

Database Management System I

Entity Relational Diagram Model
Week #2

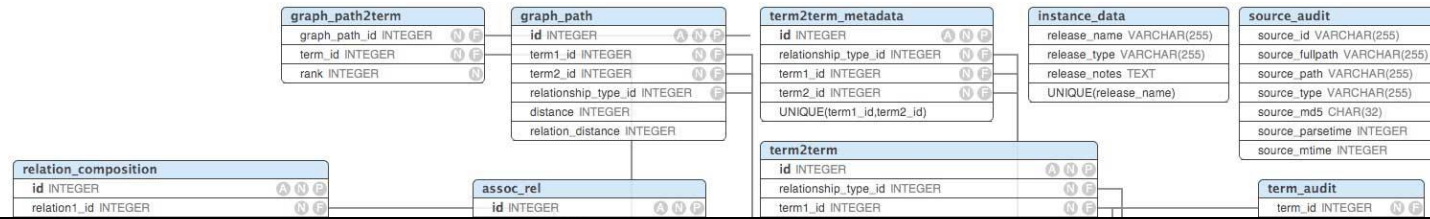
The Road Ahead

E/R Diagram

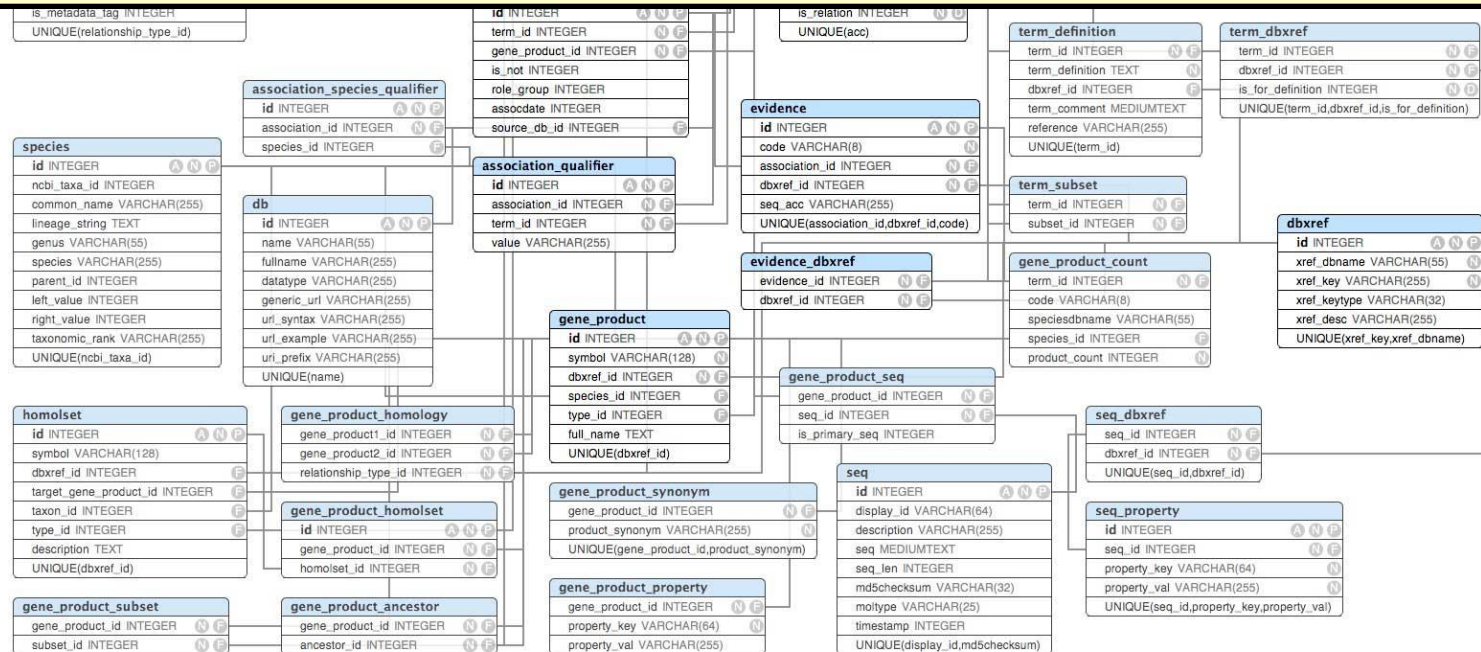
- Introduction to Entity-Relationship (ER) Diagrams
- Entity
- Relationship
- Constraints
- Subclasses
- Weak Entity Sets
- ER Design Principle
- Translating an ER Diagram into a Relational Scheme Design



A real database may have a large number of tables...



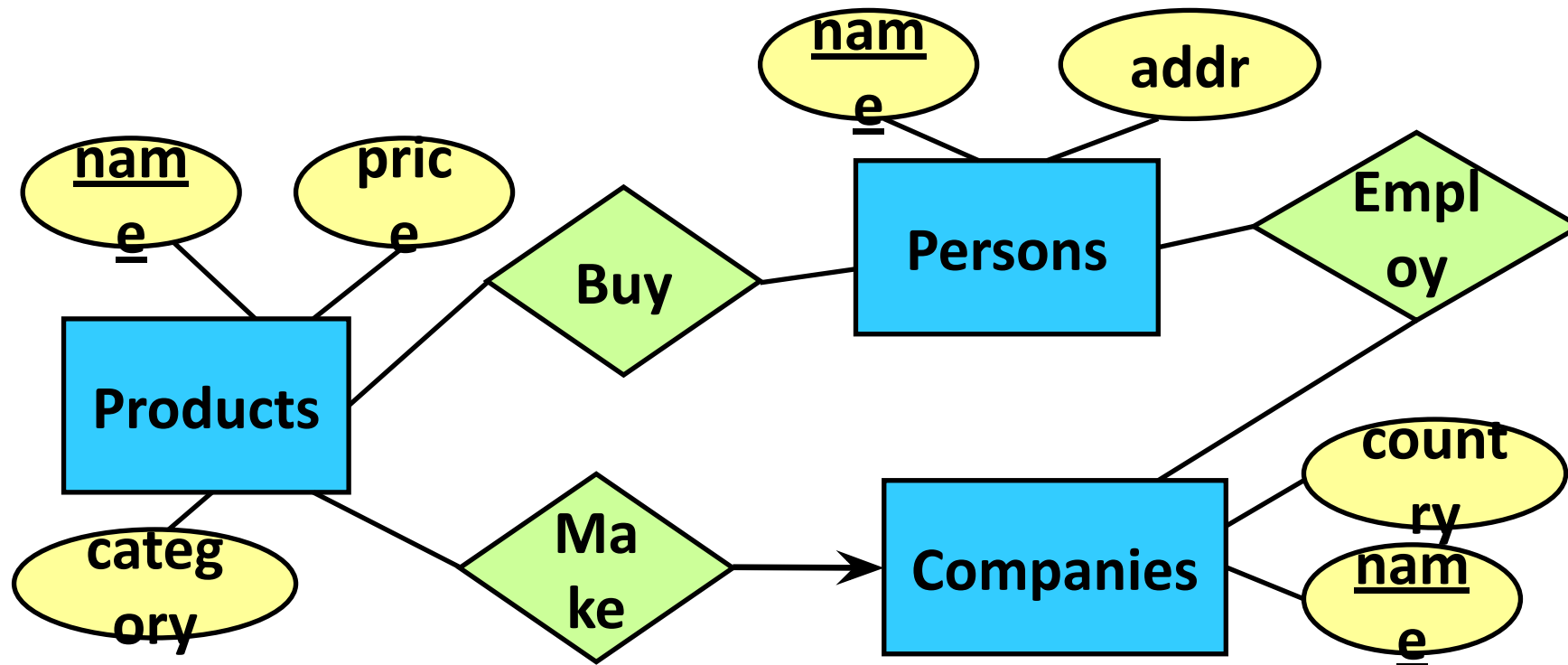
- Imagine that you are asked to design a database like this....
- How would you approach the problem?



Designing a Database for an Application

- Conceptually model the requirements of the application
 - What are the things that need to be stored?
 - How do they interact with each other?
- **Tool to use: Entity-Relationship (ER) Diagrams**
 - for modelling
- Translate the conceptual model into a set of tables
- Create the tables with a DBMS

ER Diagram

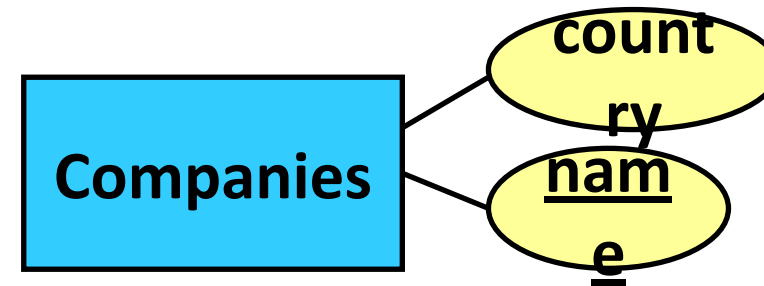
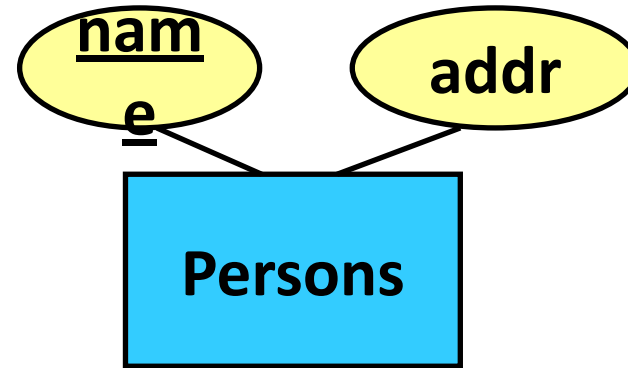
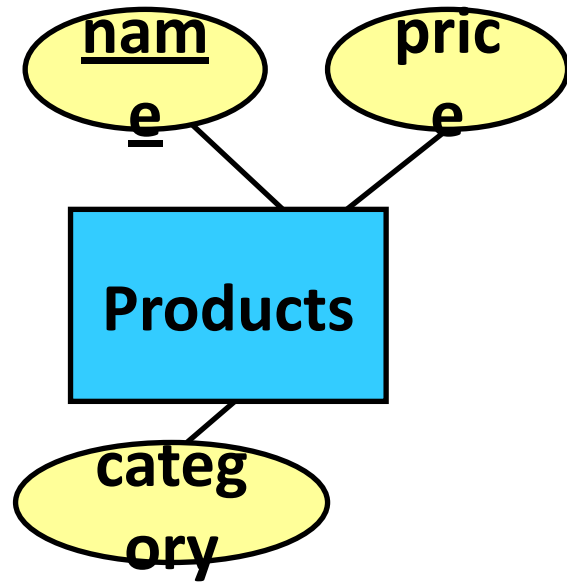


ER Diagram



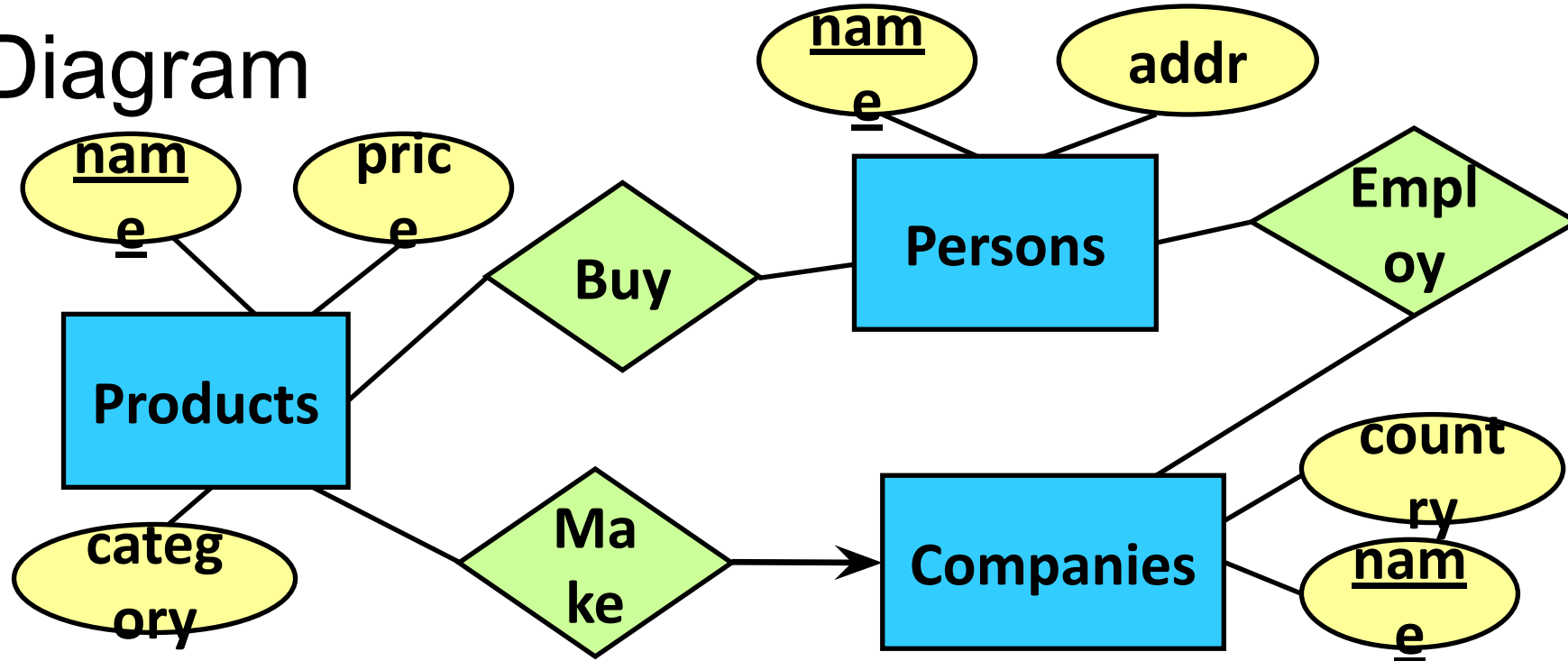
- Rectangle = **Entity**
- **Entity** = Real-world object (e.g., a bar)
- **Entity Set** = Collection of similar objects (e.g., a set of bars)

ER Diagram



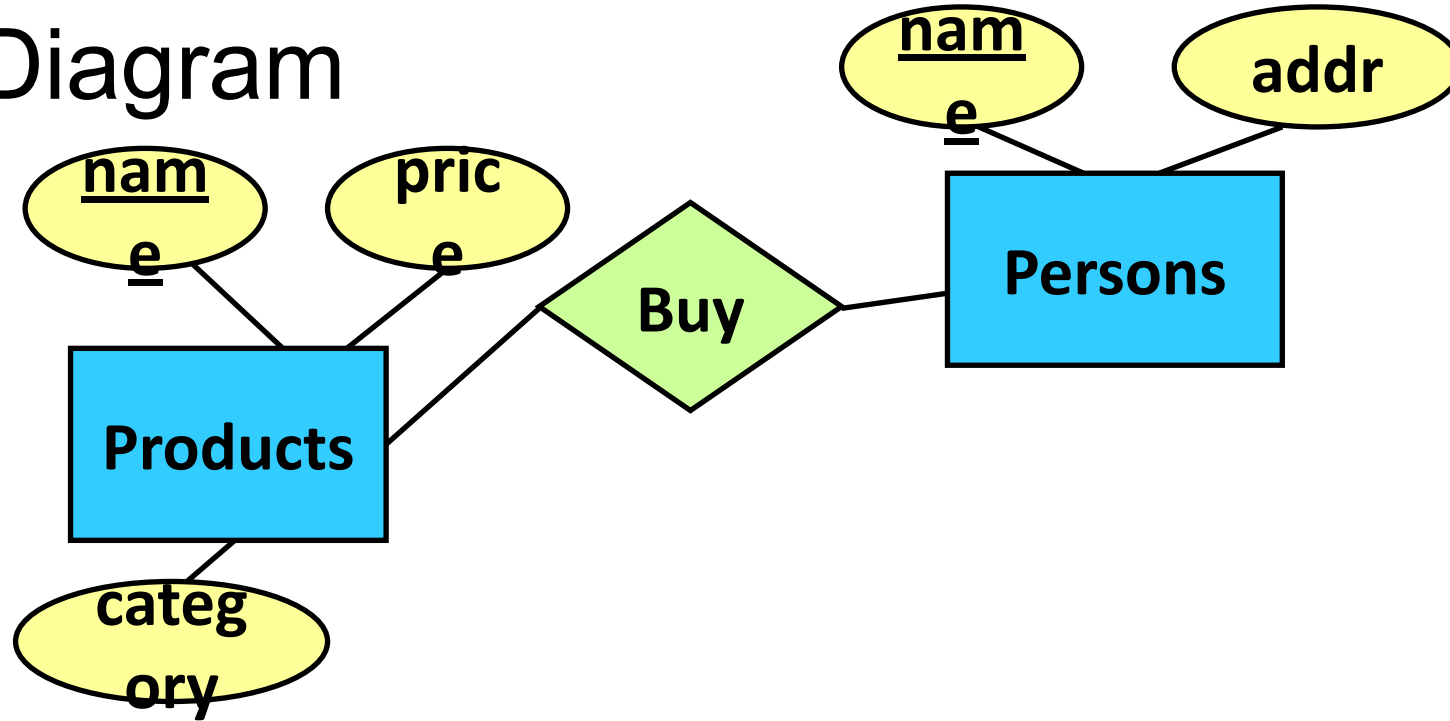
- Oval = **Attribute** = Property of an entity set

ER Diagram



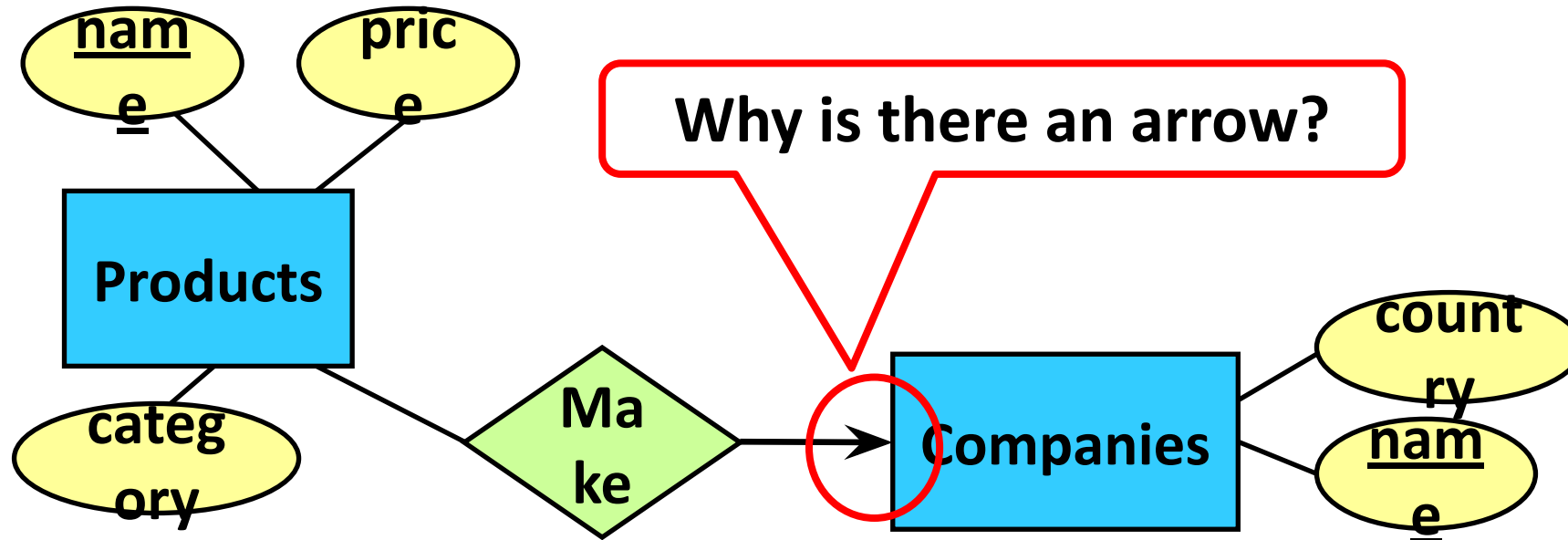
- Diamond = **Relationship** = Connection between two entity sets

ER Diagram



- Diamond = **Relationship** = Connection between two entity sets
- Persons buy products

ER Diagram

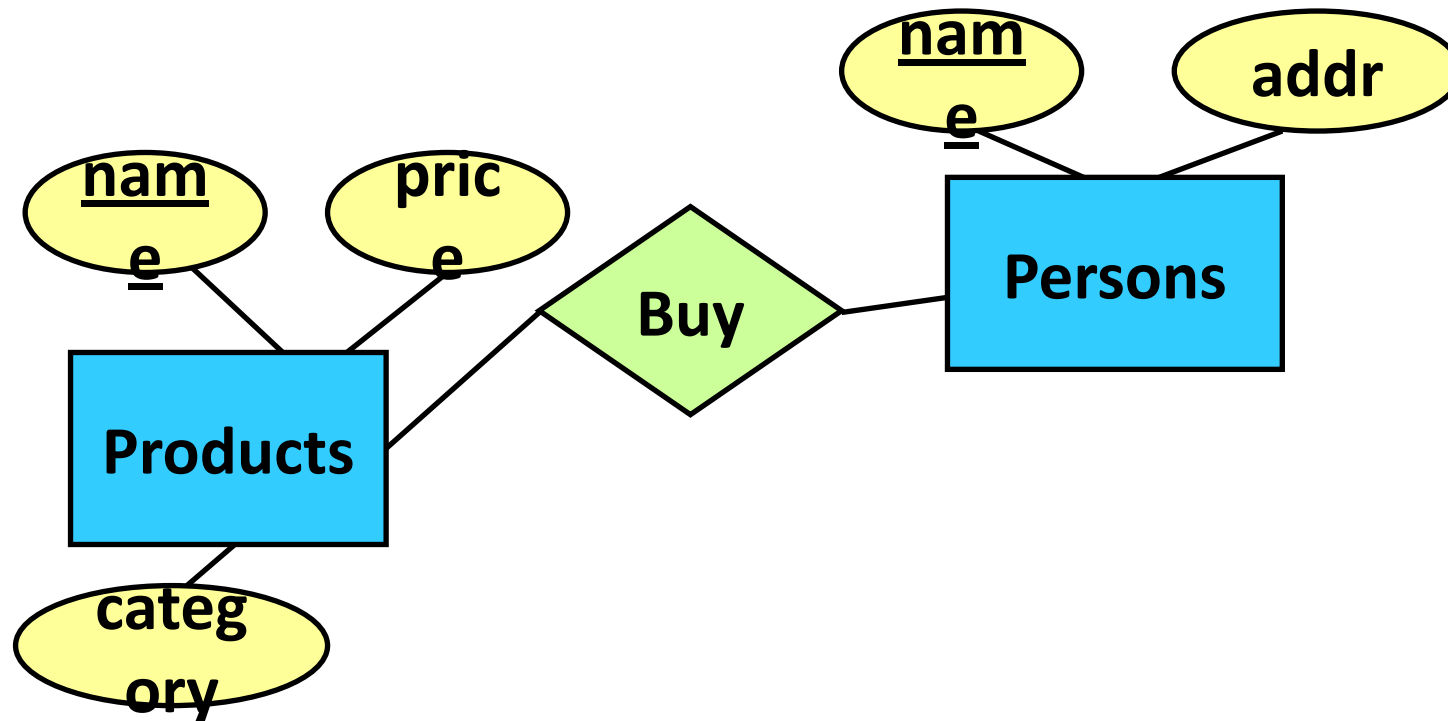


- Diamond = **Relationship** = Connection between two entity sets
- Companies make products

Types of Relationships

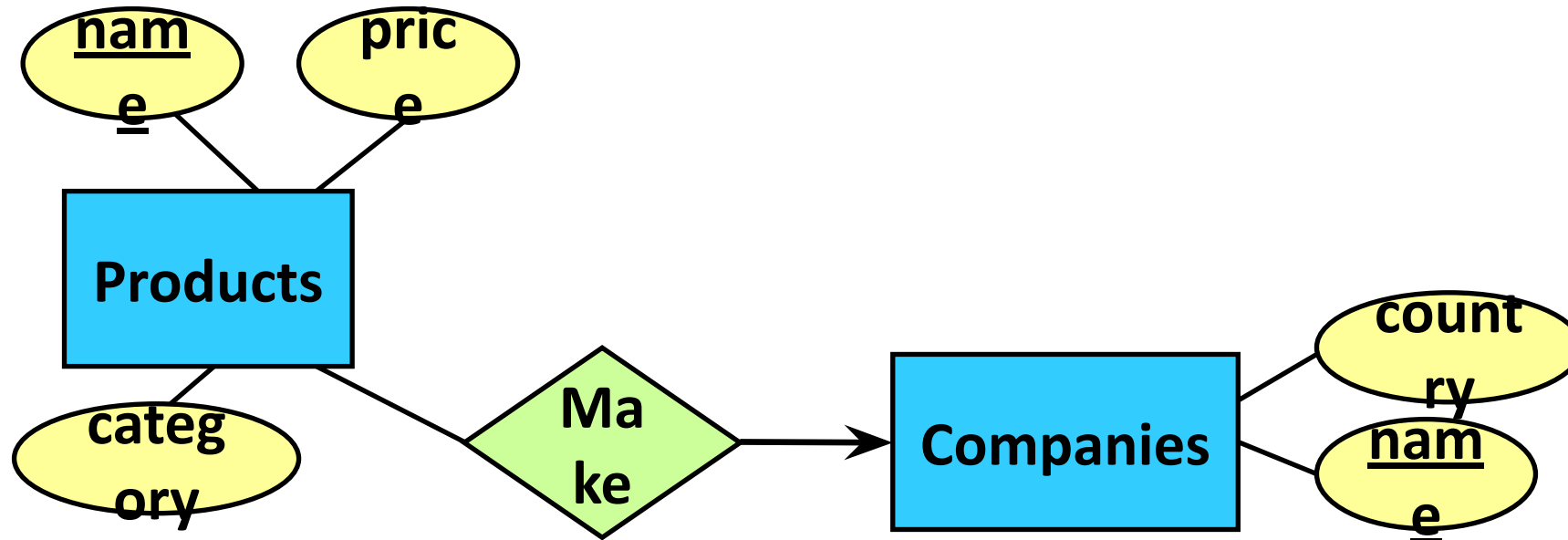
- Many-to-Many Relationships
- Many-to-One Relationships
- One-to-One Relationships

Many-to-Many Relationship



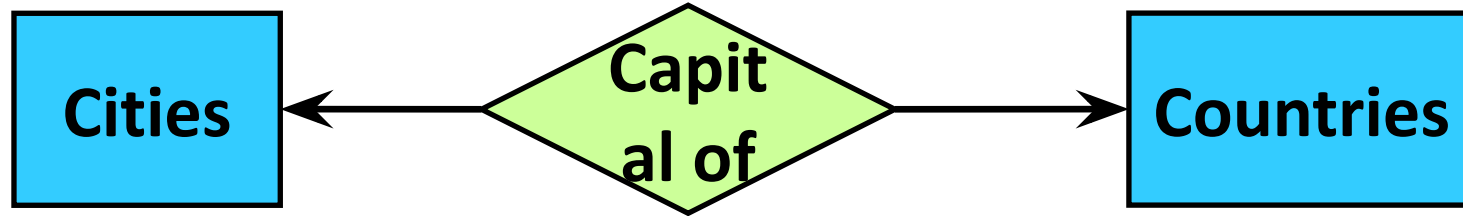
- One person can buy multiple products
- One product can be bought by multiple persons

Many-to-One Relationship



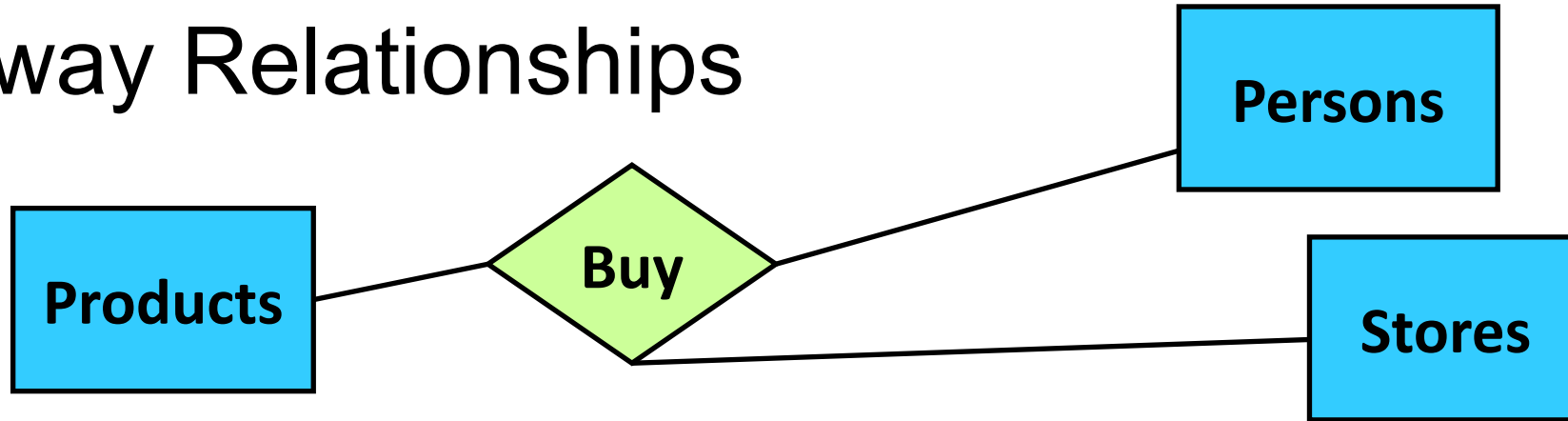
- One company can make multiple products
- But one product can only be made by one company (at most)

One-to-One Relationship



- A city can be the capital of only one country
- A country can have only one capital city

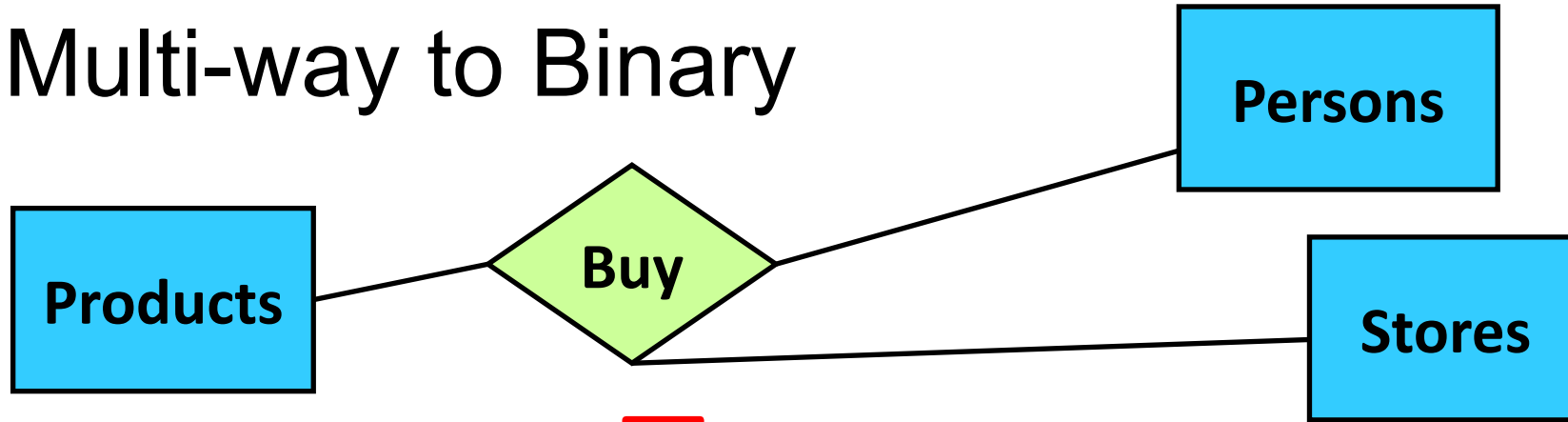
Multi-way Relationships



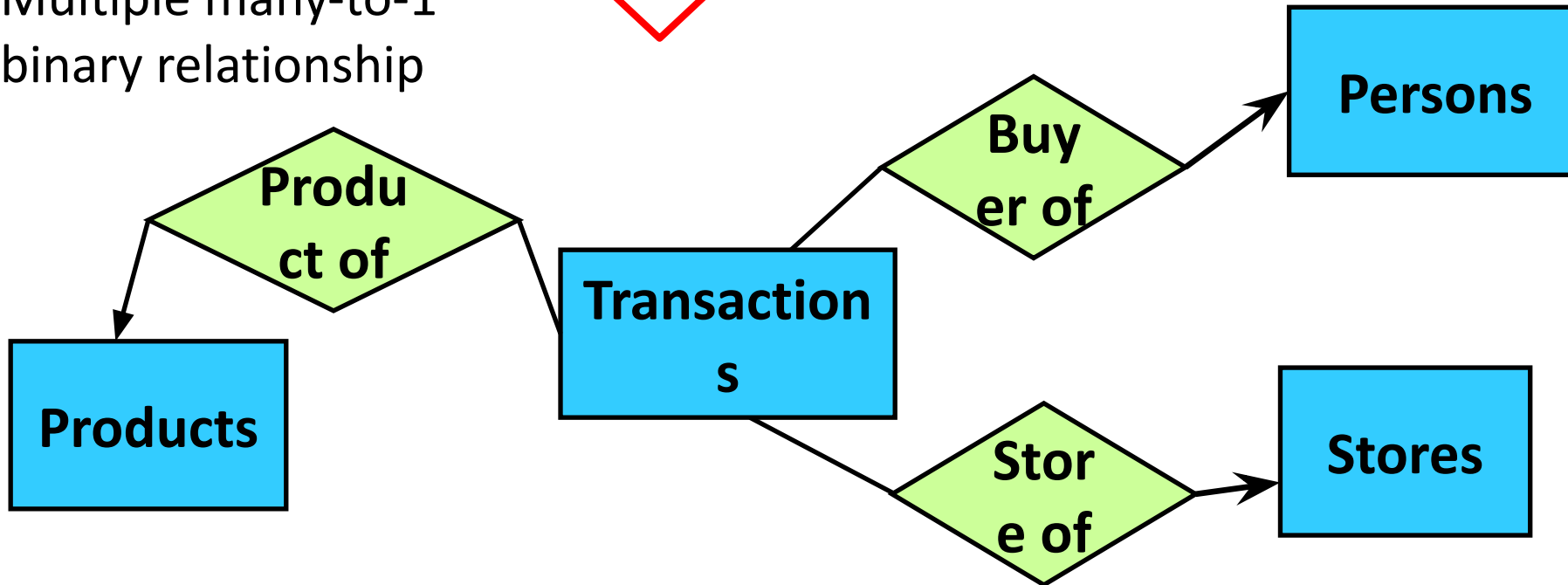
- What if we want to record the store from which the person buys the product?
- We can use a 3-way relationship

person	product	store
Ray	Milk Skim	S&S NTU
Ray	Milk Chocolate	S&S NTU
Ray	Milk Chocolate	NUS co-op
Peter	Milk Skim	S&S NTU

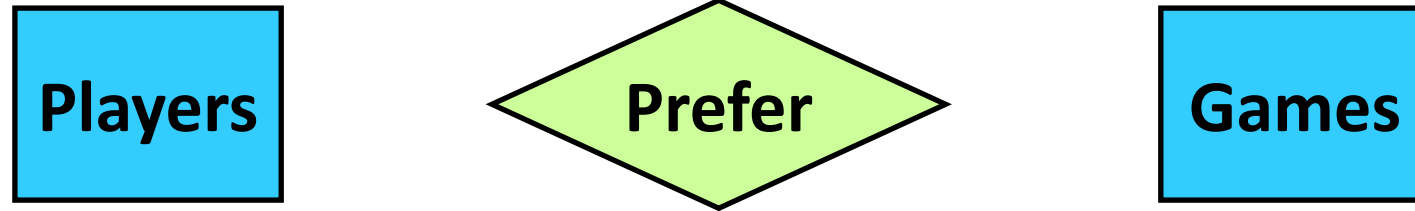
From Multi-way to Binary



Multiple many-to-1
binary relationship



Exercise



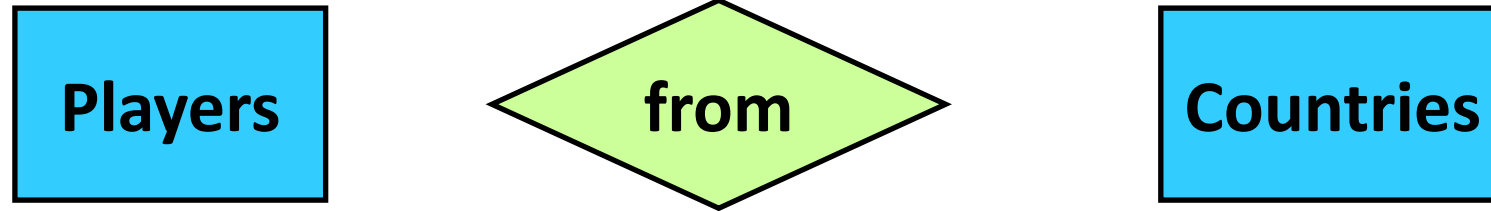
- Each player prefers only one game
 - One player can prefer one game
 - One game can be preferred by many players
 - P-G Many-to-One
- Many-to-many?
- Many-to-one?
- One-to-one?

Exercise



- No two shops sell the same product
 - One product can be sold by one shop
 - One shop can sell many products
 - P-G One-to-One
- Many-to-many?
- Many-to-one?
- One-to-one?

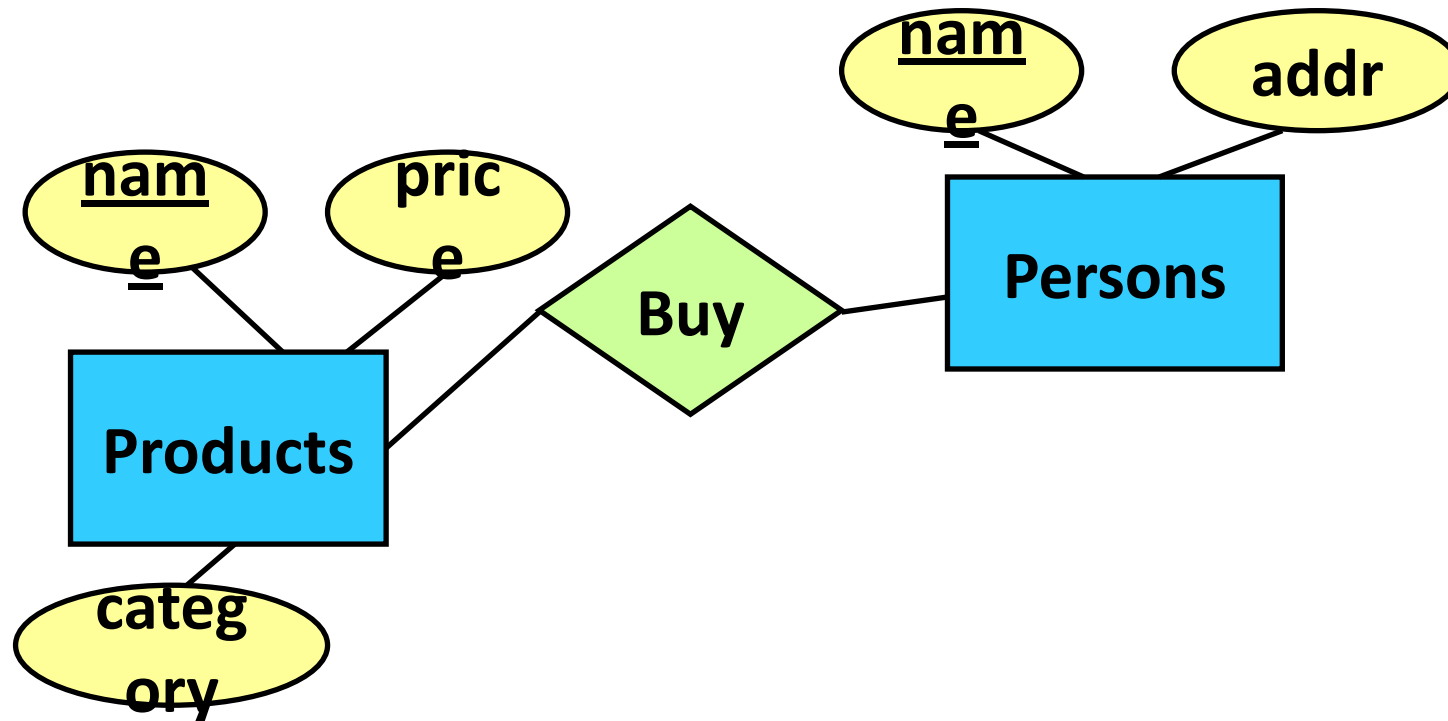
Exercise



- Any two players are from two different countries
 - One player can be from one country
 - One country can have one player only
 - P-G One-to-One
- Many-to-many?
- Many-to-one?
- One-to-one?

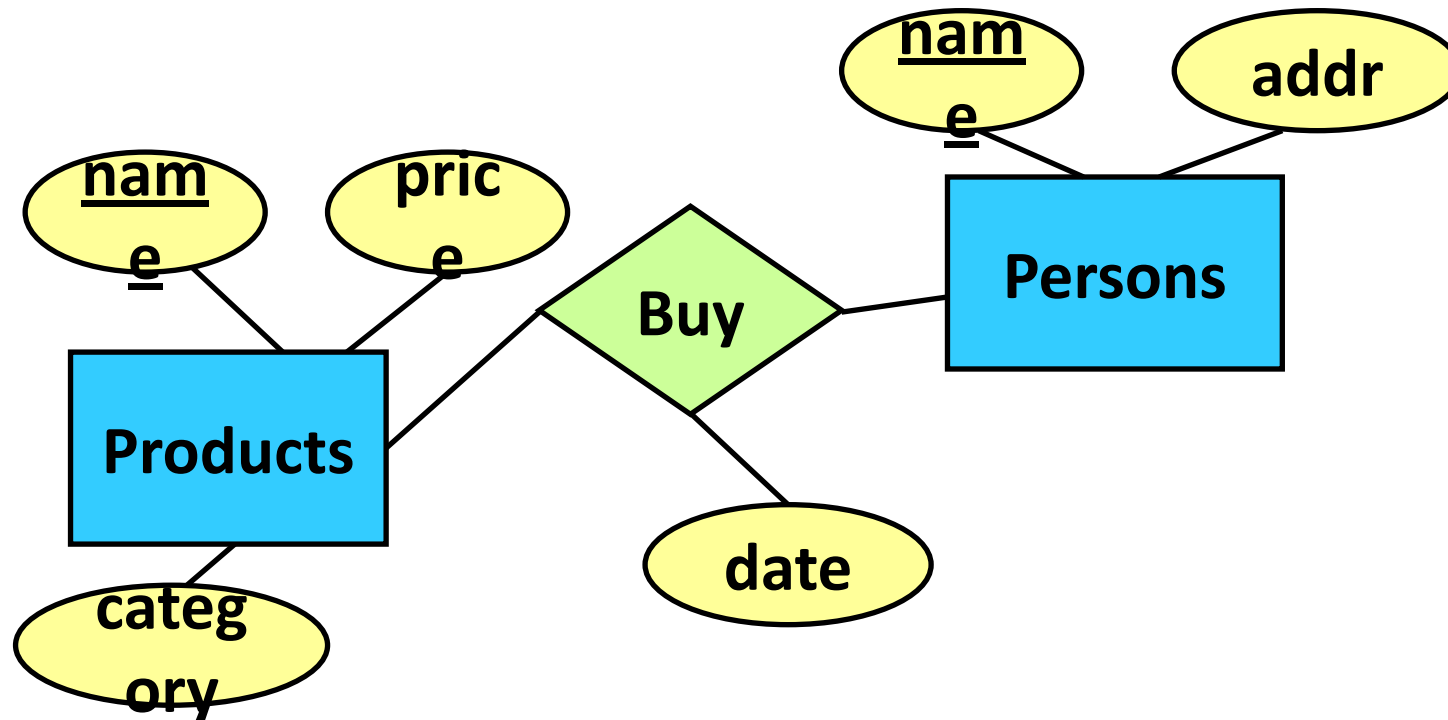
One More Thing about Relationships

- A relationship can have its own attribute

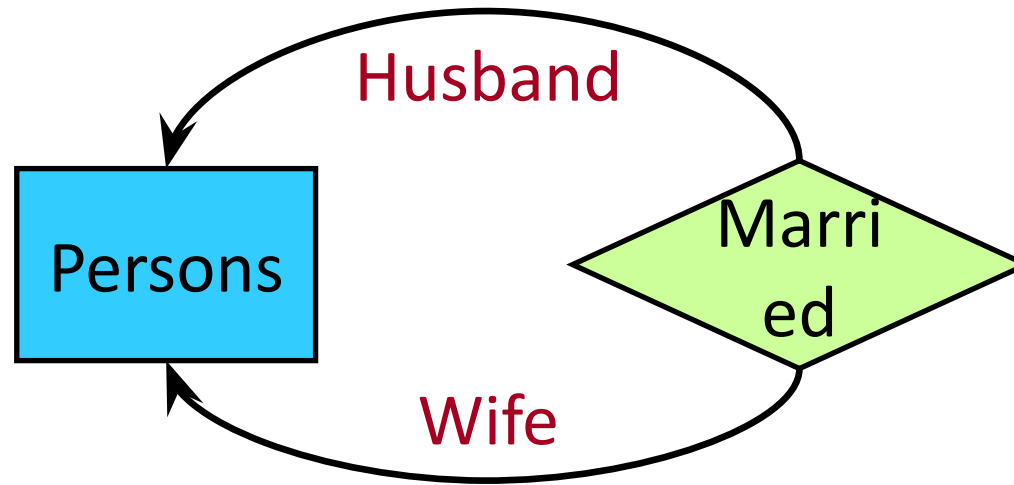


One More Thing about Relationships

- A relationship can have its own attribute
- If we want to record the date of the purchase



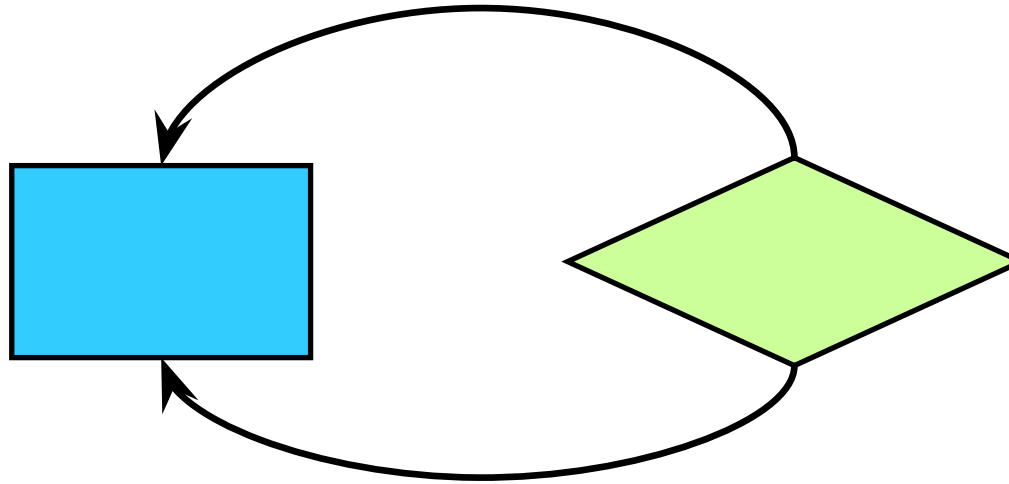
Roles



- Sometimes an entity set may appear more than once in a relationship
- Example: some persons are married to each other
- The role of the person is specified on the edge connecting the entity set to the relationship

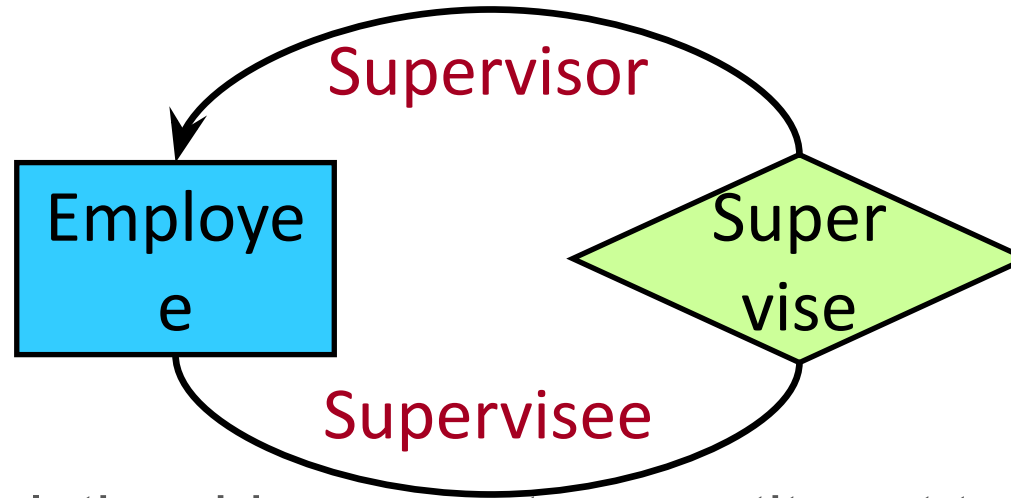
Husband	Wife
Bob	Alice
David	Cathy
...	...

Roles



- Question: A relationship connects an entity set to itself, does it have to be one-to-one?
- Answer: No

Roles



- Question: A relationship connects an entity set to itself, does it have to be one-to-one?
- Answer: No
- Example above:
 - One employee has only one supervisor, but may have many supervisee

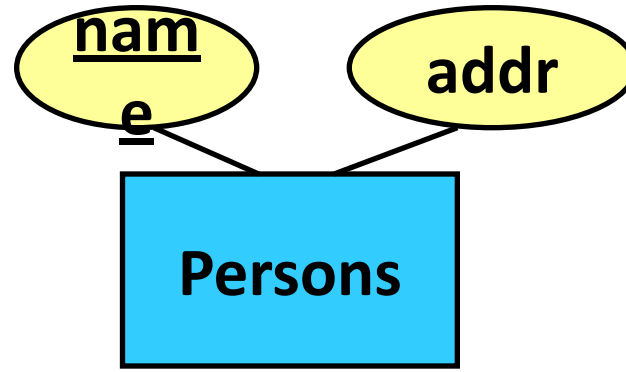
Roadmap

- Constraints
- Subclasses
- Weak Entity Sets
- ER Design Principle
- Translating an ER Diagram into a Relational Scheme Design

Constraints

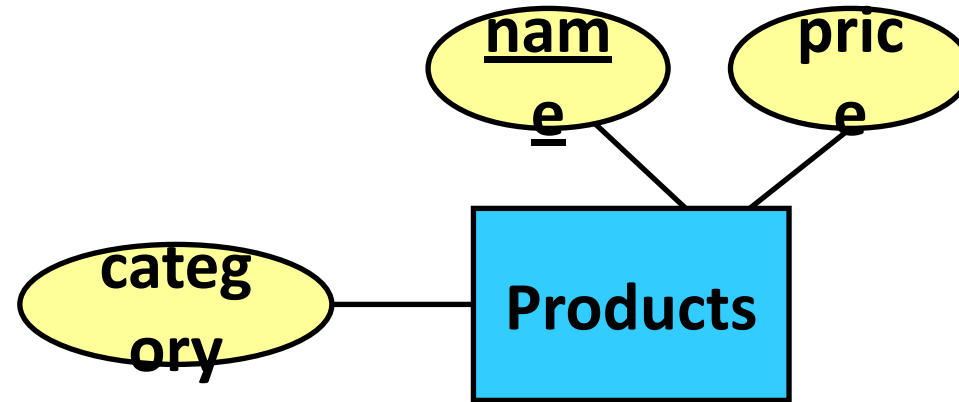
- Some conditions that entity sets and relationships should satisfy
- We will focus on three types of constraints
 - Key constraints
 - Referential integrity constraints
 - Degree constraints

Key



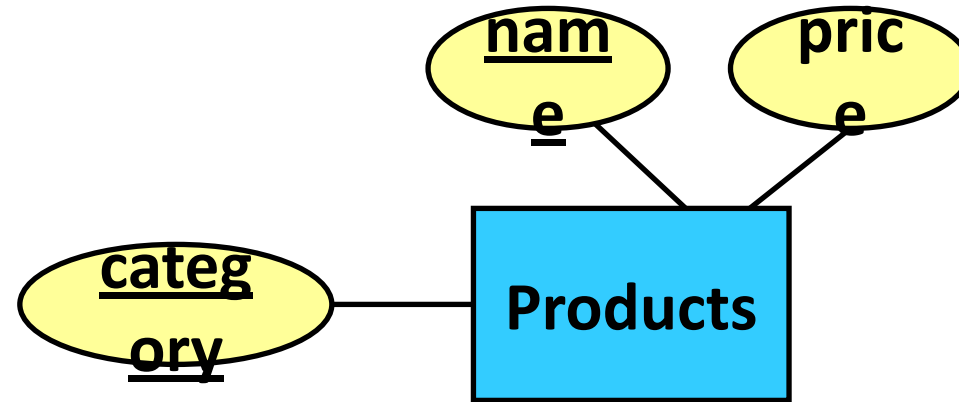
- One or more attributes that are underlined
- Meaning: They uniquely represent each entity in the entity set
- Example: The names uniquely represent the persons
- i.e., each person must have a unique name

Key



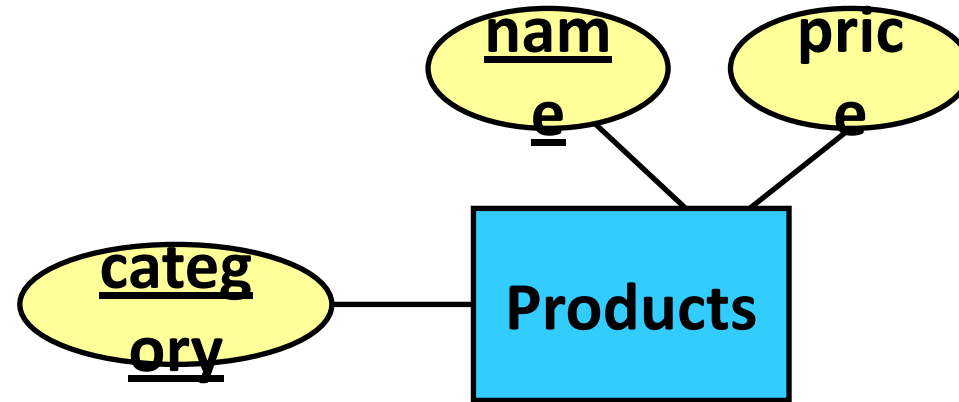
- One or more attributes that are underlined
- Meaning: They uniquely represent each entity in the entity set
- Example: Each product has a unique name

Key



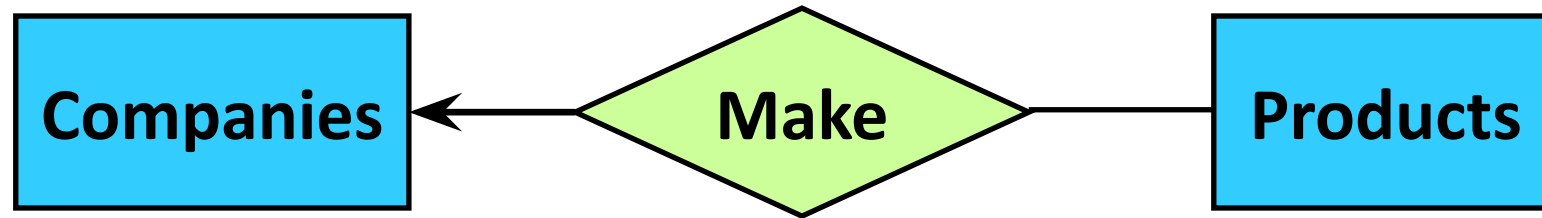
- One or more attributes that are underlined
- What now?
- Each product has a unique <name, category> combination
- But there can be products with the same name, or the same category, but not both
- Example
 - Name = "Apple", Category = "Fruit", Price = "1"
 - Name = "Apple", Category = "Phone", Price = "888"

Key



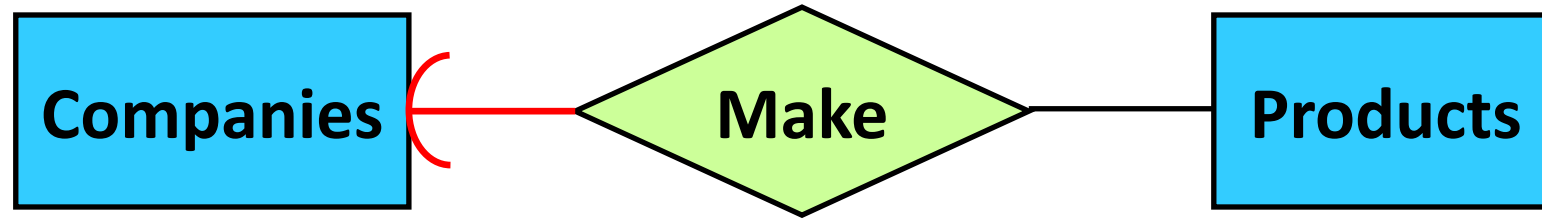
- Rule: Every entity set should have a key
 - So that we can uniquely refer to each entity in the entity set

Referential Integrity



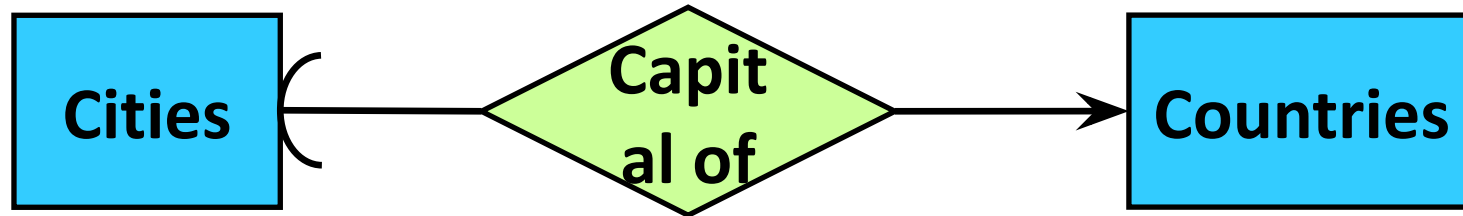
- One company may make multiple products
- One product is made by one company
- Can there be a product that is not made by any company?
- No.
- i.e., every product must be involved in the Make relationship
- This is called a referential integrity constraint.
- How do we specify this in an ER diagram?
- Use a rounded arrow instead of a pointed arrow

Referential Integrity



- One company may make multiple products
- One product is made by one company
- Can there be a product that is not made by any company?
- No.
- i.e., every product must be involved in the Make relationship
- This is called a referential integrity constraint.
- How do we specify this in an ER diagram?

Referential Integrity: Exercise



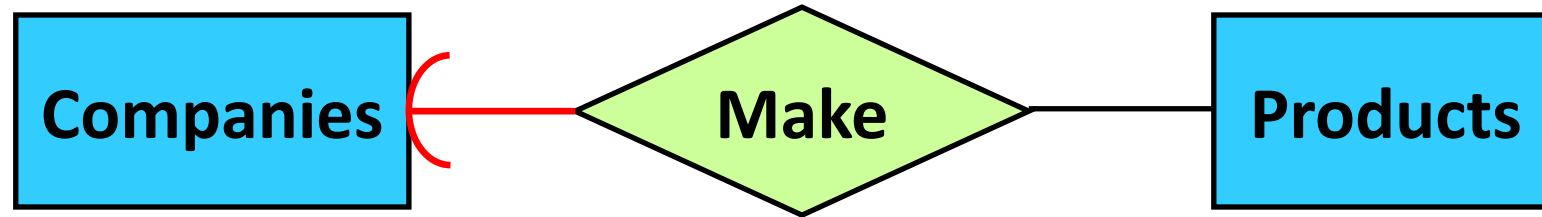
- A city can be the capital of only one country
- A country must have a capital

Referential Integrity: Exercise



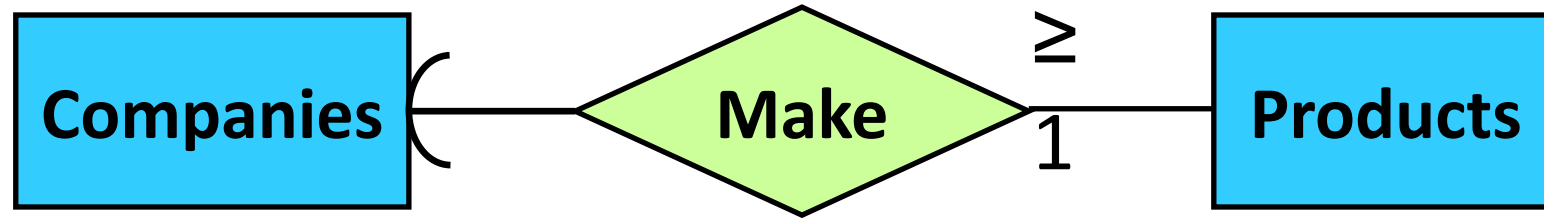
- A company must hire at least one person
- A person must be hired by exactly one company

Referential Integrity



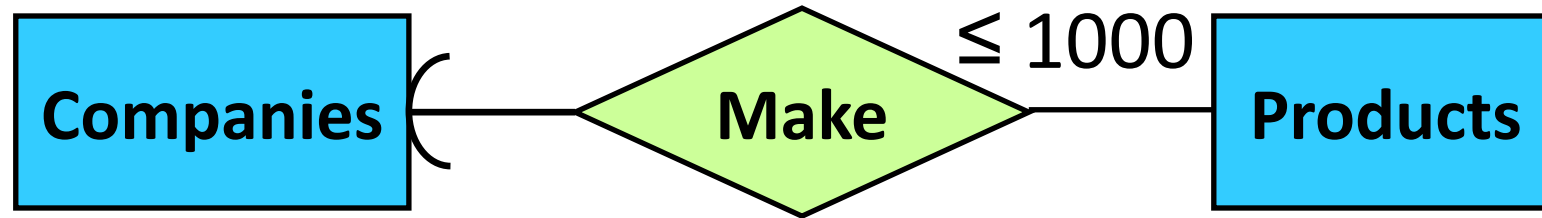
- What if every company should make at least one product?
- In general, a referential integrity constraint can only apply to the “one” side of
 - A many-to-one relationship, or
 - A one-to-one relationship
- For the “many” side, there is another type of constraints to use

Degree Constraint



- Each company should make at least 1 product

Degree Constraint

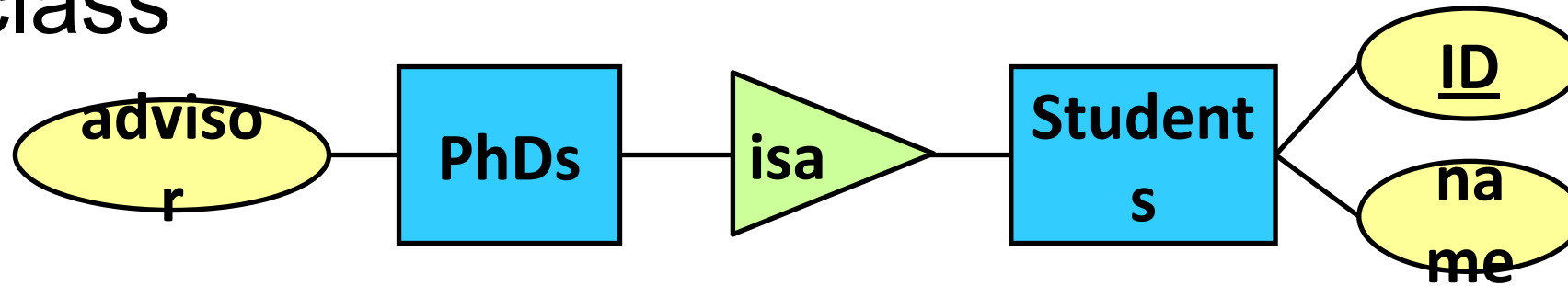


- Each company can make at most 1000 product
- Note
 - Not required in the exam
 - Key and referential integrity constraints can be easily enforced in a DBMS
 - Degree constraints are not easy to enforce

Roadmap

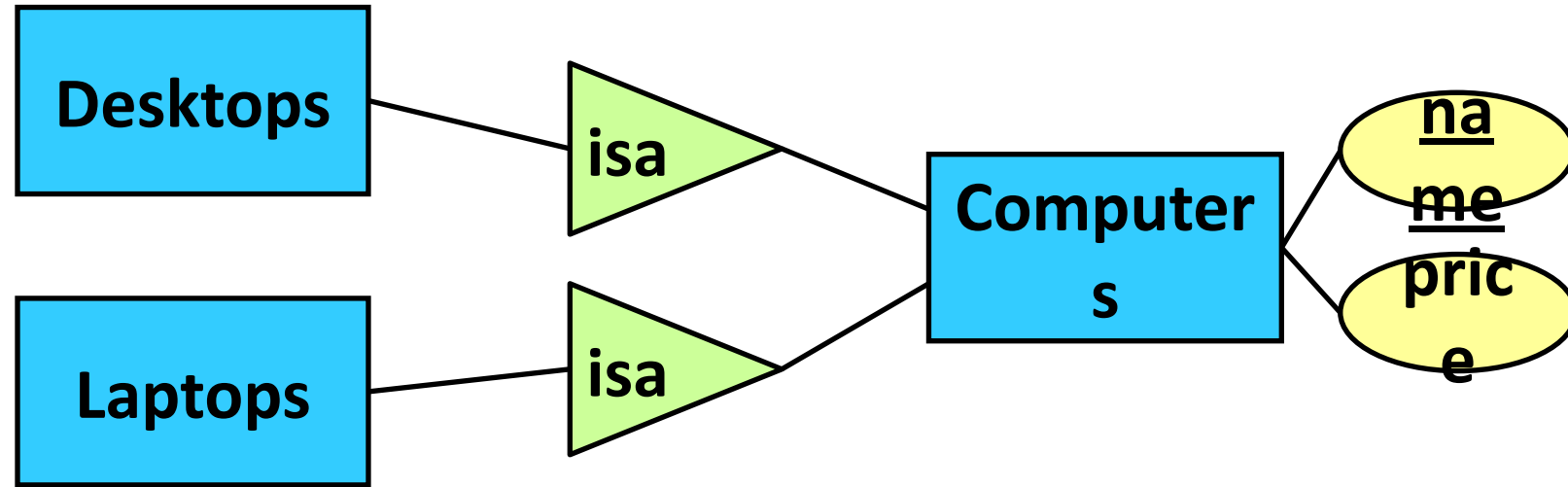
- Constraints
- Subclasses
- Weak Entity Sets
- ER Design Principle
- Translating an ER Diagram into a Relational Scheme Design

Subclass



- PhDs are a special type of Students
- **Subclass** = Special type
- The connection between a subclass and its superclass is captured by the **isa relationship**, which is represented using a triangle
- Key of a subclass = key of its superclass
- Example: Key of Phds = Students.ID
- Students is referred to as the **superclass** of PhDs

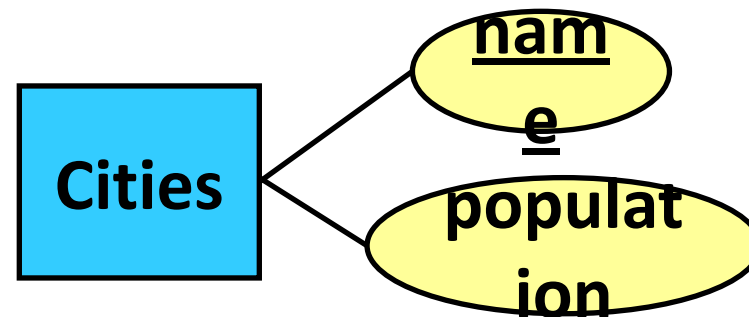
Subclass



- An entity set can have multiple subclasses
- Example
 - Superclass: Computers
 - Subclass 1: Desktop
 - Subclass 2: Laptop

Weak Entity Sets

- Weak entity sets are a special type of entity sets that
 - cannot be uniquely identified by their own attributes
 - needs attributes from other entities to identify themselves
- Example: Cities in USA
- Problem: there are cities with identical names



Madison

From Wikipedia, the free encyclopedia

Madison may refer to:

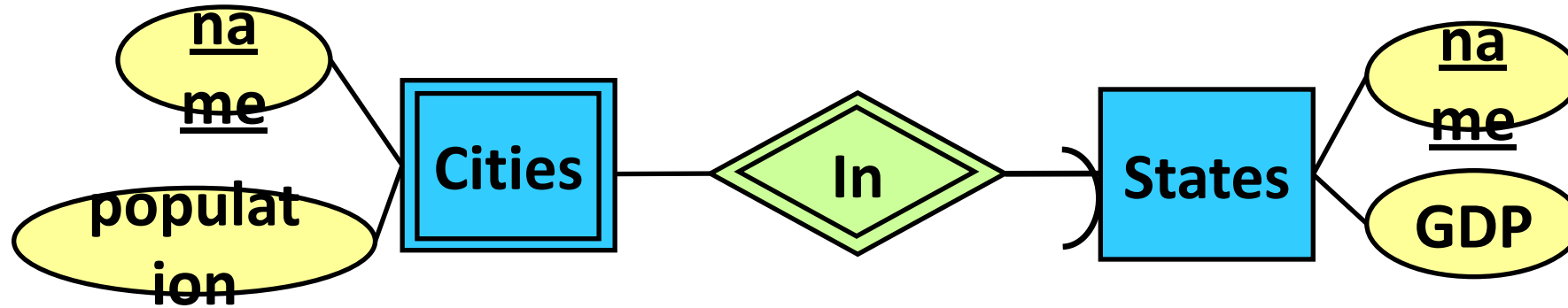
People

- [Madison \(name\)](#), a given name and a surname

Place names

- [Madison, Wisconsin](#), the largest city by the name and the state capital of Wisconsin
- [Madison, Alabama](#)
- [Madison, Arkansas](#)
- [Madison, California](#)
- [Madison, Connecticut](#)
- [Madison, Florida](#)
- [Madison, Georgia](#)
- [Madison, Illinois](#)
- [Madison, Indiana](#)
- [Madison, Kansas](#)
- [Madison, Maine](#)
 - [Madison \(CDP\), Maine](#), census-designated place within the town of Madison
- [Madison, Minnesota](#)
- [Madison, Mississippi](#)
- [Madison, Missouri](#)
- [Madison, Nebraska](#)
- [Madison, New Hampshire](#)
- [Madison, New Jersey](#)
- [Madison \(town\), New York](#)
 - [Madison \(village\), New York](#), within the town of Madison
- [Madison, North Carolina](#)
- [Madison, Ohio](#)
- [Madison, Pennsylvania](#)
- [Madison, South Dakota](#)
- [Madison, Tennessee](#)
- [Madison, Virginia](#)
- [Madison, West Virginia](#)
- [Madison \(town\), Wisconsin](#), adjacent to the city of Madison
- [Madison Lake, Minnesota](#)
- [Madison Park, Seattle, Washington State](#)

Weak Entity Sets



- Problem: there are cities with identical names
- Observation: cities in the same state would have different names
- Solution: make Cities a **weak entity set** associated with the entity set States : **Double-lined rectangle**
- The relationship In is called the **supporting relationship** of Cities : **Double-lined diamond**
- The key of Cities = (State.name, Cities.name)
 - Zero or more of its own attributes
 - Key attributes from entity sets that are reached by **supporting relationships** to other entity sets

Exercise

- Consider two entity sets: Players and Teams
- Each player has a name and a number
- Each team has a name and a manager
- Each player plays for exactly one team, and is uniquely identified within the team by his/her number
- Each team is uniquely identified by its name
- Different players may have the same name
- Draw a ER diagram that captures the above statements
- What is the key of Players?

Road Map

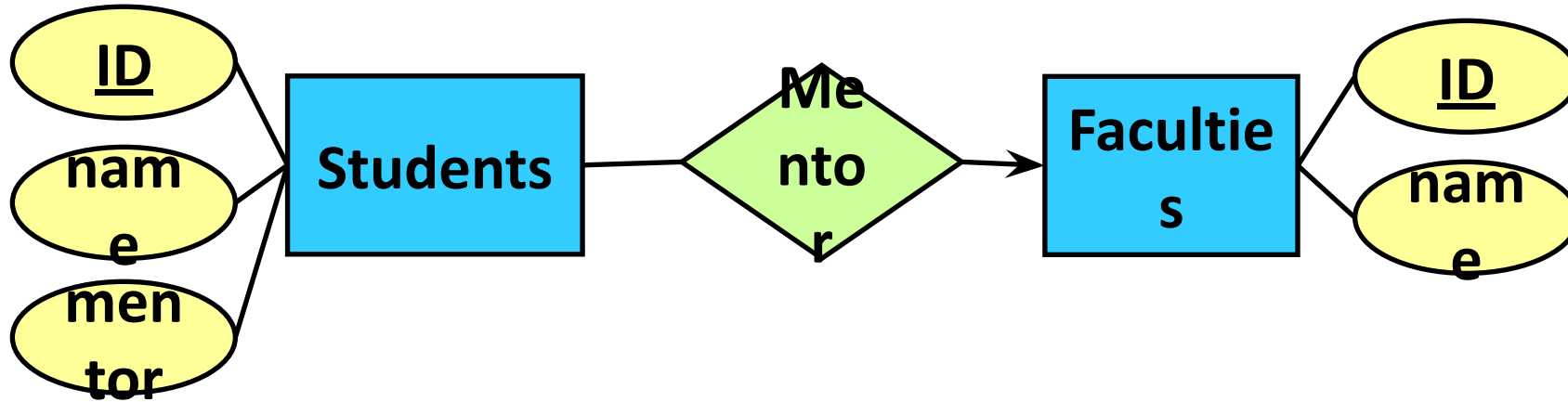
- Design Principle of ER Diagrams
- Translating an ER Diagram into a Relational Scheme Design

Design Principle 1: Be Faithful

- Be faithful to the specifications of the application
- Capture the requirements as much as possible

Design Principle 2: Avoid Redundancy

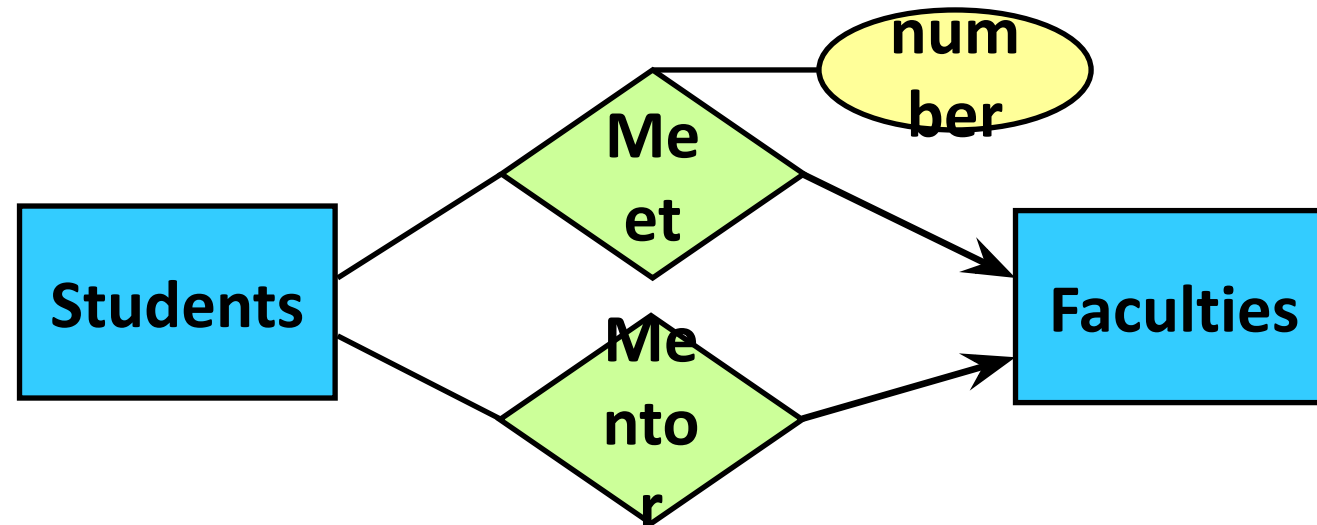
- Avoid repetition of information
- Example



- Problems that can be caused by redundancy
 - Waste of space
 - Possible inconsistency

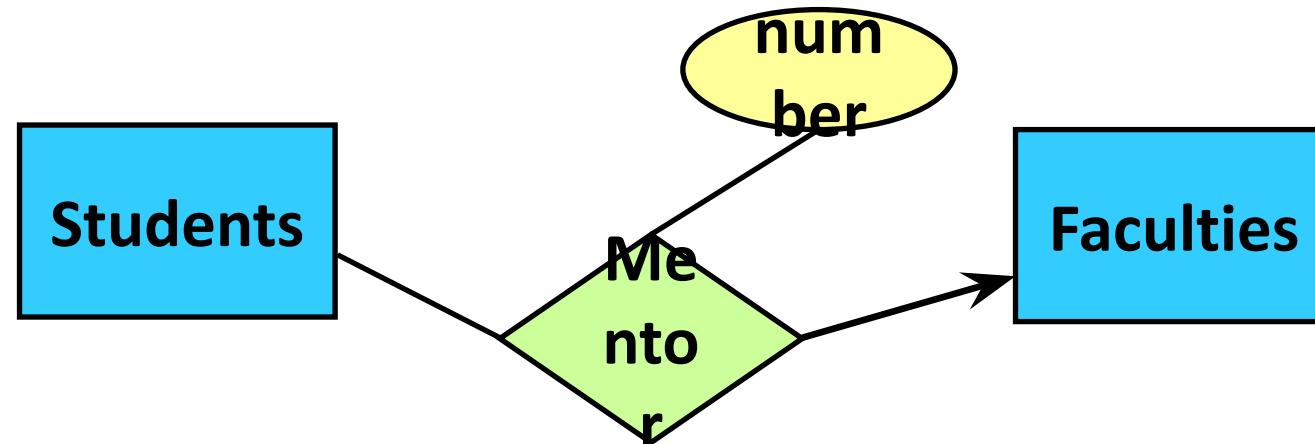
Design Principle 3: Keep It Simple

- Each student is mentored by one faculty
- One faculty can mentor multiple students
- We also record the number of times that a mentee meets with his/her mentor
- Design below: Not wrong, but can be simplified



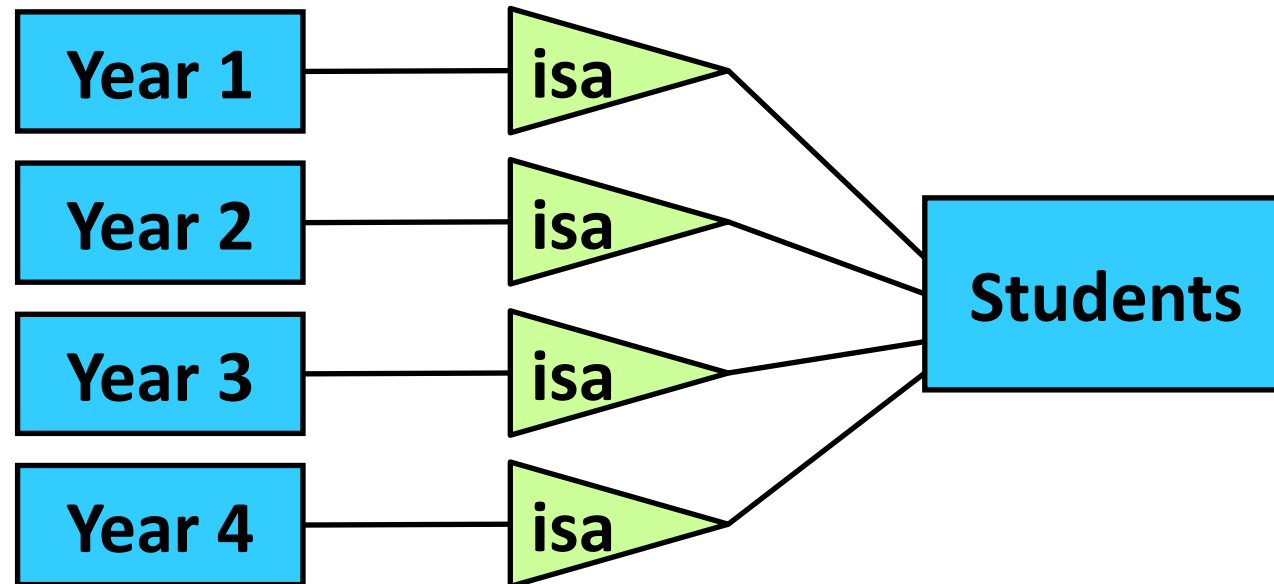
Design Principle 3: Keep It Simple

- Each student is mentored by one faculty
- One faculty can mentor multiple students
- We also record the number of times that a mentee meets with his/her mentor
- Better Design:



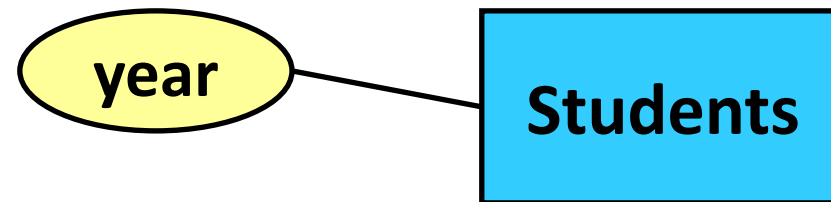
Design Principle 3: Keep It Simple

- There are four types of students: Year 1, Year 2, Year 3, Year 4
- Design below: Not wrong, but can be simplified



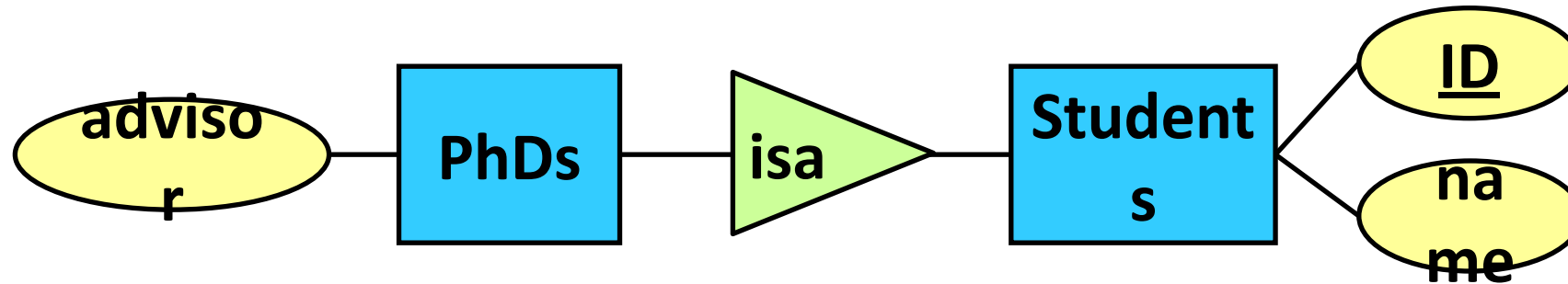
Design Principle 3: Keep It Simple

- There are four types of students: Year 1, Year 2, Year 3, Year 4
- Better Design

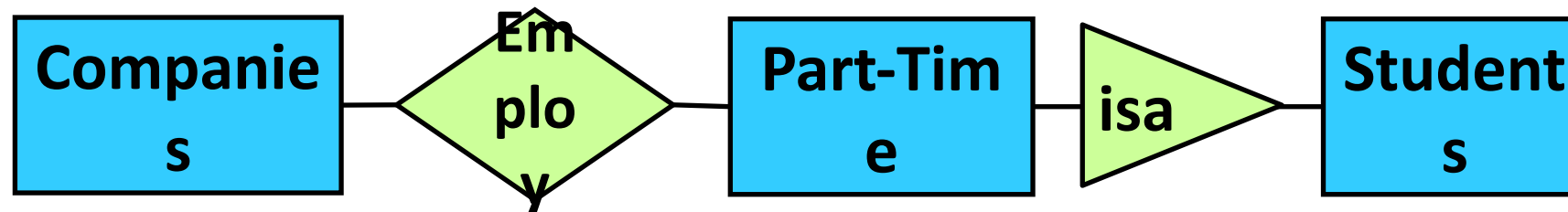


Tips: When to Use Subclasses

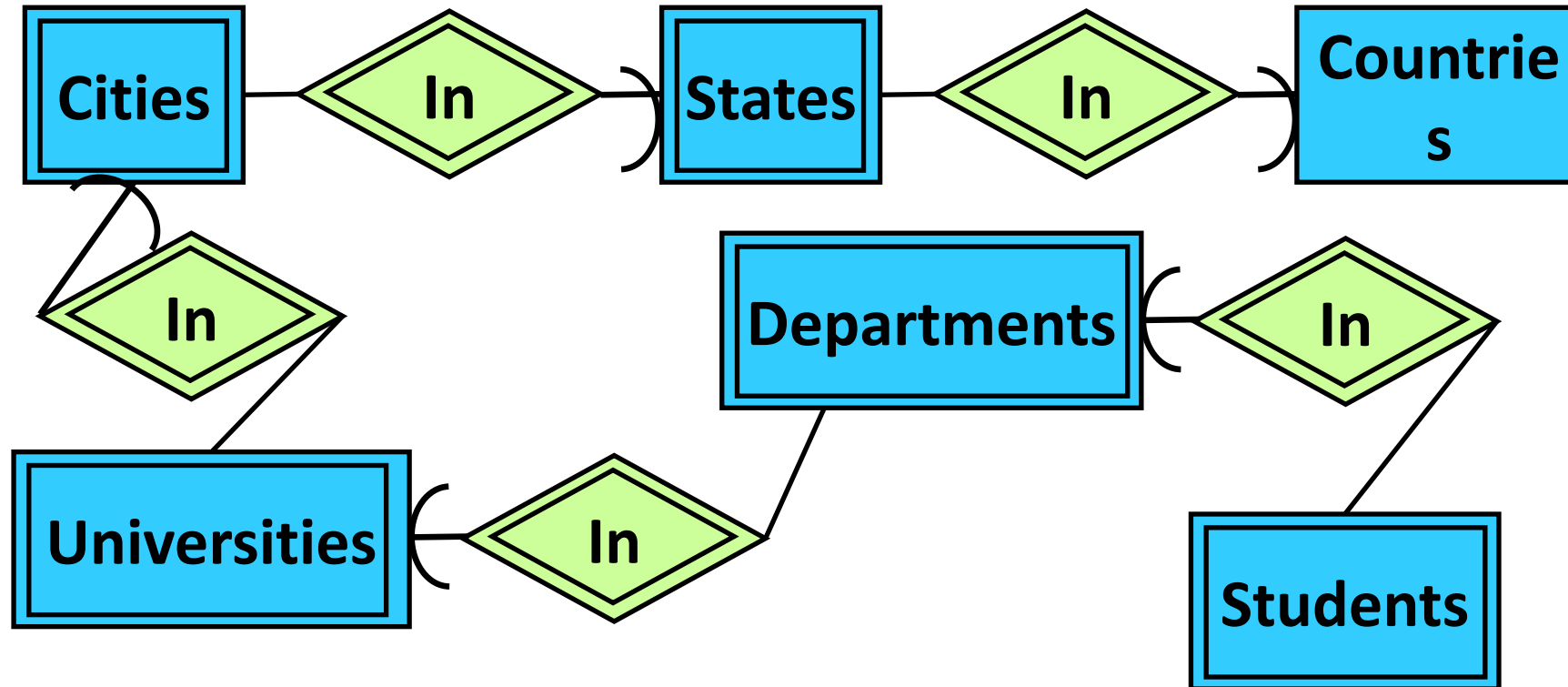
- Case 1: When a subclass has some attribute that is absent from the superclass



- Case 2: When a subclass has its own relationship with some other entity sets



Design Principle 4: Don't Over-use Weak Entity Sets

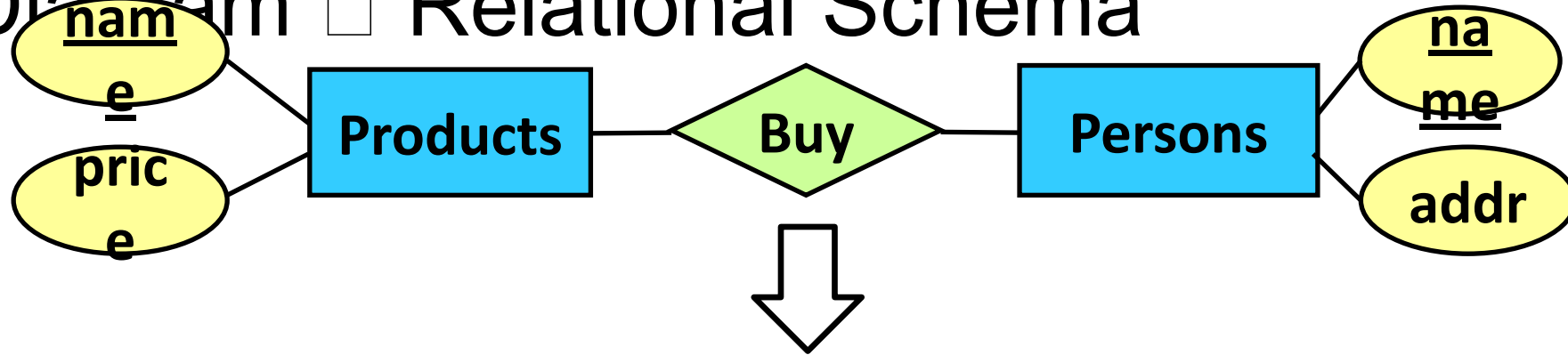


- Too many entity sets that should not be “weak”

Road Map

- Design Principle of ER Diagrams
- Translating an ER Diagram into a Relational Scheme Design

ER Diagram \square Relational Schema

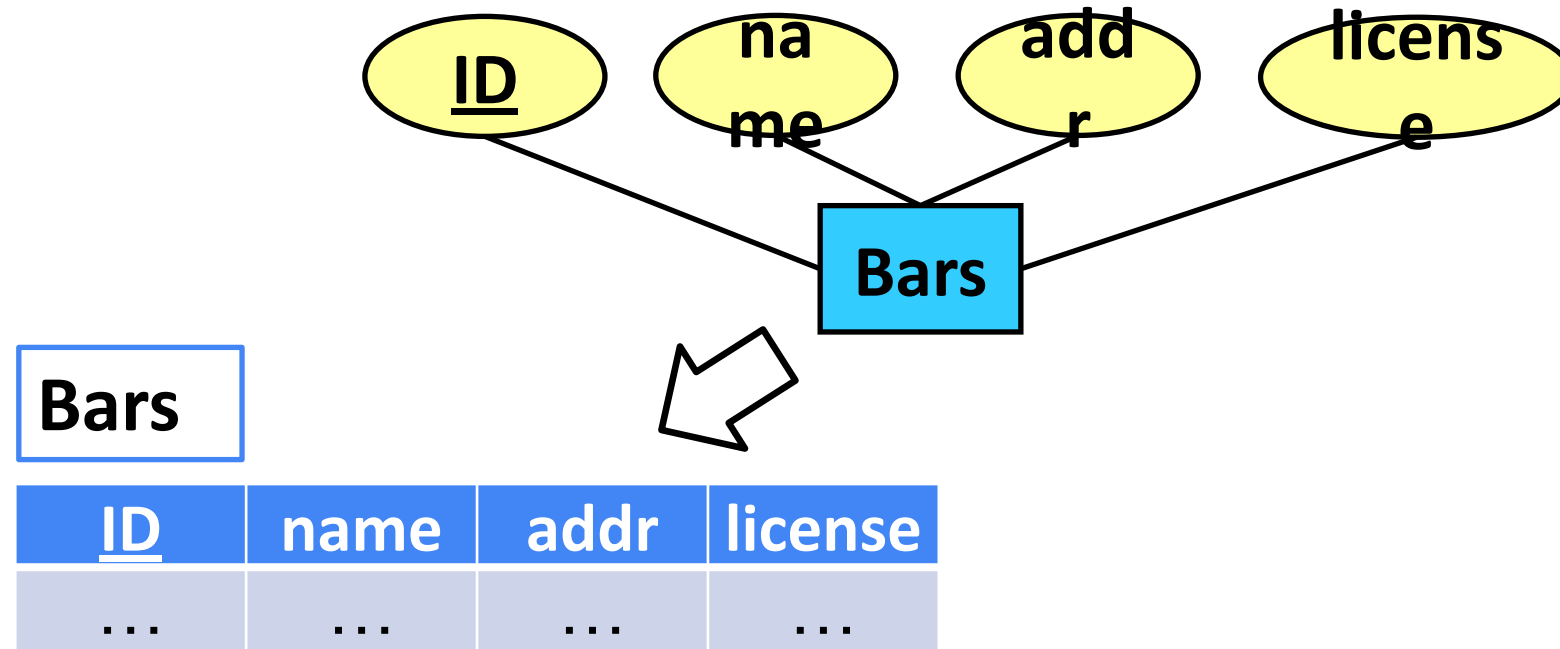


- Products (name, price)
- Persons (name, addr)
- Buy (product_name, person_name)
- Terminology
 - A **relation schema** = the name of a table + names of its attributes
 - A **database schema** = a set of relation schemas

ER Diagram \square Relational Schema

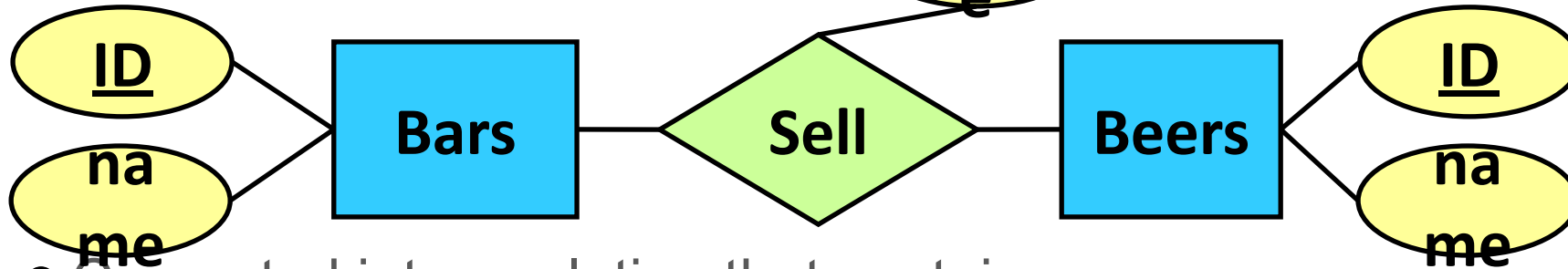
- General rules:
 - Each entity set becomes a relation
 - Each many-to-many relationship becomes a relation
- Special treatment needed for:
 - Weak entity sets
 - Subclasses
 - Many-to-one and one-to-one relationships

Entity Set \square Relation



- Each entity set is converted into a relation that contains all its attributes
- The key of the relation = the key of the entity set

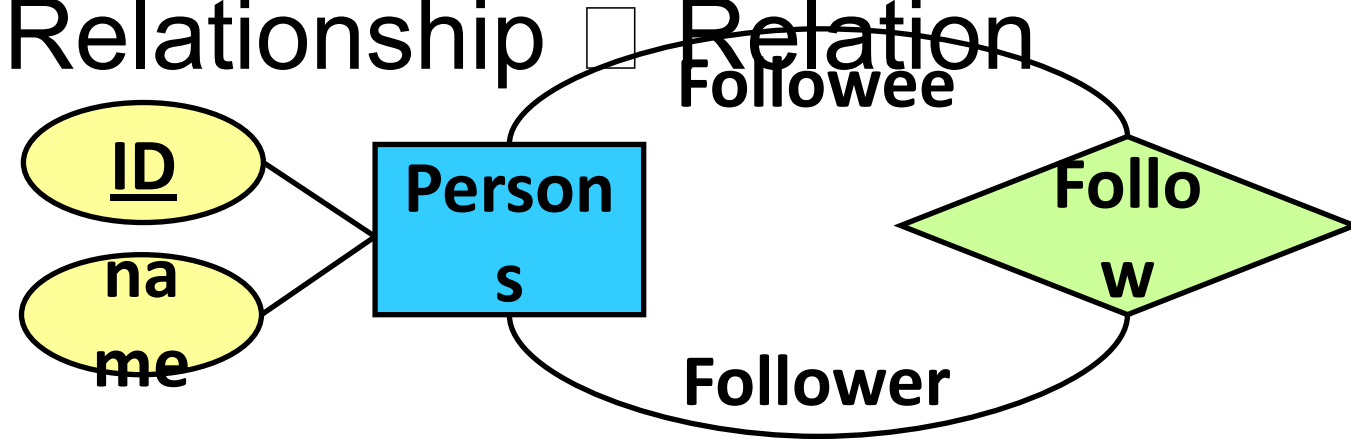
Many-to-Many Relationship Relation



- Converted into a relation that contains
 - all keys of the participating entity sets, and
 - the attributes of the relationship (if any)
- Normally, Key of relation = Keys of the participating entity sets

Sell	<u>Bars-ID</u>	<u>Beers-ID</u>	price

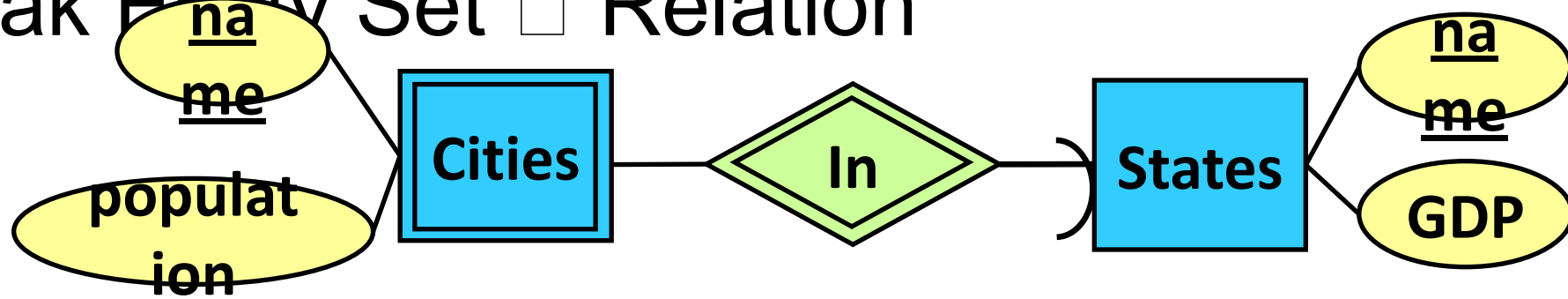
Many-to-Many Relationship



- If an entity is involved multiple times in a relationship
 - Its key will appear in the corresponding relation multiple times
 - The key is re-named according to the corresponding role

Follow	<u>Follower-ID</u>	<u>Followee-ID</u>

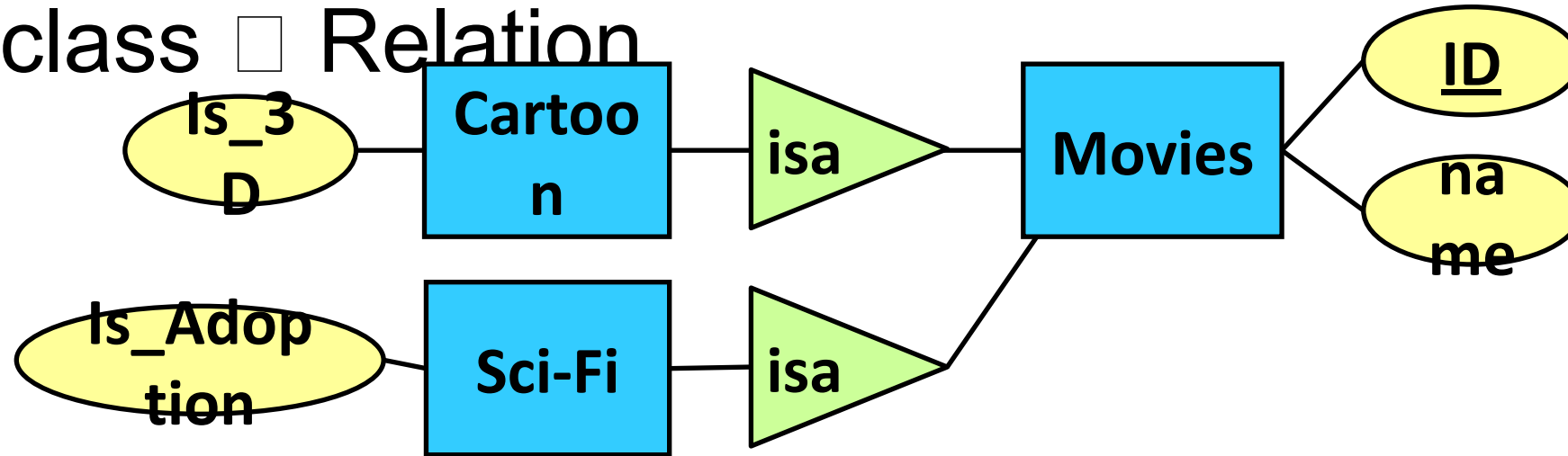
Weak Entity Set □ Relation



- Each weak entity set is converted to a relation that contains
 - all of its attributes, and
 - the key attributes of the supporting entity set
- The supporting relationship is ignored

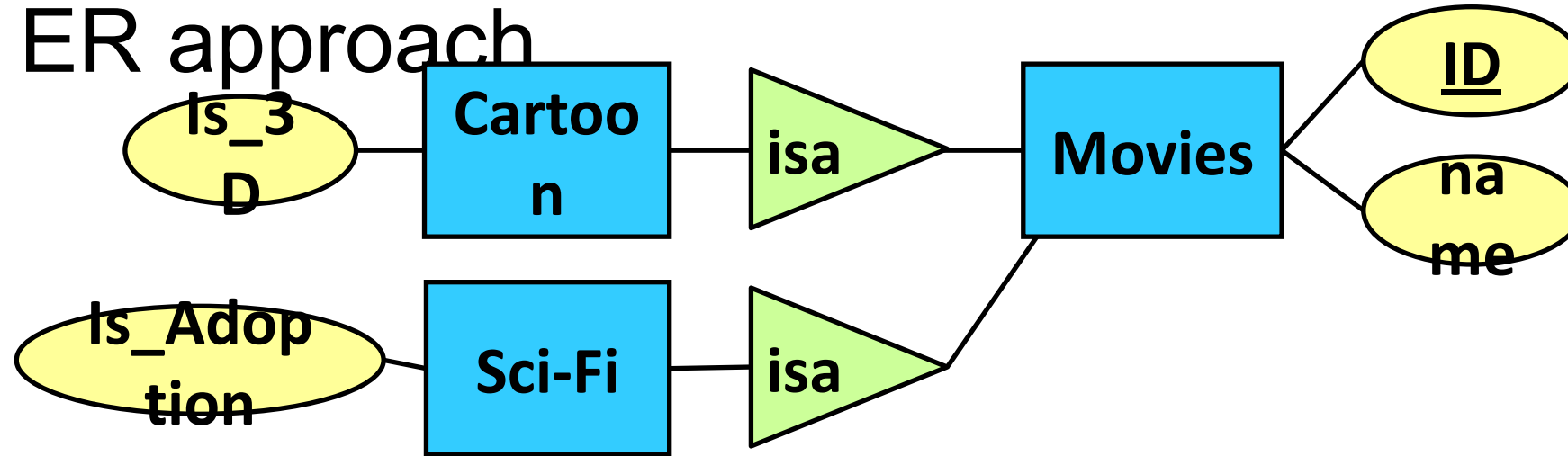
Cities	<u>state-name</u>	<u>city-name</u>	population

Subclass Relation



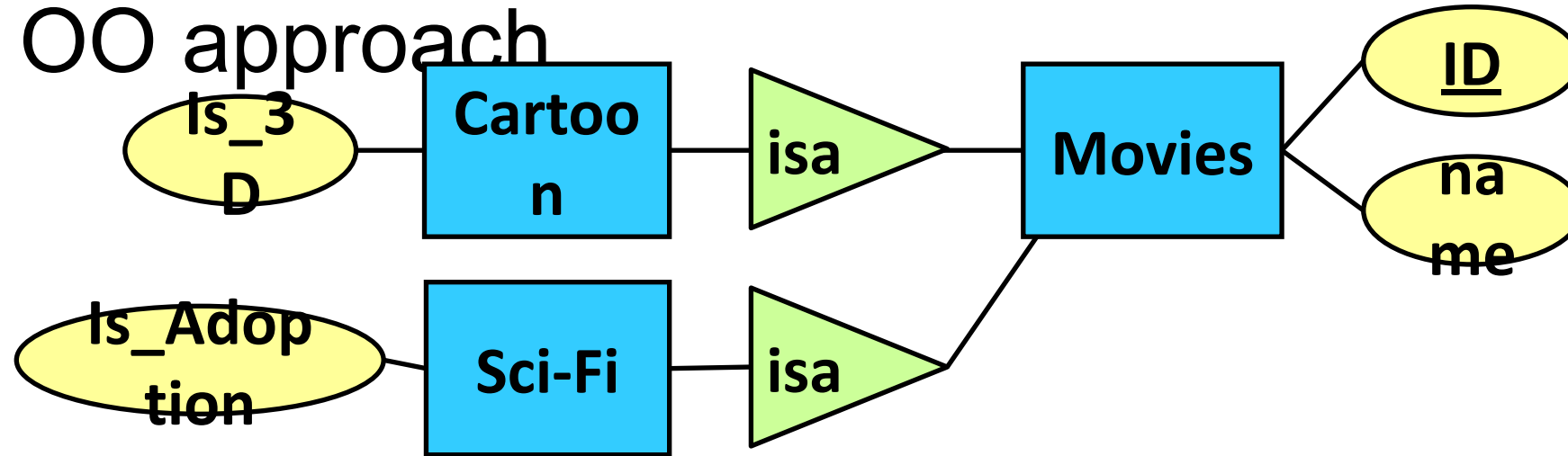
- There are three different ways
 - The ER approach
 - The OO approach
 - The NULL approach

The ER approach



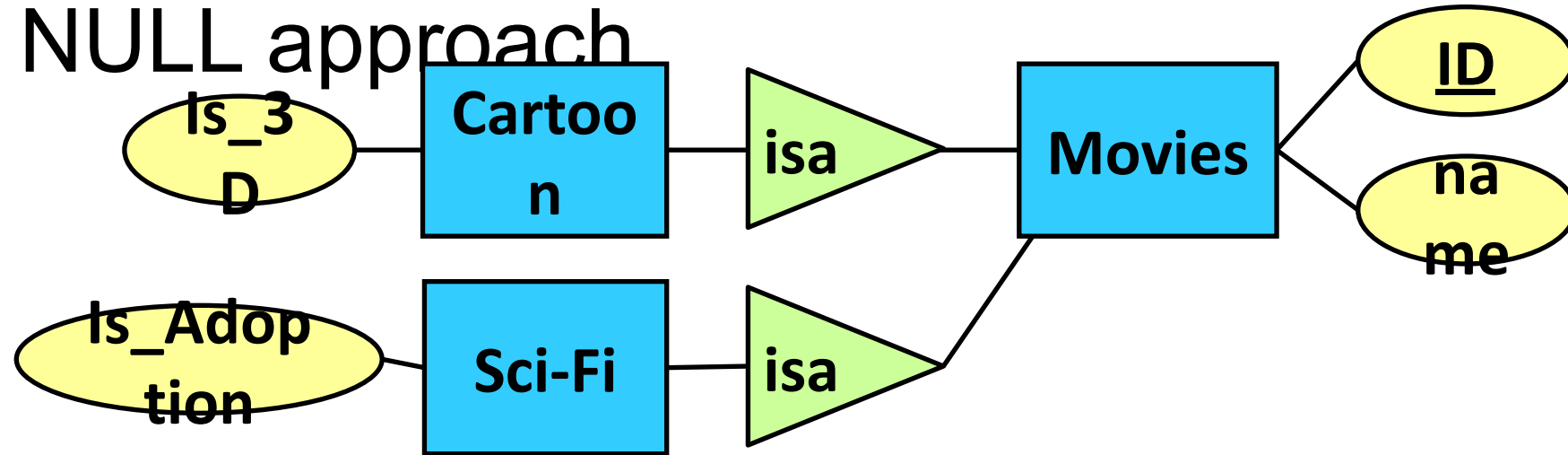
- One relation for each entity set
 - Movies(ID, name)
 - Cartoon(ID, Is_3D)
 - Sci-Fi(ID, Is_Adoption)
- A record may appear in multiple relations

The OO approach



- One relation for each entity set and each possible subclass combination
 - Movies(ID, name)
 - Cartoon(ID, name, Is_3D)
 - Sci-Fi(ID, name, Is_Adoption)
 - Sci-Fi-Cartoon(ID, name, Is_3D, Is_Adoption)
- Each record appears in only one relation

The NULL approach



- One relation that includes everything
 - Movies(ID, name, Is_3D, Is_Adoption)
- For non-cartoon movies, its “Is_3D” is set to NULL
- For non-sci-fi movies, its “Is_Adoption” is set to NULL

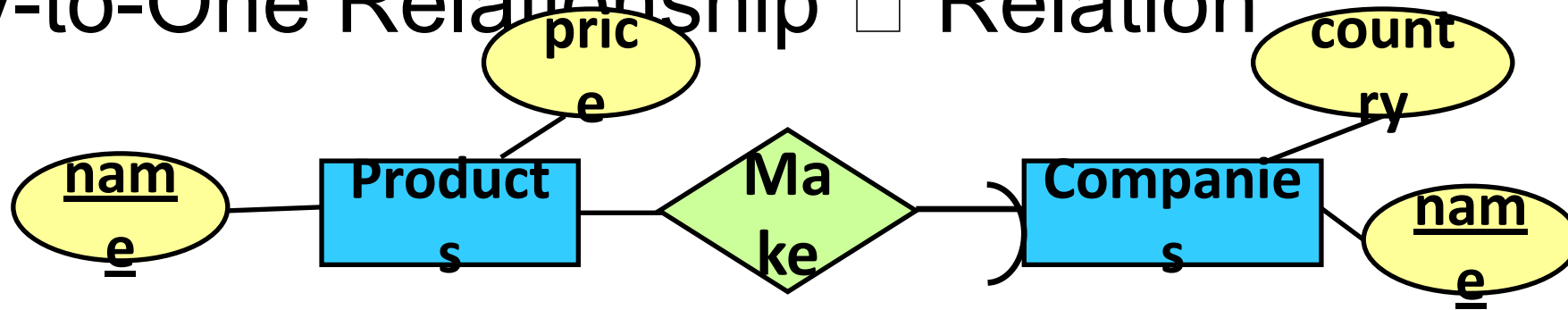
Which Approach is the Best?

- It depends
- The NULL approach
 - Advantage: Needs only one relation
 - Disadvantage: May have many NULL values
- The OO approach
 - Advantage: Good for searching subclass combinations
 - Disadvantage: May have too many tables
- The ER approach
 - A middle ground between OO and NULL

ER Diagram \square Relational Schema

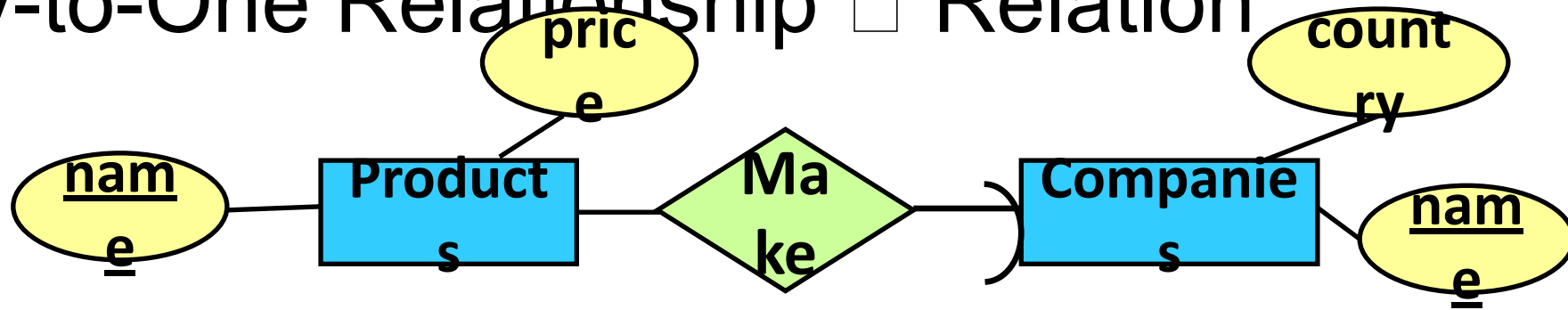
- General rules:
 - Each entity set becomes a relation
 - Each many-to-many relationship becomes a relation
- Special treatment needed for:
 - Weak entity sets
 - Subclasses
 - Many-to-one and one-to-one relationships

Many-to-One Relationship □ Relation



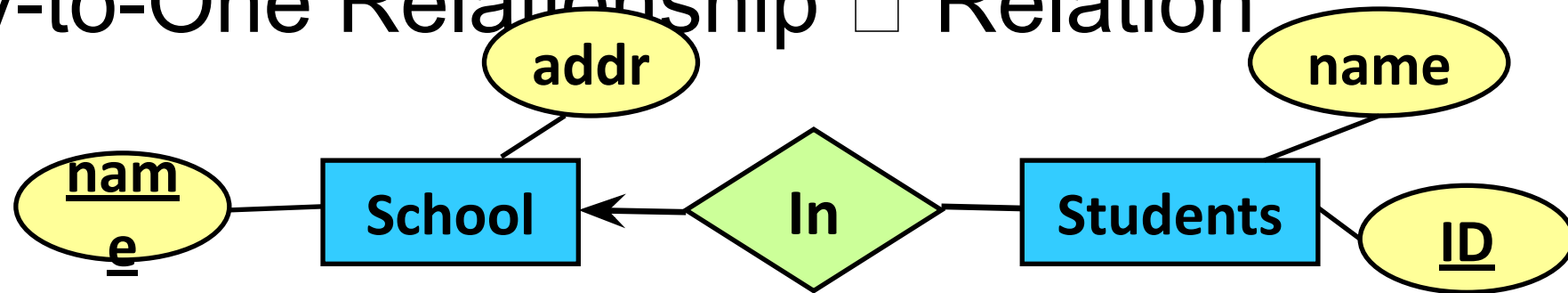
- Intuitive translation:
 - Products(Pname, price)
 - Companies(Cname, country)
 - Make(Pname, Cname)
- Observation: in “Make”, each Pname has only one Cname
- Simplification: Merge “Make” and “Products”
- Results:
 - Products(Pname, price, Cname)
 - Companies(Cname, country)

Many-to-One Relationship □ Relation



- In general, we do not need to create a relation for a many-to-one relationship
- Instead, we only need to put the key of the “one” side into the relation of the “many” side

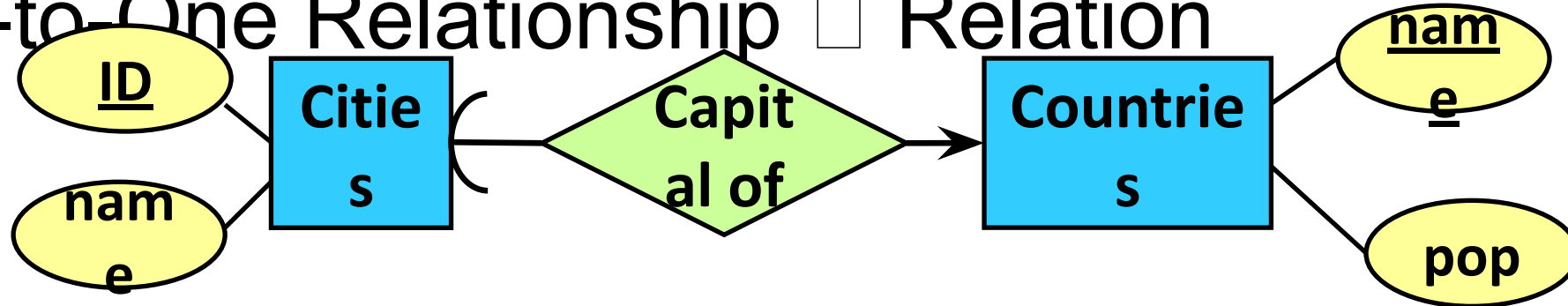
Many-to-One Relationship □ Relation



■ Translation:

- School(Hname, addr)
 - Students(ID, Sname, Hname)
- Only need to put the key of the “one” side into the relation of the “many” side

One-to-One Relationship □ Relation



- No need to create a relation for a one-to-one relationship
- Only need to put the key of one side into the relation of the other
- Solution 1
 - Cities(TID, Tname)
 - Countries(Cname, pop, TID)
- Solution 2
 - Cities(TID, Tname, Cname)
 - Countries(Cname, pop)

Summary

- Entity
- Relationship
- Constraints (Key, FK)
- Subclasses
- Weak Entity Sets
- Translating an ER Diagram into a Relational Scheme Design