema
→ + Partitioning
→ + Indexing



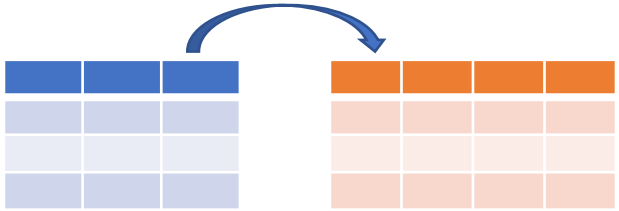
The Database Design Process

Today

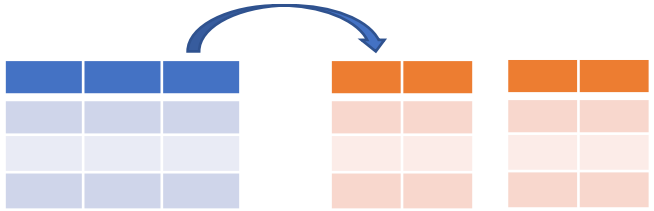
Conceptual Model



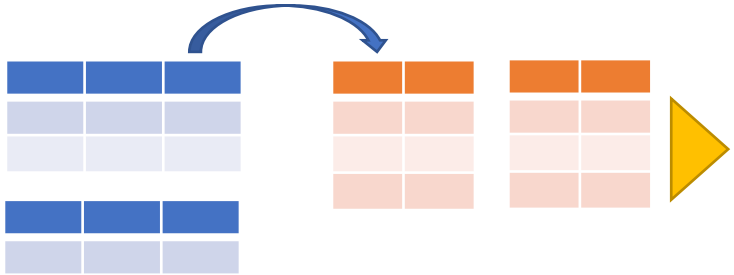
Relational Model
→ + Schema
→ + Constraints



Conceptual Schema
→ + Normalization

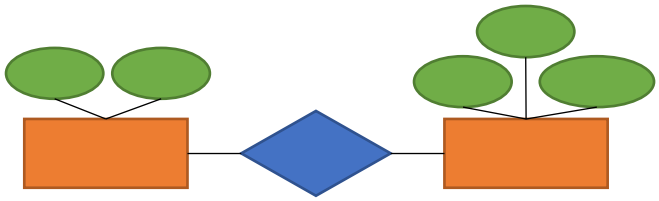


Physical Schema
→ + Partitioning
→ + Indexing

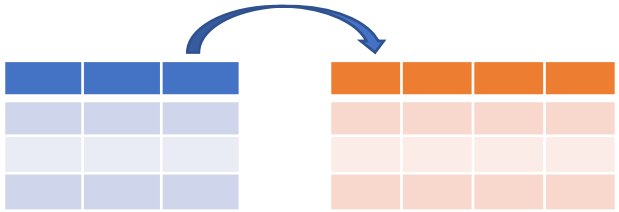


The Database Design Process

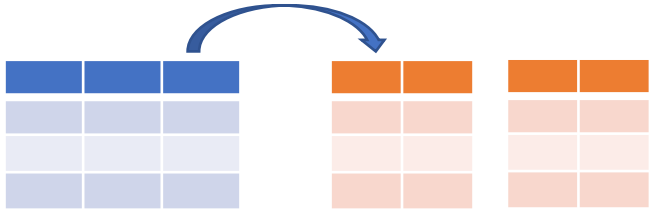
Conceptual Model



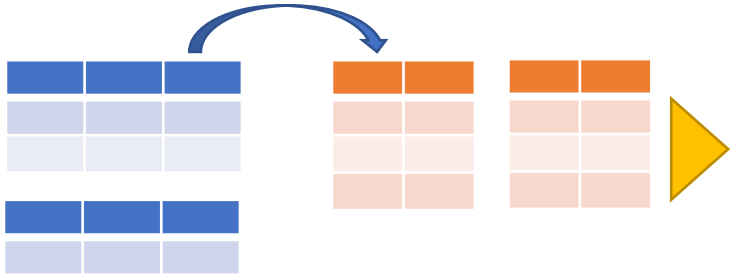
Relational Model
→ + Schema
→ + Constraints



Conceptual Schema
→ + Normalization

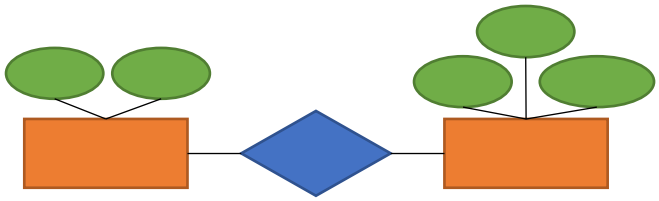


Physical Schema
→ + Partitioning
→ + Indexing

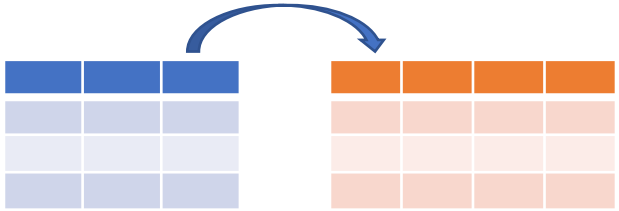


The Database Design Process

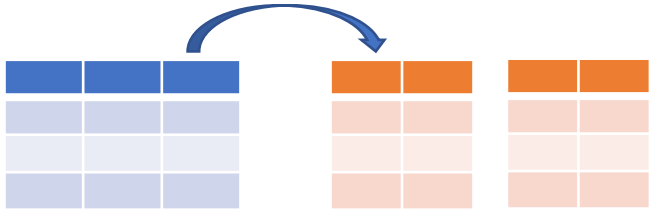
Conceptual Model



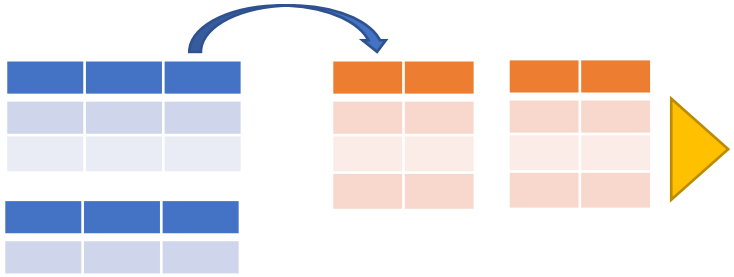
Relational Model
→ + Schema
→ + Constraints



Conceptual Schema
→ + Normalization

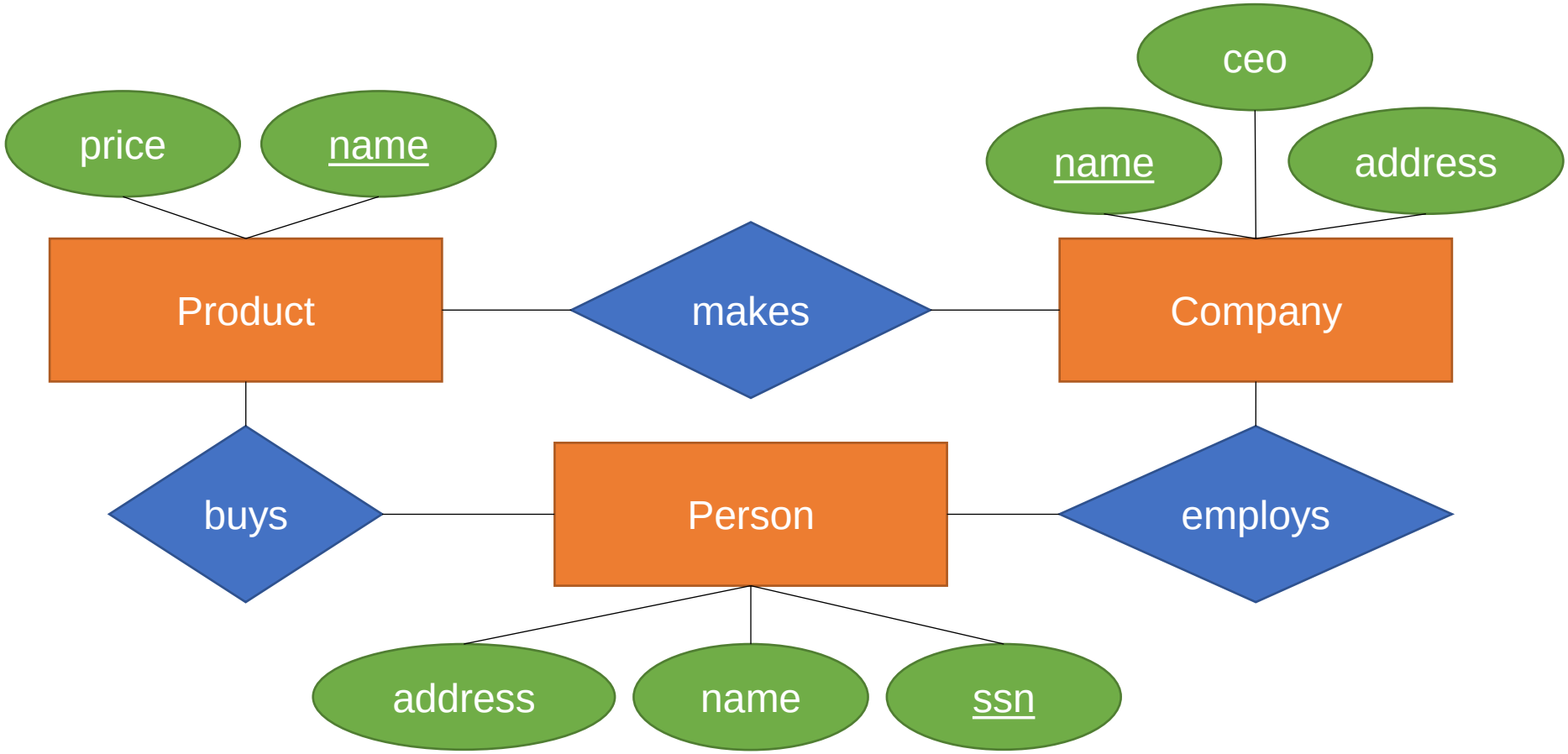


Physical Schema
→ + Partitioning
→ + Indexing



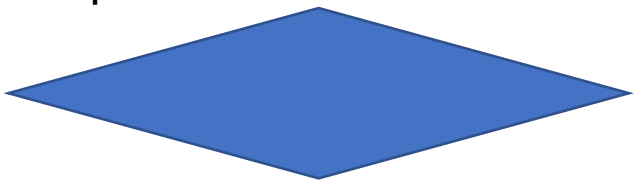
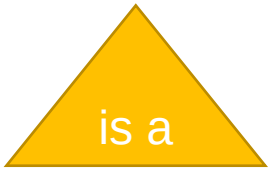
ER Diagrams

- A visual model let's us see the entire schema in one view



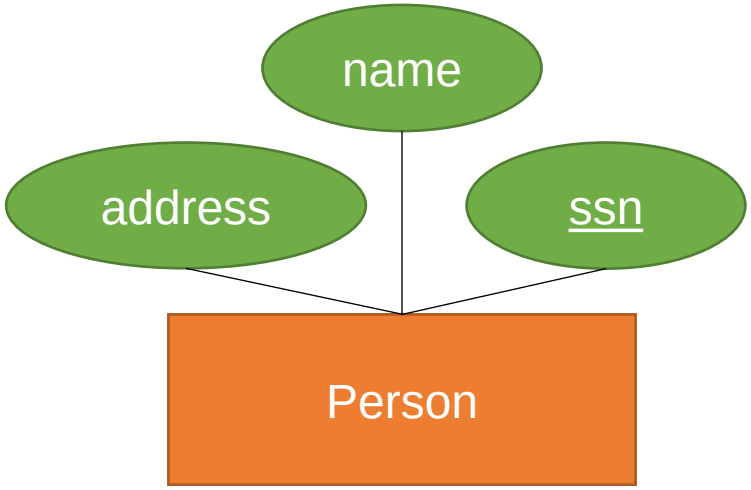
ER Diagram Building Blocks

- These are all the blocks we will learn about

Entity set 	Attribute 
Relationship 	Subclass 

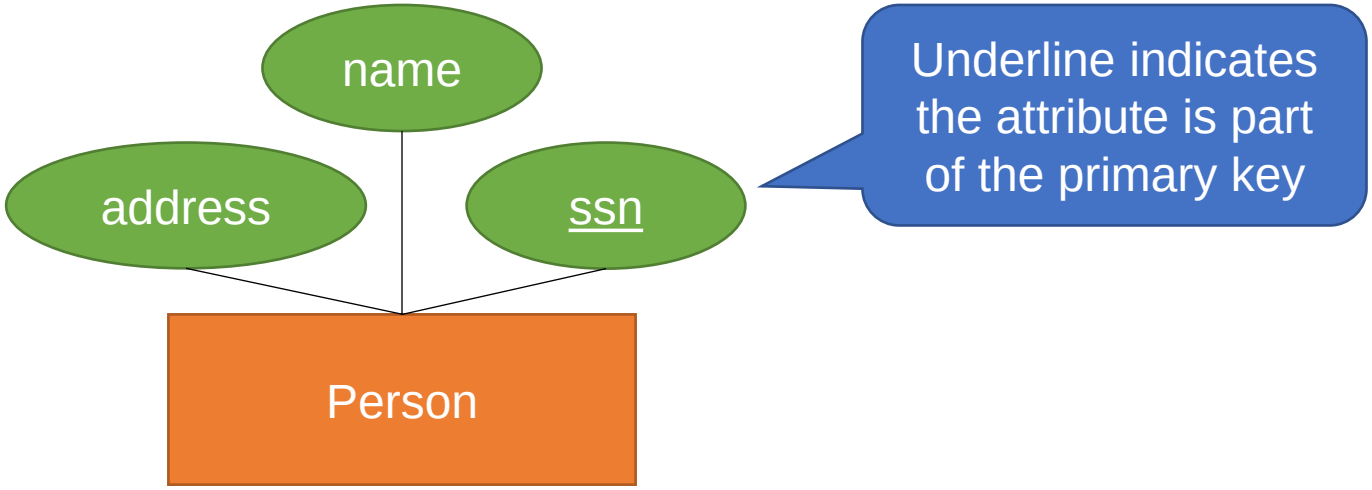
Entity Sets

- An "**entity set**" is like a **class**
- An **attribute** is like a **field**
- An "**entity**" is like a **object**
  - Corresponds to a row



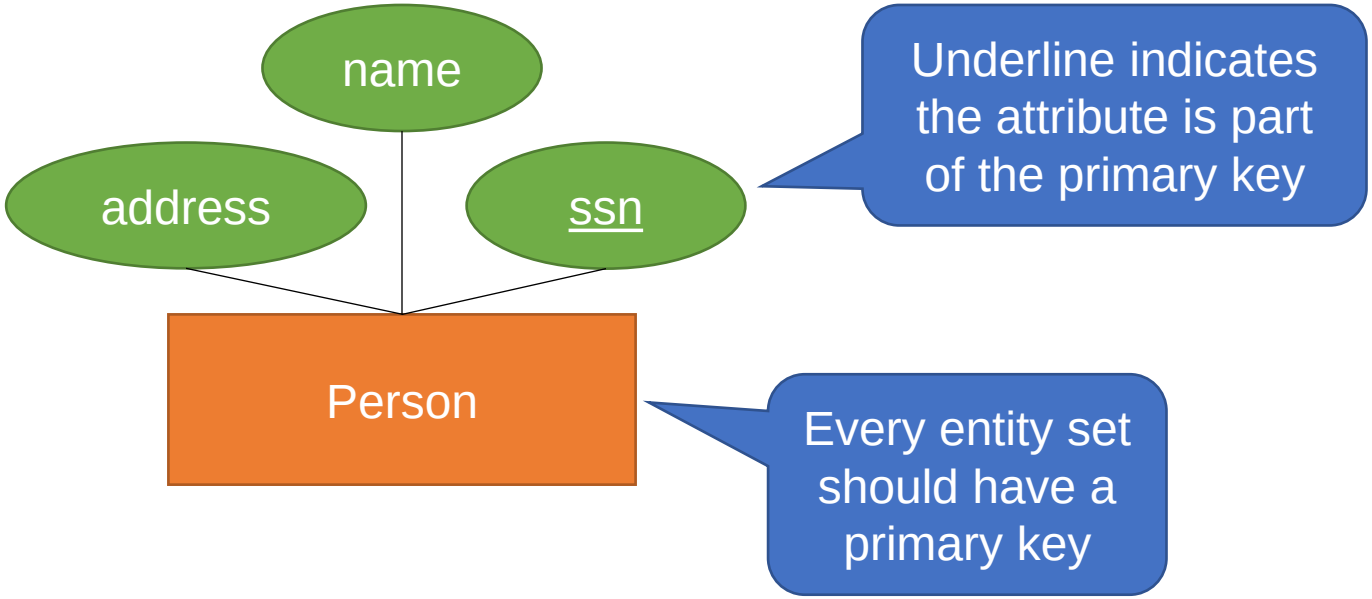
Entity Sets

- An "**entity set**" is like a **class**
- An **attribute** is like a **field**
- An "**entity**" is like a **object**
  - Corresponds to a row



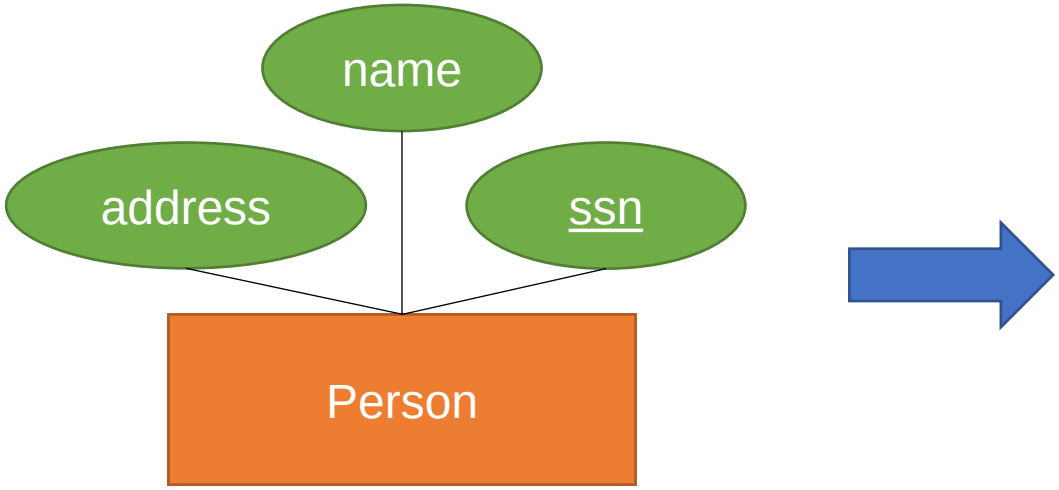
Entity Sets

- An "**entity set**" is like a **class**
- An **attribute** is like a **field**
- An "**entity**" is like a **object**
  - Corresponds to a row



Entity Sets

- An "**entity set**" is like a **class**
- An **attribute** is like a **field**
- An "**entity**" is like a **object**
  - Corresponds to a row

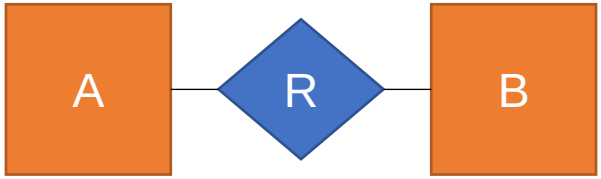



```
CREATE TABLE Person (
    ssn INT PRIMARY KEY,
    name TEXT,
    address TEXT);
```

Relations

Relationship

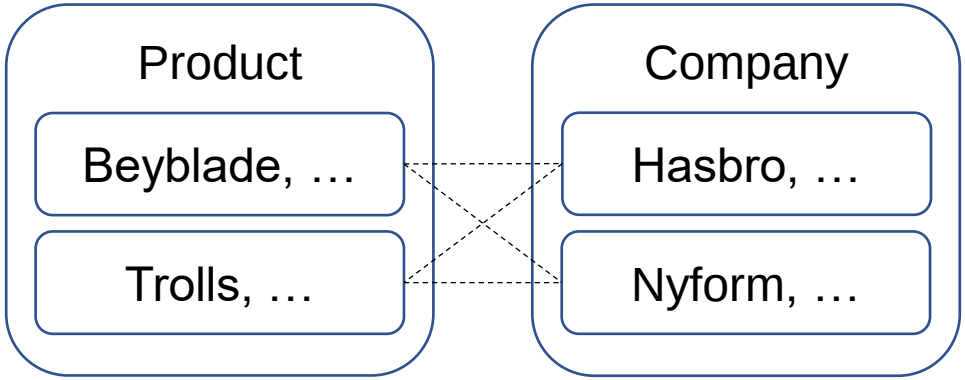
If A and B are sets, then a relation R is a subset of $A \times B$

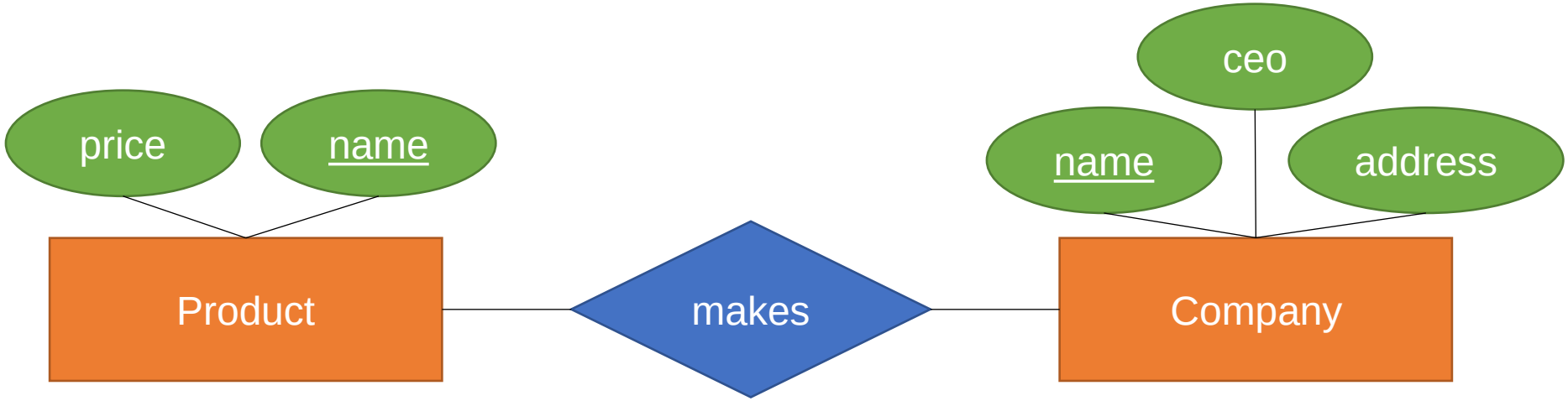


Relations

Relationship

If A and B are sets, then a relation R is a subset of $A \times B$

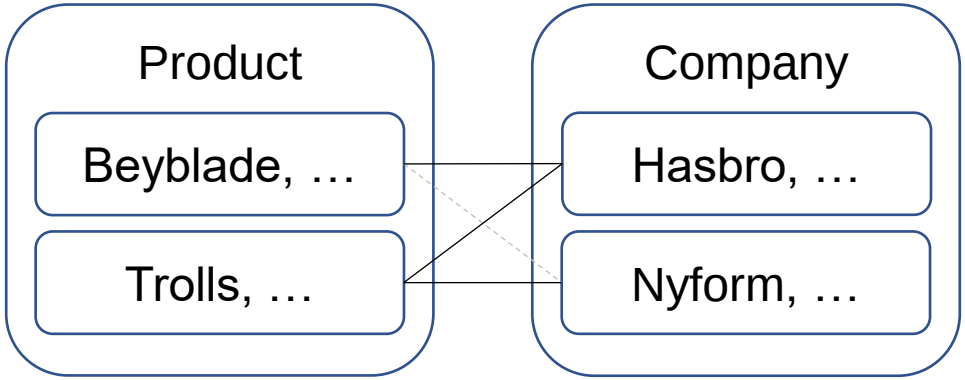


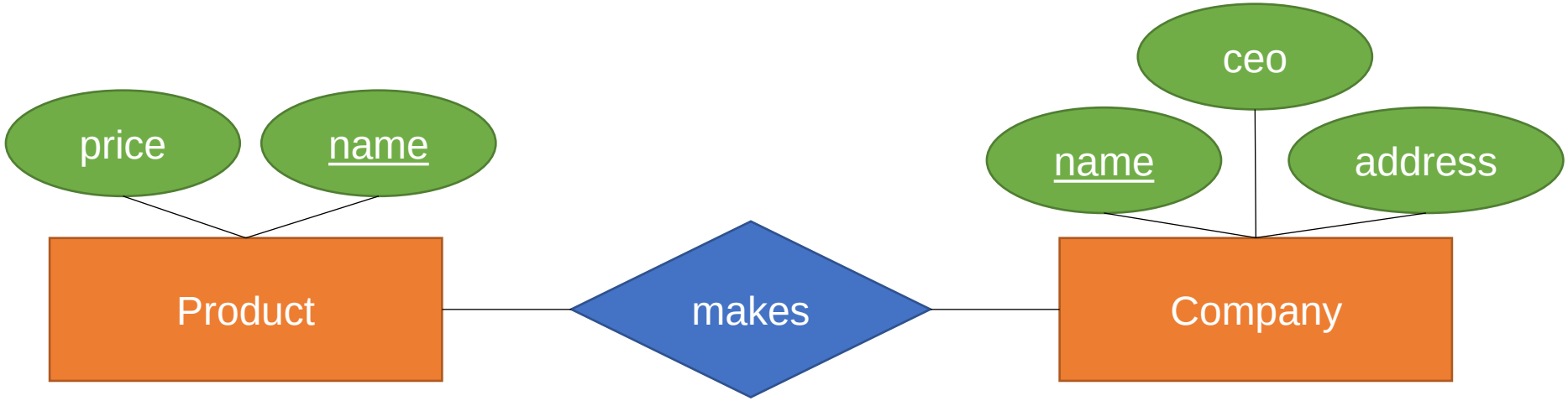


Relations

Relationship

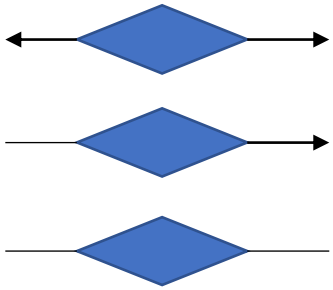
If A and B are sets, then a relation R is a subset of $A \times B$

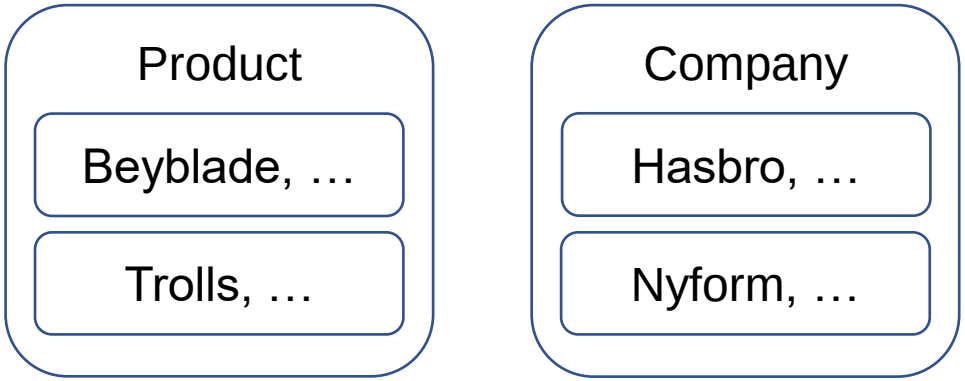




Relation Multiplicity

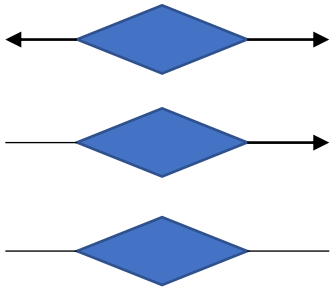
- One-to-one
- Many-to-one
- Many-to-many

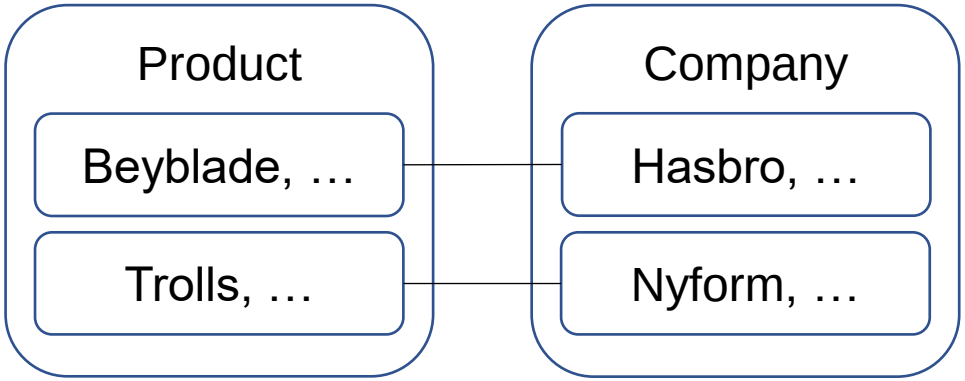




Relation Multiplicity

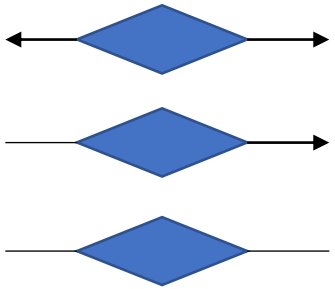
- **One-to-one**
- Many-to-one
- Many-to-many





Relation Multiplicity

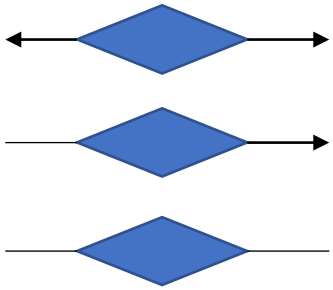
- **One-to-one**
- Many-to-one
- Many-to-many

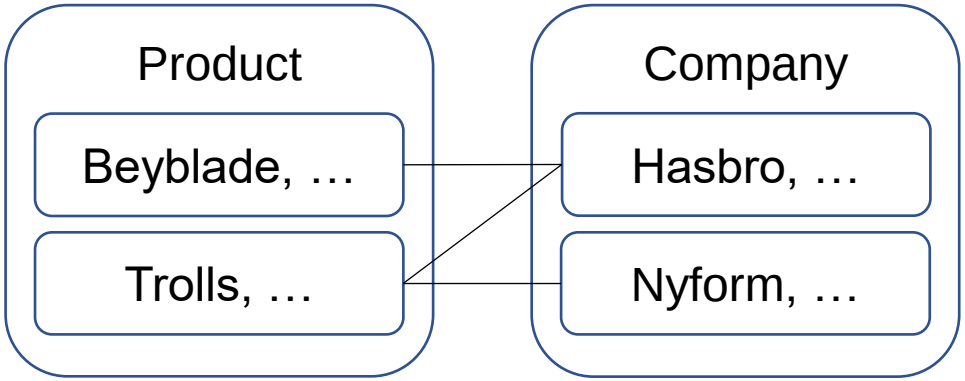


```
CREATE TABLE Product (
    name VARCHAR(100) PRIMARY KEY,
    ...);
CREATE TABLE Company (
    name VARCHAR(100) PRIMARY KEY,
    ...);
CREATE TABLE Makes (
    cname VARCHAR(100) UNIQUE REFERENCES Company,
    pname VARCHAR(100) UNIQUE REFERENCES Product,
    ...);
```

Relation Multiplicity

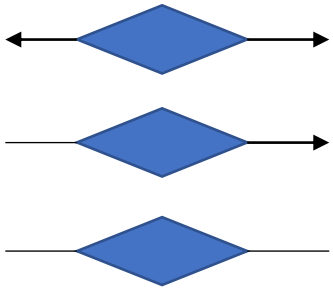
- One-to-one
- Many-to-one
- **Many-to-many**





Relation Multiplicity

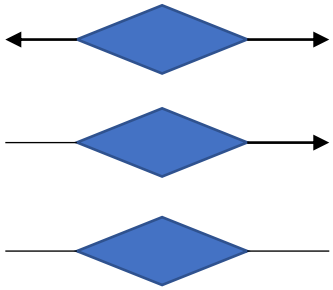
- One-to-one
- Many-to-one
- **Many-to-many**

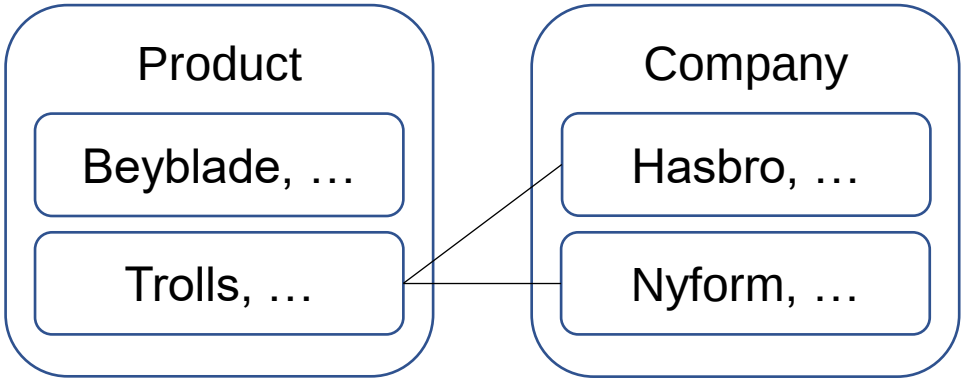


```
CREATE TABLE Product (
    name VARCHAR(100) PRIMARY KEY,
    ...);
CREATE TABLE Company (
    name VARCHAR(100) PRIMARY KEY,
    ...);
CREATE TABLE Makes (
    cname VARCHAR(100) UNIQUE REFERENCES Company,
    pname VARCHAR(100) UNIQUE REFERENCES Product,
    PRIMARY KEY (cname, pname),
    ...);
```

Relation Multiplicity

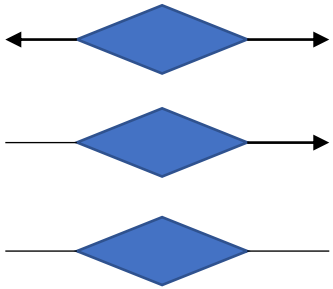
- One-to-one
- **Many-to-one**
- Many-to-many

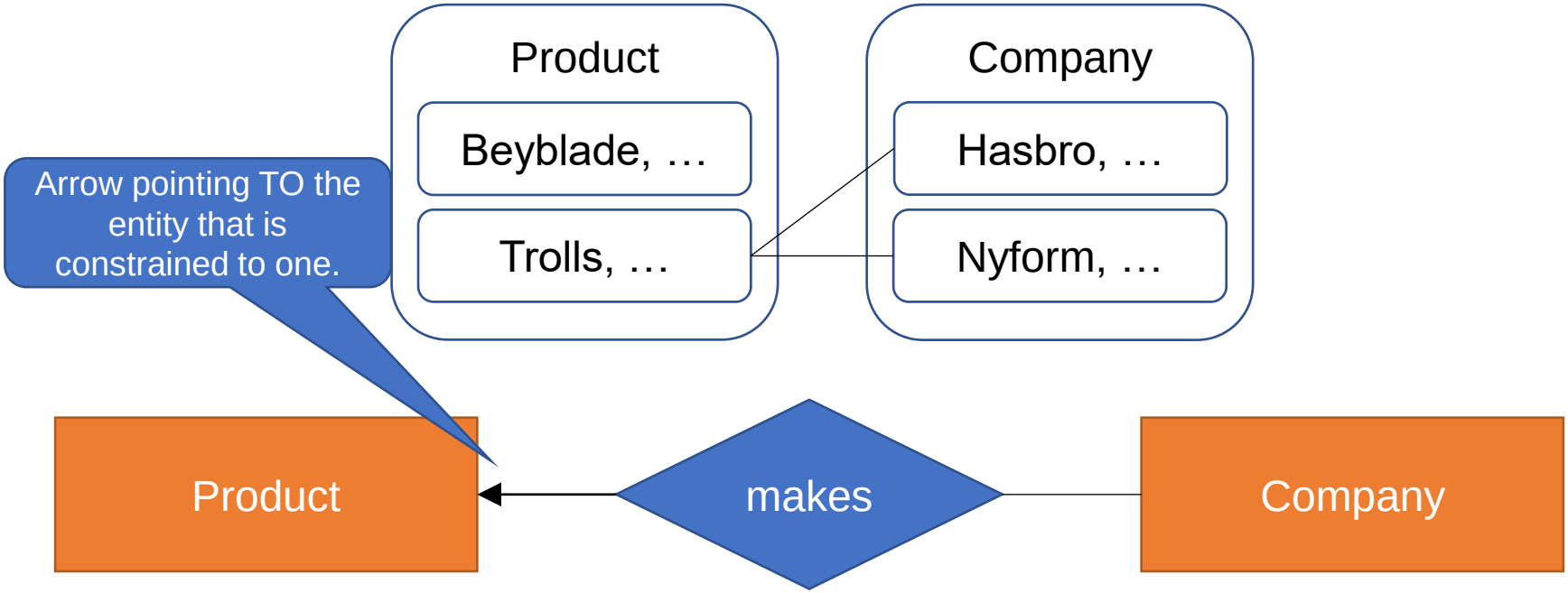




Relation Multiplicity

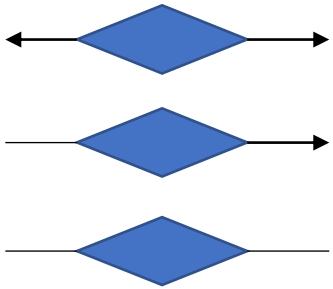
- One-to-one
- **Many-to-one**
- Many-to-many

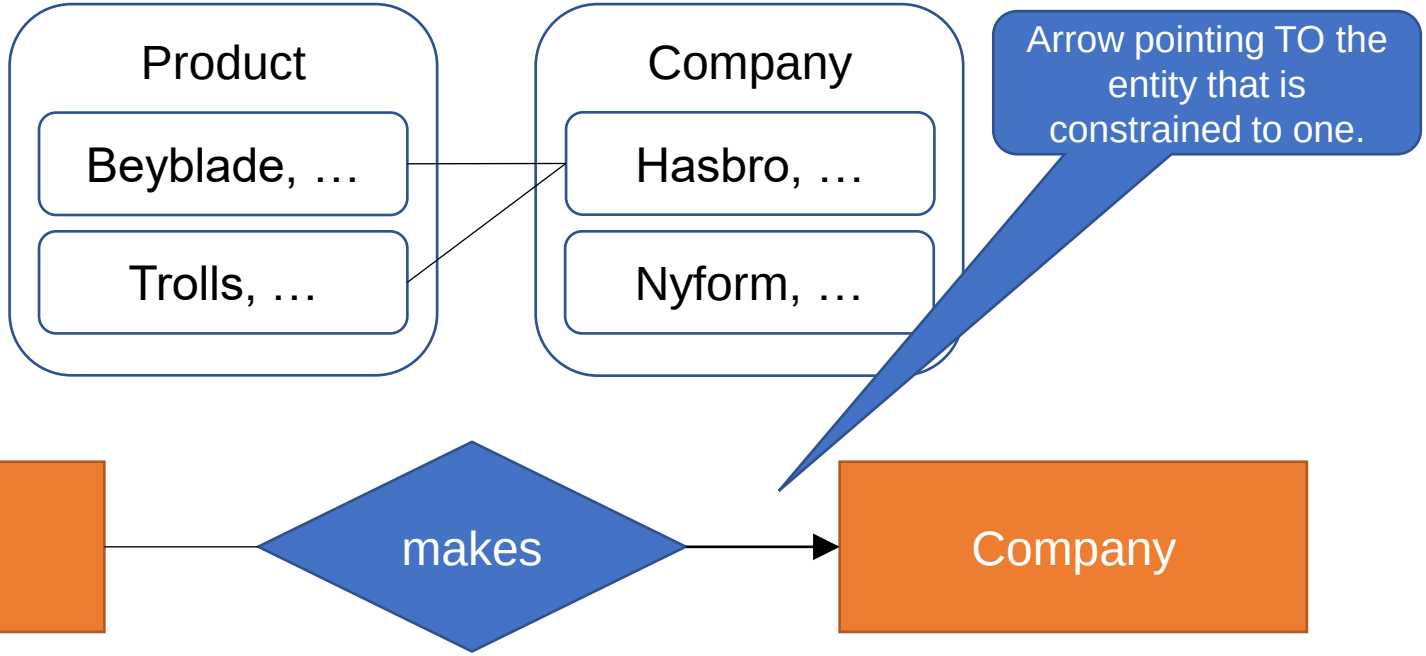




Relation Multiplicity

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



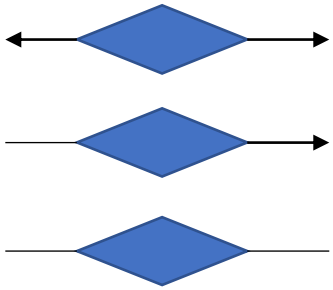


Announcements

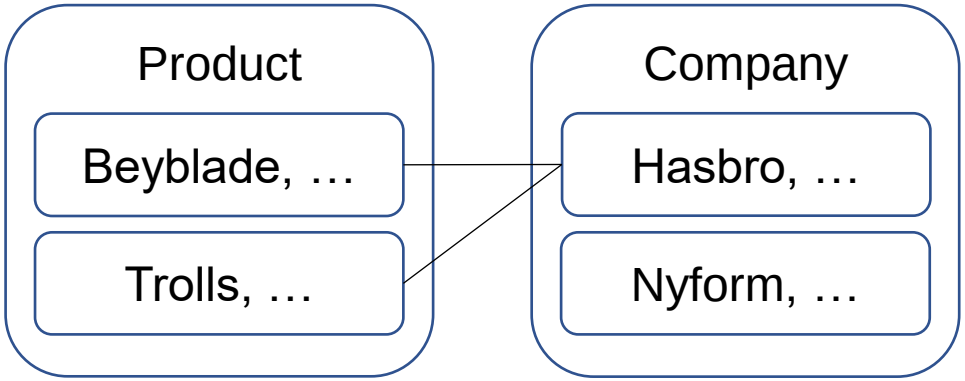
- Quiz 1+2 out on Friday 6am through Canvas.
- You have until Saturday 1pm to complete the quiz (over 24 hours because of the weekend time)
- You can resubmit your answers to the quiz, but your previous answers will not be saved if you enter a new submission
- Recommend you do the work outside of Canvas and paste your answers in

Relation Multiplicity

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

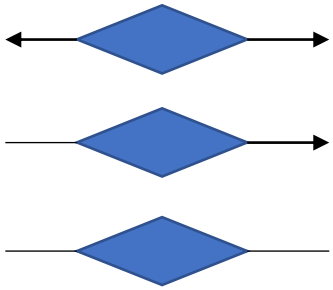


Do I need a Makes table?

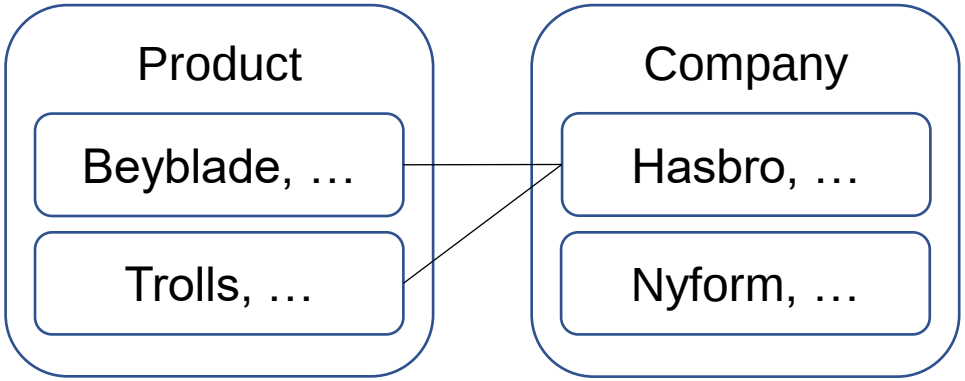


Relation Multiplicity

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

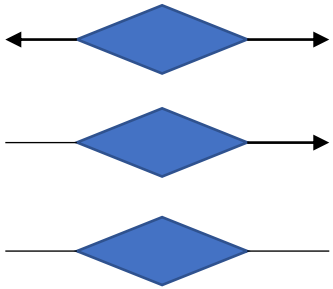


Do I need a Makes table? **No!**

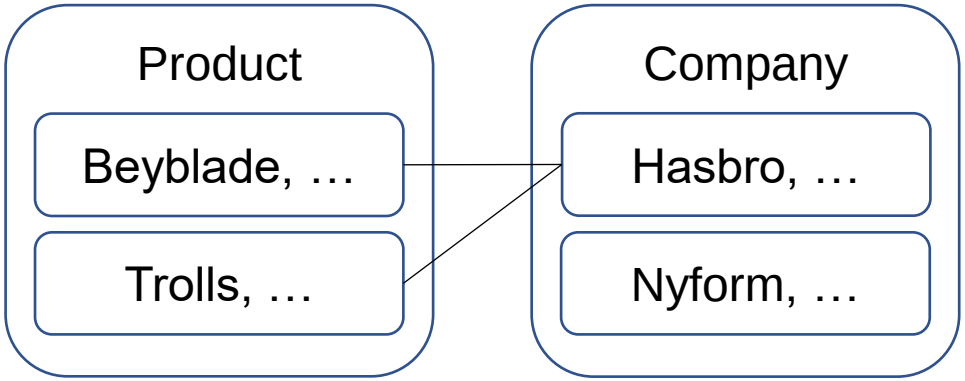


Relation Multiplicity

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

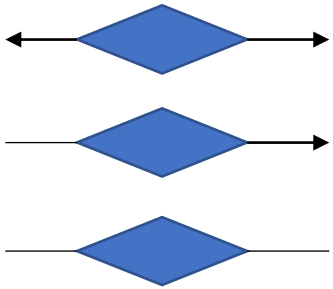


Do I need a Makes table? **No!** Key observation: Each product can be made by **only one** company, so a field in Product can represent ”makes”

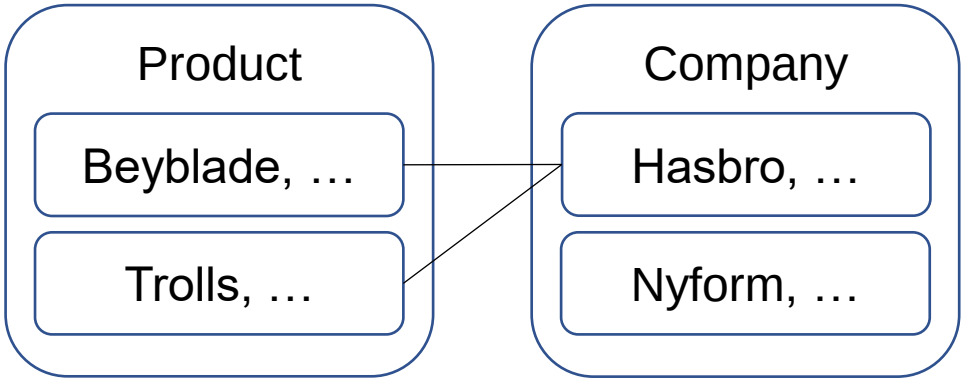


Relation Multiplicity

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



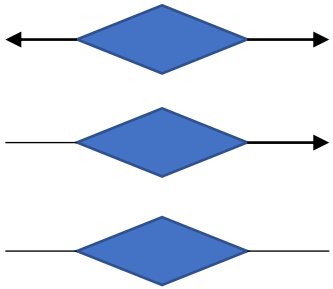
```
CREATE TABLE Company (
    name VARCHAR(100) PRIMARY KEY,
    ...);
CREATE TABLE Product (
    name VARCHAR(100) PRIMARY KEY,
    cname VARCHAR(100)
      REFERENCES Company
    ...);
```



We don't need a makes table, store Company.cname directly in Product, with a reference.

Relation Multiplicity

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

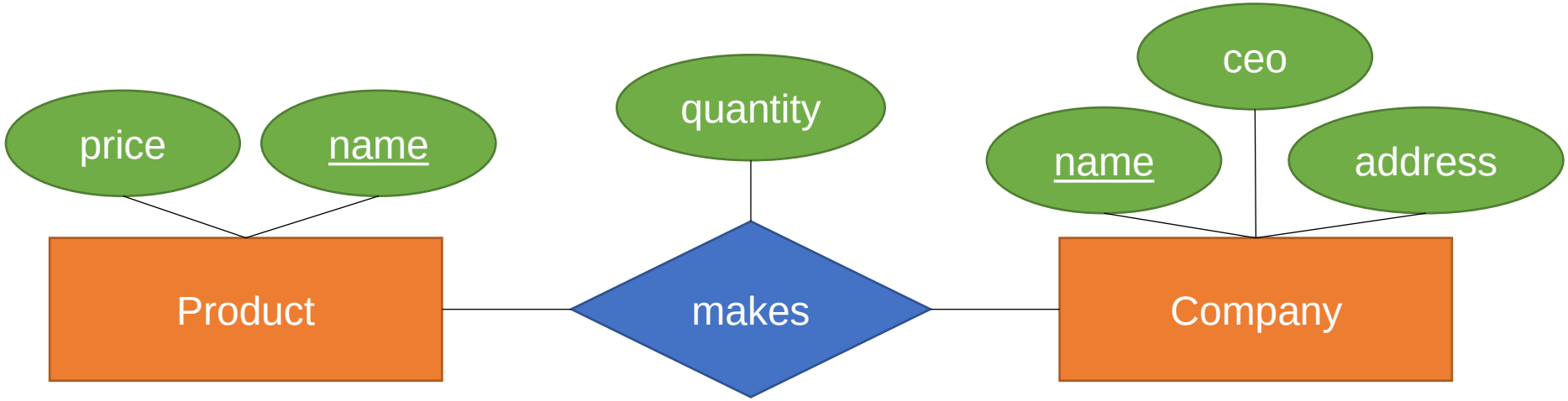


```
CREATE TABLE Company (
    name VARCHAR(100) PRIMARY KEY,
    ...);
CREATE TABLE Product (
    name VARCHAR(100) PRIMARY KEY,
    cname VARCHAR(100)
      REFERENCES Company
    );
```

You should reduce the number of tables when you can, in the Many-to-one and one-to-one case

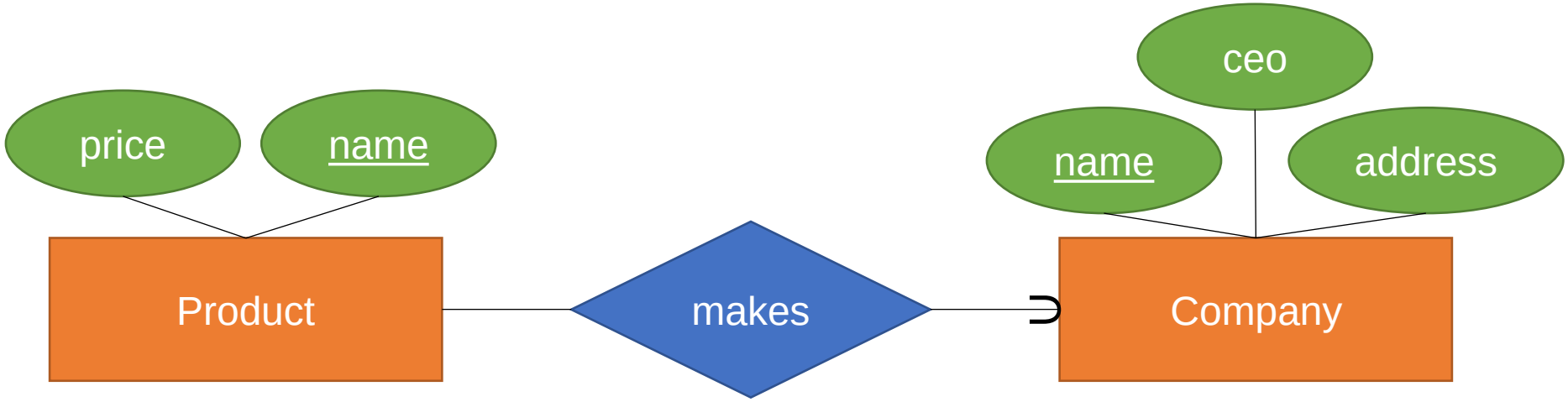
Relation Attributes

- Relations can have attributes too!



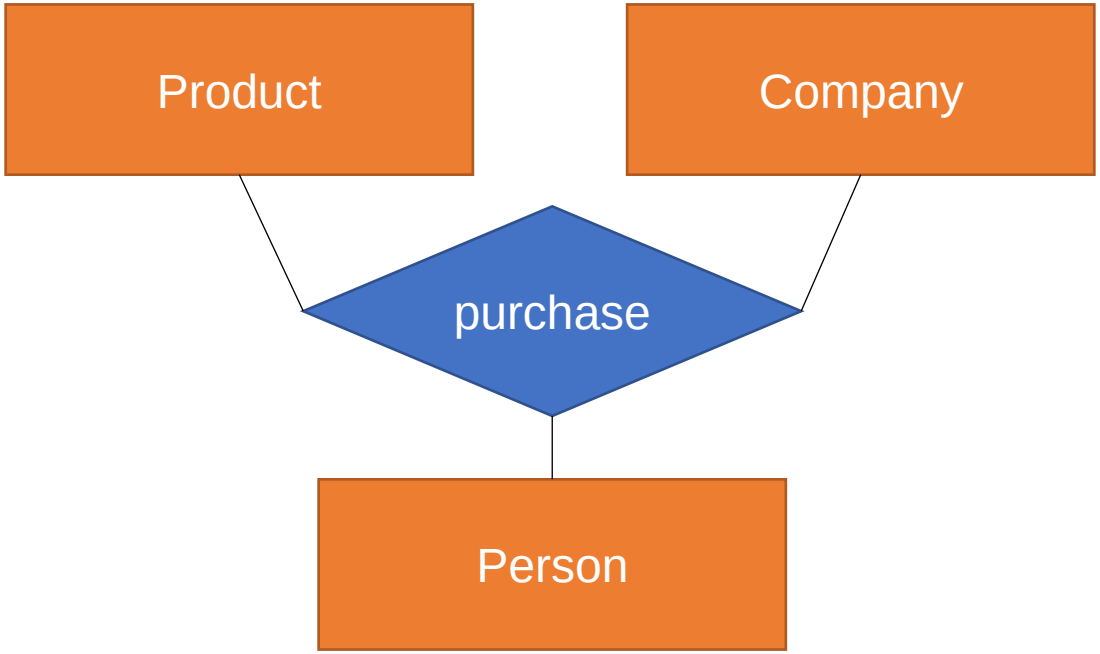
Exactly-One Reference

- Rounded arrow means the relationship is not optional (exactly one vs. at most one)

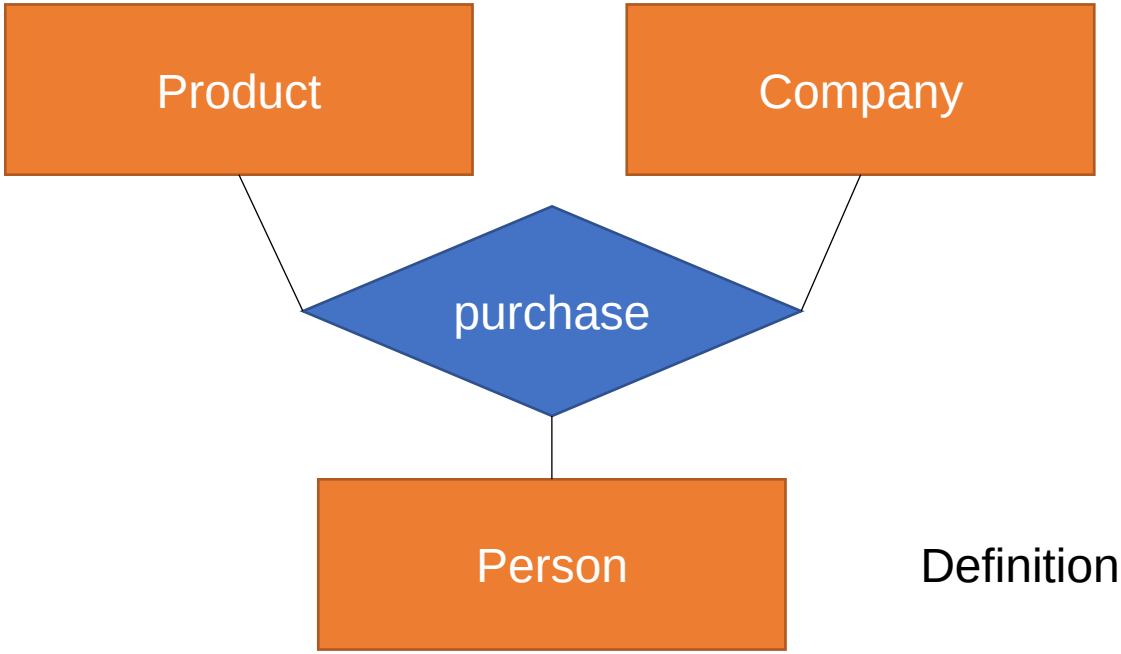



```
CREATE TABLE Company (
    name VARCHAR(100) PRIMARY KEY,
    ...);
CREATE TABLE Product (
    name VARCHAR(100) PRIMARY KEY,
    cname VARCHAR(100) NOT NULL
      REFERENCES Company
    ...);
```

Multi-Way Relations



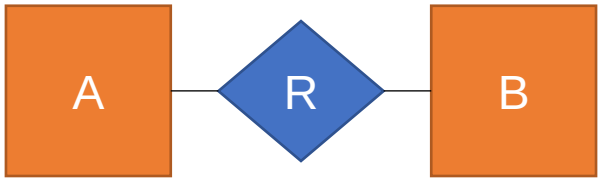
Multi-Way Relations



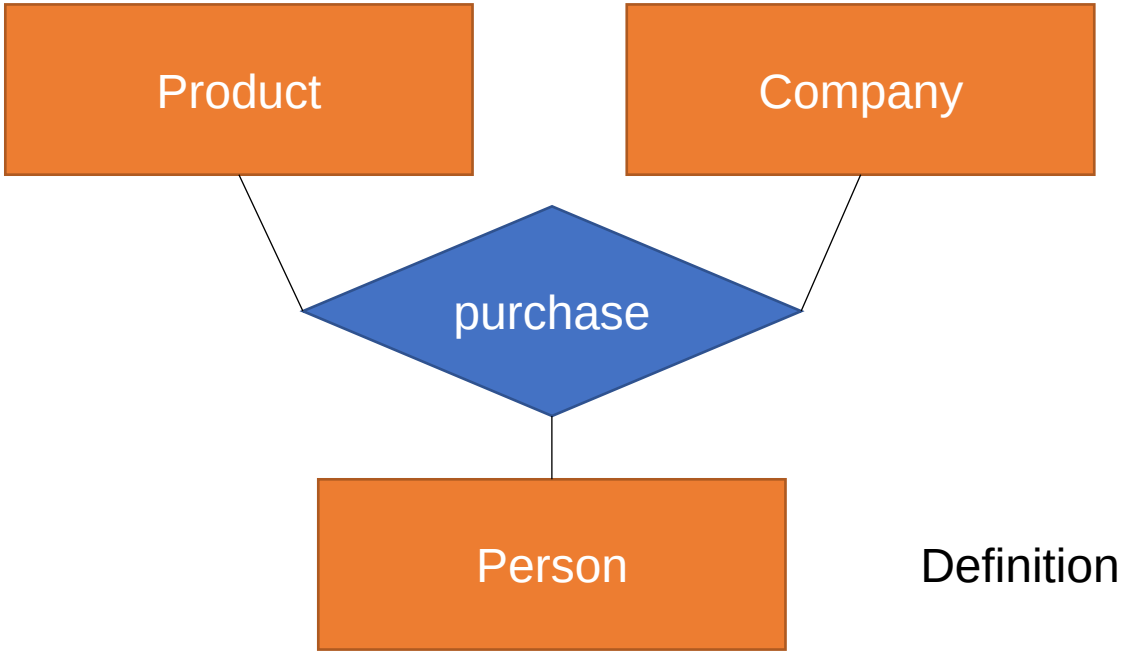
Definition of a relation generalizes!

Relationship

If A and B are sets, then a relation R is a subset of $A \times B$



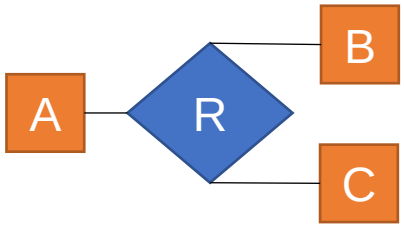
Multi-Way Relations



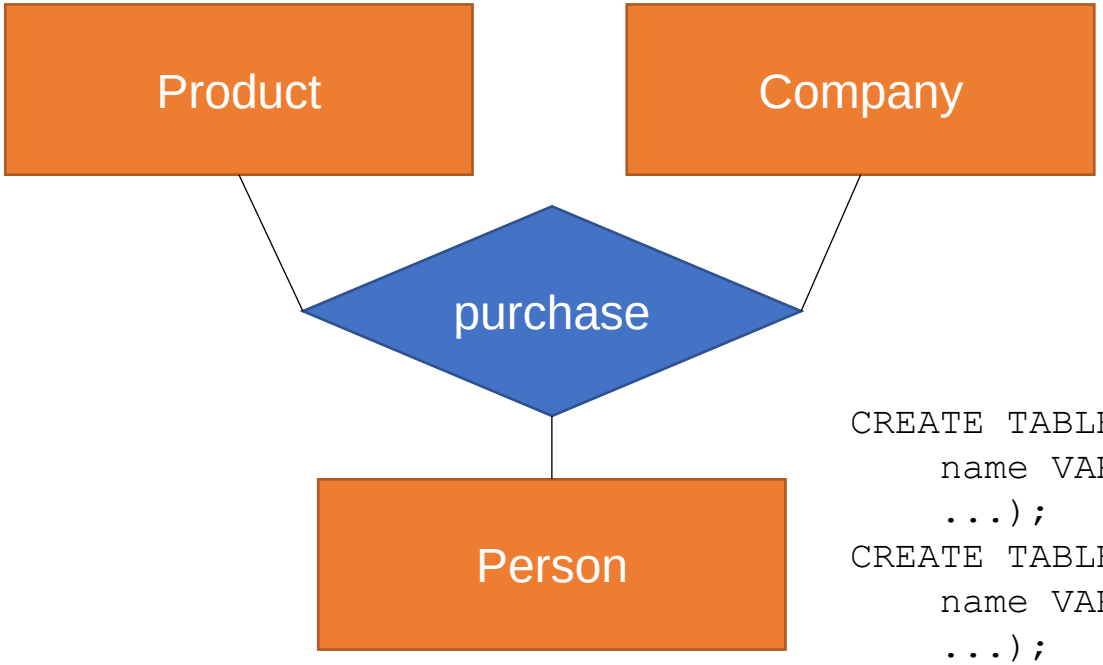
Definition of a relation generalizes!

Relationship

If A, B, and C are sets, then a relation R is a subset of $A \times B \times C$

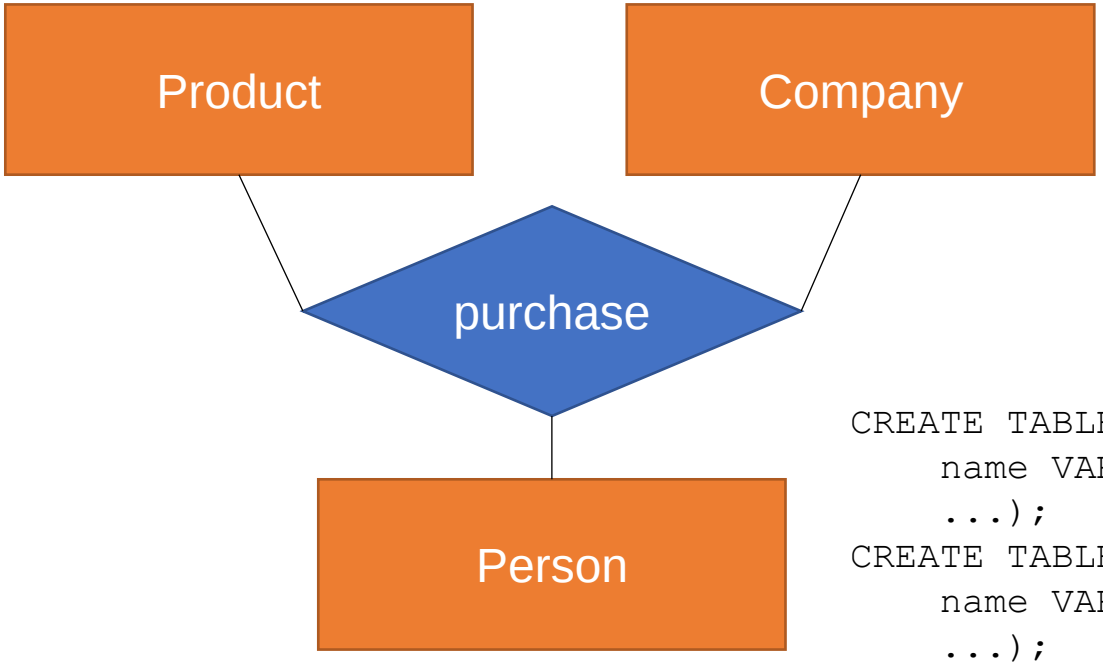


Multi-Way Relations




```
CREATE TABLE Product (
    name VARCHAR(100) PRIMARY KEY,
    ...);
CREATE TABLE Company (
    name VARCHAR(100) PRIMARY KEY,
    ...);
CREATE TABLE Person (
    ssn INT PRIMARY KEY,
    ...);
CREATE TABLE Purchase (
    cname VARCHAR(100) REFERENCES Company,
    pname VARCHAR(100) REFERENCES Product,
    ssn INT REFERENCES Person,
    PRIMARY KEY (cname, pname, ssn),
    ...);
```

Multi-Way Relations



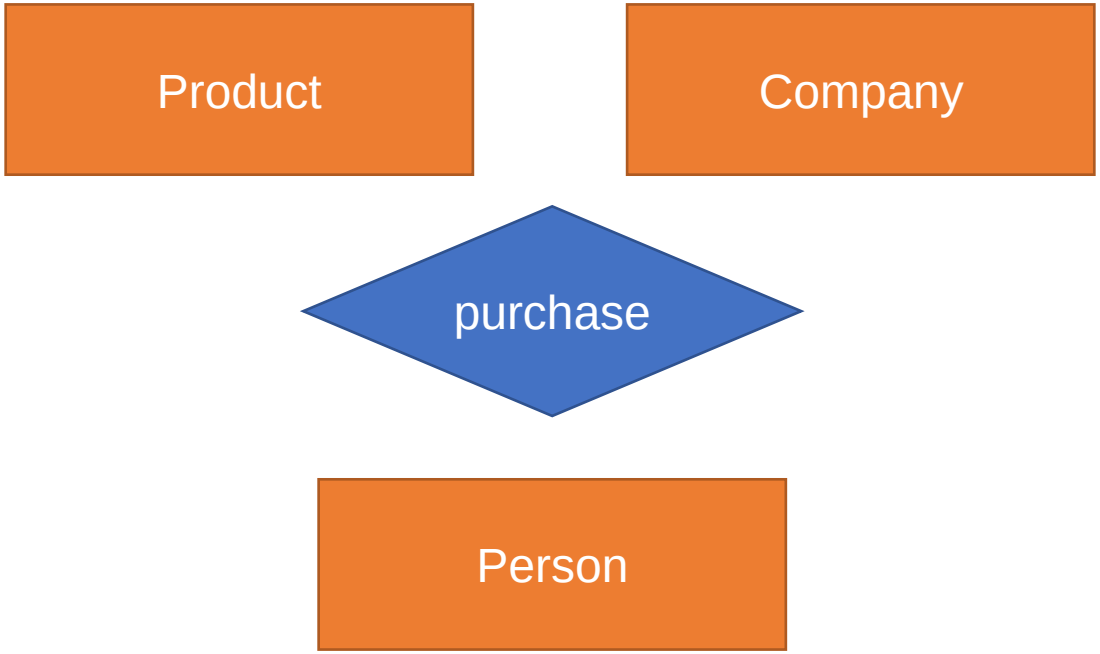

```
CREATE TABLE Product (
    name VARCHAR(100) PRIMARY KEY,
    ...);
CREATE TABLE Company (
    name VARCHAR(100) PRIMARY KEY,
    ...);
CREATE TABLE Person (
    ssn INT PRIMARY KEY,
    ...);
CREATE TABLE Purchase (
    cname VARCHAR(100) REFERENCES Company,
    ) REFERENCES Product,
    ES Person,
    me, pname, ssn),
```

Purchase

Person.id	Product.pname	Company.cname
Ryan	Soap	Dial
Mary	Cereal	Capn Crunch

It’s Your Turn!

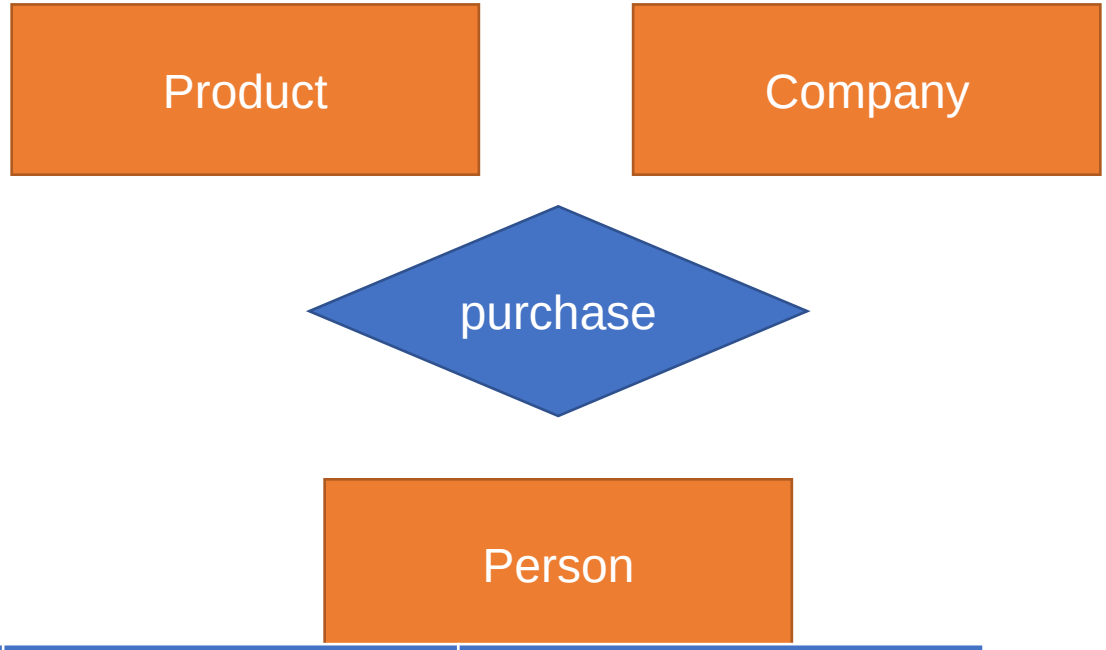
I want it to be true that each (person, product) pair comes from a single company.



How would you draw it?
Remember that the arrows read like an implication/function
Discuss!

It's Your Turn!

I want it to be true that each (person, product) pair comes from a single company.



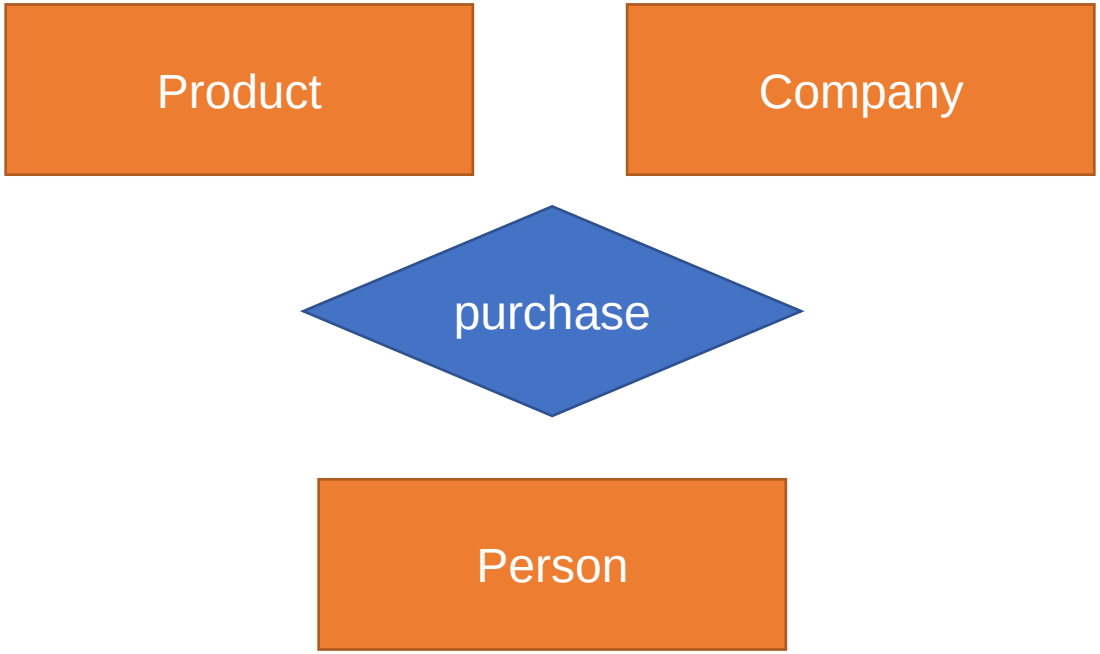
Purchase

Person.id	Product.pname	Company.cname
Ryan	Soap	Dial
Ryan	Soap	Dove
Mary	Soap	Dove

This row not allowed

It’s Your Turn!

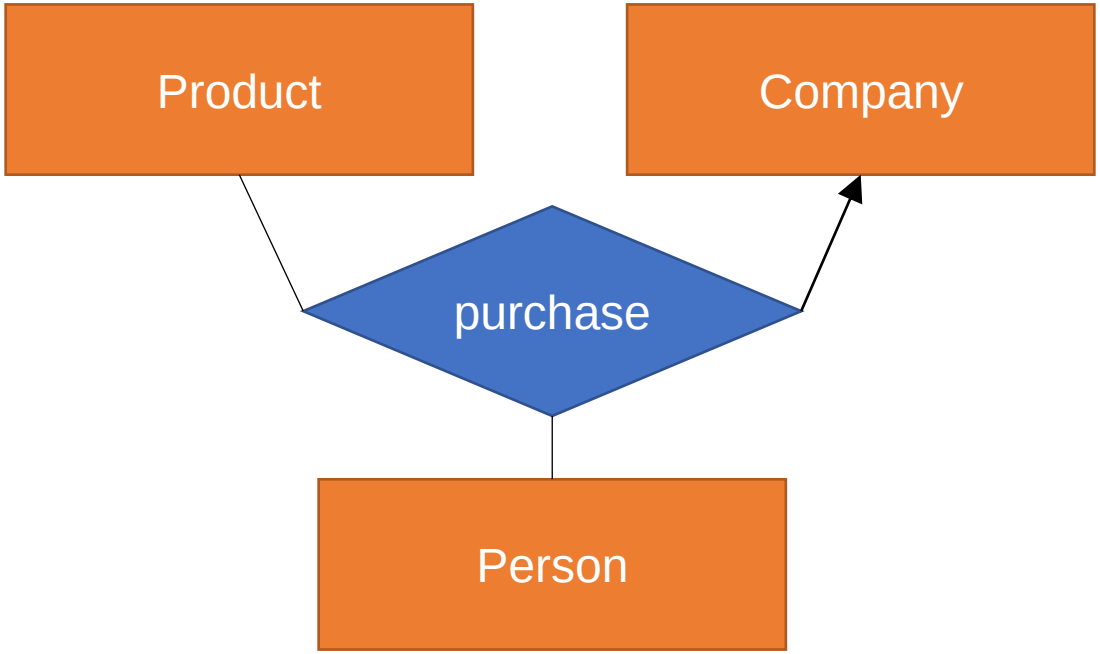
I want it to be true that each (person, product) pair comes from a single company.



How would you draw it?
Remember that the arrows read like an implication/function
Discuss!

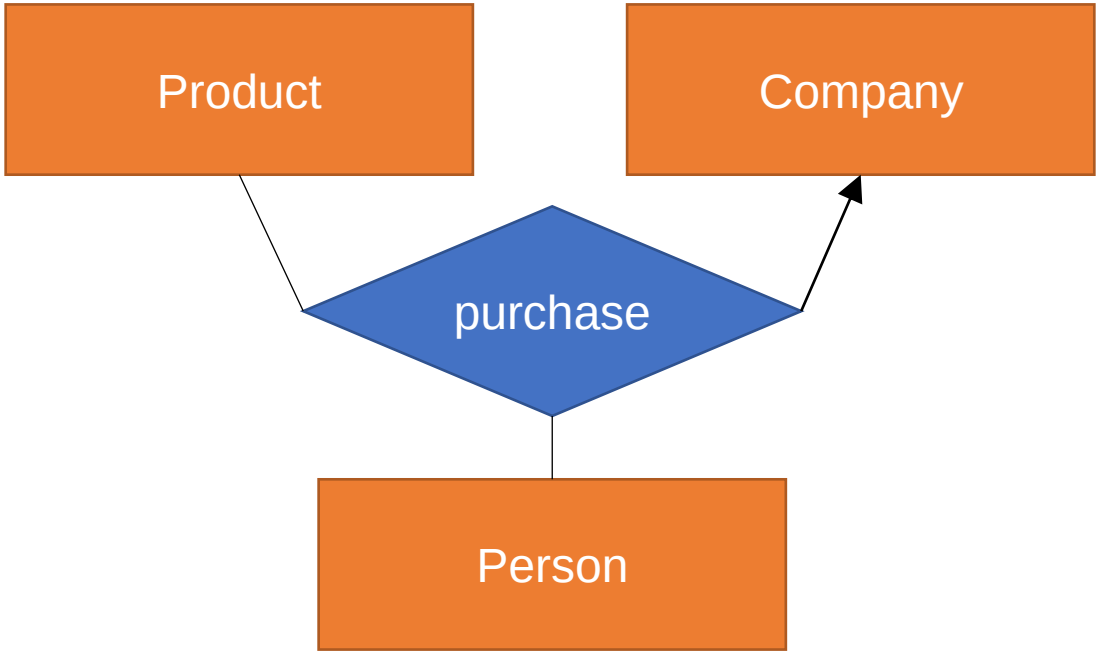
Multi-Way Relations

I want it to be true that each (person, product) pair comes from a single company.



Multi-Way Relations

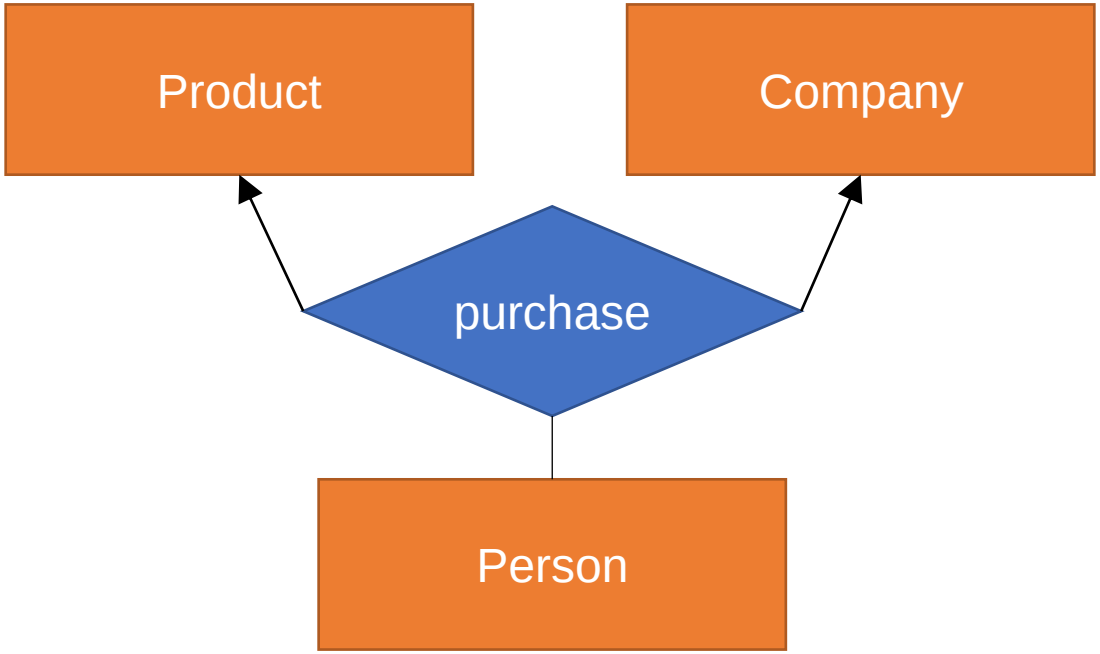
I want it to be true that each (person, product) pair comes from a single company.



Person	Product	Company
Ryan	Soap	Dial
Ryan	Soap	Dove

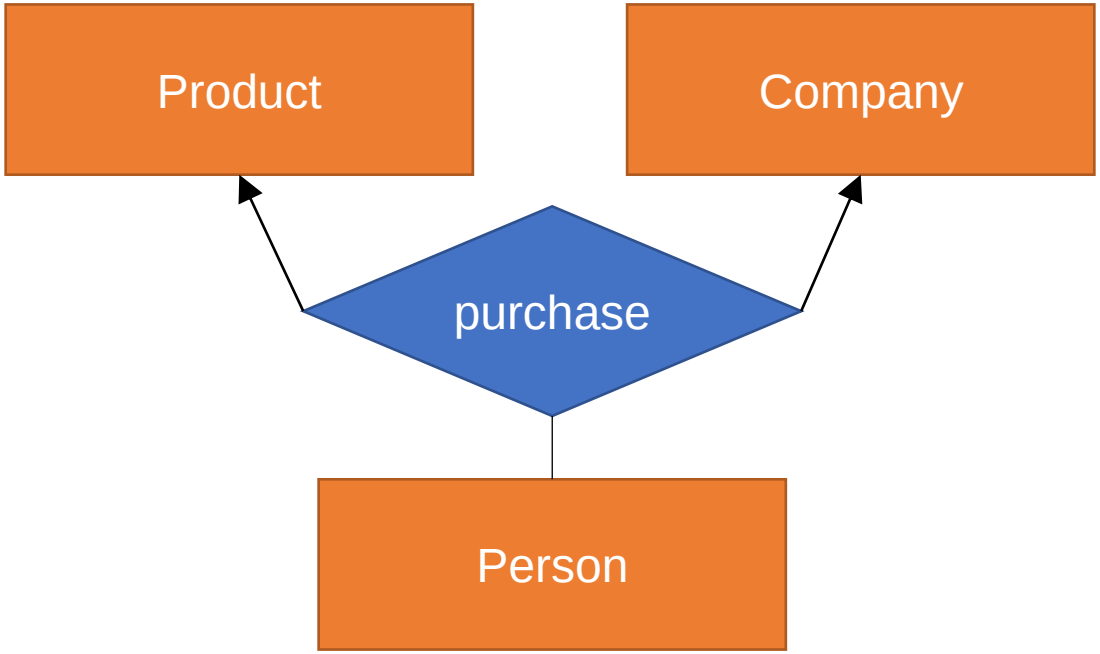
Multi-Way Relations

What does this mean?



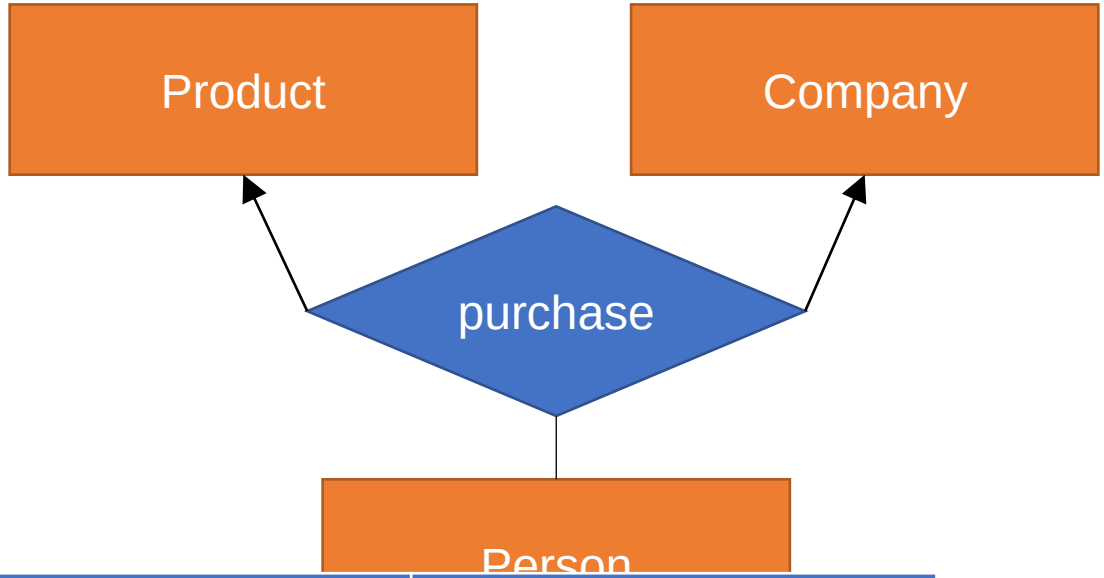
Multi-Way Relations

What does this mean?
Each (person, product) pair comes from one company
Each (person, company) pair comes from one product



Multi-Way Relations

What does this mean?
Each (person, product) pair comes from one company
Each (person, company) pair comes from one product

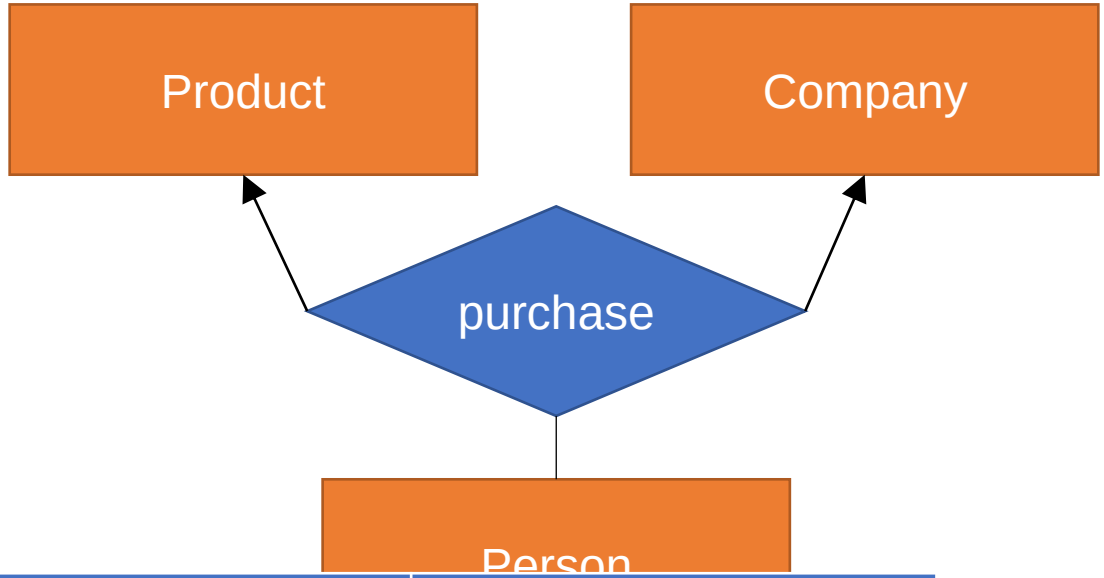


Person.id	Product.pname	Company.cname
Ryan	Soap	Dial
Mary	Soap	Dove
Mary	Shampoo	Dove

This row not allowed

Multi-Way Relations

What does this mean?
Each (person, product) pair comes from one company
Each (person, company) pair comes from one product

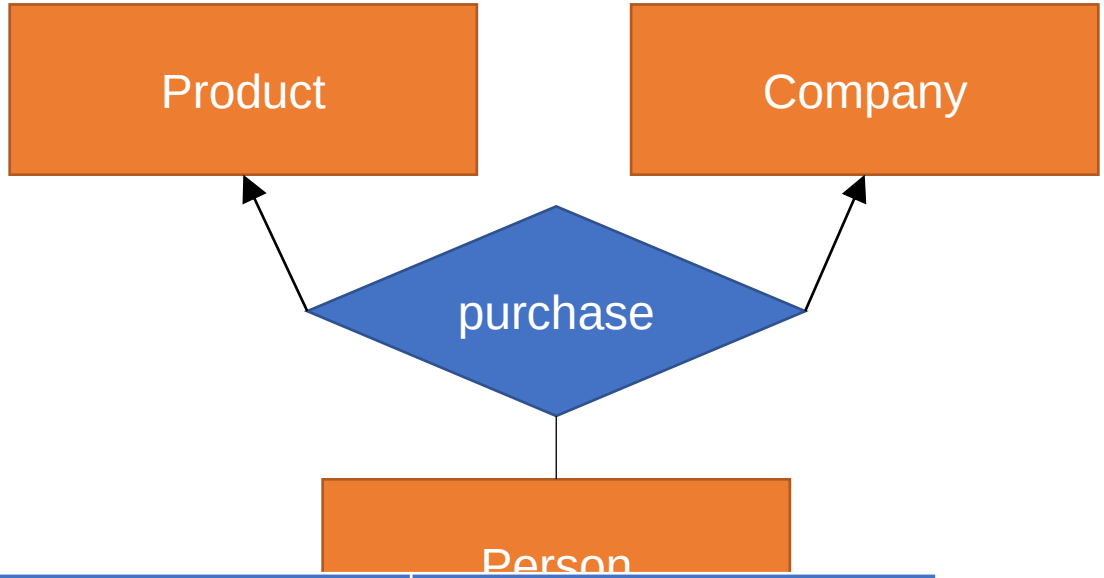


Person.id	Product.pname	Company.cname
Ryan	Soap	Dial
Mary	Soap	Dove
Ryan	Chips	Kettle

Allowed?

Multi-Way Relations

What does this mean?
Each (person, product) pair comes from one company
Each (person, company) pair comes from one product

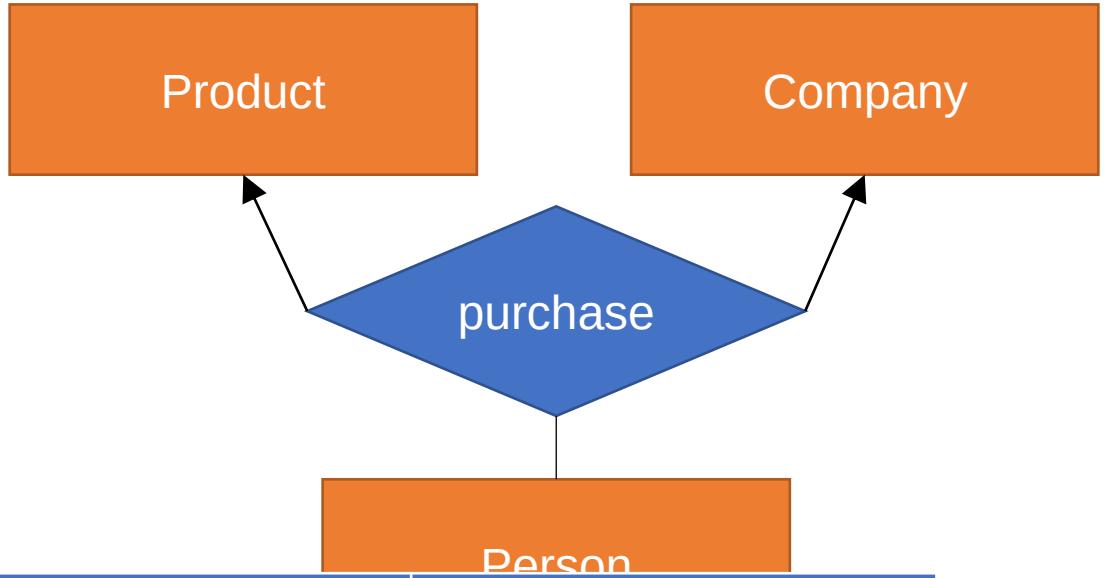


Person.id	Product.pname	Company.cname
Ryan	Soap	Dial
Mary	Soap	Dove
Ryan	Chips	Kettle

OK

Multi-Way Relations

What does this mean?
Each (person, product) pair comes from one company
Each (person, company) pair comes from one product

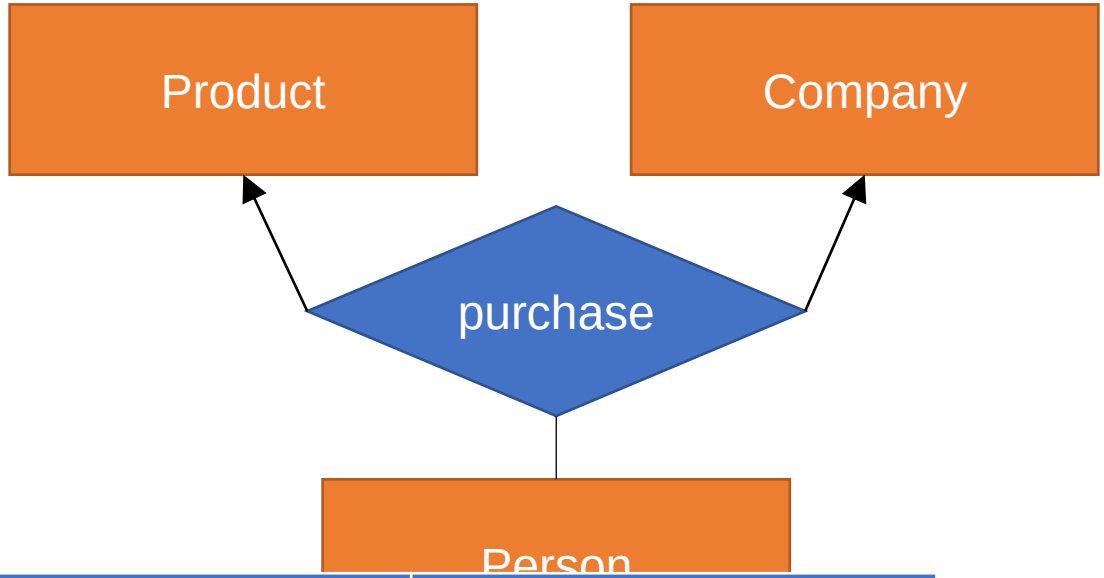


Person.id	Product.pname	Company.cname
Ryan	Soap	Dial
Mary	Soap	Dove
Ryan	Chips	Kettle
Mary	Chips	Kettle

Allowed?

Multi-Way Relations

What does this mean?
Each (person, product) pair comes from one company
Each (person, company) pair comes from one product



Person.id	Product.pname	Company.cname
Ryan	Soap	Dial
Mary	Soap	Dove
Ryan	Chips	Kettle
Mary	Chips	Kettle

OK

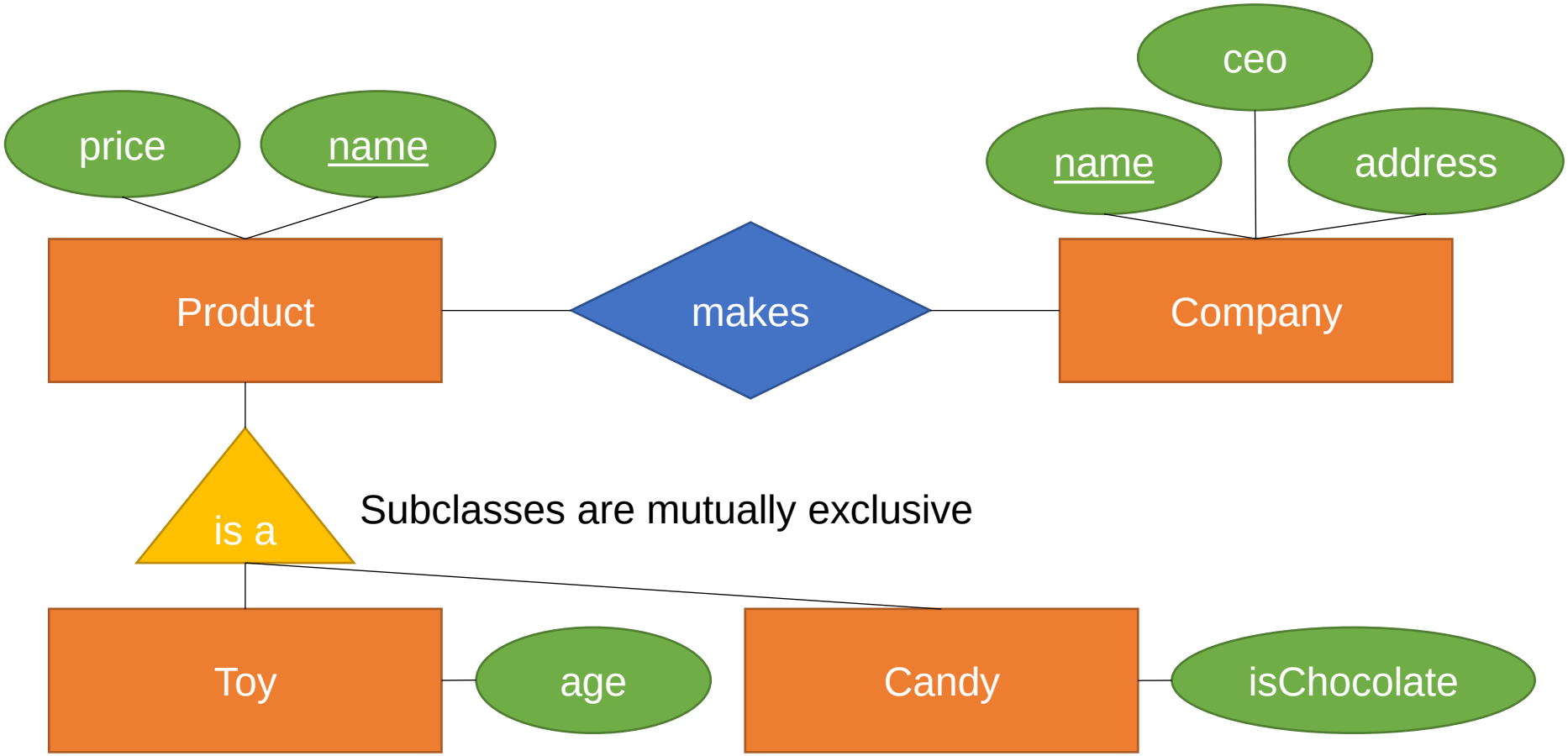
Rules of Thumb in Database Design

Design Principles (common sense):

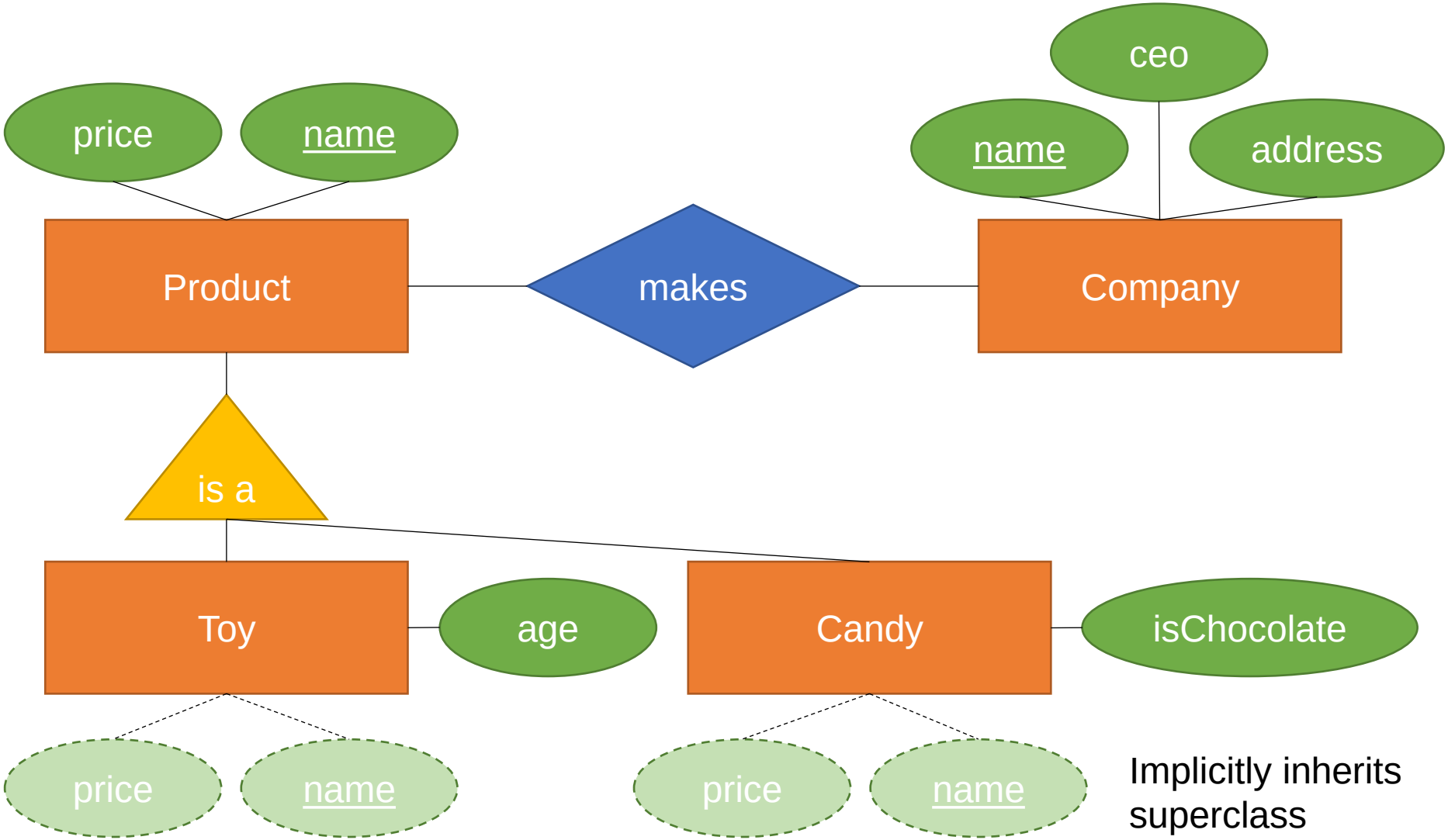
- Pick the right entities
- Don’t over complicate things
- Follow the application spec

Subclassing

- Distinguish special entities in an entity set
- Mimics heuristics in object oriented programming

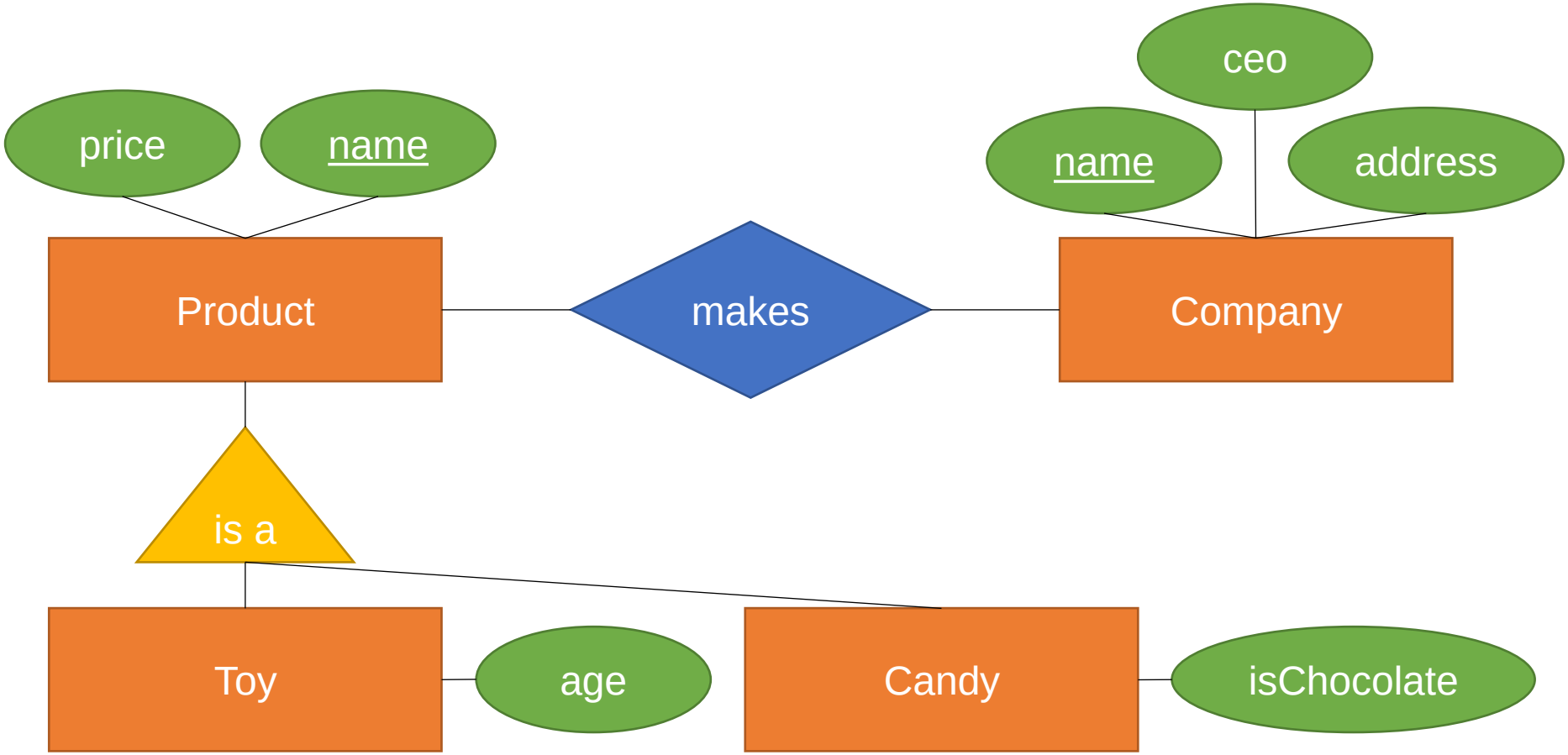


Subclassing



Implicitly inherits superclass attributes and key

Subclassing



Company(…)
Makes(…)
Product(price, <u>name</u>)

Toy(<u>name</u>, age)
Candy(<u>name</u>, isChocolate)

Announcements

- Quiz 1+2 out on Friday 6am through Canvas.
- You have until Saturday 1pm to complete the quiz (over 24 hours because of the weekend time)
- You can resubmit your answers to the quiz, but your previous answers will not be saved if you enter a new submission
- Recommend you do the work outside of Canvas and paste your answers in

Announcements

- HW 3 Due Tuesday at 11pm
- Quiz 1 will be graded by Monday
- Section worksheet from yesterday on website, great example of building an E/R diagram

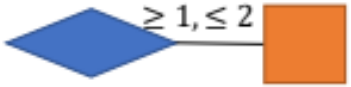
Relationships

- **one-one**: ssn - UW student id
- **one-many**: ssn - phone#
- **many-many**: store - product
- **is-a**: computer - PC and computer - Mac
- **has-a**: country - city
  - What country does the city of Cambridge belong to?

"at most one" 

"exactly one" 

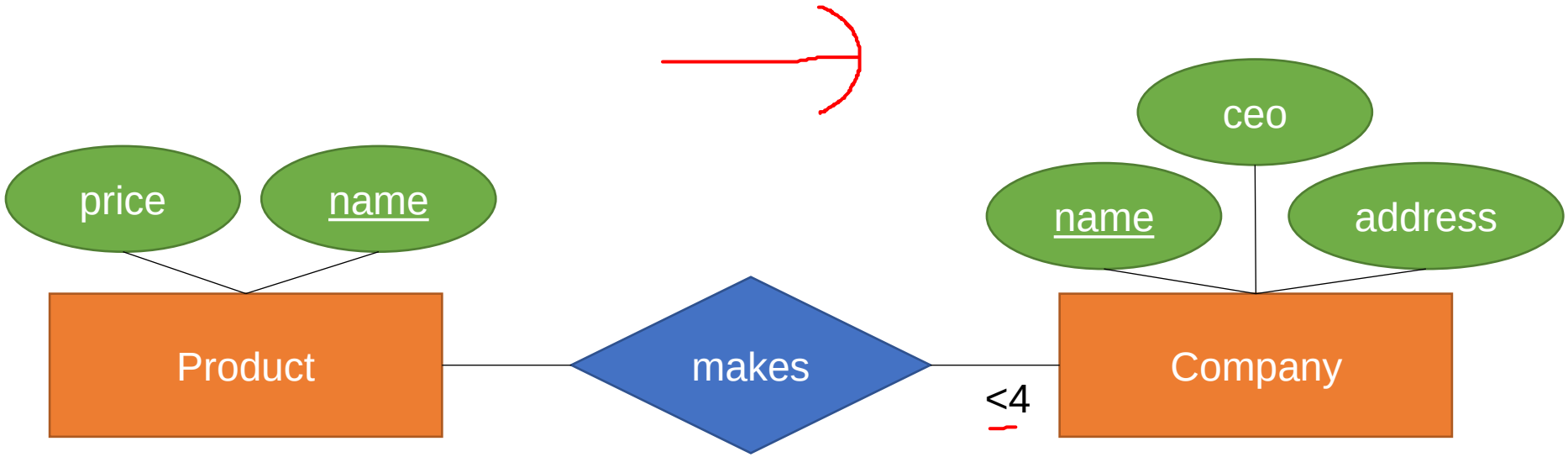
no constraint 

other constraint 

"subclass" 

Misc Constraints

- Normal arrows are shorthand versions of (<=1)
  - “at most one”
- Rounded arrows are shorthand versions of (=1)
  - “exactly one” - often requires NOT NULL in create table



Each product can be made by, at most, 3 companies

Other Constraints

- `CHECK (condition)`
  - Single attribute range
  - Single tuples
  - Can combine checks at end of create table statement

```
CREATE TABLE User (
    uid INT PRIMARY KEY,
    firstName TEXT,
    lastName TEXT,
    age INT CHECK (age > 12 AND age < 120),
    email TEXT,
    phone TEXT,
    CHECK (email IS NOT NULL OR phone IS NOT NULL)
);
```

Referential Constraint Maintenance

`ON UPDATE/ON DELETE`

- `NO ACTION` → (default) error out
- `CASCADE` → update/delete referencers
- `SET NULL` → set referencers' field to NULL
- `SET DEFAULT` → set referencers' field to default
  - Assumes default was set, e.g.

```
CREATE TABLE Table (
   id INT DEFAULT 42 REFERENCES OtherTable,
   ...
);
```

Referential Constraint Maintenance

```
CREATE TABLE Company (
    name VARCHAR(100) PRIMARY KEY);
CREATE TABLE Product (
    name VARCHAR(100) PRIMARY KEY,
    cname VARCHAR(100)
      REFERENCES Company
      ON UPDATE CASCADE
      ON DELETE SET NULL);
```

Company

name
Hasbro
Nyform

Product

name	cname
Beyblade	Hasbro
Troll	Hasbro

Referential Constraint Maintenance

```
CREATE TABLE Company (
    name VARCHAR(100) PRIMARY KEY);
CREATE TABLE Product (
    name VARCHAR(100) PRIMARY KEY,
    cname VARCHAR(100)
      REFERENCES Company
      ON UPDATE CASCADE
      ON DELETE SET NULL);
```

Company

name
Hasbro
Nyform

Product

name	cname
Beyblade	Hasbro
Troll	Hasbro

```
UPDATE Company
   SET name = 'lmao'
 WHERE name = 'Hasbro';
```

Referential Constraint Maintenance

```
CREATE TABLE Company (
    name VARCHAR(100) PRIMARY KEY);
CREATE TABLE Product (
    name VARCHAR(100) PRIMARY KEY,
    cname VARCHAR(100)
      REFERENCES Company
      ON UPDATE CASCADE
      ON DELETE SET NULL);
```

Company

name
lmao
Nyform

Product

name	cname
Beyblade	lmao
Troll	lmao

```
UPDATE Company
   SET name = 'lmao'
 WHERE name = 'Hasbro';
```

Referential Constraint Maintenance

```
CREATE TABLE Company (
    name VARCHAR(100) PRIMARY KEY);
CREATE TABLE Product (
    name VARCHAR(100) PRIMARY KEY,
    cname VARCHAR(100)
      REFERENCES Company
      ON UPDATE CASCADE
      ON DELETE SET NULL);
```

Company

name
~~lmao~~
Nyform

Product

name	cname
Beyblade	lmao
Troll	lmao

```
DELETE FROM Company
 WHERE name = 'lmao';
```

Referential Constraint Maintenance

```
CREATE TABLE Company (
    name VARCHAR(100) PRIMARY KEY);
CREATE TABLE Product (
    name VARCHAR(100) PRIMARY KEY,
    cname VARCHAR(100)
      REFERENCES Company
      ON UPDATE CASCADE
      ON DELETE SET NULL);
```

Company

name
Nyform

Product

name	cname
Beyblade	NULL
Troll	NULL

```
DELETE FROM Company
 WHERE name = 'lmao';
```

Assertions

- Hard to support
- Usually impractical
- Usually not supported
  - Simulated with triggers

```
CREATE ASSERTION myAssert CHECK
    (NOT EXISTS (
        SELECT Product.name
          FROM Product, Purchase
         WHERE Product.name = Purchase.prodName
         GROUP BY Product.name
        HAVING count(*) > 200));
```

Triggers

- Triggers activate on a specified event

```
CREATE TRIGGER LowCredit ON Purchasing.PurchaseOrderHeader
AFTER INSERT AS
  IF (ROWCOUNT_BIG() = 0) RETURN;
  IF EXISTS (SELECT *
             FROM Purchasing.PurchaseOrderHeader AS p
             JOIN inserted AS i
             ON p.PurchaseOrderID = i.PurchaseOrderID
             JOIN Purchasing.Vendor AS v
             ON v.BusinessEntityID = p.VendorID
             WHERE v.CreditRating = 5
            )
    BEGIN
      RAISERROR ('A vendor''s credit rating is too
                  low to accept new purchase orders.', 16, 1);
      ROLLBACK TRANSACTION;
      RETURN
    END;
GO
```

= you don't need to study this for the class

Takeaways

- ER diagrams can sketch out **high-level designs**
- Certain rules of thumb for ER-to-SQL conversions help **preserve design semantics**
- SQL allows you to make **rules specific to your application**