CMPSC 431W
Database Management Systems

Conceptual Database Design

ER Model

Slides Credit: Jiannan Wang@SFU, Dan Suciu@UW

Review

- Database – Collection of data to support applications.
- Relational Model – Common data model used by DBMS
- Relational Algebra/Calculus – Formal query languages
- SQL – Concrete Way (DSL) to talk with DBMS

- Given an application, how do we design the database to support it?

Motivation

- How to figure out this **database design**?
  - Customer = {customerID, firstName, lastName, income, birthDate}
  - Account = {accNumber, type, balance, branchNumber$^{\text{FK-Branch}}$}
  - Owns = {customerID$^{\text{FK-Customer}}$, accNumber$^{\text{FK-Account}}$}
  - Transaction = {transNumber, accNumber$^{\text{FK-Account}}$, amount}
  - Employee = {ssn, firstName, lastName, salary, branchNumber$^{\text{FK-Branch}}$}
  - Branch = {branchNumber, branchName, managerSSN$^{\text{FK-Employee}}$, budget}
- What **tables** to create?
- Which **attributes** should be added to each table?
- What are the **relationships** between tables?
- What **constraints** that tables have to follow?

Database Design

- Why do we need it? Why we need a good one?
  - Agree on **structure of the database** before deciding on a particular implementation.
  - Databases may be in operation for years. Updating structures of the data (schema) in production is **very expensive**.
- Consider issues such as:
  - What entities to model
  - How entities are related
  - What constraints exist in the domain
- Several formalisms exist: **ER diagrams**, UML, etc.

Database Design Process

1. Requirement Analysis
2. Conceptual Database Design  **This is where E/R Model fits in**
3. Logical Database Design
4. Schema Refinement
5. Physical Database Design
6. Application and Security Design

Requirement Analysis

- What data is going to be store?
- What are we going to do with the data?
- Who should access the data?

- Involves both technical and non-technical people.
- Usually very sloppy.
- Gap in understanding: **super tricky**

Conceptual Database Design

- Design a **high-level description** of the database.
- Sufficiently **precise** that technical people can understand it
- But, *not so precise that non-technical people can't participate*

- Should enable a **straightforward translation** into a data model supported by DBMS (e.g., relational model).
- E/R Model fits in here.

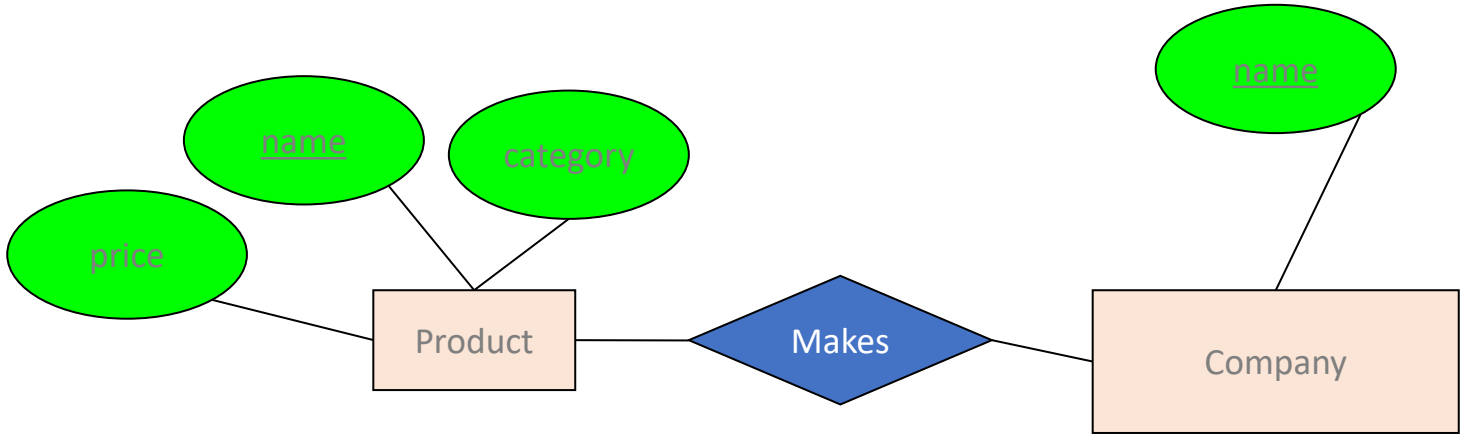
Going Beyond

- Logical Database Design
  - Converting conceptual design into **database schema**. (DBMS-specific)
- Schema Refinement
  - Identify **potential problems** and refine DB schema. (DB design theory)
- Physical Database Design
  - Ensure meeting **performance criteria**.
- Application and Security Design
  - Reflect back to **application** itself and **accessibility**.

ER Diagram

- Used from **requirement analysis** to **logical database design**.
- A *visual syntax* for DB design which is **precise enough** for technical points, but **abstracted enough** for non-technical people

- ER -> **Entity** and **Relationship**



Entities and Entity Sets

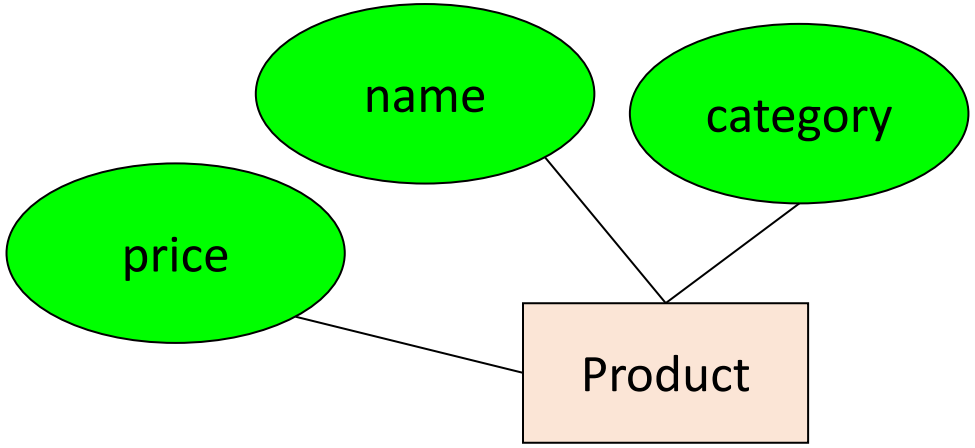
- An entity is an individual object
  - e.g., a specific person or product

- An entity set is a collection of “similar” entities
  - Not essentially disjoint to each other.
  - Shown as **rectangles** in ER diagrams
  - Person, Product

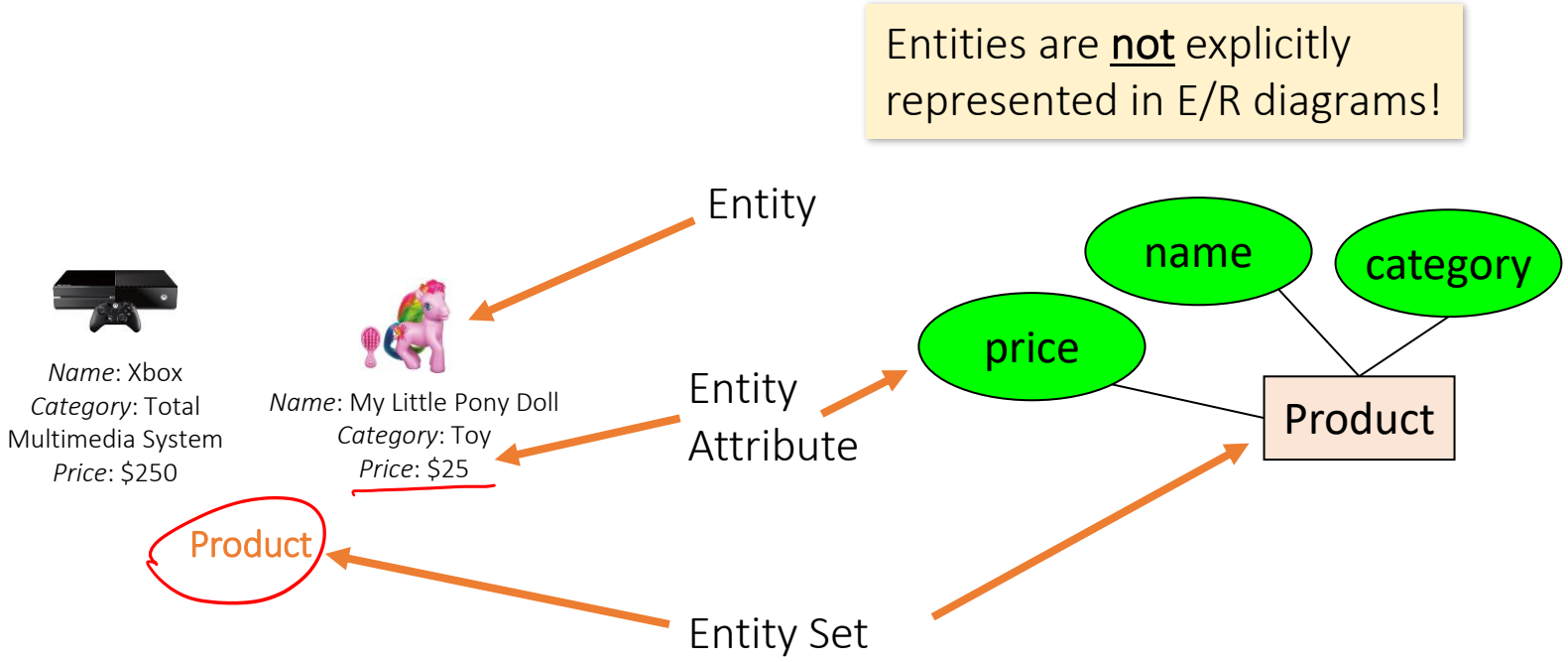
Attributes

- How do we define “similar” for entity sets.
- Similar entities has **the same set of attributes**.

- Represented by **ovals attached to an entity set** in ER diagram



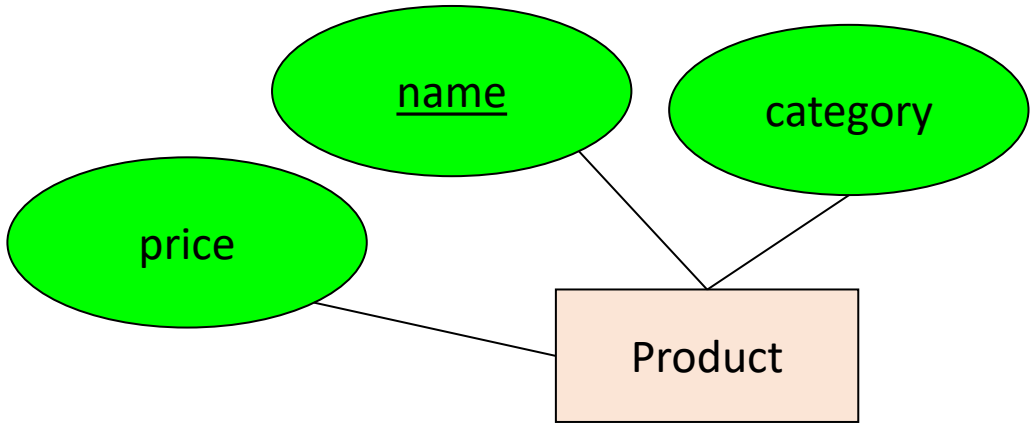
Example


Entities are not explicitly represented in E/R diagrams!
Entity
name
category
price
Product
Name: Xbox
Category: Total Multimedia System
Price: $250
Name: My Little Pony Doll
Category: Toy
Price: $25
Entity Attribute
Product
Entity Set

Keys

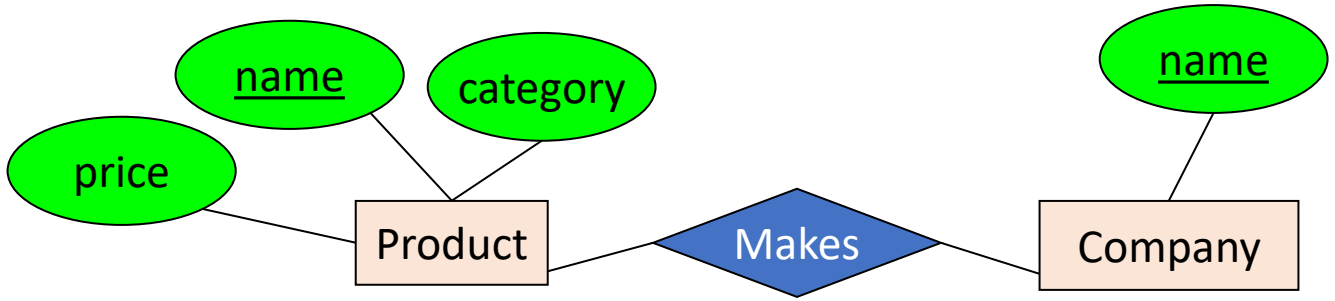
- A *key* is a set of attributes that uniquely identifies an entity.
- Every entity set must have a key

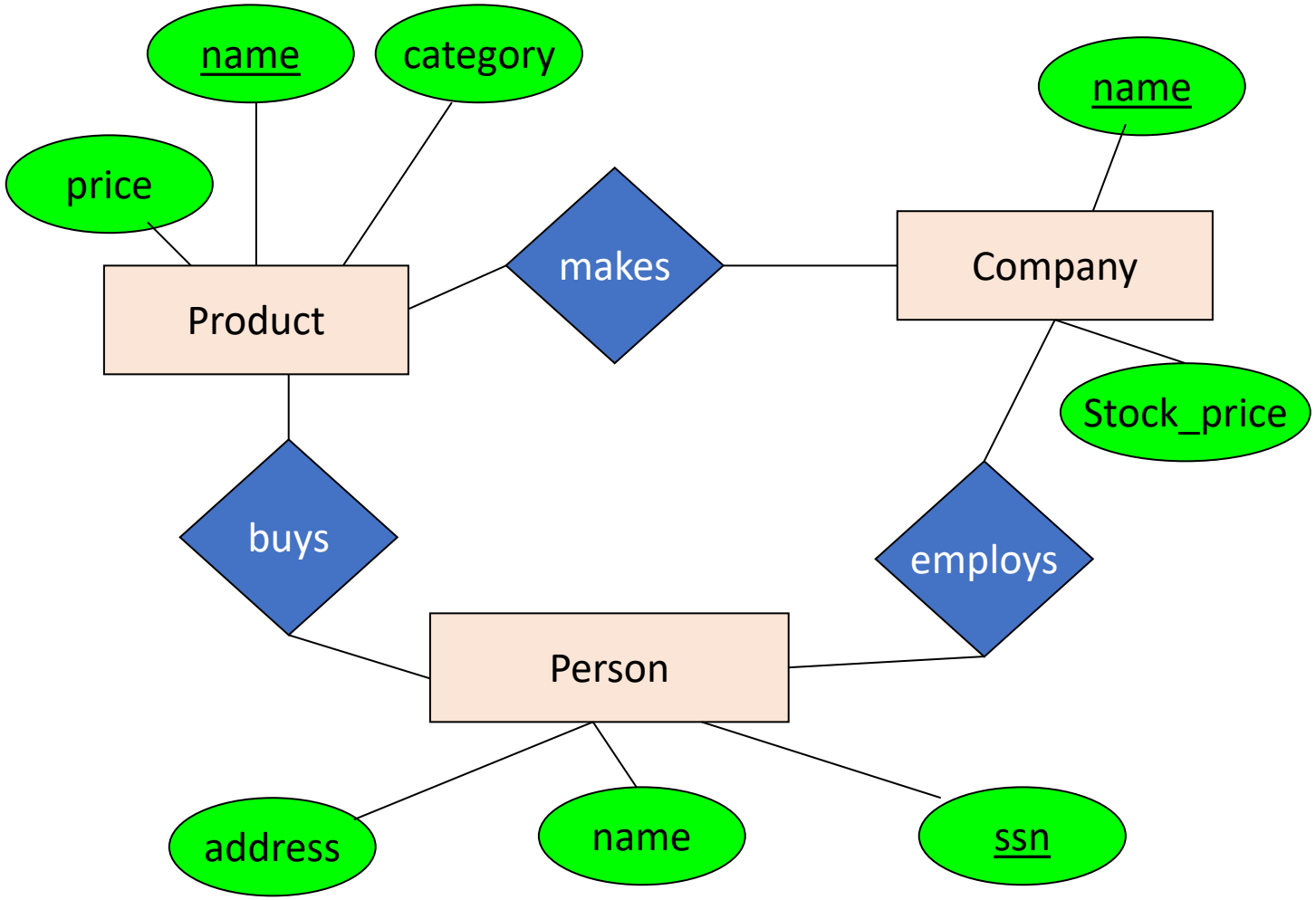
- Denote elements of the **primary key** by underlining.



The R in E/R: **Relationships**

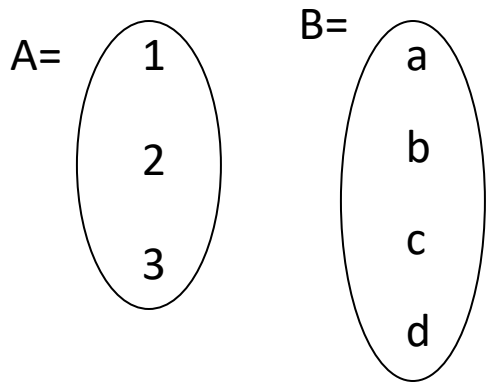
- A **relationship** is between two entities




name
category
price
Product
makes
Company
name
Stock_price
buys
employs
Person
address
name
ssn

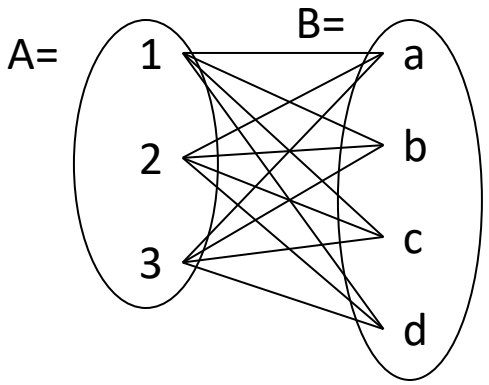
What is a Relationship?

- ***A mathematical definition:***
  - Let A, B be sets
    - *A={1,2,3}, B={a,b,c,d}*



What is a Relationship?

- ***A mathematical definition:***

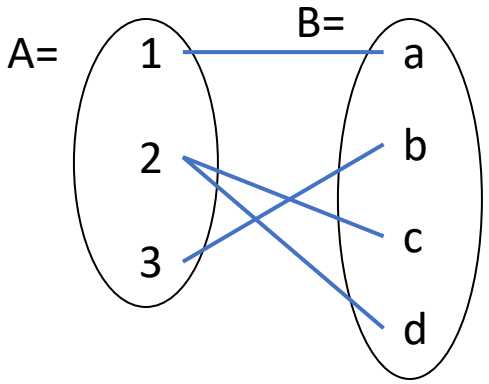


  - Let A, B be sets
    - *A={1,2,3}, B={a,b,c,d}*

  - A x B (the ***cross-product***) is the set of all pairs (a,b)
    - $A \times B = \{(1,a), (1,b), (1,c), (1,d), (2,a), (2,b), (2,c), (2,d), (3,a), (3,b), (3,c), (3,d)\}$

What is a Relationship?

- ***A mathematical definition:***



  - Let A, B be sets
    - *A={1,2,3}, B={a,b,c,d},*
  - A x B (the ***cross-product***) is the set of all pairs (a,b)
    - $A \times B = \{(1,a), (1,b), (1,c), (1,d), (2,a), (2,b), (2,c), (2,d), (3,a), (3,b), (3,c), (3,d)\}$
  - **We define a <u>relationship</u> to be a subset of A x B**
    - $R = \{(1,a), (2,c), (2,d), (3,b)\}$

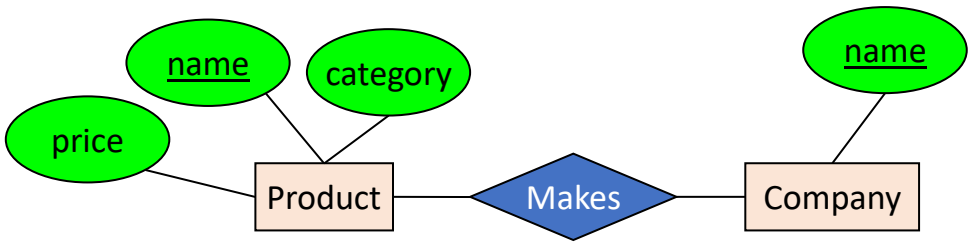
What is a Relationship?

Company

name
Apple
Microsoft

Product

name	category	price
iPhone 12	Electronics	$700
iPad Pro	Electronics	$300
Office	Software	$120



A relationship between **entity sets P and C** is a ***subset of all possible pairs of entities in P and C,*** with tuples uniquely identified by ***P and C's keys***

What is a Relationship?

Company

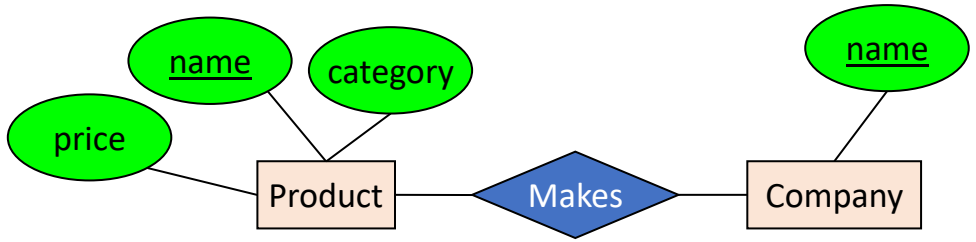
name
Apple
Microsoft

Product

name	category	price
iPhone 12	Electronics	$700
iPad Pro	Electronics	$300
Office	Software	$120

Company C × Product P

C.name	P.name	P.category	P.price
Apple	iPhone 12	Electronics	$700
Apple	iPad Pro	Electronics	$300
Apple	Office	Software	$120
Microsoft	iPhone 12	Electronics	$700
Microsoft	iPad Pro	Electronics	$300
Microsoft	Office	Software	$120



A relationship between **entity sets P and C** is a ***subset of all possible pairs of entities in P and C,*** with tuples uniquely identified by ***P and C's keys***

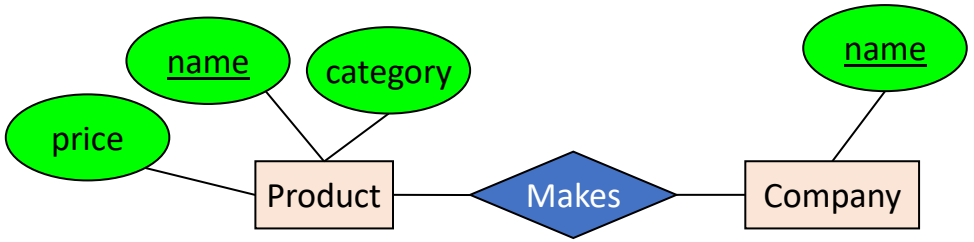
What is a Relationship?

Company

name
Apple
Microsoft

Product

name	category	price
iPhone 12	Electronics	$700
iPad Pro	Electronics	$300
Office	Software	$120



Company C × Product P

C.name	P.name	P.category	P.price
Apple	iPhone 12	Electronics	$700
Apple	iPad Pro	Electronics	$300
Apple	Office	Software	$120
Microsoft	iPhone 12	Electronics	$700
Microsoft	iPad Pro	Electronics	$300
Microsoft	Office	Software	$120

Makes

C.name	P.name
Apple	iPhone 12
Apple	iPad Pro
Microsoft	Office

A relationship between **entity sets P and C** is a ***subset of all possible pairs of entities in P and C,*** with tuples uniquely identified by ***P and C's keys***

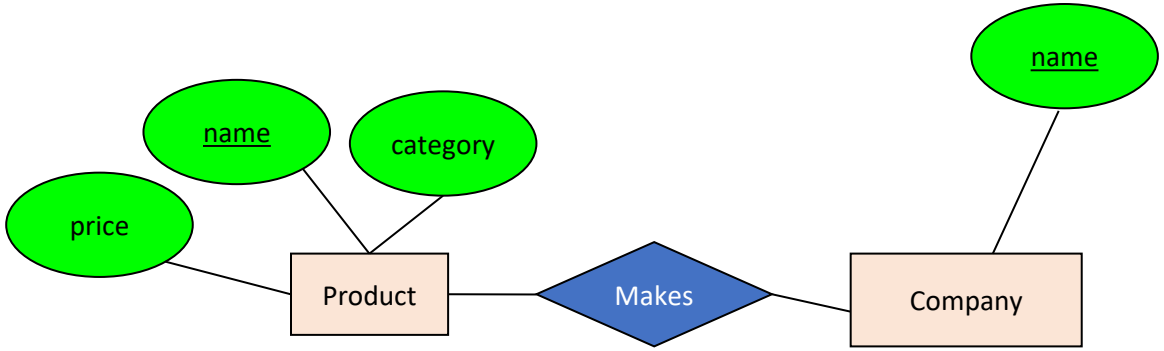
What is a Relationship?

- There can only be **one relationship for every unique combination of entities**

- This also means that **the relationship is uniquely determined by the keys of its entities**

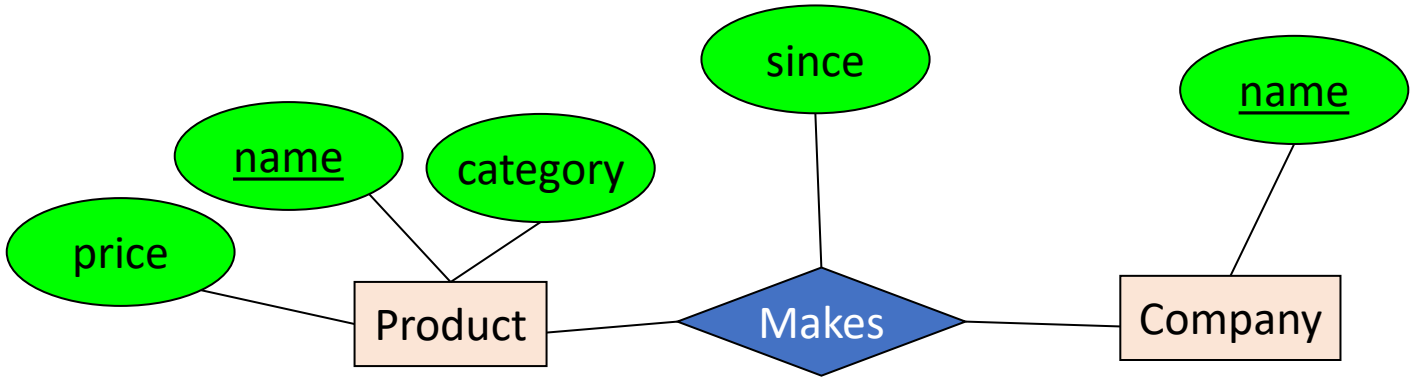
- Example: the "key" for Makes (to right) is
  {Product.name, Company.name}

This follows from our mathematical definition of a relationship- it's a SET!



Relationships with Attributes

- Relationships may have attributes as well.
  - Called **descriptive attributes**.



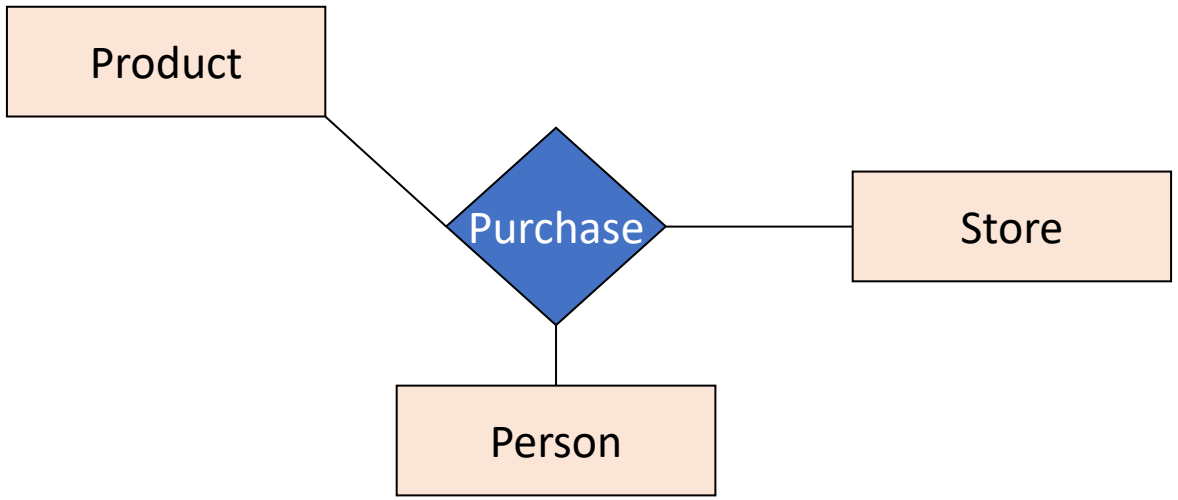
For example: “since” records when company started making a product

Makes

C.name	P.name	Since
Apple	iPhone 12	2020.11.01
Apple	iPhone 12	2021.01.01

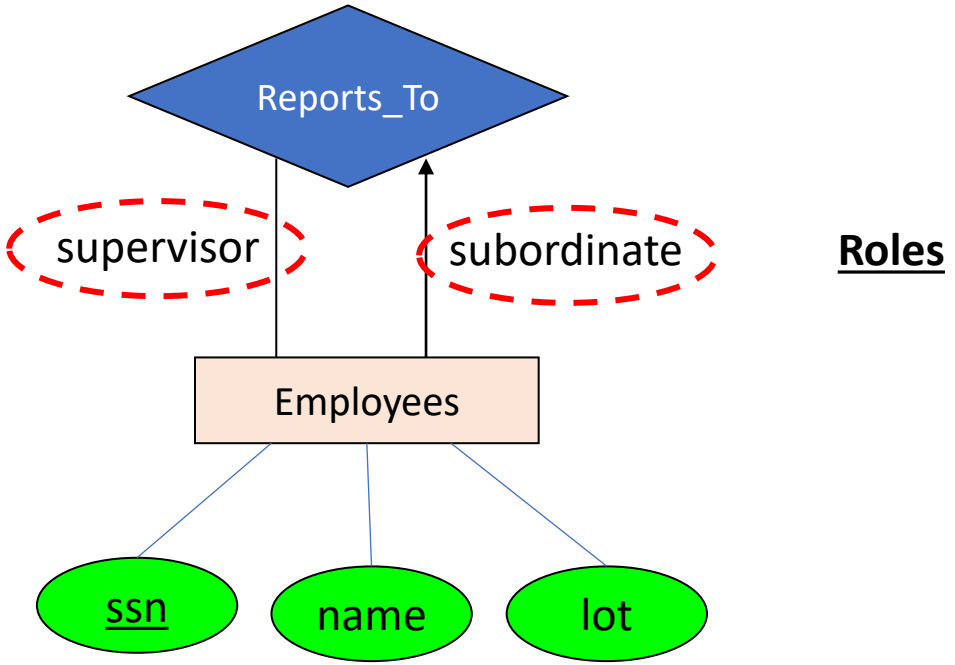
Multi-way Relationship

- How do we model a purchase relationship in multiple ways?
  - e.g., between buyers, products and stores



Roles in Relationship

- What about a relationship from an entity set to itself?
  - E.g, Employee A manages employee B.



Exercise: ER diagram for Geography

Entities

- Country: name, area, population, gdp
- City: name, population, longitude, latitude
- River: name, length
- Sea: name, max depth

Relationships

- City belongs to Country
- River crosses Country
- River ends in Sea

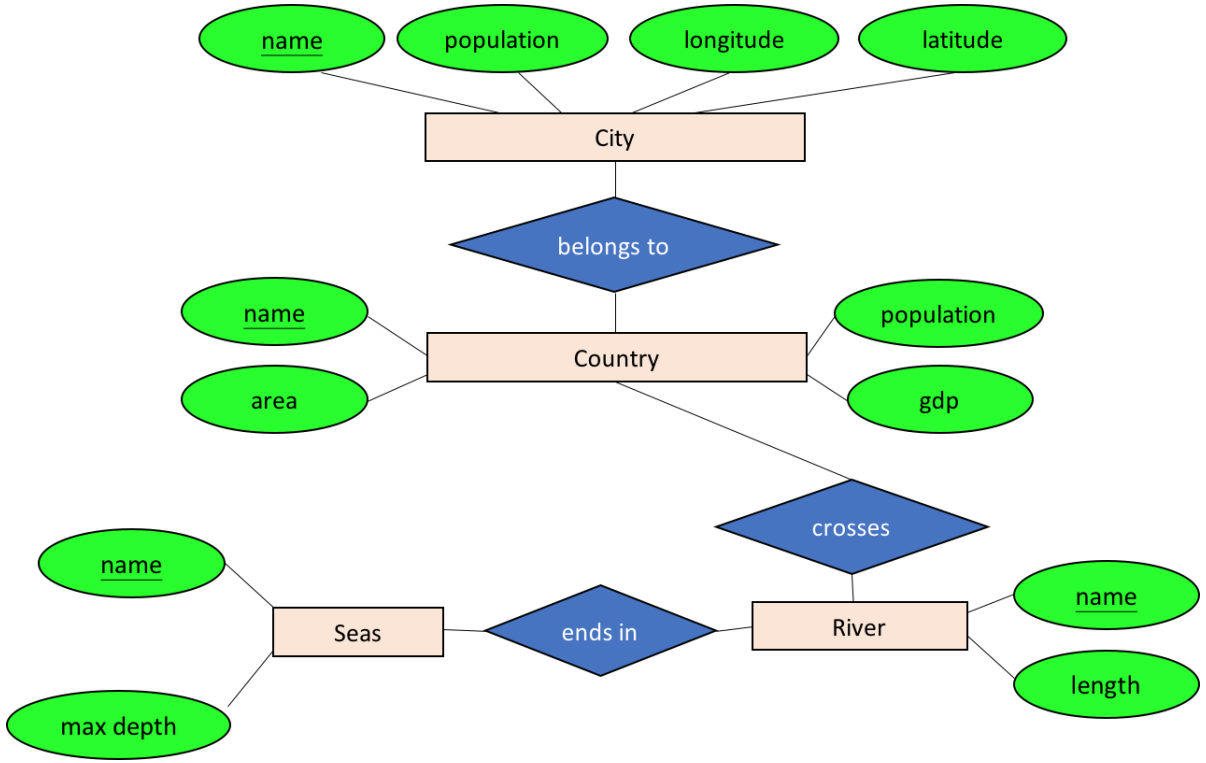
Exercise: ER diagram for Geography

Entities

- Country: name, area, population, gdp
- City: name, population, longitude, latitude
- River: name, length
- Sea: name, max depth

Relationships

- City belongs to Country
- River crosses Country
- River ends in Sea



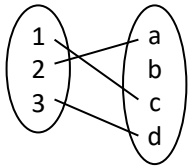
Integrity Constraints

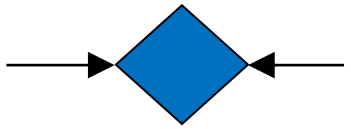
- Rules in database to describe limit of occurrences of certain events.
  - Student cannot register for a course if it is already at capacity.
  - In other words, we always have: $currentRegistration \leq Capacity$
- In SQL, we already see some of them:
  - UNIQUE, FOREIGN KEY, PRIMARY KEY, CHECK, etc.

- How do we describe some of them in an ER diagram?
- Restrictions in relationships or entities?

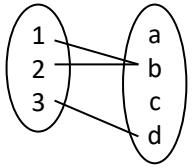
Key Constraints of Relationships

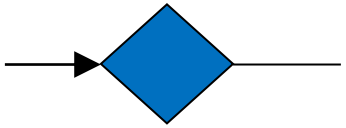
One-to-one:



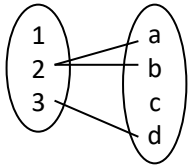


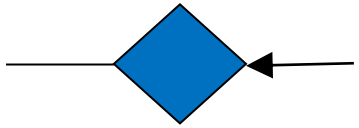
Many-to-one:



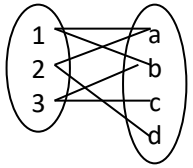


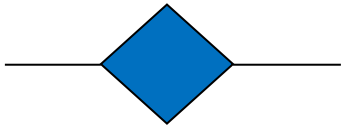
One-to-many:





Many-to-many:



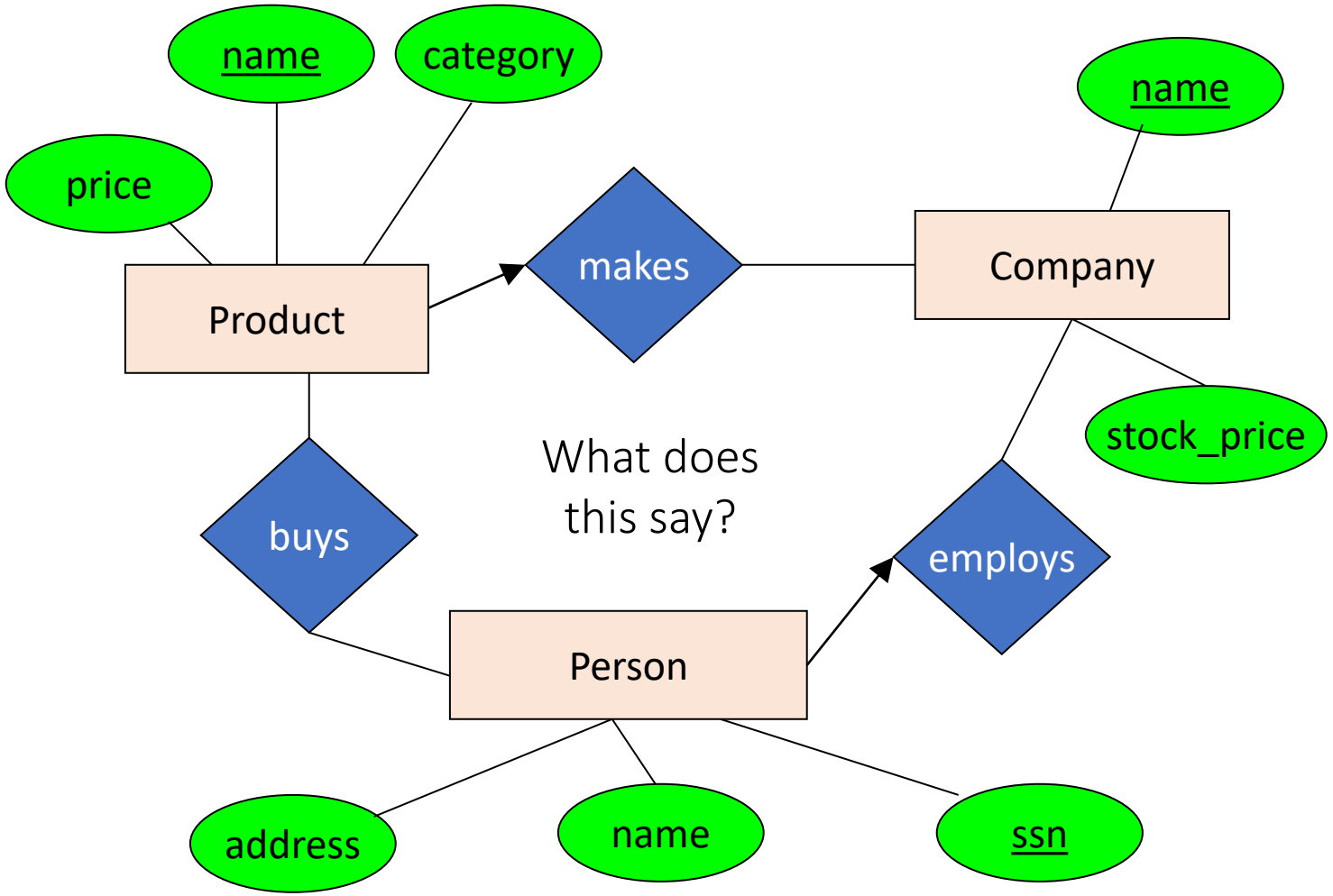


Indicated using arrows

Many-to-one: Many on the left can associate with one on the right

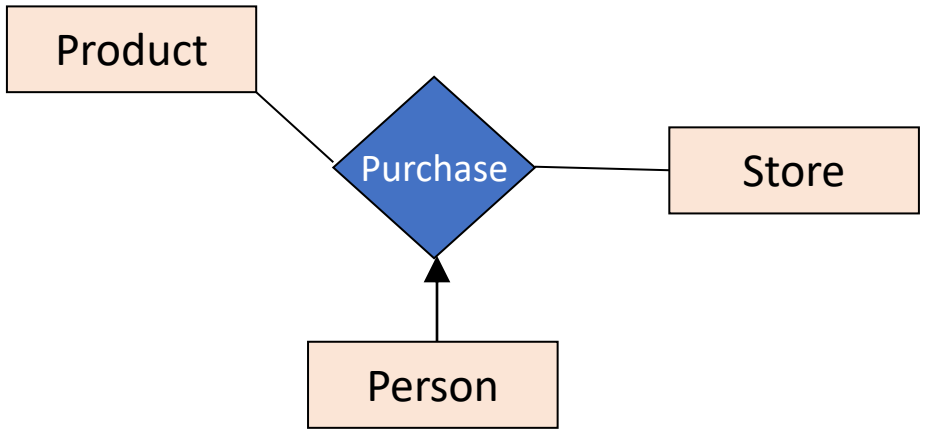
X -> Y means **there exists a function mapping from X to Y** (*recall the definition of a function)*

Implies each entity in X must appears **at most one time** in the relationship.


name
category
price
name
makes
Product
Company
What does
this say?
stock_price
buys
employs
Person
address
name
ssn

Arrows in Multiway Relationships

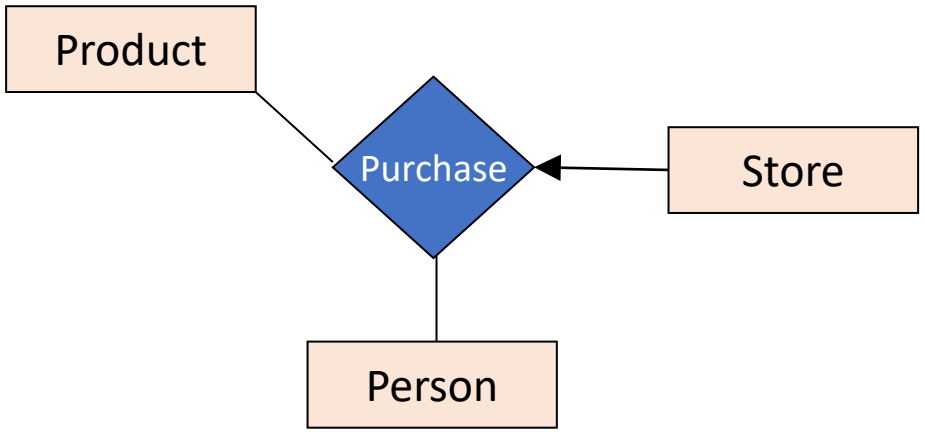
Q: What does the arrow mean ?



A: Given a person, we can determine what they bought and the store where they bought it

Arrows in Multiway Relationships

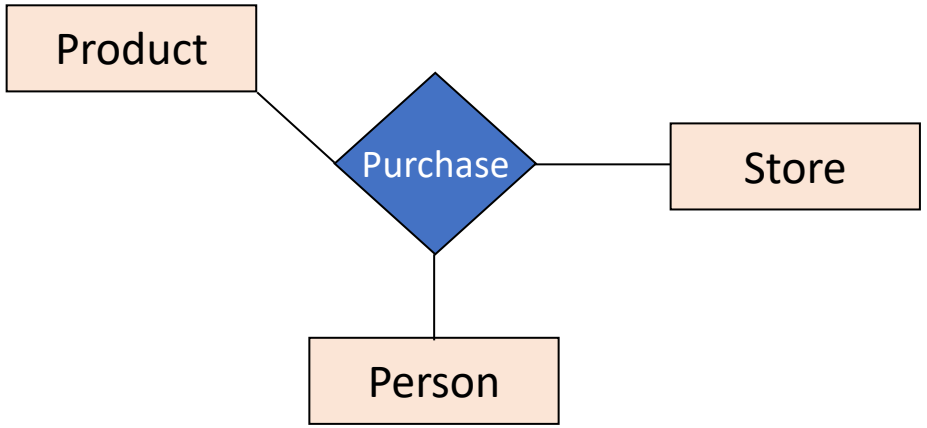
Q: What does the arrow mean ?



A: Given a store, we can determine who shopped there and the product they bought each store sells one product and to one person, ever.

Arrows in Multiway Relationships

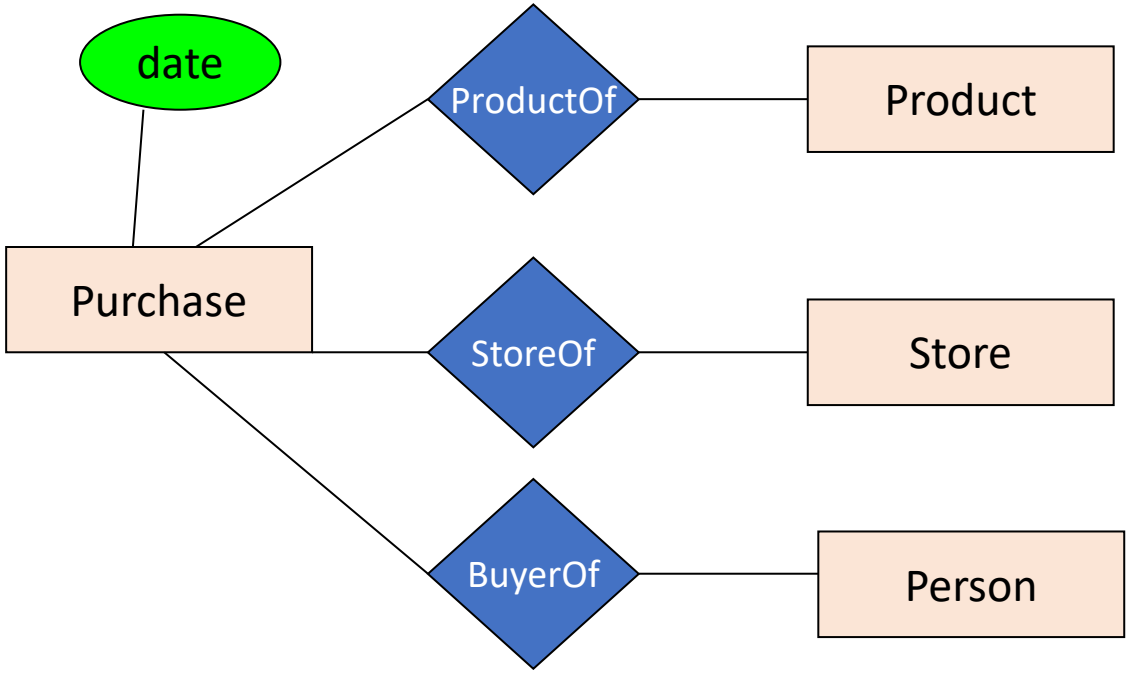
Q: How do we say that every person shops in at most one store?



A: Cannot. This is the best approximation. (Why only approximation ?)

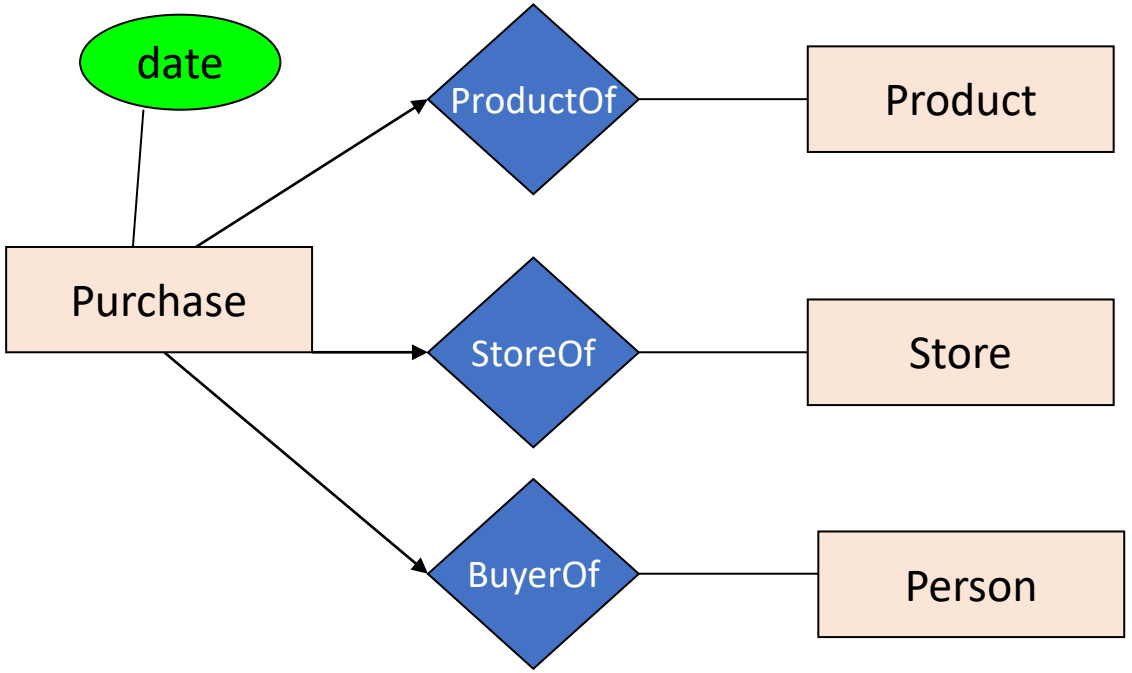
To achieve it, need something else (**aggregation** / **extra entity**)

Multiway Relationships to Binary



From what we had on previous slide to this – what to do?

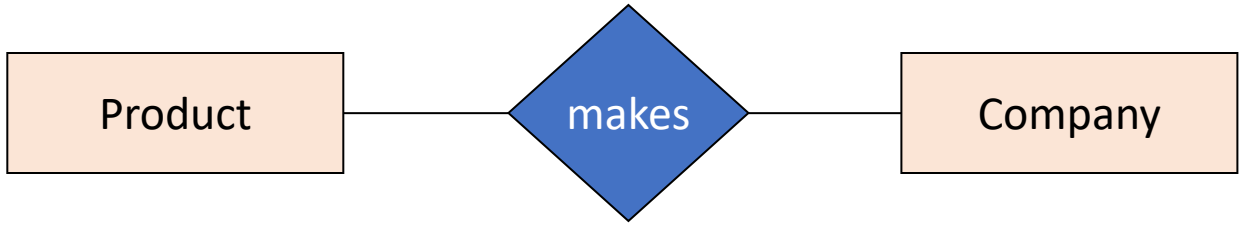
Multiway Relationships to Binary



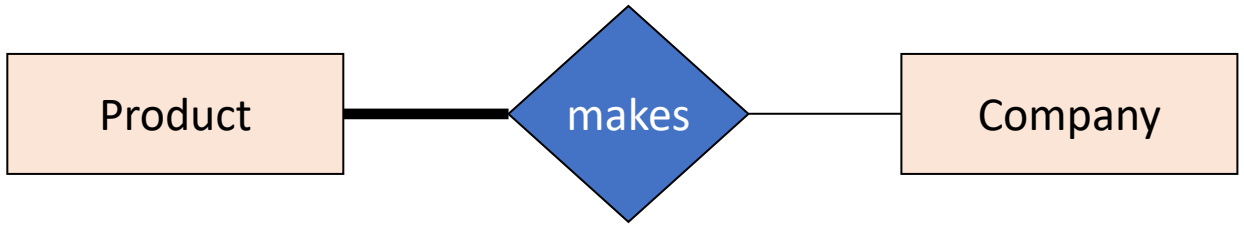
From what we had on previous slide to this – what to do?

Participation Constraints

- Partial participation vs. Total participation



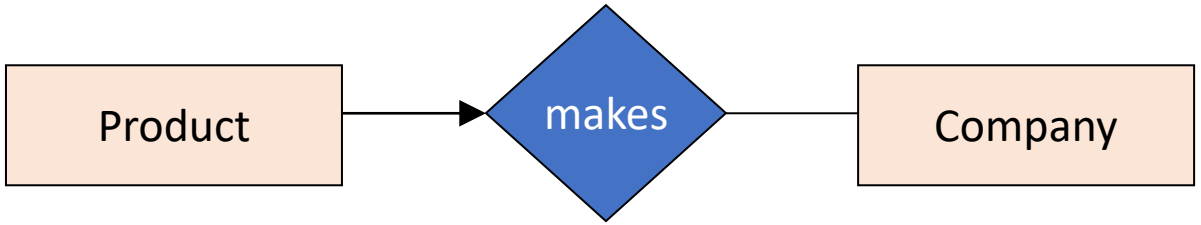
Are there products made by no company?
Companies that don't make a product?



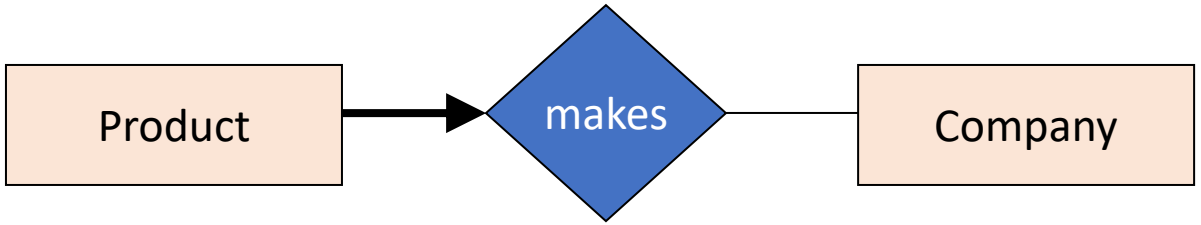
Bold line indicates ***total participation*** (i.e. here: all products are made by a company)
Total participation enforce each entity must appear **at least one time** in the relationship.

Single-value Constraints

- Combining key constraint and total participation constraint.



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



Each product made by *exactly* one company.

Exercise: ER diagram for Geography

Entities

- Country: name, area, population, gdp
- City: name, population, longitude, latitude
- River: name, length
- Sea: name, max depth

Relationships w/ Constraints

- Each city belongs to a single country
- Each river crosses one or several countries
- Each river ends in a single sea

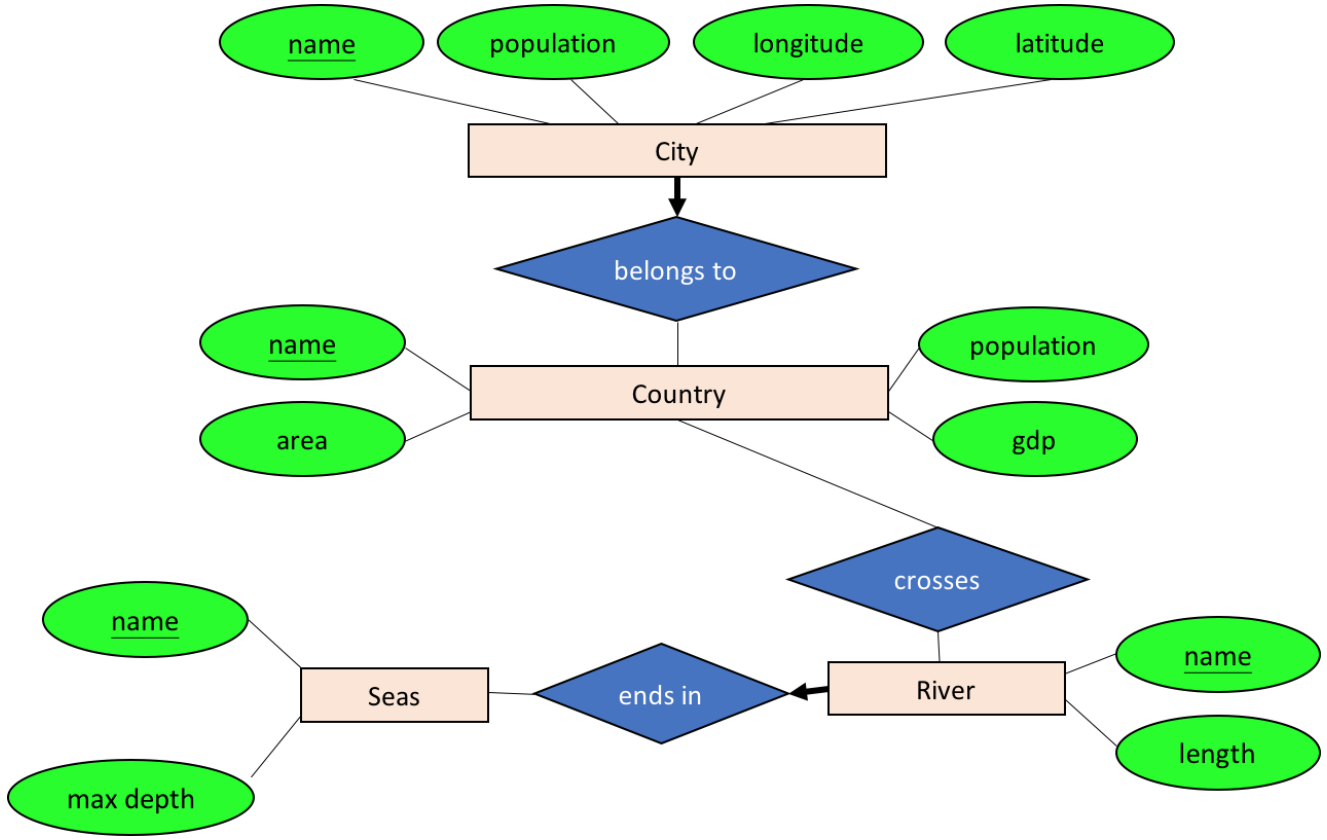
Exercise: ER diagram for Geography

Entities

- Country: name, area, population, gdp
- City: name, population, longitude, latitude
- River: name, length
- Sea: name, max depth

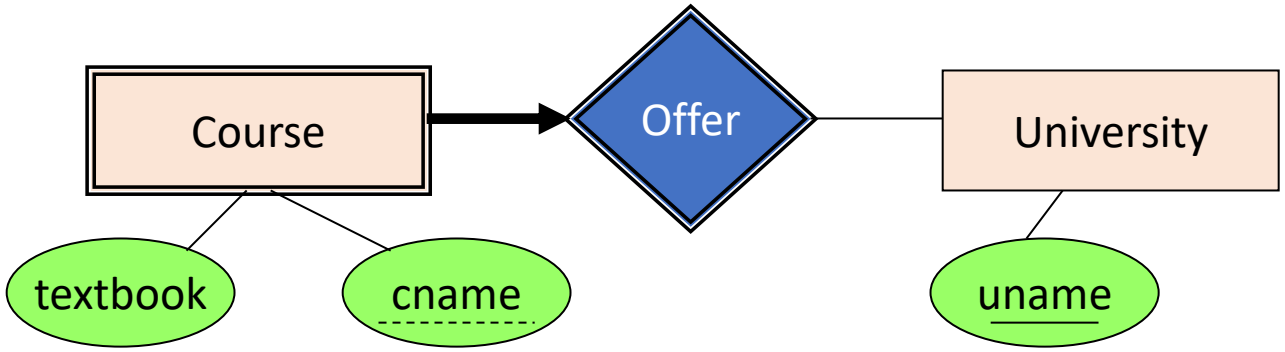
Relationships w/ Constraints

- Each city belongs to a single country
- Each river crosses one or several countries
- Each river ends in a single sea



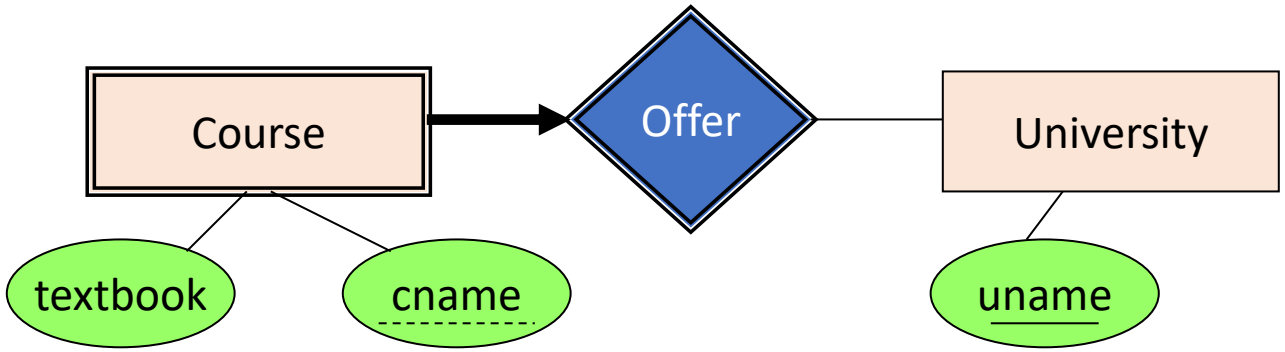
Weak Entities and Weak Entity Sets

- An entity may only be of interest when associated with another entity in another entity set.
  - Classes offered by different university. Course name only matters when associated with a university.
  - "**Database class**" vs. "**Database class** offered by **Penn State**"
- An entity set is ***weak*** when its key comes from other entity sets to which they are related.



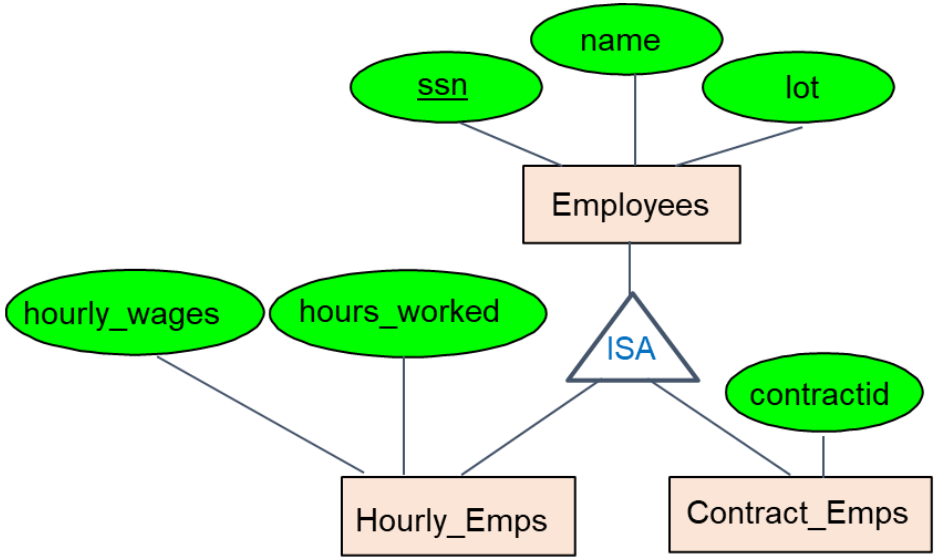
Weak Entities and Weak Entity Sets

- An entity set is *weak* when its key comes from other entity sets to which they are related.



- cname is a *partial key* (denote with dashed underline).
- University is called the *supporting entity set*
- Offer is called the supporting relationship

Class Hierarchies



- Key Concept in OO: subclasses and inherence
  - Employee -> Contracted/Hourly
- Describe specialized entities.
  - To add descriptive attributes specific to a subclass.
  - To identify entities that participate in a particular relationship

- We declare A **ISA** B (every A entity is also considered to be a B entity)
  - Hourly Employee **is an** employee

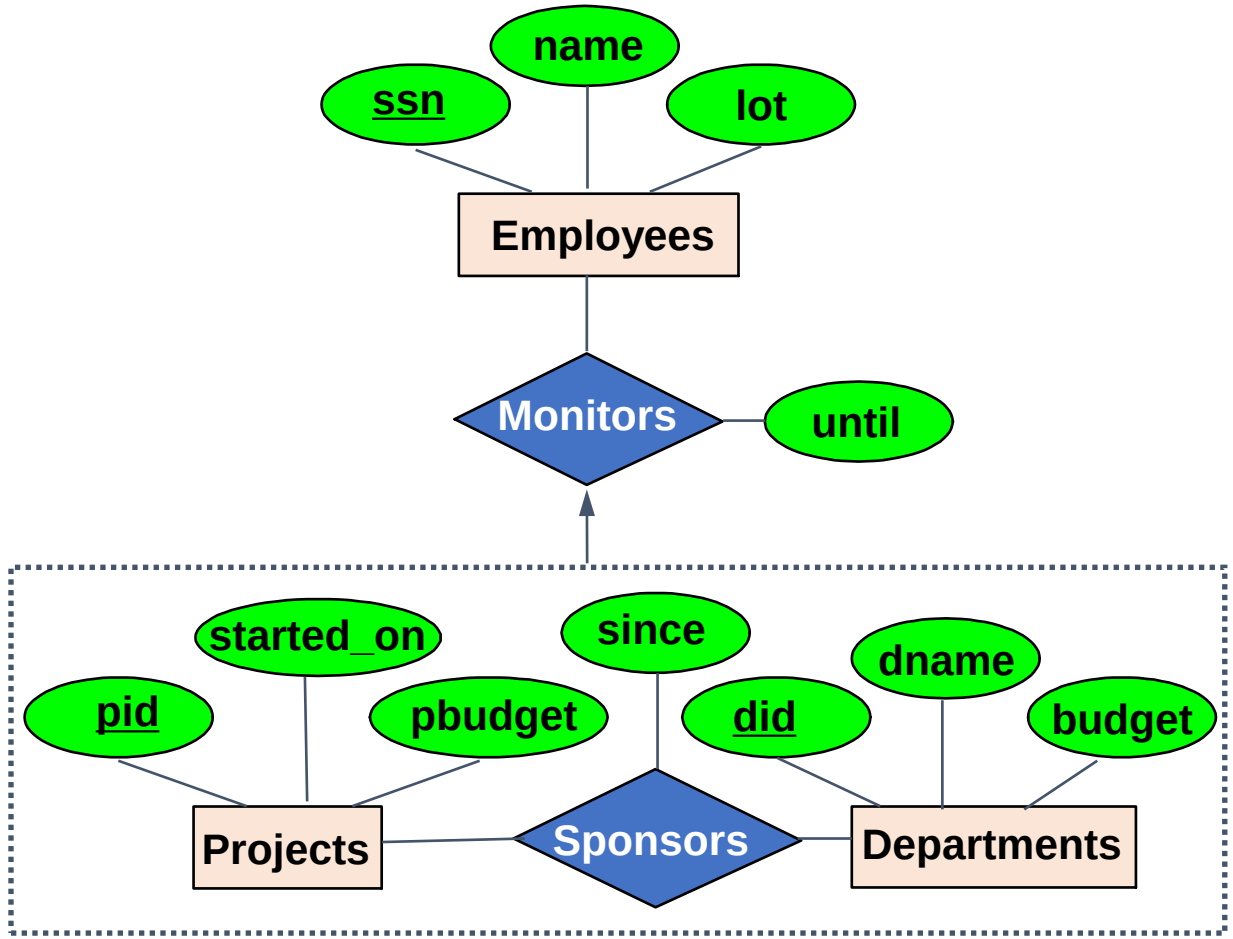
- **Overlap constraints**: Can Joe be a Hourly_Emps as well as a Contract_Emps entity? (Allowed/disallowed)
- **Covering constraints**: Does every Employees entity also have to be an Hourly_Emps or a Contract_Emps entity? (Yes/no)

Aggregations

- Model a relationship involving **another relationship set**.
- Allows us to treat a relationship set as an entity set for the purposes of participation in (other) relationships.

***Why not multi-way relationship*?**

- "Monitors" and "sponsors" are two truly distinct relationships.
- Also, can say that each sponsorship is monitored by at most one employee.



Design Choices

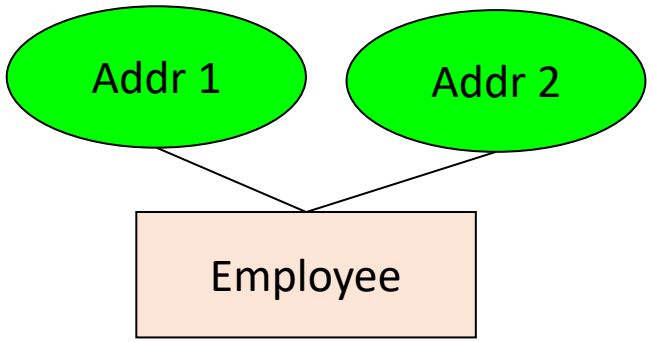
- ER Modeling can be tricky!

- Should a concept be modeled as an entity or an attribute?
- Should a concept be modeled as an entity or a relationship?
- Identifying relationships: Binary or Multiway? Aggregation?

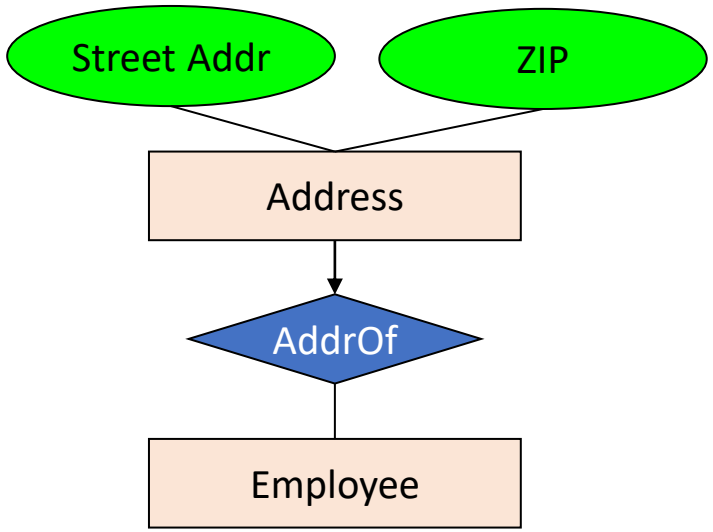
- Note constraints of the ER Model:
  - A lot of data semantics can (and should) be captured.
  - But some constraints cannot be captured in ER diagrams.
    - Put it in text during conceptual design.
    - Refine things in our logical (relational) design. (DB design theory.)

Entity vs. Attribute?

Should address (A)
be an attribute?

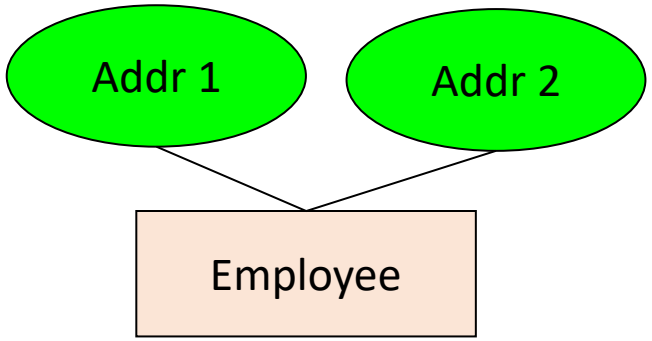


Or (B) be an entity?



Entity vs. Attribute?

Should address (A) be an attribute?

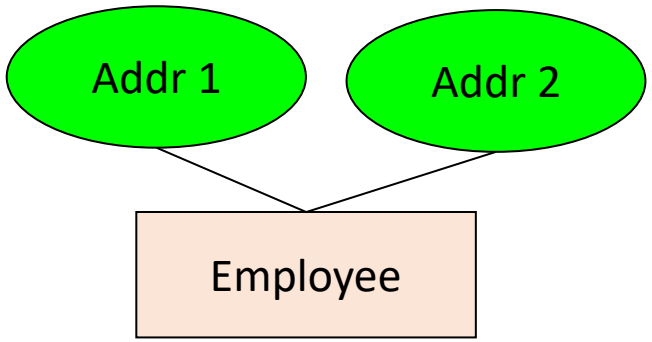


How do we handle employees with multiple addresses here?

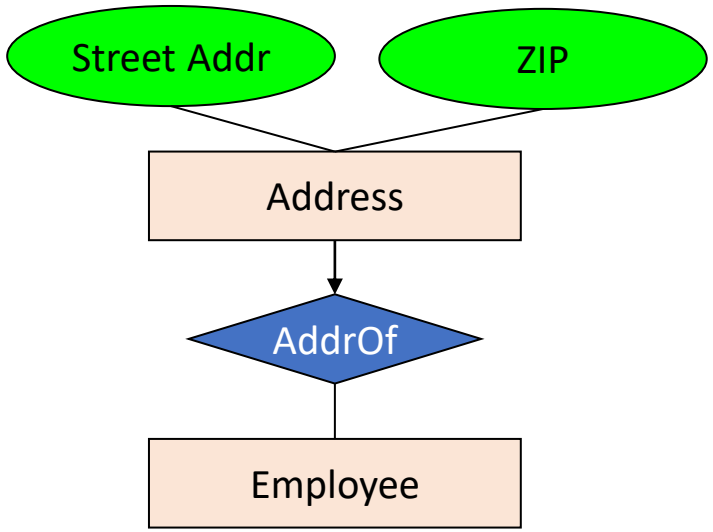
How do we handle addresses where internal structure of the address (e.g. zip code, state) is useful?

Entity vs. Attribute?

Should address (A)
be an attribute?



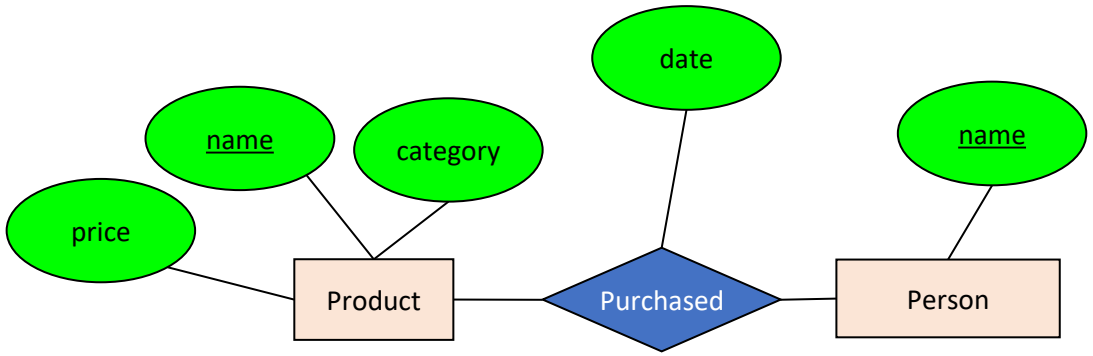
Or (B) be an entity?



In general, when we want to record several values,
we choose new entity

Relationship vs. Entity?

- **Q:** What does this say?



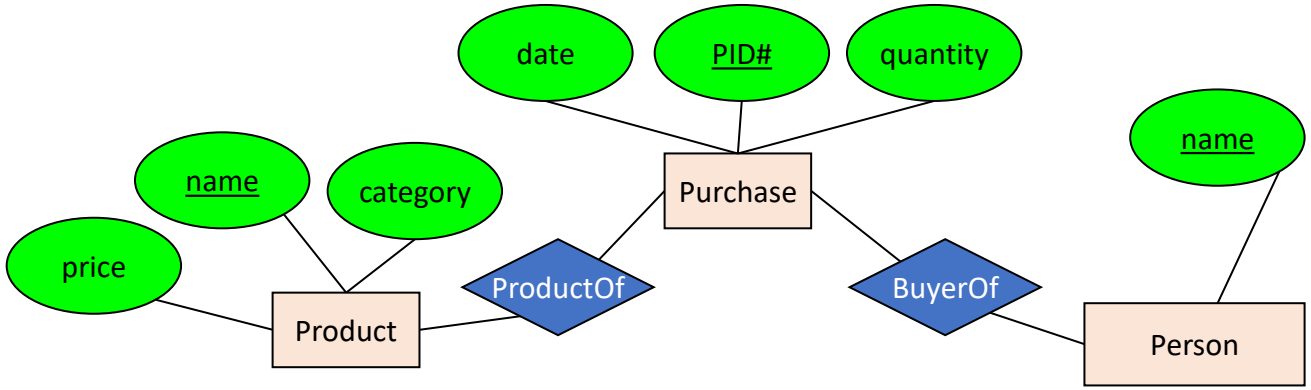
- **A:** A person can only buy a specific product once

Purchase

Person.name	Product.name	Date
Dong	iPhone 12	2020.10.01
Dong	iPhone 12	2020.12.01

Relationship vs. Entity?

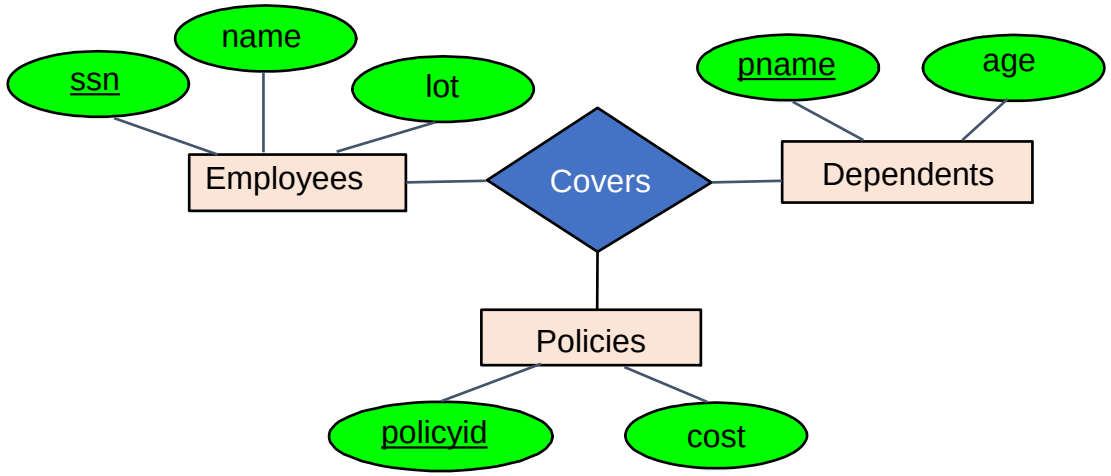
- What about this way?



- *Now we can have multiple purchases per product, person pair!*

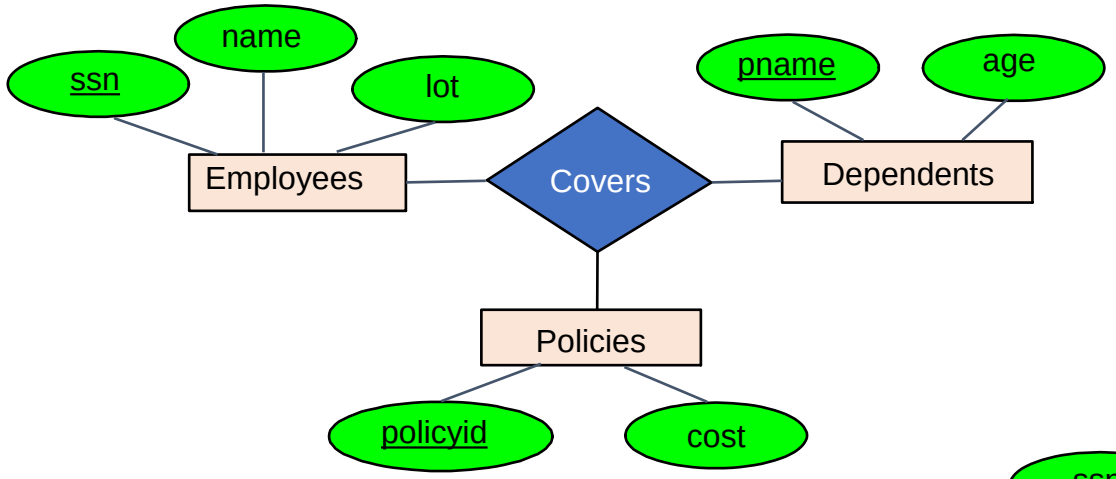
We can always use **a new entity** instead of a relationship. For example, to permit multiple instances of each entity combination!

Multiway vs. Binary?

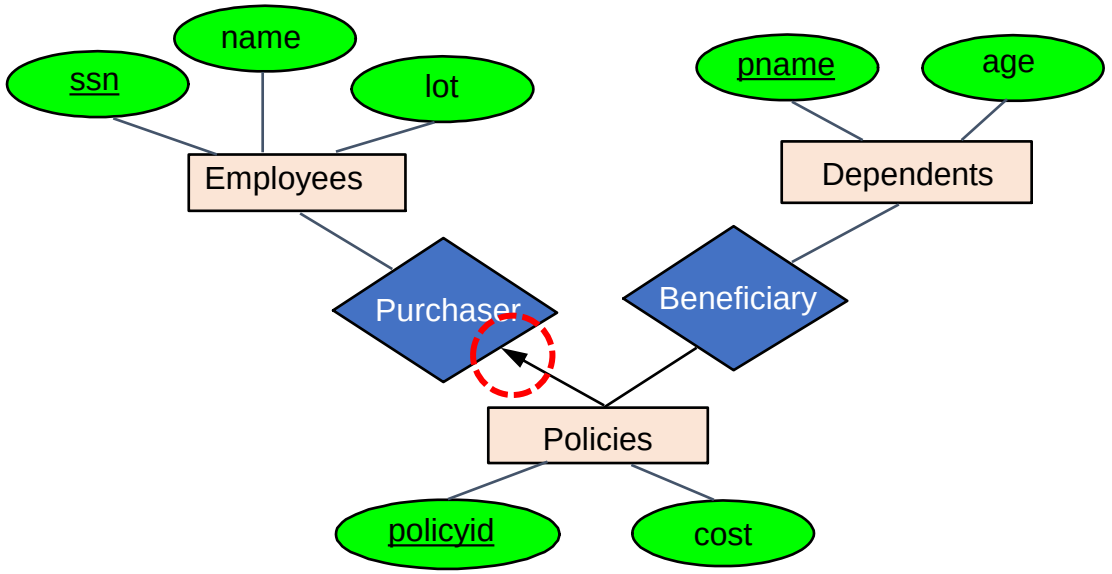


What if I want to make sure each policy is own by just 1 employee (cannot be jointly owned) but may cover multiple dependents?

Multiway vs. Binary?



What if I want to make sure each policy is own by just 1 employee (cannot be jointly owned) but may cover multiple dependents?



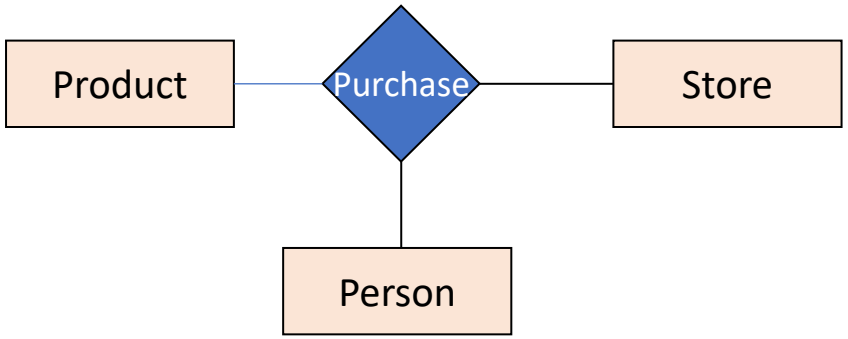
Multiway vs. Binary?

- Relationships inherently involving more than two entities?

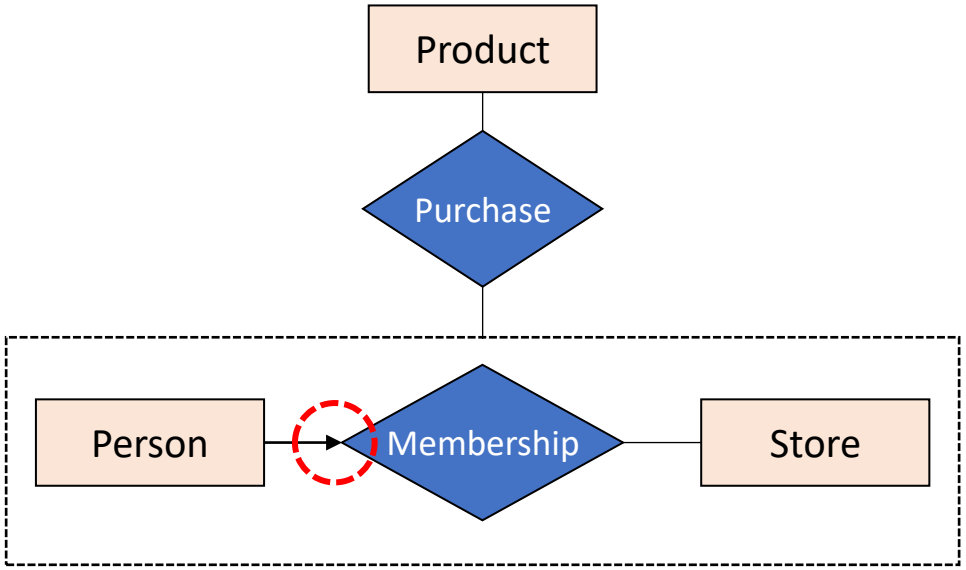
- Example: *A contract specifies that a supplier will supply some quantity of a part to a department*.
- A multiway relationship **Contracts** relates entity sets **Parts**, Departments and **Suppliers**, and has descriptive attribute **quantity**.

- No combination of binary relationships is an adequate substitute.
- S "can-supply" P, D "needs" P, and D "deals-with" S does not imply that D has agreed to buy P from S.
- How do we record **quantity**?
  - Can't represent this concept cleanly with binary relationships.

Multiway vs. Aggregation?

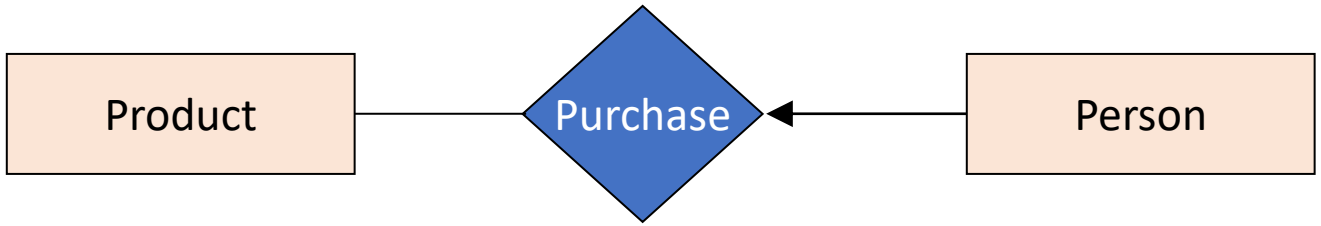


What if I want to make sure each person only buys in one store but can buy multiple products?

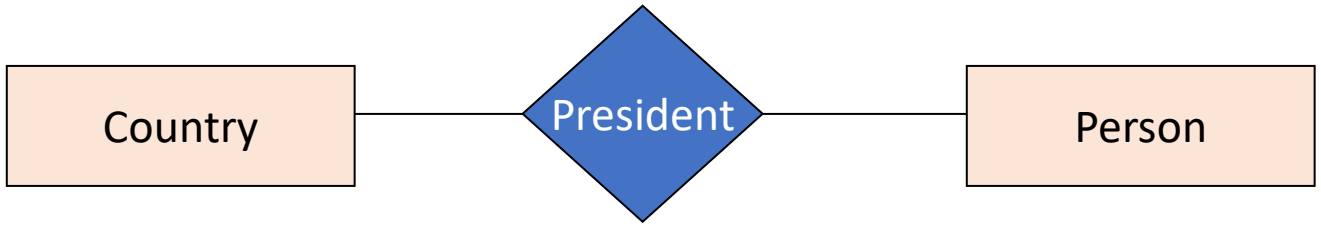


Design Principles: What's Wrong?

What's wrong with these examples?

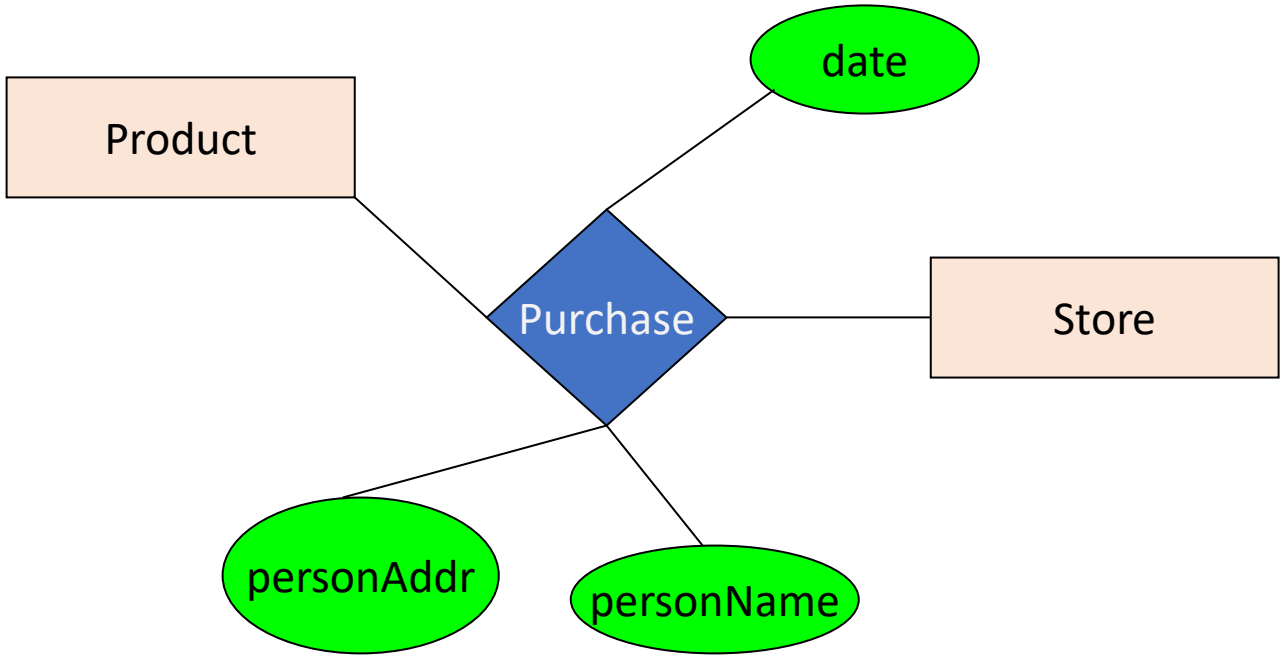


Each person buys only one product, then out



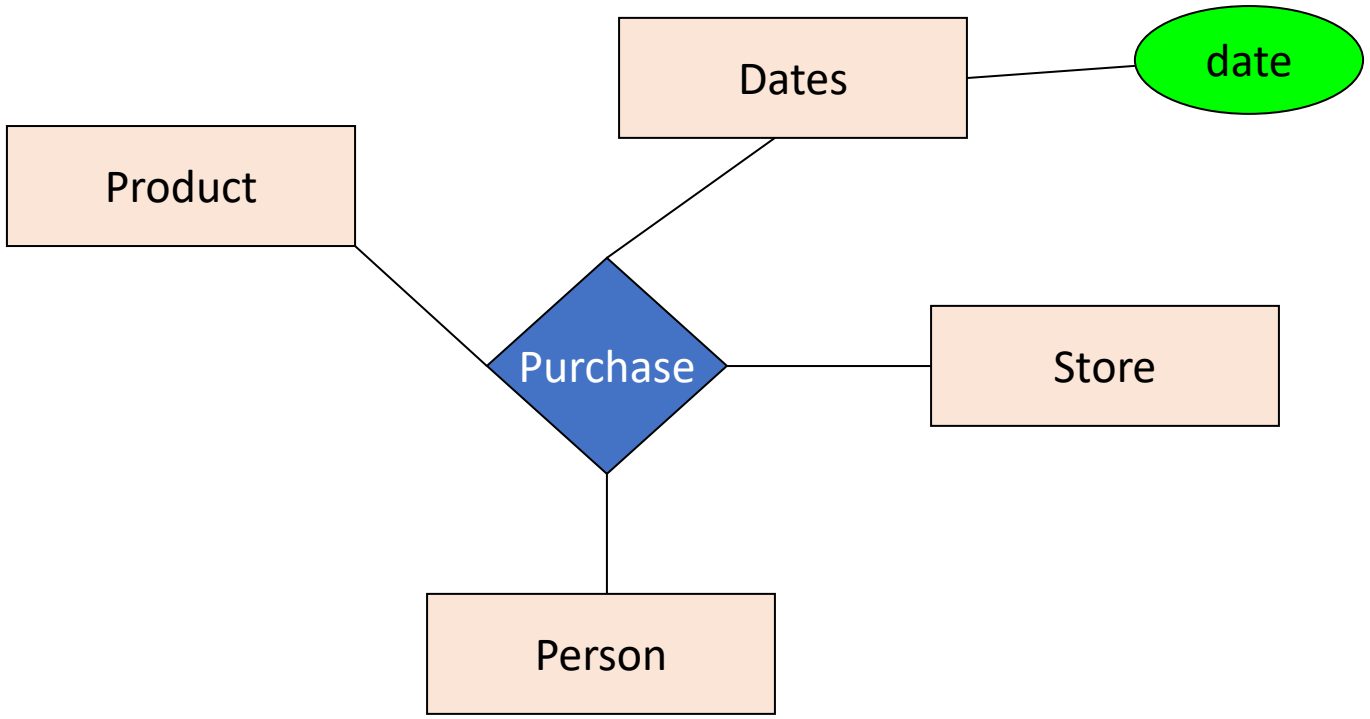
Multiple presidents, also may want to require country to have president

Design Principles: What's Wrong?



maybe people should be entities!

Design Principles: What's Wrong?



dates don't need to be an entity by themselves

Next Step…

- From conceptual DB design to Logical DB Design.
- Starting from a simple rule:
  - Entity Set → Relation
  - Relationship → Relation

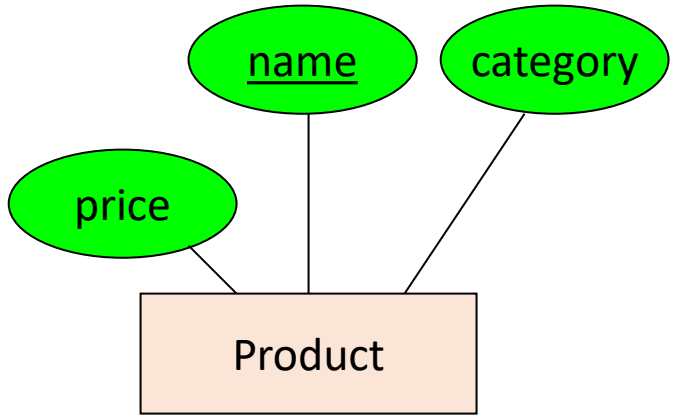
- DB schema need further refinement!
- Powerful tool of database design theory
  - Functional dependencies
  - Normal Forms

Summary

- **Conceptual design** follows **requirements analysis**
  - Yields a high-level description of data to be stored
- Basic constructs: **entities**, **relationships**, and **attributes** (of entities and relationships).
- Some additional constructs: **weak entities**, **class hierarchies**, and **aggregation**.
- Several kinds of integrity constraints:
  - **key constraints**
  - **participation constraints**
  - **overlap/covering constraints** for class hierarchies.
- ER design is **subjective and tricky**! It involves various design choices:
  - Entity vs. attributes, entity vs. relationship, binary vs. multiway relationship, aggregation vs. multiway relationship, …
- Next step: **logical database design**, towards a relational DB schema.

Logical DB Design: ER → Relational

- Entity Sets to Tables

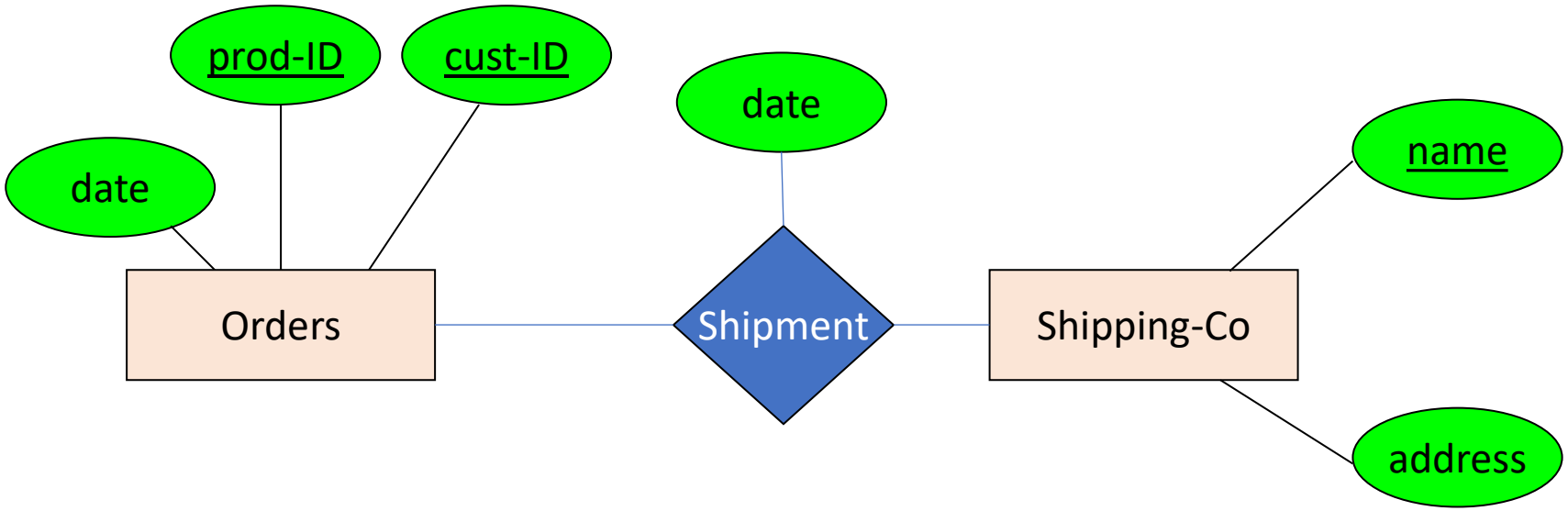


name	category	price
iPhone 12	Electronics	$700
iPad Pro	Electronics	$300
Office	Software	$120

```
CREATE TABLE Product(
    name VARCHAR(40),
    category VARCHAR(40),
    price DECIMAL(2, 10),
    PRIMARY KEY (name))
```

Product(name,category, price)

Logical DB Design: ER → Relational



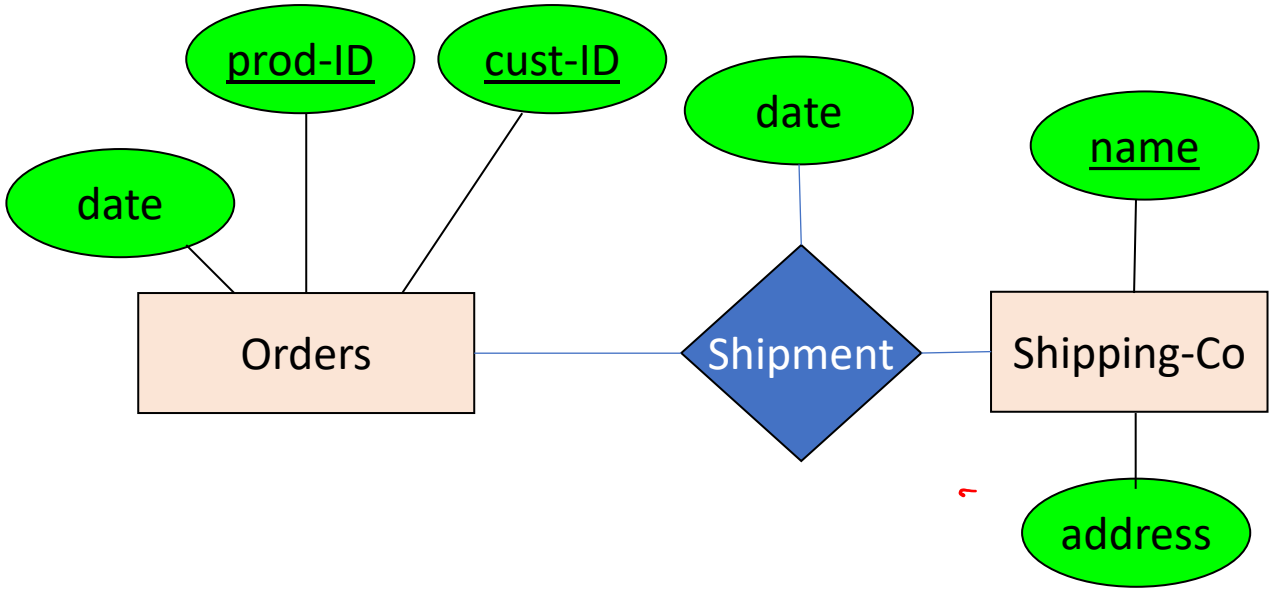
Orders(<u>prod-ID</u>, <u>cust-ID</u>, date)

Shipment(<u>prod-ID</u>, <u>cust-ID</u>, <u>name</u>, date)

Shipping-Co(<u>name</u>, address)

prod-ID	cust-ID	name	date
IPH12P	dongx	UPS	01/12/2021
IPH12P	dongx	FEDEX	02/12/2021

Many-to-Many Relationship to Tables



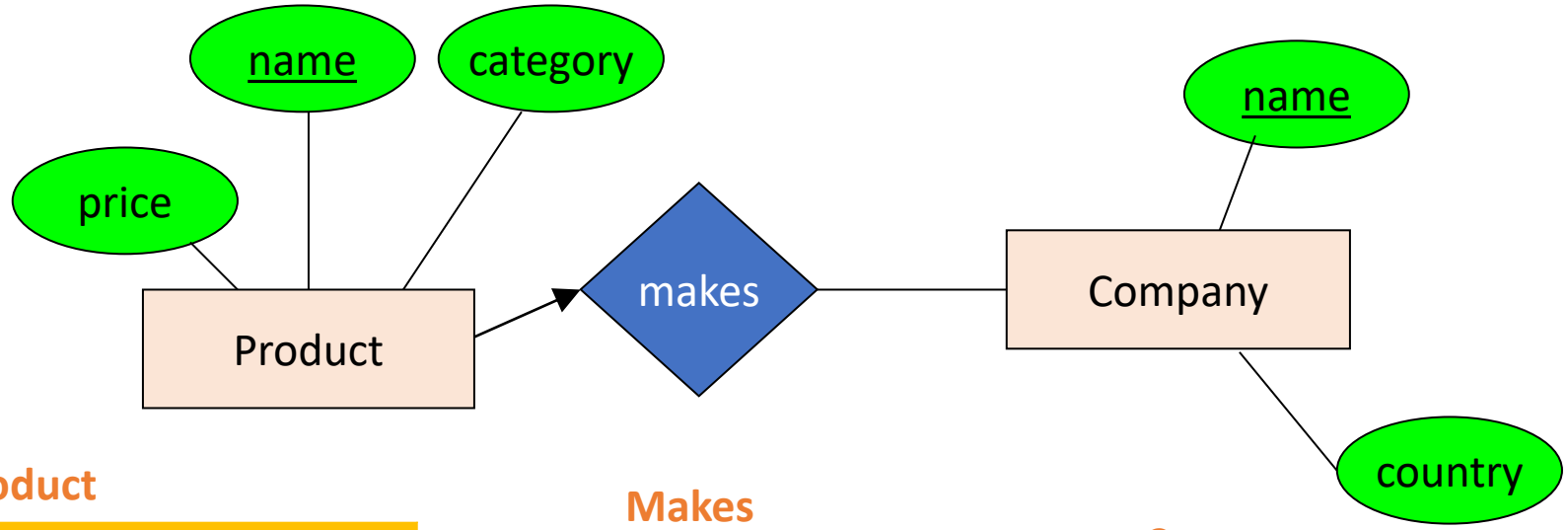
prod-ID	cust-ID	name	date
IPH12P	dongx	UPS	01/12/2021
IPH12P	dongx	FEDEX	02/12/2021

Shipment(prod-ID, cust-ID, name, date)

```
CREATE TABLE Shipment(
  prod-ID CHAR(6),
  cust-ID VARCHAR(60),
  name VARCHAR(10),
  date DATETIME,
  PRIMARY KEY (prod-ID, cust-ID, name),
  FOREIGN KEY (prod-ID, cust-ID)
    REFERENCES Order(prod-ID, cust-ID),
  FOREIGN KEY (name)
    REFERENCE Shipping-Co(name))
```

Key Constraints: Combining Relations

- For many to one relationships



Product

name	category	price
iPhone 12	Electronics	$700
iPad Pro	Electronics	$300
Office	Software	$120

Makes

P.name	C.name
iPhone 12	Apple
iPad Pro	Apple
Office	Microsoft

Company

name	country
Apple	US
Microsoft	US

Key Constraints: Combining Relations

Product

name	category	price
iPhone 12	Electronics	$700
iPad Pro	Electronics	$300
Office	Software	$120

Make

P.name	C.name
iPhone 12	Apple
iPad Pro	Apple
Office	Microsoft

Company

name	country
Apple	US
Microsoft	US

pname	cname	category	price
iPhone 12	Apple	Electronics	$700
iPad Pro	Apple	Electronics	$300
Office	Microsoft	Software	$120

name	country
Apple	US
Microsoft	US

- No separate relations for many-one relationship

Key Constraints: Combining Relations

Product

name	category	price
iPhone 12	Electronics	$700
iPad Pro	Electronics	$300
Office	Software	$120

Make

P.name	C.name
iPhone 12	Apple
iPad Pro	Apple
Office	Microsoft

pname	cname	category	price
iPhone 12	Apple	Electronics	$700
iPad Pro	Apple	Electronics	$300
Office	Microsoft	Software	$120

```
CREATE TABLE Product(
  pname VARCHAR(50),
  cname VARCHAR(60),
  category VARCHAR(60),
  price DECIMAL(2, 10),
  PRIMARY KEY (pname),
  FOREIGN KEY (cname)
    REFERENCES Company(name))
```

Company

name	country
Apple	US
Microsoft	US

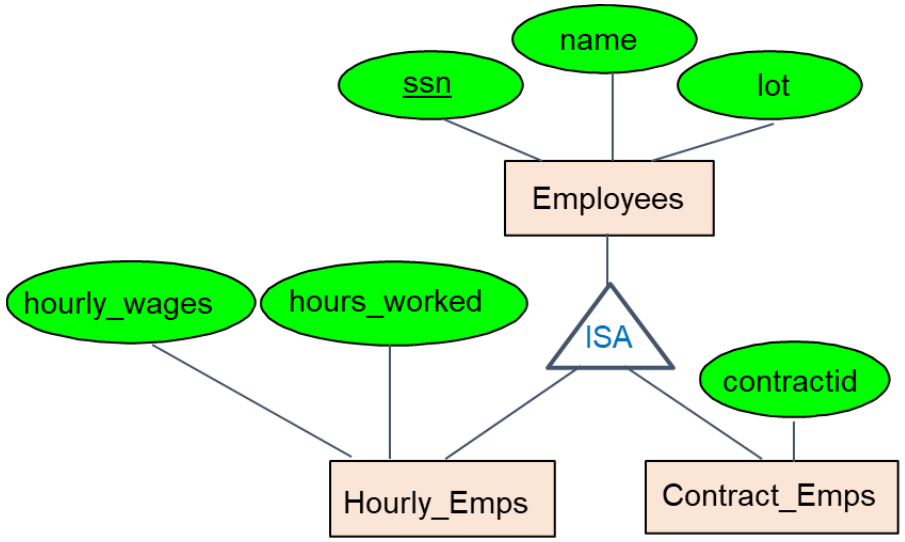
Capturing Participation Constraints

- We can capture participation constraints **involving one entity set (exactly once semantics)** in a binary relationship, but little else.

```
CREATE TABLE Product(
  pname VARCHAR(50),
  cname VARCHAR(60) NOT NULL,
  category VARCHAR(60),
  price DECIMAL(2, 10),
  PRIMARY KEY (pname),
  FOREIGN KEY (cname)
    REFERENCES Company(name))
```

Translating ISA Hierarchies

- **General approach***:*
  - 3 relations: Employees, Hourly_Emps and Contract_Emps.
  - Employees(ssn, name, lot)
  - Hourly_Emps(ssnFK, hourly wages, hours_worked)
  - Contract_Emps(ssnFK, contractid)
  - Queries involving all employees easy, those involving just Hourly_Emps require a join to get some attributes.
- **Alternative:** Just Hourly_Emps and Contract_Emps.
  - Hourly_Emps(ssn, name, lot, hourly wages, hours_worked)
  - Contract_Emps(ssn, name, lot, contractid)
  - **Each employee must be in one of these two subclasses.**



Translating Aggregation

- Treat a relationship table as an entity!
- Key constraints -> also combine relations.

Monitors(pid^{FK_P}, did^{FK_D}, ssn^{FK_E}, until)

Sponsor_Monitors(pid^{FK_P}, did^{FK_D}, ssn^{FK_E}, since, until)