

CZ2007 Introduction to Database Systems (Week 2)

Topic 1: Entity Relationship Diagram (3)



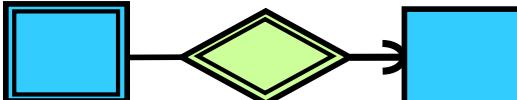
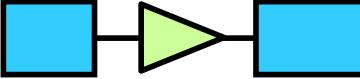
Dr. Ng Wee Keong

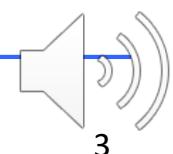
Associate Professor

This presentation is copyright property of NTU. It is intended for students of CZ2007 only.



So far, we learned:

- Elements of ER Diagrams
 - Entities Sets 
 - Relationships 
 - Attributes 
 - Weak Entities Sets 
 - Subclasses 
- How do we design an ER Diagram for an application?



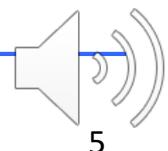
This Lecture

- ER diagram design principles ←
- ER diagram → relational schema



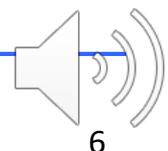
From Applications to ER Diagrams

- Identify the **objects** involved in your application
- Model each type of objects as an entity set
- Identify the attributes of each entity set
- Identify the relationships among the entity sets
- Refine your design
- Example: A database for NTU
 - **Objects:** Students, Faculty, Schools, Courses...
 - **Entity sets:** Students, Faculty, Schools, Courses...
 - **Relationships:** course-enrollment, course-lecturer...



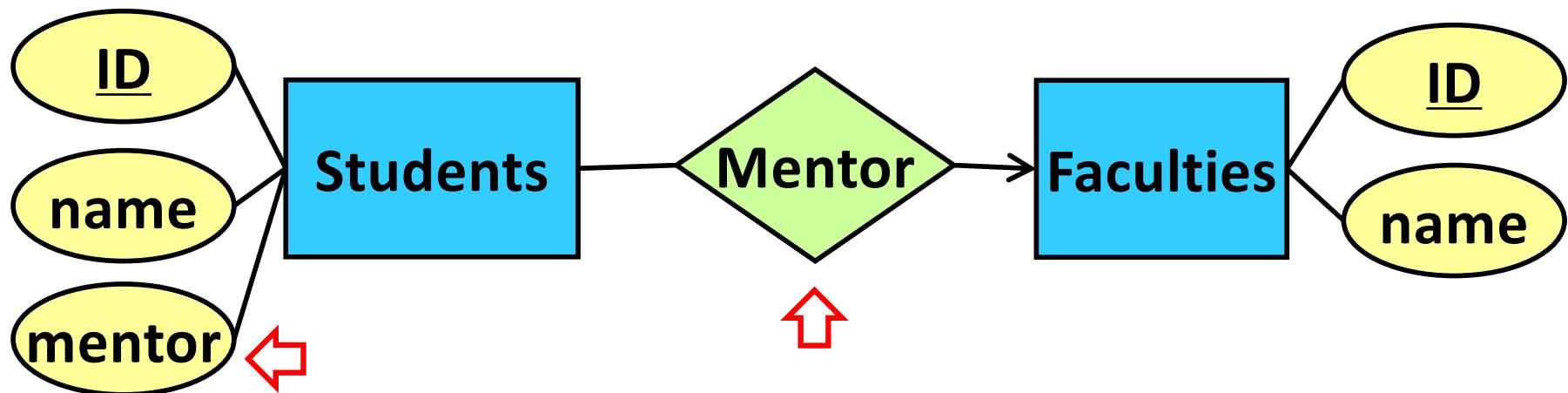
Design Principle 1: Be Faithful

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

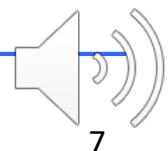


Design Principle 2: Avoid Redundancy

- Avoid repetition of information
- Example

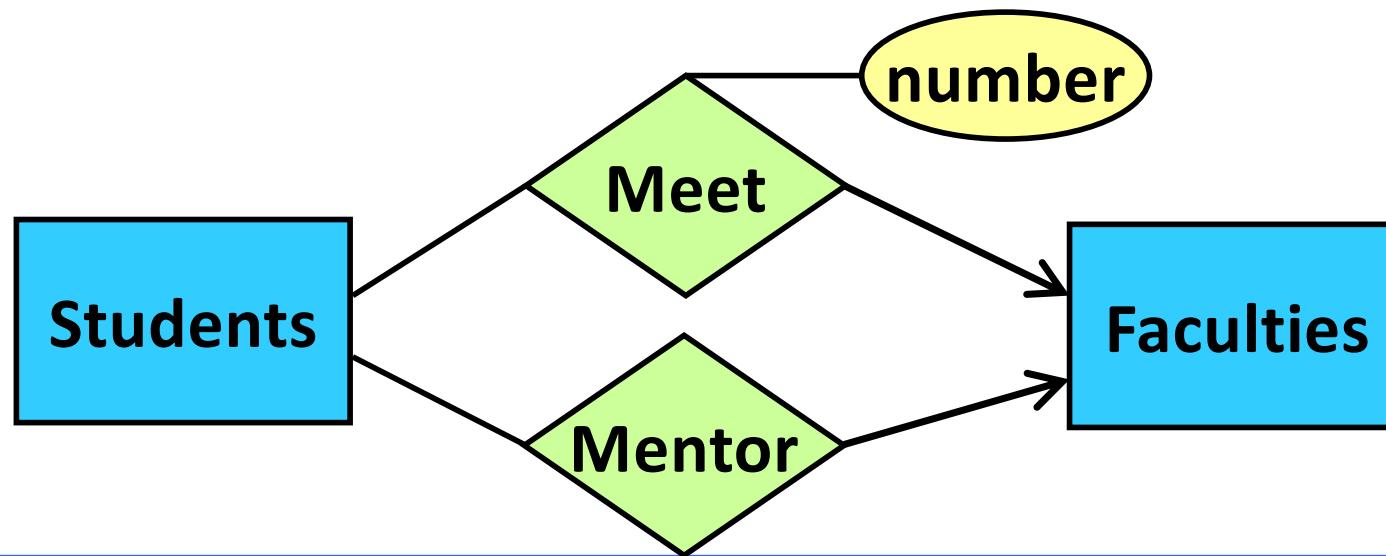


- Problems that can be caused by redundancy
 - Waste of space
 - Possible inconsistency



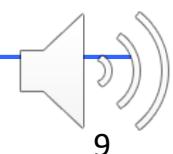
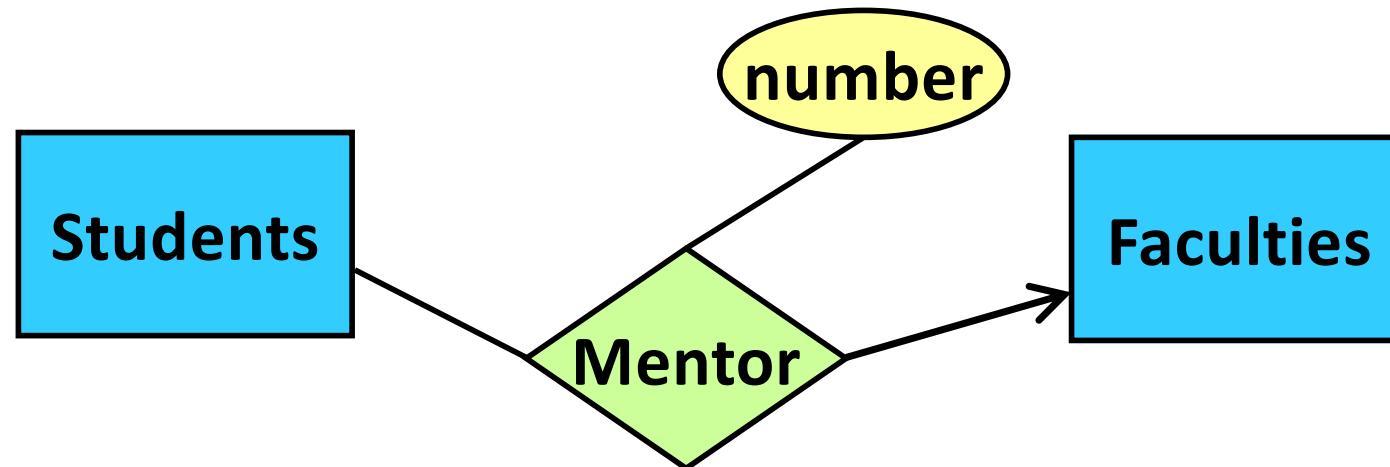
Design Principle 3: Keep It Simple

- 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



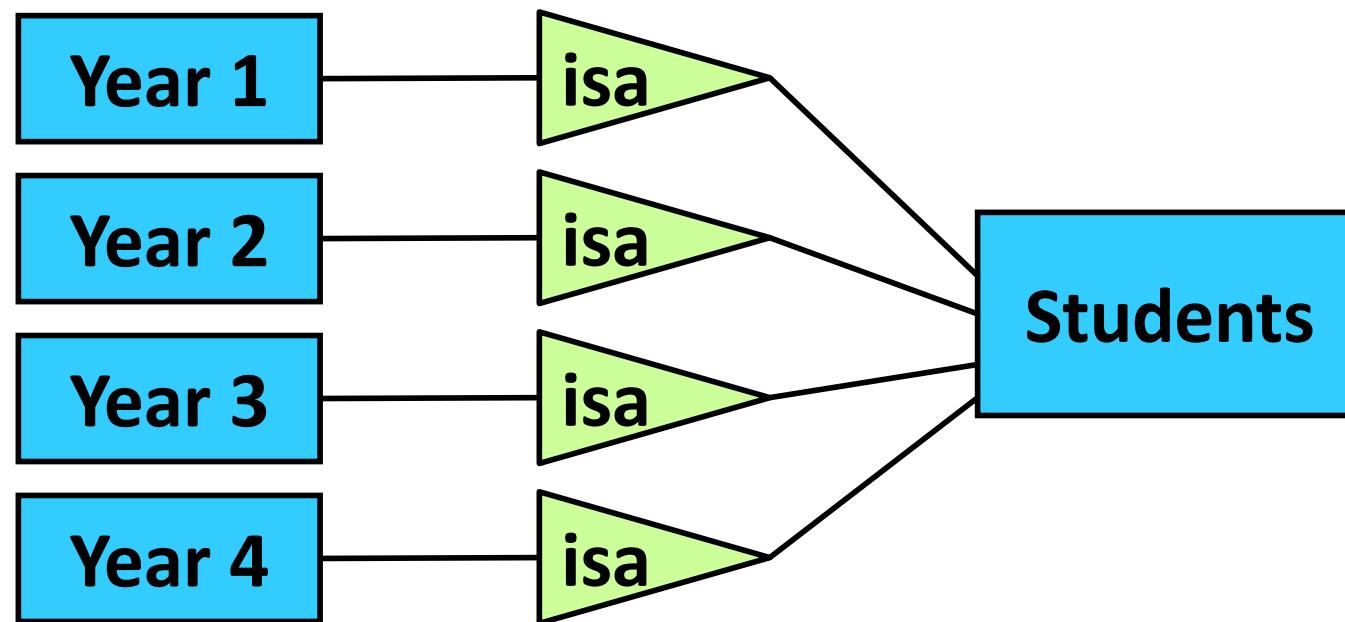
Design Principle 3: Keep It Simple

- 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:



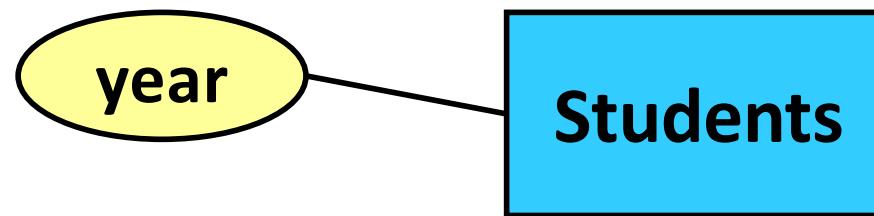
Design Principle 3: Keep It Simple

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



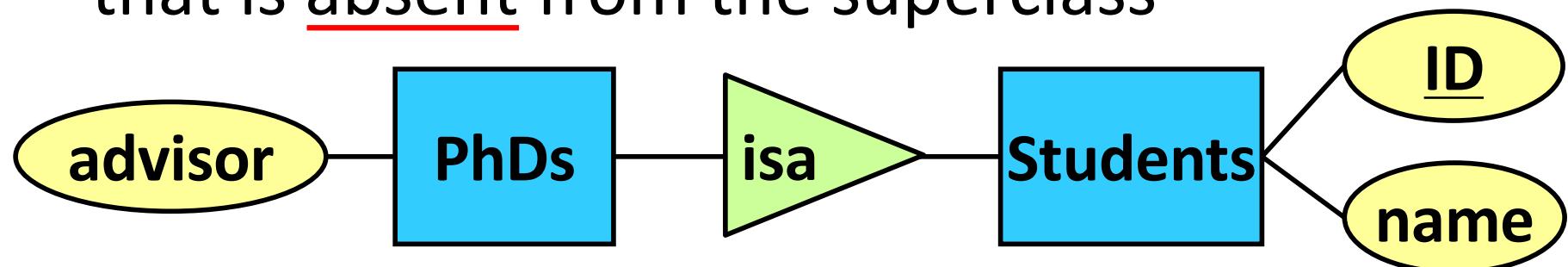
Design Principle 3: Keep It Simple

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

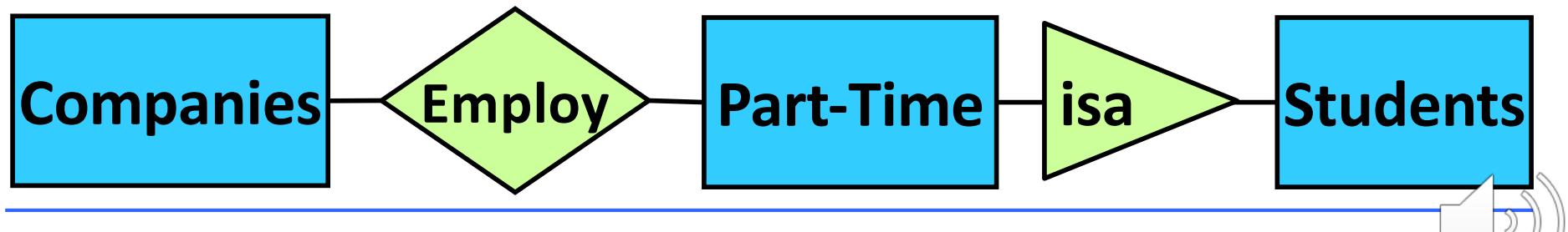


Tips: When to Use Subclasses

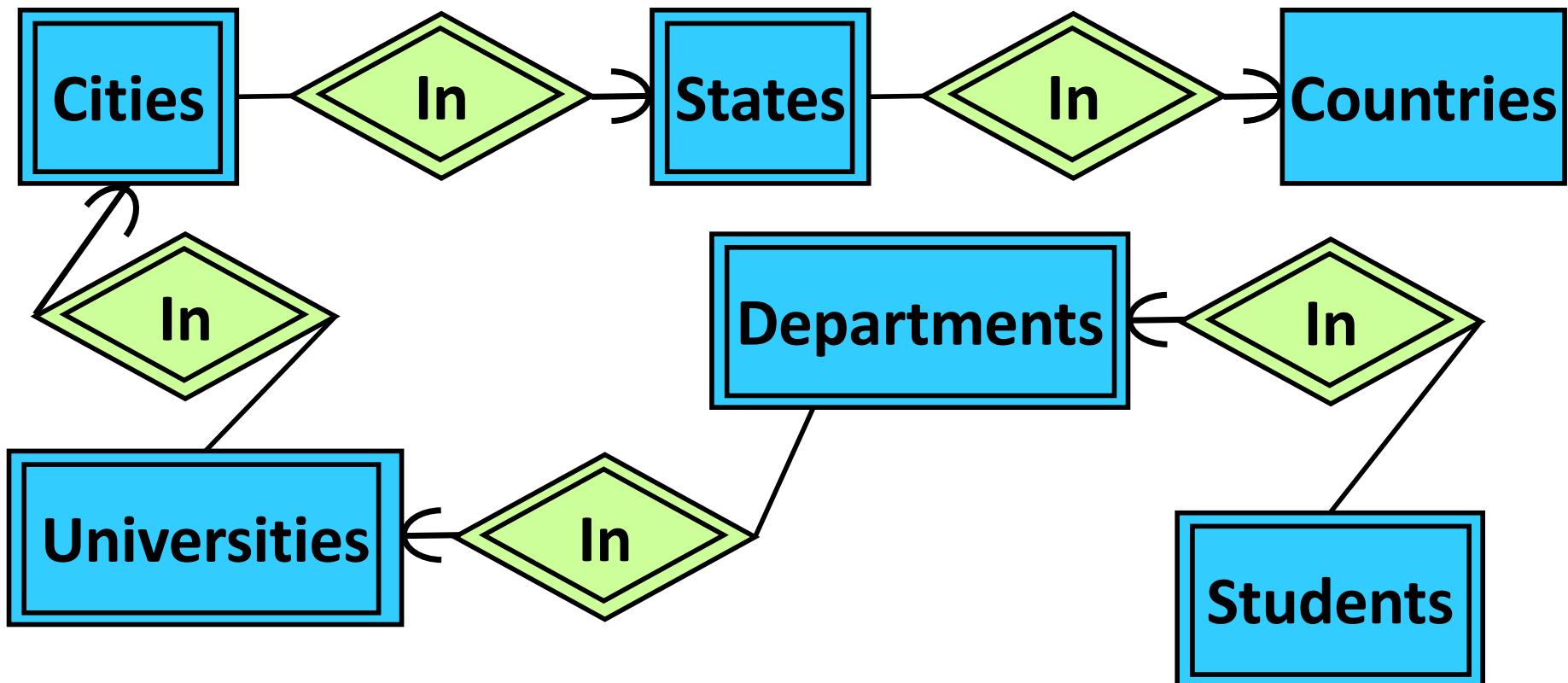
- Case 1: When a subclass has some attribute that is absent from the superclass



- 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”

This Lecture

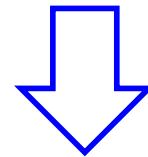
- ER diagram design principles
- ER diagram → relational schema 



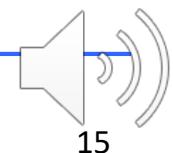
Road Map

- We have discussed
 - Elements of ER Diagrams: Entity Sets, Relationships, Attributes...
 - Design principles of ER Diagrams
- These all concern the conversion below:

Real-World Application



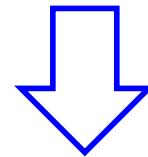
Entity-Relationship (ER) Diagrams



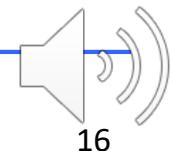
Road Map

- But how do we convert the ER diagrams to a set of tables?
- We will discuss this in the next few slides

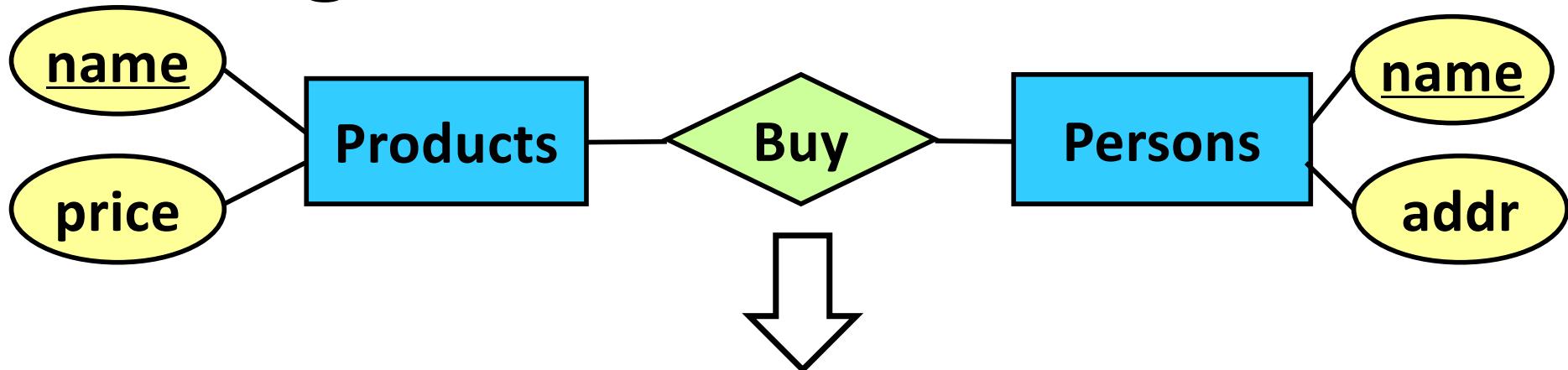
Entity-Relationship (ER) Diagrams



Tables (Database Schema)



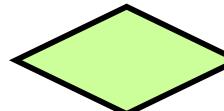
ER Diagram → Relational Schema



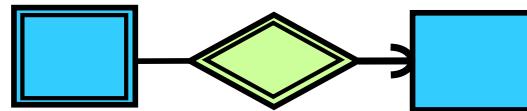
- Products (name, price)
- Persons (name, addr)
- Buy (product_name, person_name)
- Terminology
 - A **relation schema** = name of a table + names of its attributes
 - A **database schema** = a set of relation schemas

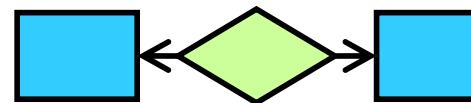
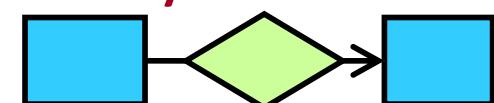
ER Diagram → Relational Schema

■ General rules:

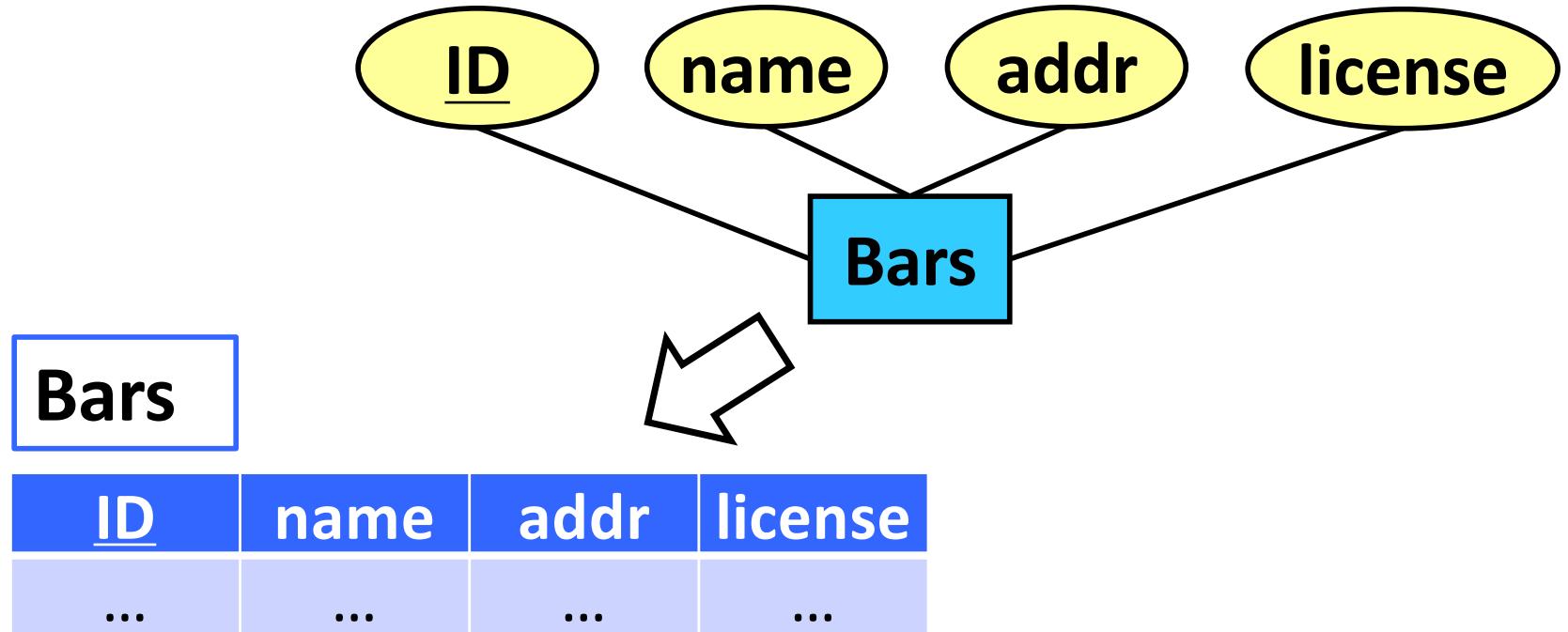
- Each entity set  becomes a relation 
- Each many-to-many relationship  becomes a relation 

■ Special treatment needed for:

- Weak entity sets 
- Subclasses 
- Many-to-one and one-to-one relationships

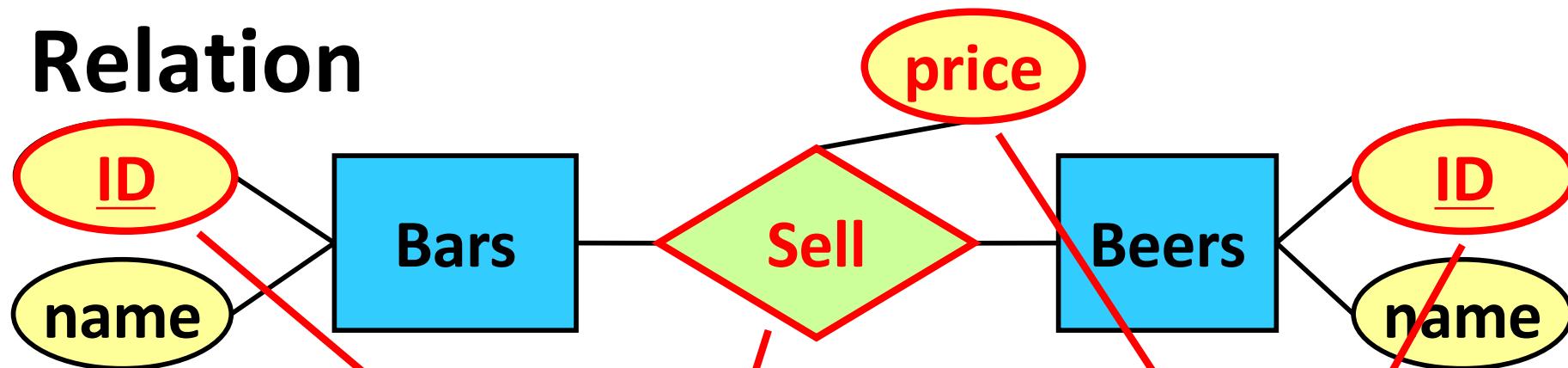


Entity Set → Relation



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

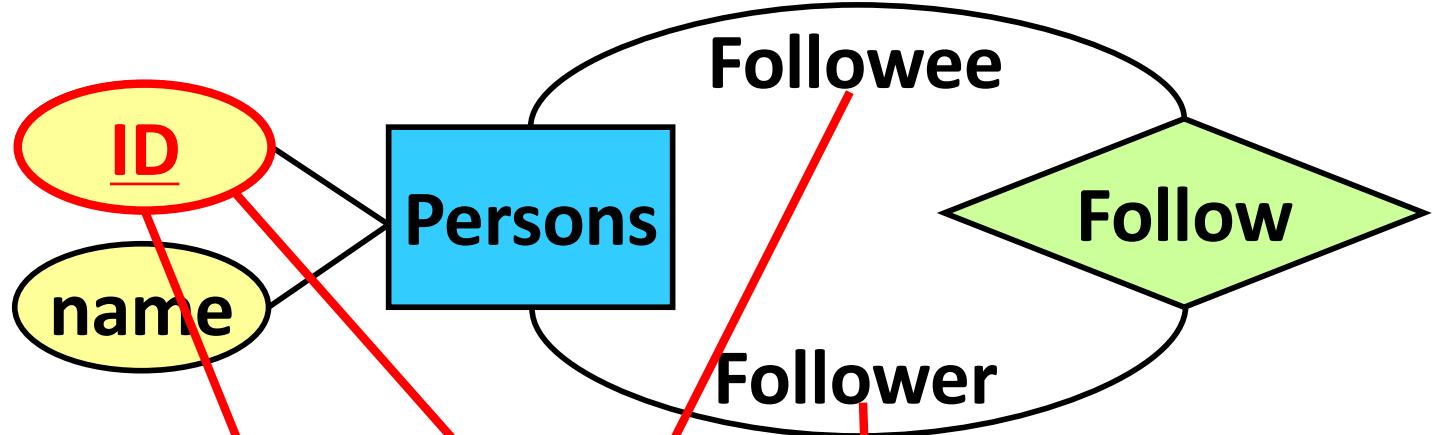
Many-to-Many Relationship → Relation



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

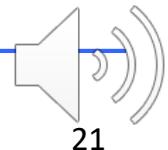
Sell	<u>Bars-ID</u>	<u>Beers-ID</u>	<u>price</u>

Many-to-Many Relationship → Relation

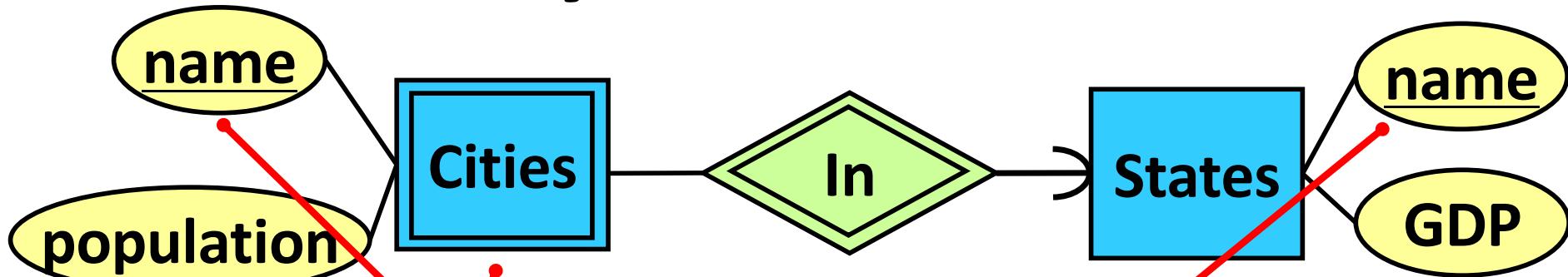


- If an entity is involved multiple times in a relationship
 - Its key will appear in the corresponding relation multiple times
 - The key is re-named according to the corresponding role

Follow	Followee-ID	Follower-ID



Weak Entity Set → Relation

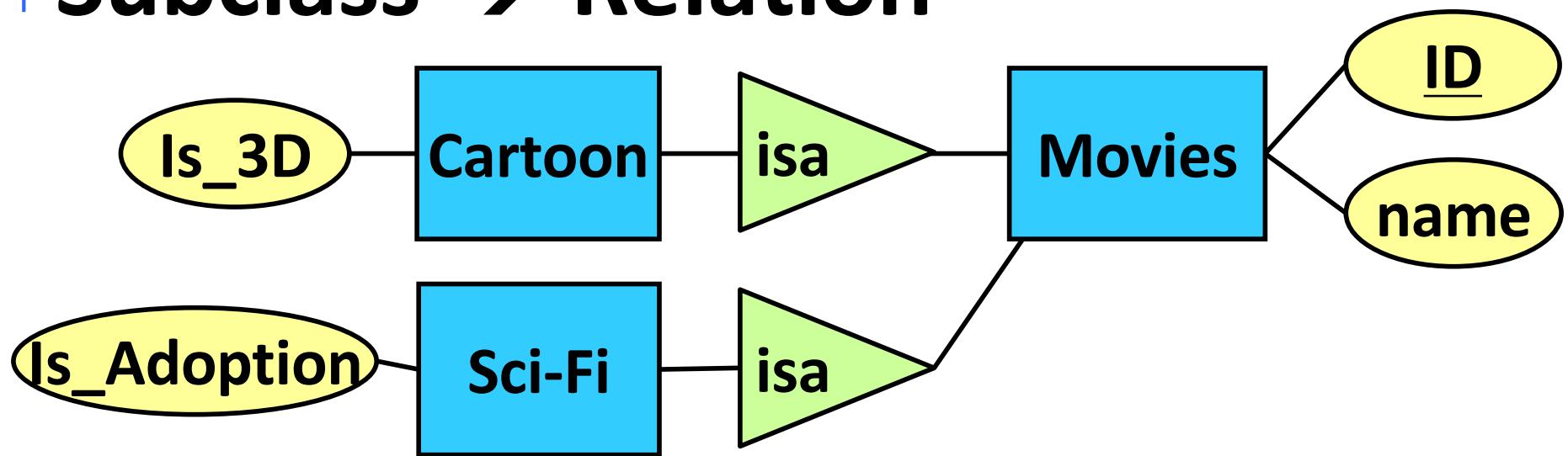


- Each weak entity set is converted to a relation that contains
 - all of its attributes, and
 - the key attributes of the supporting entity set
- The supporting relationship is ignored

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

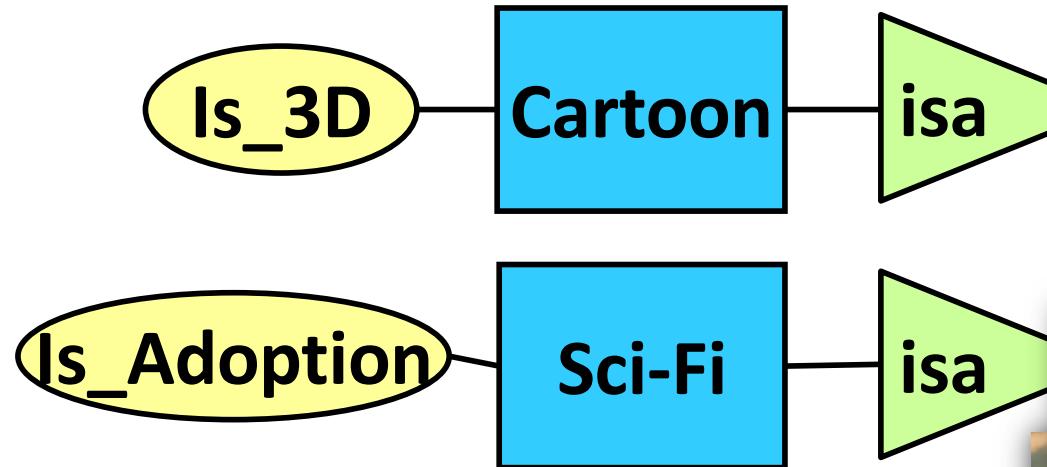
			...

Subclass → Relation



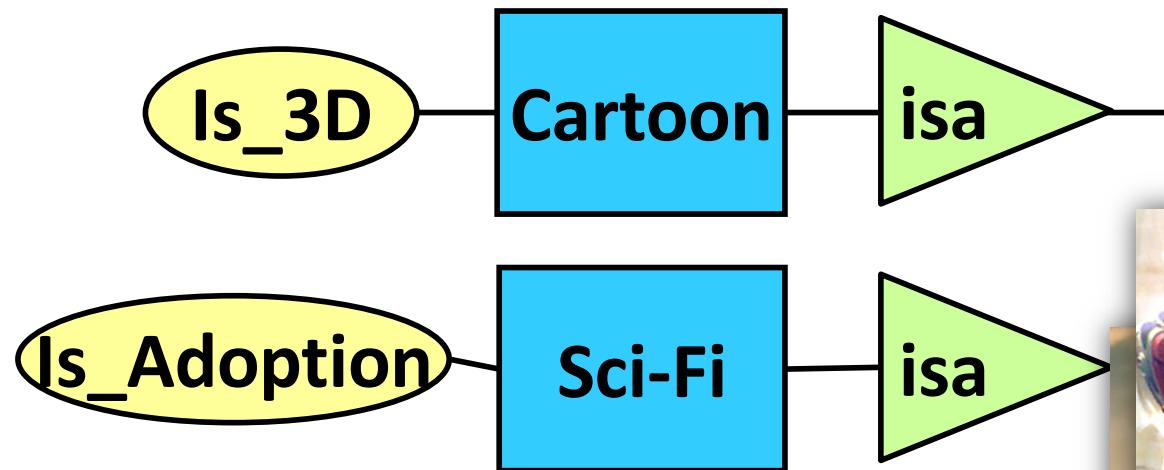
- There are three different ways
 - The ER approach **One record in multiple relations**
 - The OO approach **One record in one relation; potentially many multiple relations**
 - The NULL approach **One big relation, lots of NULLs**

The ER approach



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

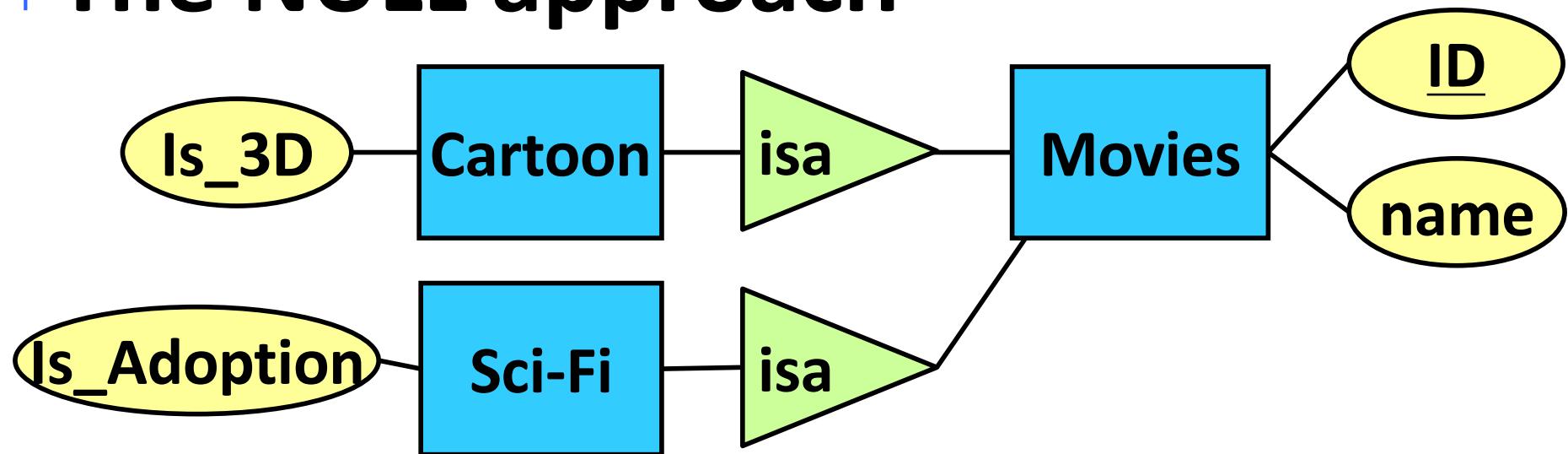
The OO approach



- One relation for each entity set **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



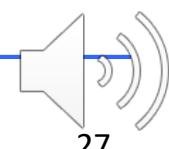
- One relation that includes everything
 - `Movies(ID, name, Is_3D, Is_Adoption)`
- For non-cartoon movies, its “Is_3D” is `NULL`
- For non-sci-fi movies, its “Is_Adoption” is `NULL`



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

S1, S2, S3, S4
6+4+1 combinations:
 $\{S1,S2\}$
 $\{S1,S3\}$
 $\{S1,S4\}$
 $\{S2,S3\}$
 $\{S2,S4\}$
 $\{S3,S4\}$
 $\{S1,S2,S3\}$
 $\{S1,S2,S4\}$
 $\{S1,S3,S4\}$
 $\{S2,S3,S4\}$
 $\{S1,S2,S3,S4\}$



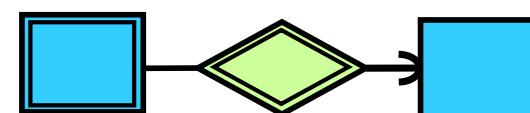
ER Diagram → Relational Schema

- General rules:

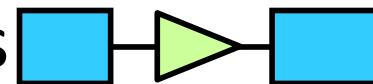
- Each entity set becomes a relation
 - Each many-to-many relationship becomes a relation

- Special treatment needed for:

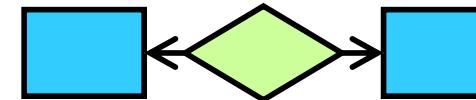
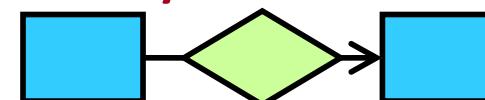
Weak entity sets



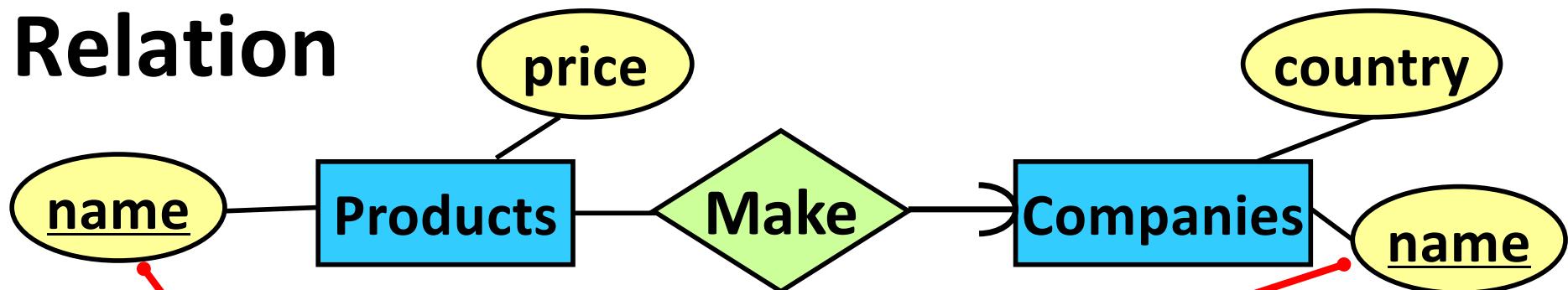
Subclasses



→ Many-to-one and one-to-one relationships



Many-to-One Relationship → Relation



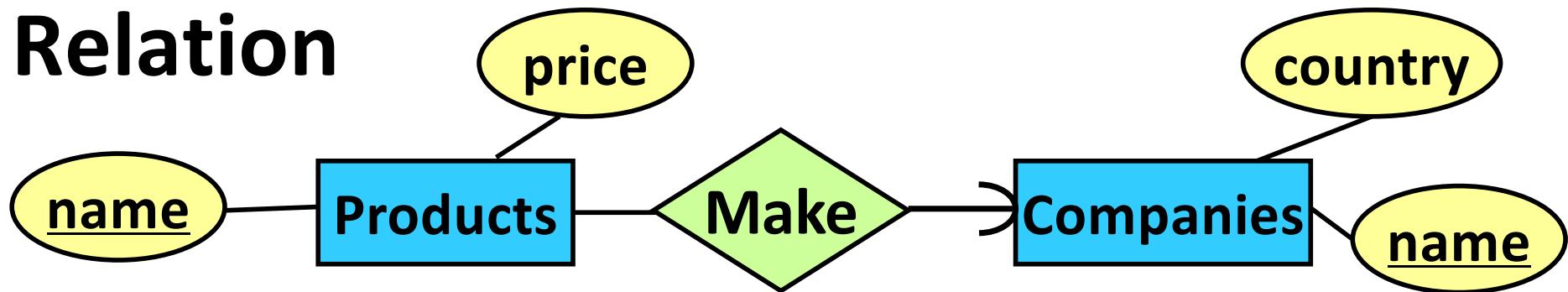
- Intuitive translation:
 - $\text{Products}(\underline{\text{Pname}}, \text{price})$
 - $\text{Companies}(\underline{\text{Cname}}, \text{country})$
 - $\text{Make}(\underline{\text{Pname}}, \underline{\text{Cname}})$ => $\text{Make}(\underline{\text{Pname}}, \text{Cname})$
- Observation: in “Make”, each Pname has
- Simplification: Merge “Make” and “Produ
- Results:
 - $\text{Products}(\underline{\text{Pname}}, \text{price}, \text{Cname})$
 - $\text{Companies}(\underline{\text{Cname}}, \text{country})$

A relational table with two columns: 'Products' and 'Companies'. The 'Products' column contains eight entries, each represented by a small circle. The 'Companies' column contains four entries, each represented by a diamond shape. Red lines connect the entries in the 'Products' column to the entries in the 'Companies' column, showing that multiple products can be associated with a single company. The last entry in the 'Products' column is highlighted with a red arrow pointing to it.

Products	Companies
○	◊
○	◊
○	◊
○	◊
○	◊
○	◊
○	◊
○	◊

Many-to-One Relationship →

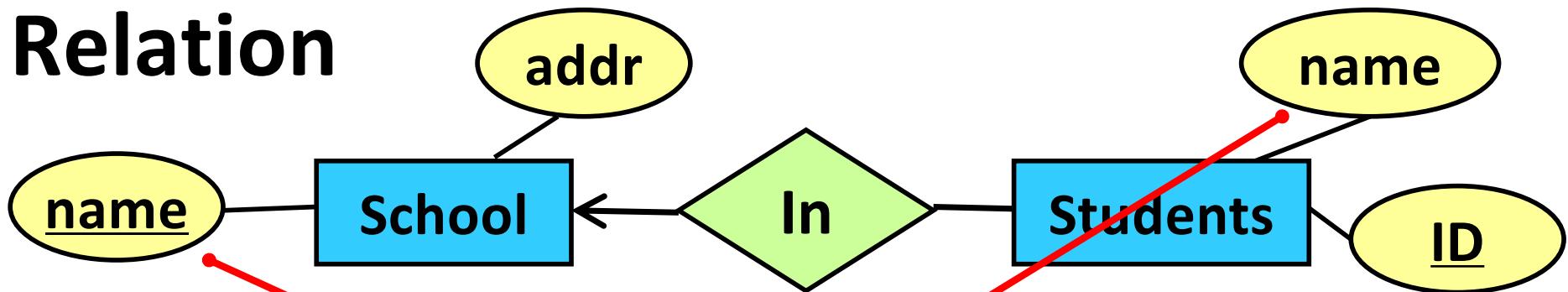
Relation



- 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

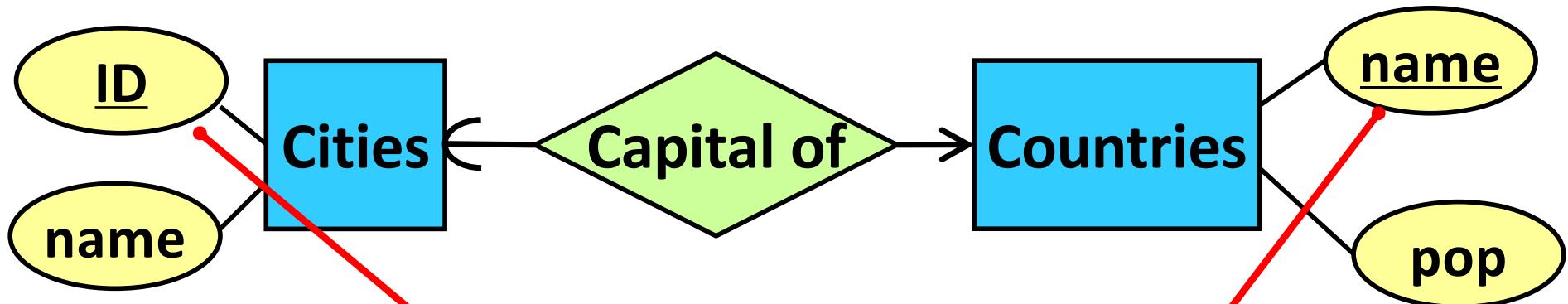
Many-to-One Relationship →

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

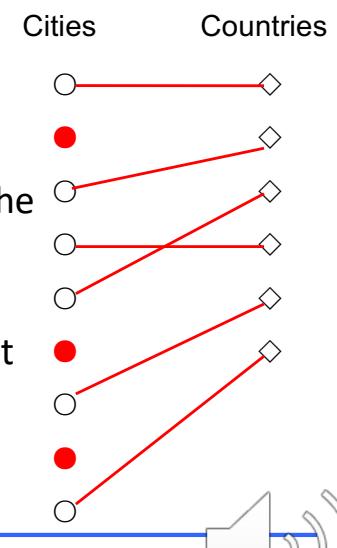


- No need to create a relation for a one-to-one relationship
- Only need to put the key of one side into the relation of the other
- Solution 1
 - Cities(CityID, Cityname)
 - Countries(Countryname, pop, CityID)
- Solution 2
 - Cities(CityID, Cityname, Countryname)
 - Countries(Countryname, pop)

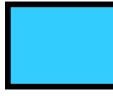
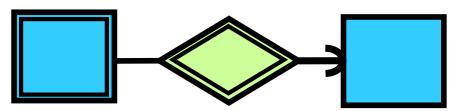
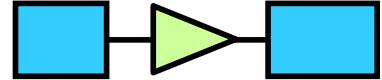
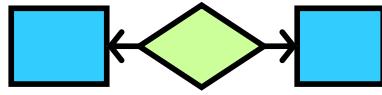
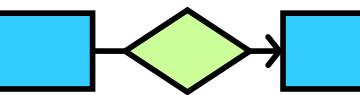
Sol 1 vs Sol 2: Which one is better?

A city can be the capital of only one country
Will not be null

A country must have a capital
May be null



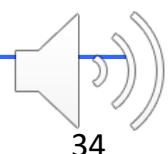
Summary

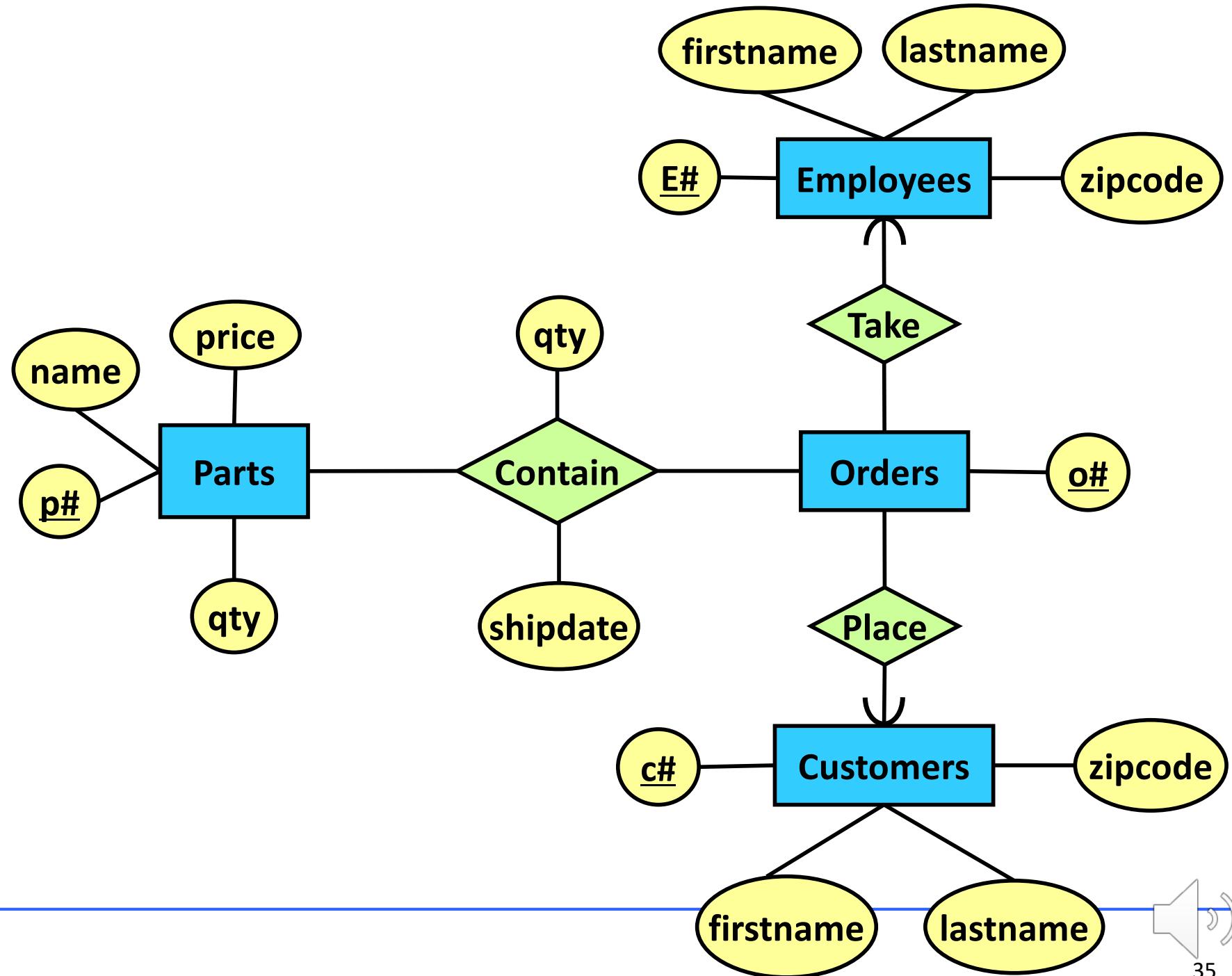
- General rules:
 - Each entity set  becomes a relation 
 - Each many-to-many relationship becomes a relation 
- Special treatment needed for:
 - Weak entity sets 
 - Subclasses 
 - Many-to-one and one-to-one relationships 



ER Diagram Design: Example

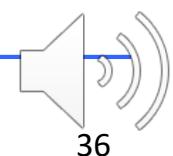
- Consider a mail order database in which employees take orders for parts from customers. The requirements are:
- Each employee is identified by a unique employee number, and has a first name, a last name, and a zip code.
- Each customer is identified by a unique customer number, and has a first name, last names, and a zip code.
- Each part being sold is identified by a unique part number. It has a part name, a price, and a quantity in stock.
- Each order placed by a customer is taken by one employee and is given a unique order number. Each order may contain certain quantities of one or more parts. The shipping date of each part is also recorded.





Relational Schema

- Parts(PID, name, price, stock)
- Employees(EID, fname, lname, ZIP)
- Customers(CID, fname, lname, ZIP)
- Orders(OID, EID, CID) – from two many-to-one relationships
- Contain(OID, PID, qty, shipdate) – from many-to-many relationships



Next lecture:

Topic 2: Functional Dependencies (1)

