SLOT 6

CHAPTER 4

HIGH-LEVEL DATABASE MODEL

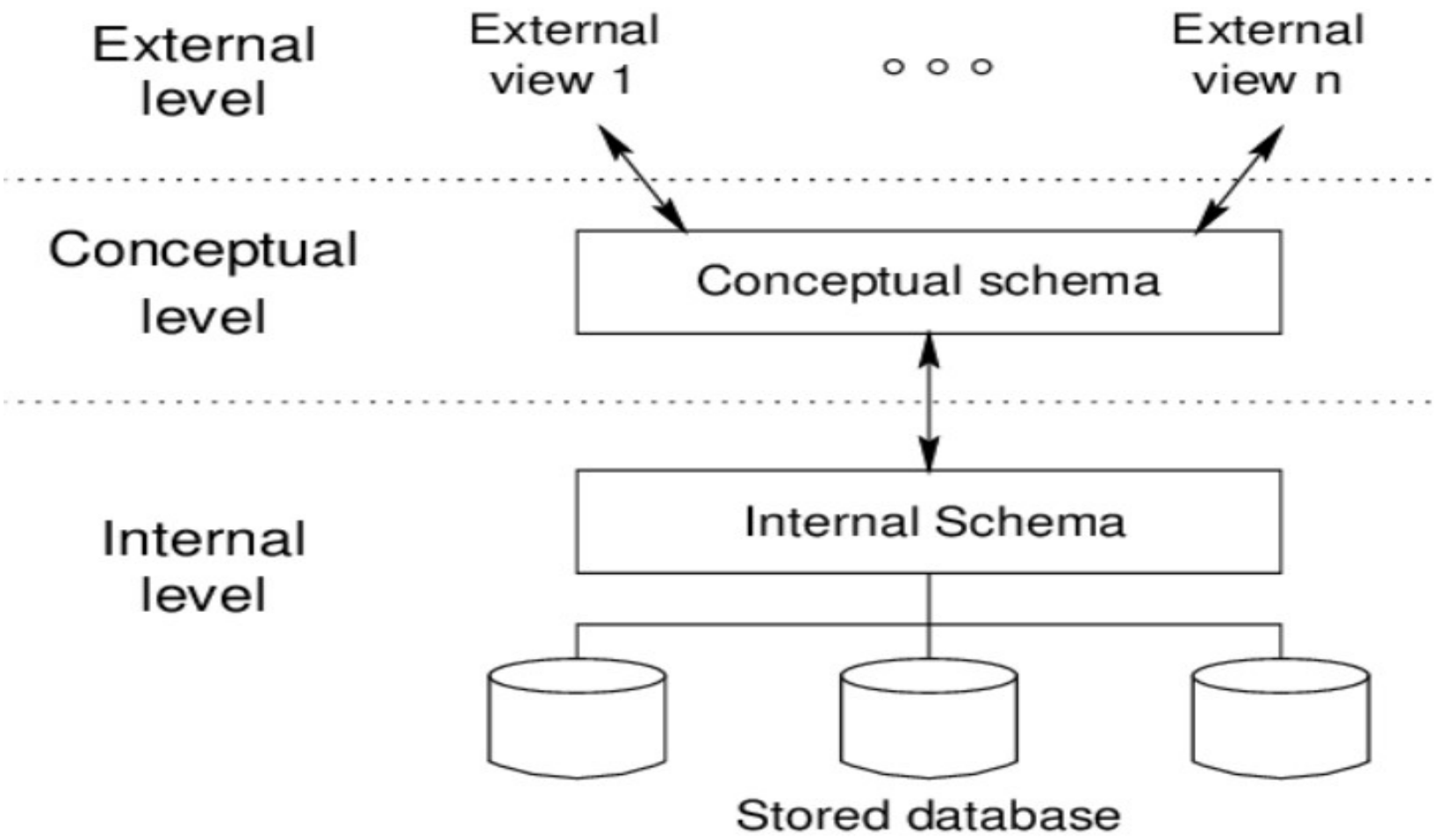
OBJECTIVES

Understand concepts of:

- ☐ Database Design Process
- ☐ Data modeling basing on entity relationship
- ☐ Design a suitable database adapted business requirements in reality

CONTENT

DATA MODEL - OVERVIEW



Database modeling and implementation process

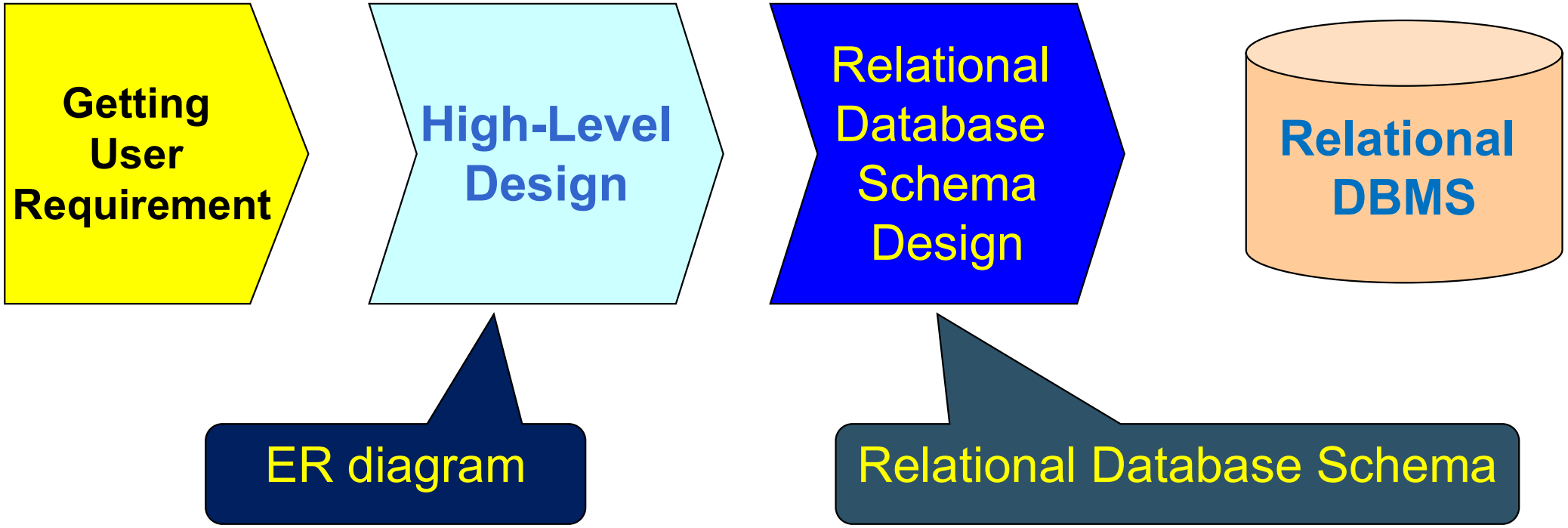


Figure 4.1: The database modeling and implementation process

Steps in Database Design

1. **Requirements Analysis**
   - user needs; what must database do?
2. **Conceptual Design**
   - high level description (Entity Relationship diagram)
3. **Logical Design**
   - translate ERD into DBMS data model
4. **Schema Refinement**
   - consistency, normalization
5. **Physical Design**
   - indexes, disk layout
6. **Security Design**
   - who accesses what, and how

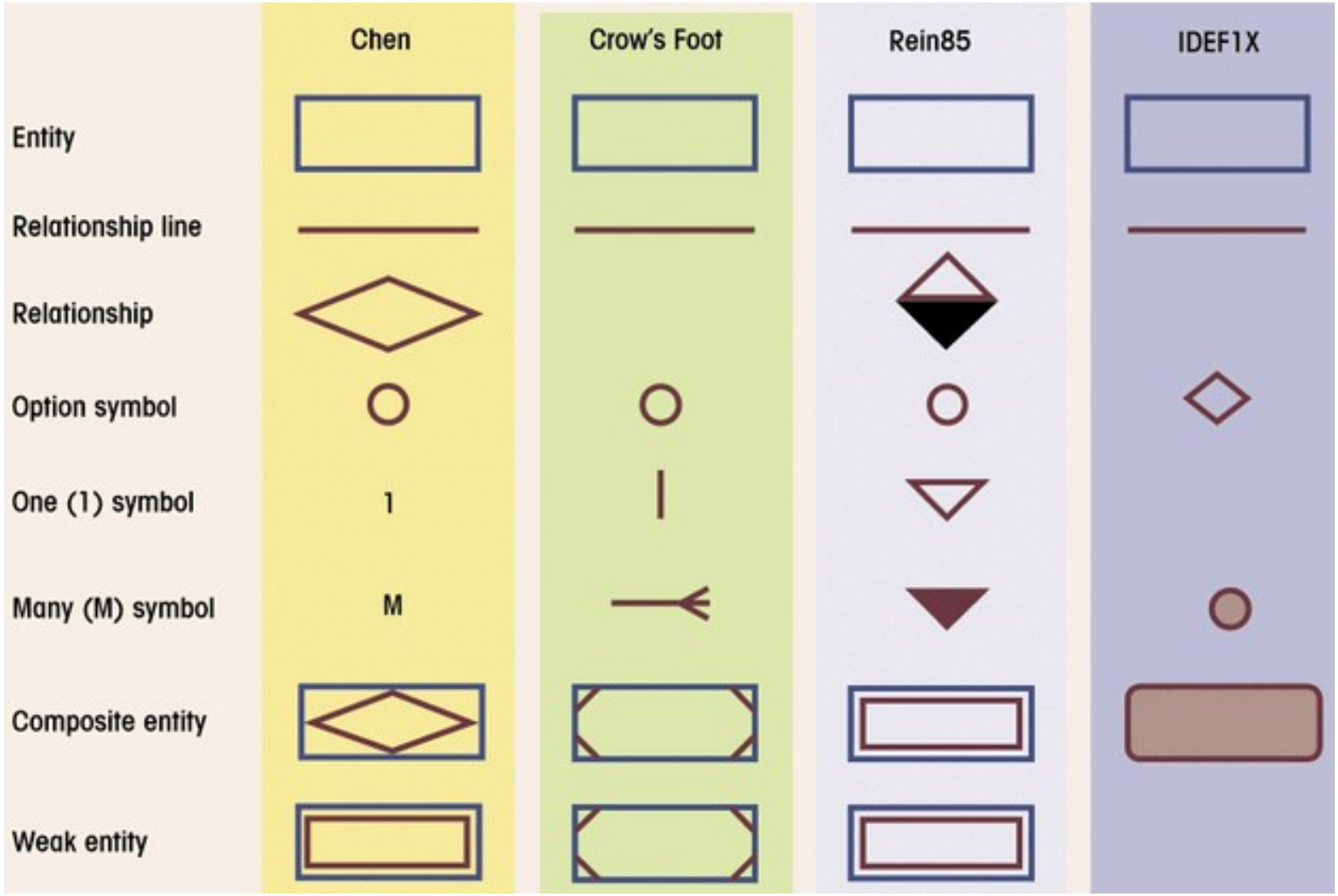
ERD – How to construct

- Gather all the data that needs to be modeled.
- Identify data that can be modeled as real world entities.
- Identify the attributes for each entity.
- Sort entity sets as weak or strong entity sets.
- Sort entity attributes as key attributes, multi-valued attributes, composite attributes, derived attributes.
- Identify the relations between the different entities.
- Using the different symbols draw the entities, their attributes and their relationships. Use appropriate symbols while drawing attributes.

Entity Relationship Diagram - Notations

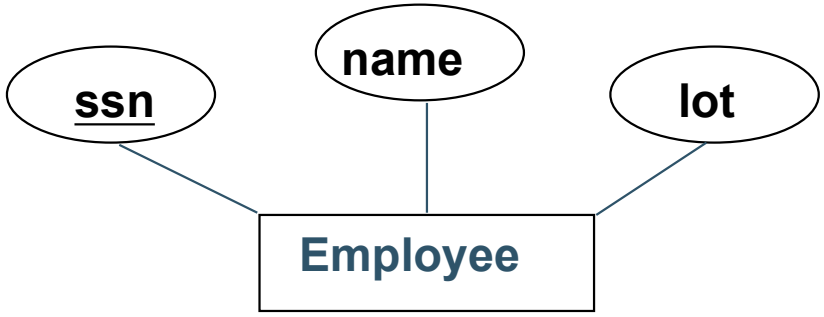
Component	Symbol	Example
Entity	Entity	Student
Weak Entity	Weak Entity	Assignments
Attribute	Attribute	Roll_num
Relationship	Relationship	Saves in
Key Attribute	Attribute	Acct_num

Comparison of E-R Modeling notations



ERD - ENTITY

- ***Entity:***
  - Real-world thing, distinguishable from other objects.
  - Noun phrase
  - Entity described by set of *attributes*.
- ***Entity Set*: A collection of similar entities. E.g., all employees.**
  - All entities in an entity set have the same set of attributes. (Until we consider hierarchies, anyway!)
  - Each attribute has a domain.

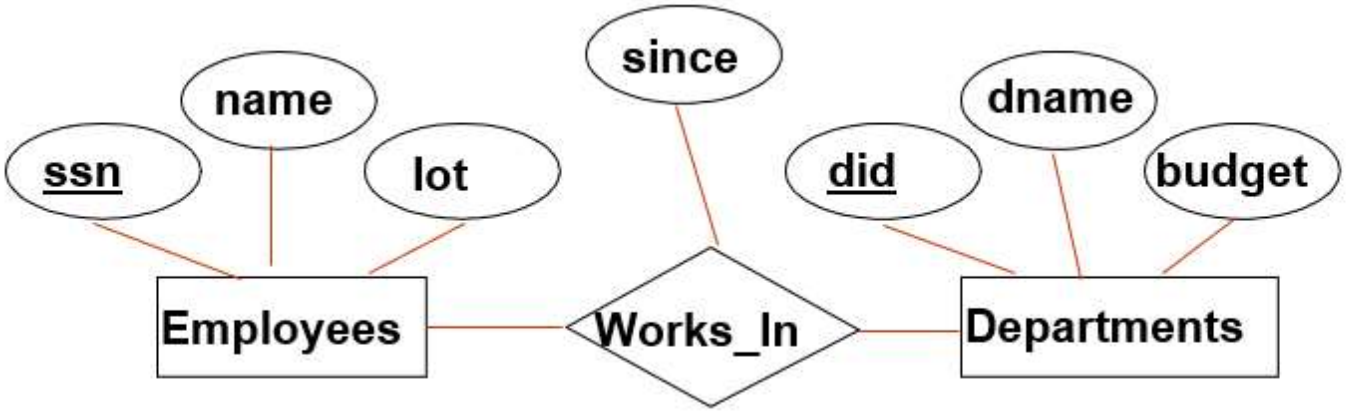


ERD - RELATIONSHIP

***Relationship*:** Association among two or more entities. Relationships can have their own attributes (descriptive attributes).

verb phrases

- 1-1
- 1- M/ M -1
- M - M



Degree Constraints

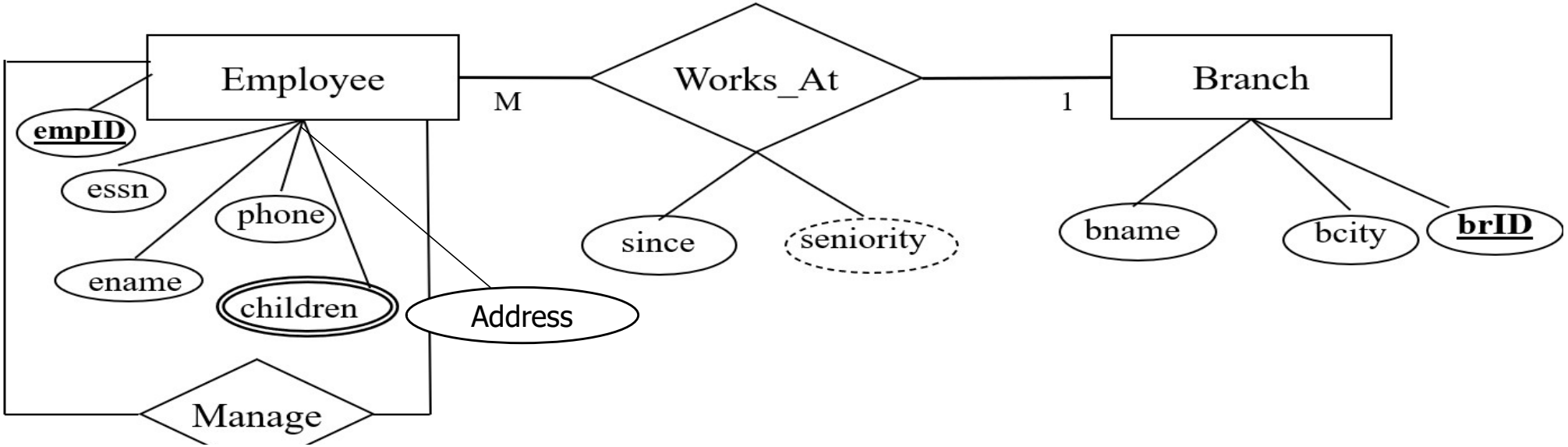
- Recursive relationship
- Unary, Binary, Ternary relationship (quan hệ 1 ngôi, hai ngôi,...)

A referential integrity constraints (ràng buộc toàn vẹn tham chiếu)

- A value appearing in one context must also appear in another

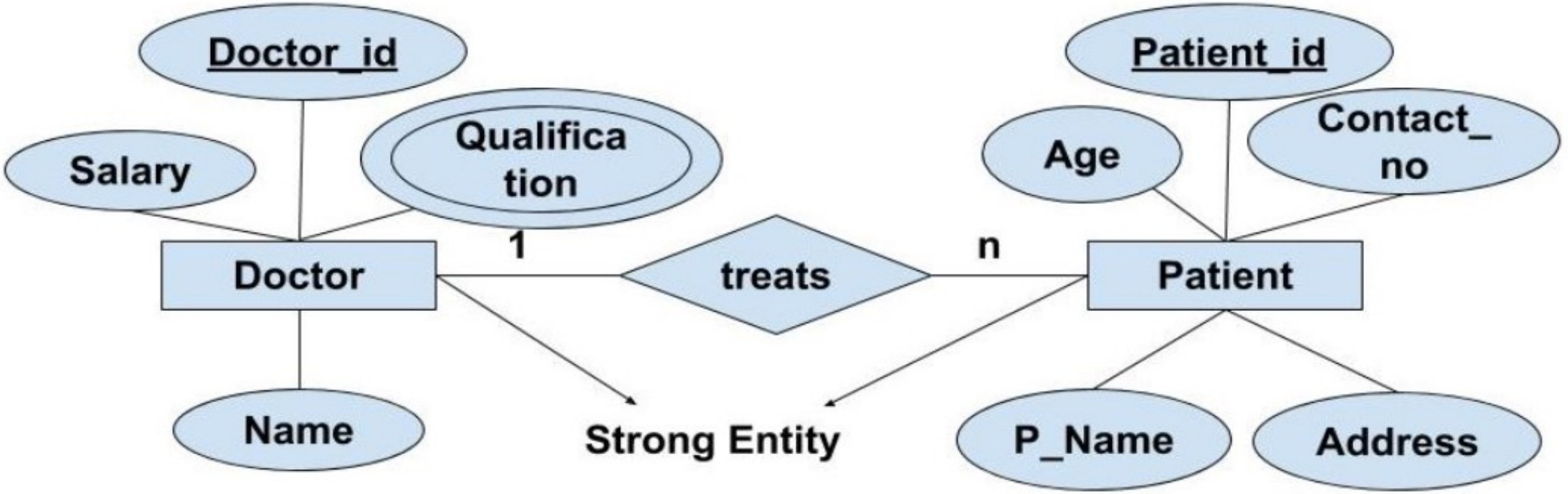
Type of Attributes

- **Key** attribute (khoá) → EmpID
- **Multivalued** attribute (đa trị) → children
- **Derived** attribute (suy ra) → seniority
- **Composite** attribute (tổng hợp) → Address



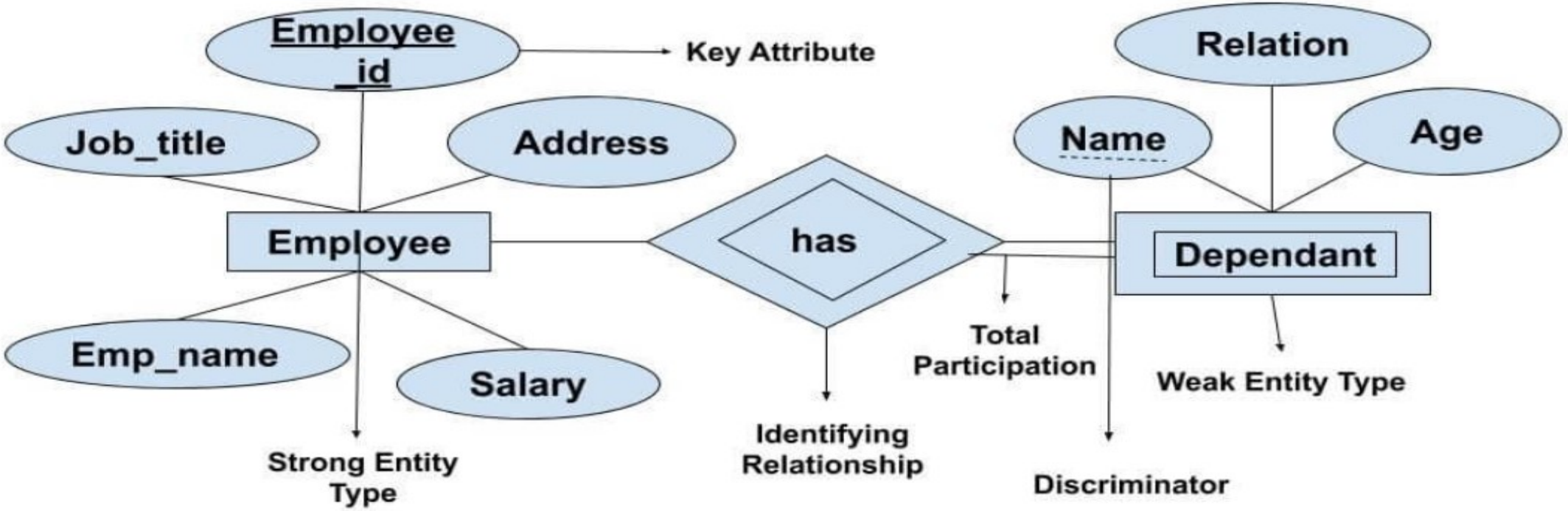
Strong Entity

➡A strong entity is an entity type whose existence doesn't depend on any other entity.



Weak Entity Sets

Weak entity type doesn't have a key attribute. Weak entity type can't be identified on its own. It depends upon some other strong entity for its distinct identity

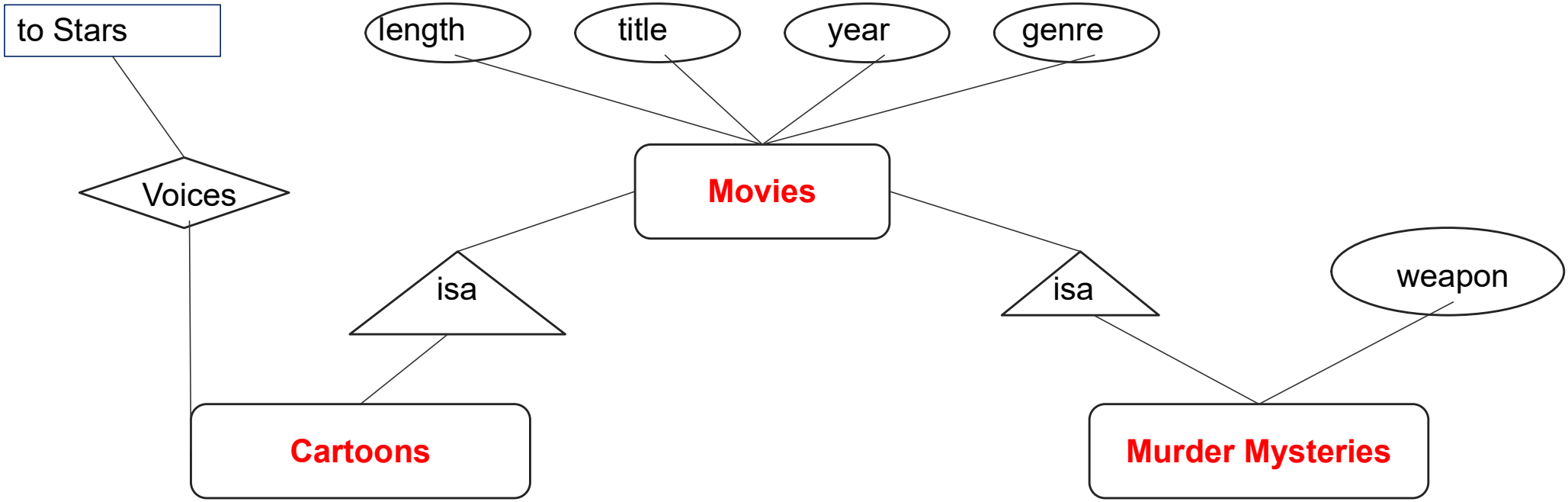


Requirements for Weak Entity Sets

- R is a relationship from E to F
- R is called *supporting relationship* if
  - R must be a binary, many-one (M - 1) relationship from E to F
  - R must have referential integrity from E to F
  - The attributes that F supplies for the key of E must be key attributes of F

Subclasses in E/R Model

- Consider Cartoons and Murder Mysteries are the special kinds of movies, with some special properties



Example COMPANY Database – Construct ERD

- Requirements:
  - The company is organized into **DEPARTMENTs**. Each department has a **name**, **number** and an employee who ***manages*** the department. We keep track of the **start date** of the department manager.
  - Each department ***controls*** a **number** of **PROJECTs**. Each **project** has a **name**, **number** and is located at a single **location**.

Example COMPANY Database (Cont.)

- We store each **EMPLOYEE**'s **social security number**, **address**, **salary**, **sex**, and **birthdate**. Each employee *works for* one department but may *work on* several projects. We keep track of the **number of hours per week** that an employee currently works on each project. We also keep track of the **direct supervisor** of each employee.
- Each employee may *have* a **number** of **DEPENDENTs**. For each dependent, we keep track of their **name**, **sex**, **birthdate**, and **relationship** to employee.

Relational database

Employee (SSNumber, E_Name E_Address, Salary, Sex, Birthday)

Department(DNumber, DName, Startdate, SSNumber)

Project (P_Number, Name, P_Location, DNumber)

Dependents (SSNumber, DependentName, Sex, Birthday, Relationship)

WorkFor(**SSNumber, DNumber**)

WorkOn(**SSNumber,P_Number,** NumberHour,Supervisor)

ERD

Design ERD by yourself

EX1

Create an Entity-Relationship Diagram (ERD) as a basis for designing the data structure to manage the university.

The Ministry of Education and Training conducts surveys of universities. Each university has multiple faculties, and each faculty comprises several departments, with each department having numerous instructors.

To enhance the quality of education, universities stipulate that each instructor can only teach one subject, and each subject is managed by one instructor. Within the universities, each faculty oversees multiple classes, and each class consists of many students. Throughout their years at the university, each student must take multiple subjects, and each subject can have many participating students.

HINT

- **Identify entities**: university, Faculty, Department, Lecturer, Course, Class, Student.
- **Identify the attributes of the entities** and create the following table:

List of entities and attributes

Entities	Attribute
University	**UniversityID**, Name
Faculty	...
Department	
Lecturer	
...	

HINT

Identify relationships between entities:

- The relationship between School and Faculty is a second-order relationship, type 1-M.
- The relationship between Faculty and Department is a second-order relationship, type 1-M.
- …
- In particular, the relationship between Student and Subject is a second-order relationship, type M-M (with separate attributes such as "TestTime", "Score").

From ER Diagram to Relational Model

Overview:

- 1 entity = 1 relation
- attributes of entity ~ attributes of relation
- key of entity ~ key of relation

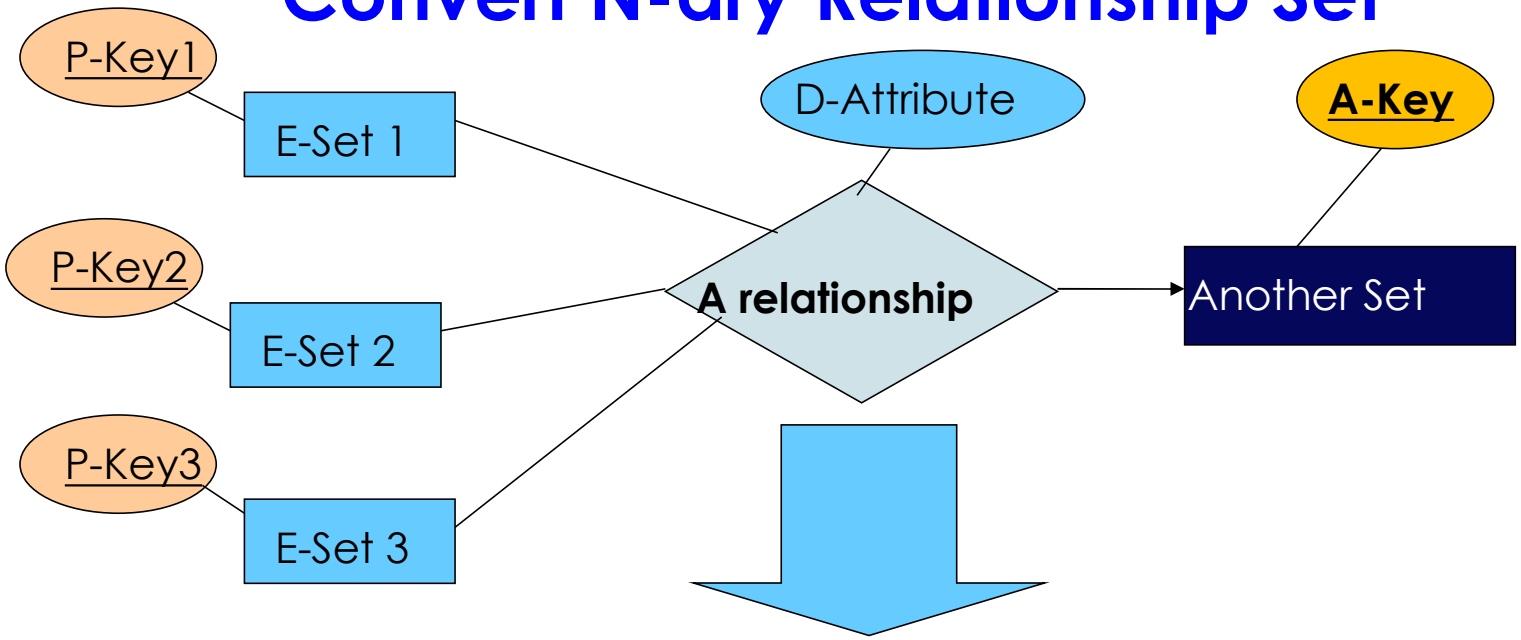
- Convert 1-1 relationship
- Convert 1- M relationship
  - Put key attribute of one-side to n-side
- Convert M-M relationship
  - Generate 1 relation, Primary key of this relation combined from two relations. Attributes of new relation ~ attributes of relationship (if have)

From ER Diagram to Relational Model

Convert 1-1 relationship

- For one-to-one relationship without total participation
  - Build a table with two columns, one column for each participating entity set's primary key. Add successive columns, one for each descriptive attributes of the relationship set (if any).
- For one-to-one relationship with one entity set having total participation
  - Augment one extra column on the right side of the table of the entity set with total participation, put in there the primary key of the entity set without complete participation as per to the relationship.

Convert N-ary Relationship Set

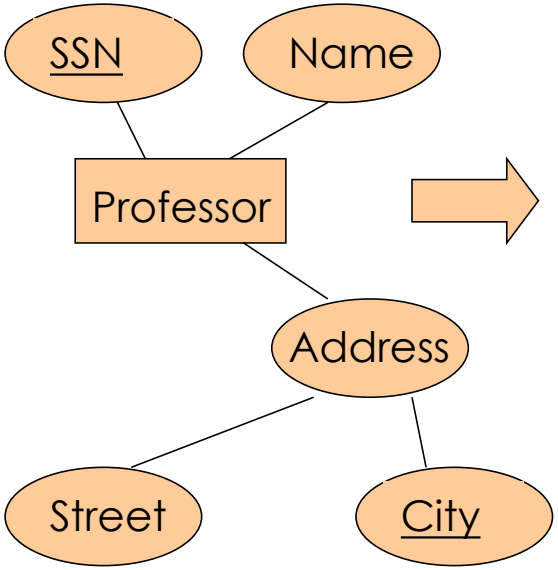


P-Key1	P-Key2	P-Key3	A-Key	D-Attribute
9999	8888	7777	6666	Yes
1234	5678	9012	3456	No

* Primary key of this table is *P-Key1 + P-Key2 + P-Key3*

Representing Composite Attribute

- Relational Model Indivisibility Rule Applies
- One column for each component attribute
- NO column for the composite attribute itself



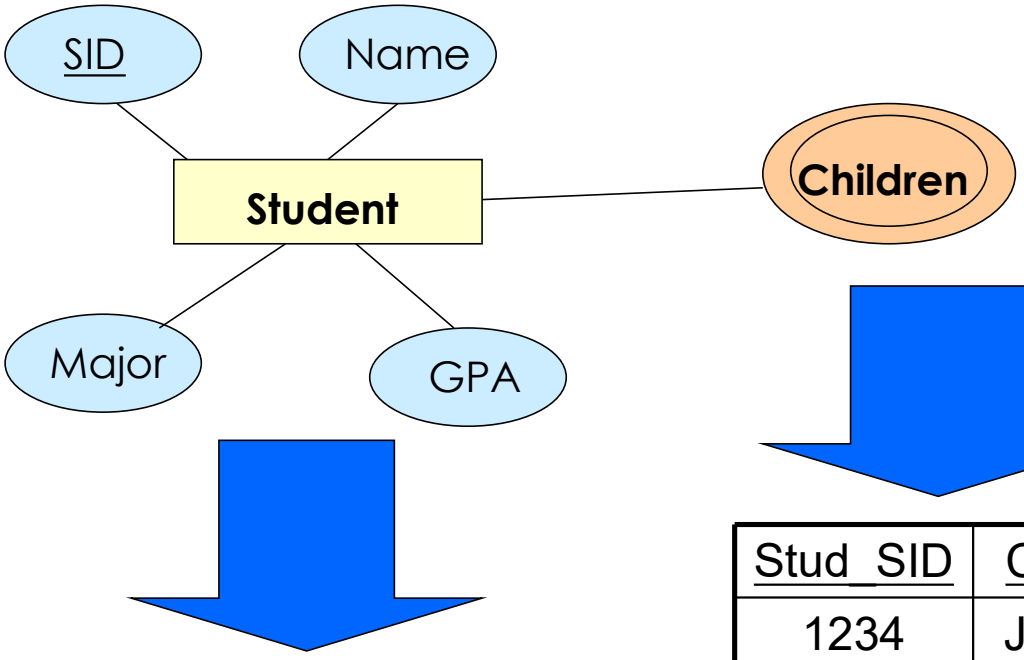
SSN	Name	Street	City
9999	Dr. Smith	50 1st St.	Fake City
8888	Dr. Lee	1 B St.	San Jose

Representing Multivalue Attribute

For each **multivalue attribute** in an entity set/relationship set

- Build a new relation schema with two columns
- One column for the primary keys of the entity set/relationship set that has the multivalue attribute
- Another column for the multivalue attributes. Each cell of this column holds only one value. So each value is represented as an unique tuple
- Primary key for this schema is the union of all attributes

Example – Multivalue attribute



The primary key for this table is Student_SID + Children, the union of all attributes

SID	Name	Major	GPA
1234	John	CS	2.8
5678	Homer	EE	3.6

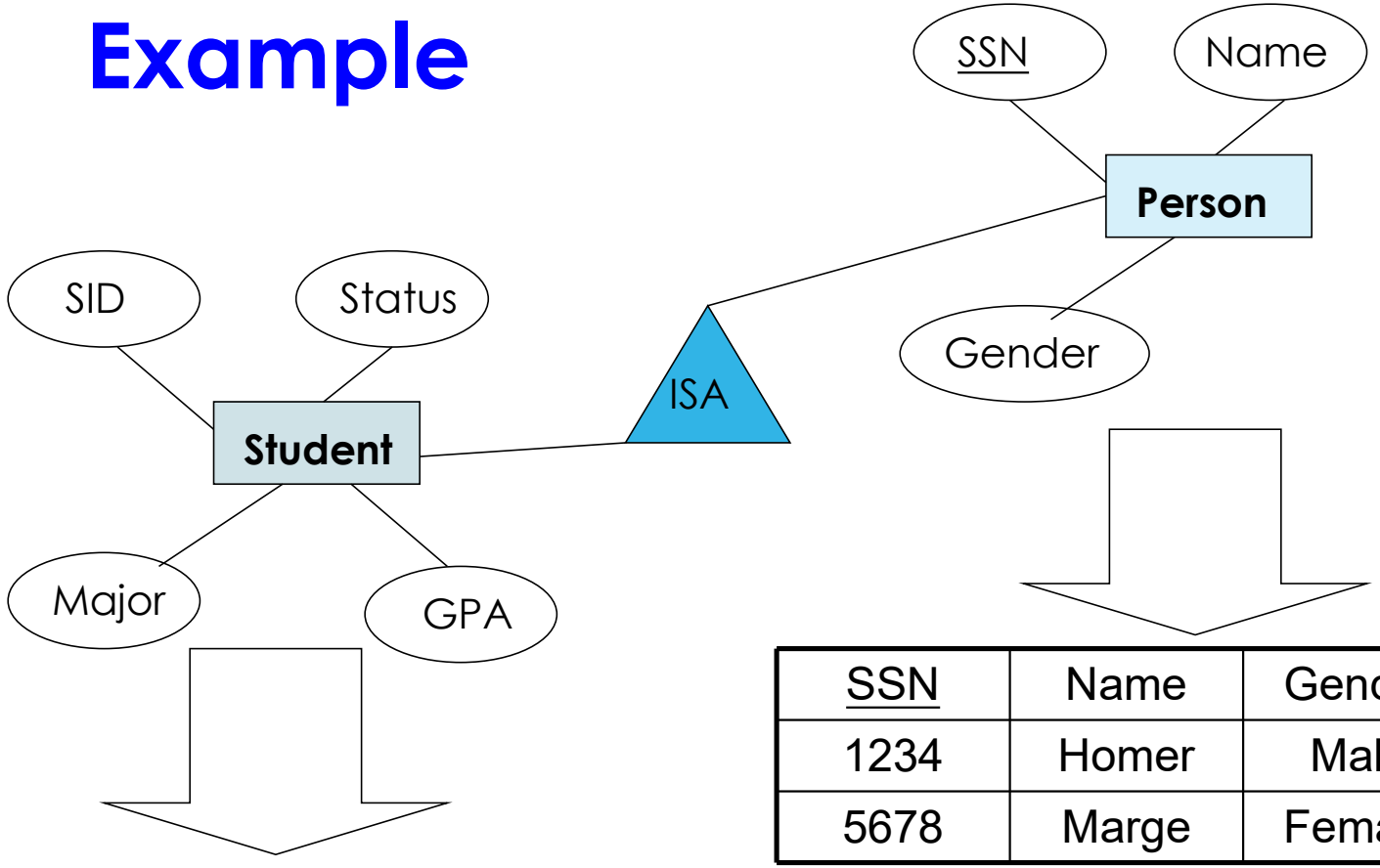
Stud_SID	Children
1234	Johnson
1234	Mary
5678	Bart
5678	Lisa
5678	Maggie

Representing Class Hierarchy

Two general approaches depending on disjointness and completeness

- For non-disjoint and/or non-complete class hierarchy:
  - create a table for each super class entity set according to normal entity set translation method.
  - Create a table for each subclass entity set with a column for each of the attributes of that entity set plus one for each attributes of the primary key of the super class entity set
  - This primary key from super class entity set is also used as the primary key for this new table

Example



SSN	Name	Gender
1234	Homer	Male
5678	Marge	Female

SSN	SID	Status	Major	GPA
1234	9999	Full	CS	2.8
5678	8888	Part	EE	3.6

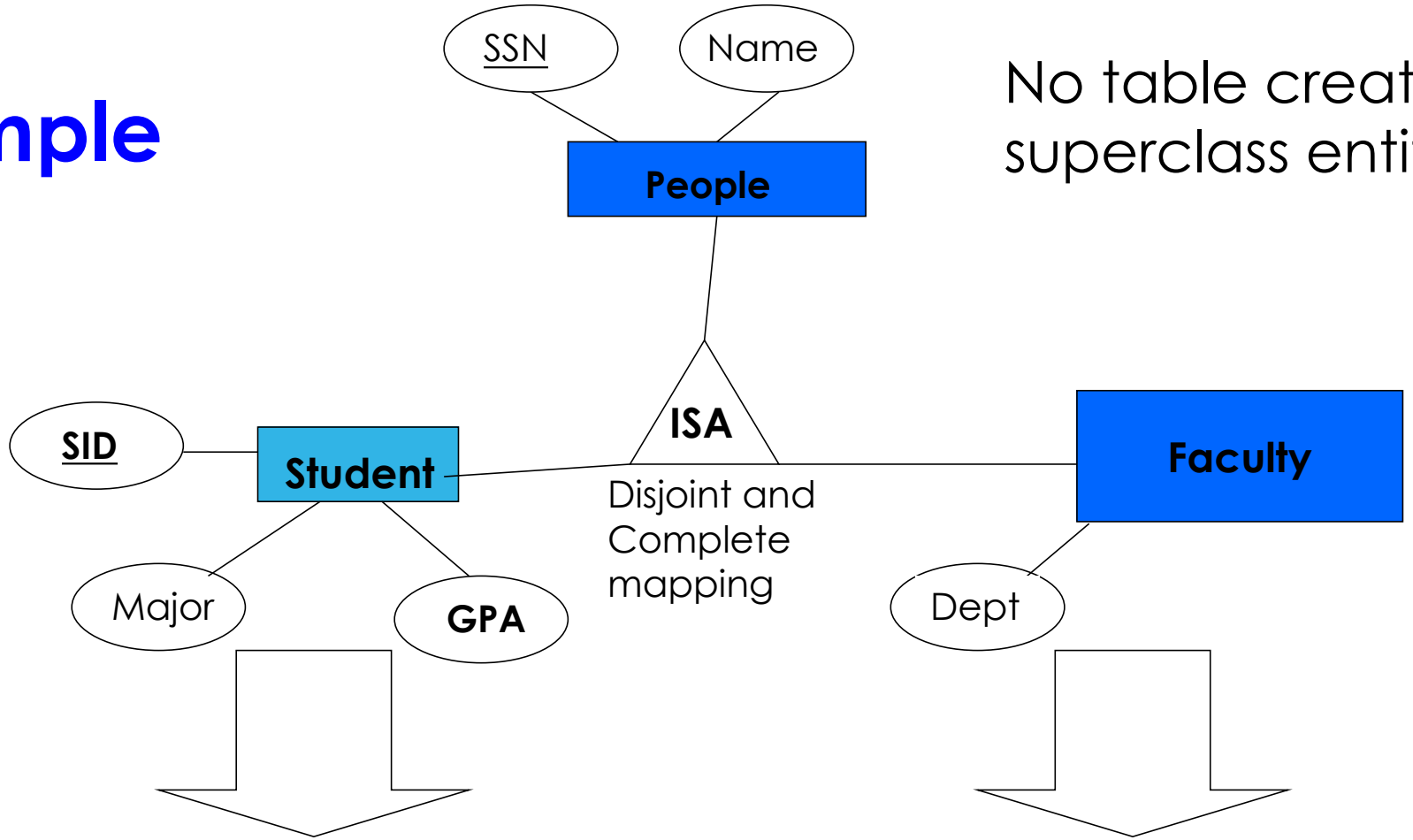
Representing Class Hierarchy

Two general approaches depending on disjointness and completeness

- For disjoint **AND** complete mapping class hierarchy:
- DO NOT create a table for the super class entity set
- Create a table for each subclass entity set include all attributes of that subclass entity set and attributes of the superclass entity set

Simple and Intuitive enough, need example?

Example

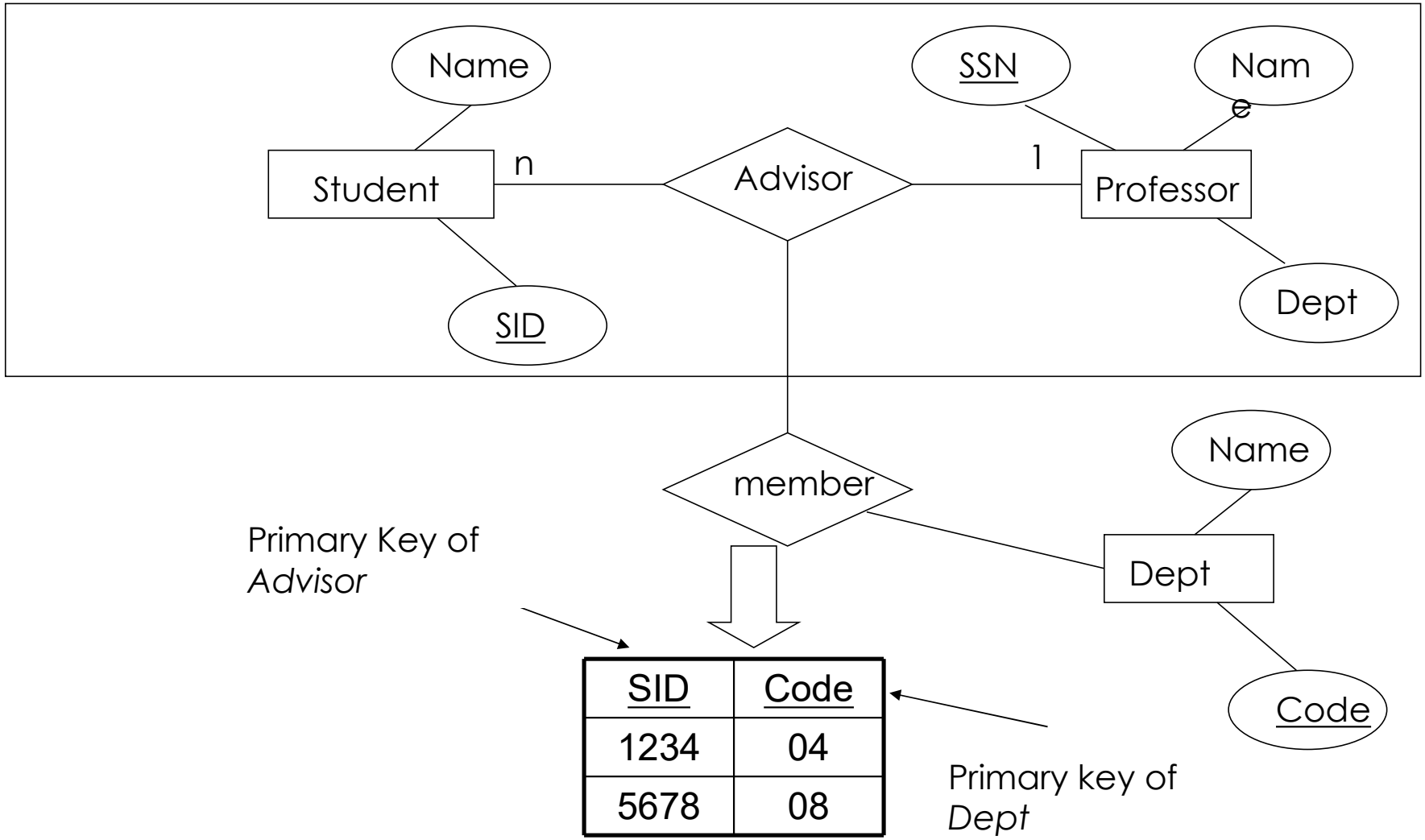


No table created for superclass entity set

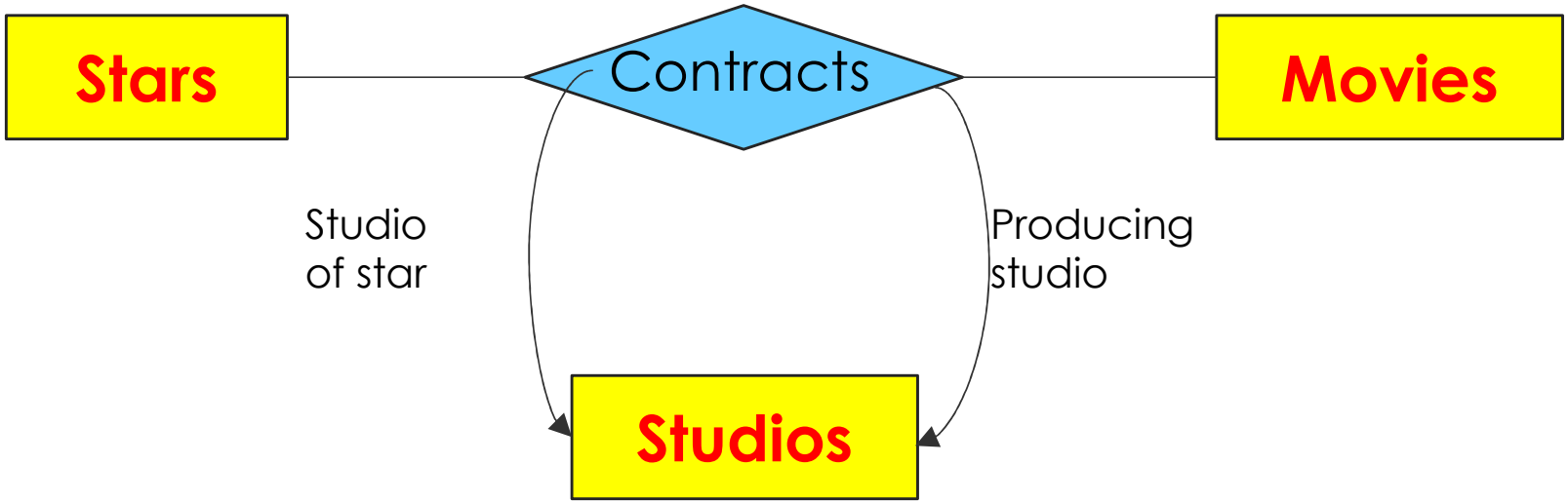
SSN	Name	SID	Major	GPA
1234	John	9999	CS	2.8
5678	Mary	8888	EE	3.6

SSN	Name	Dept
1234	Homer	C.S.
5678	Marge	Math

Representing Aggregation

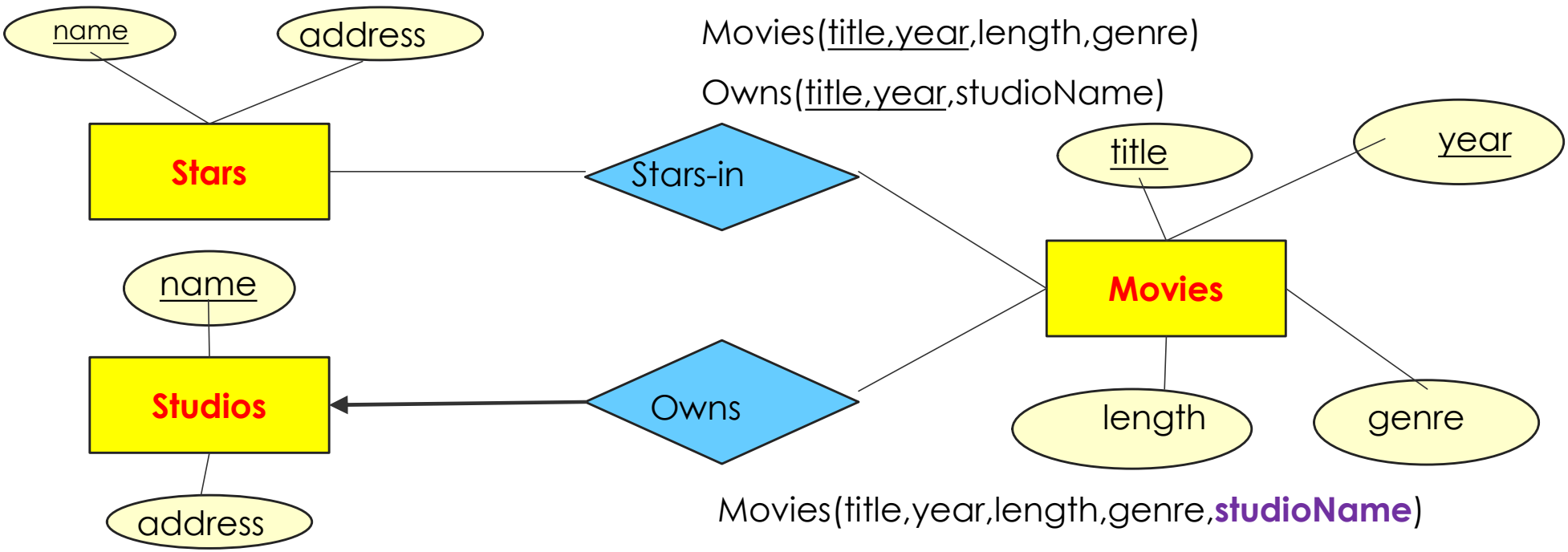


From E/R Relationship to Relations



Contracts(starName, title,year, studioOfStar_name, producingStudio_name)

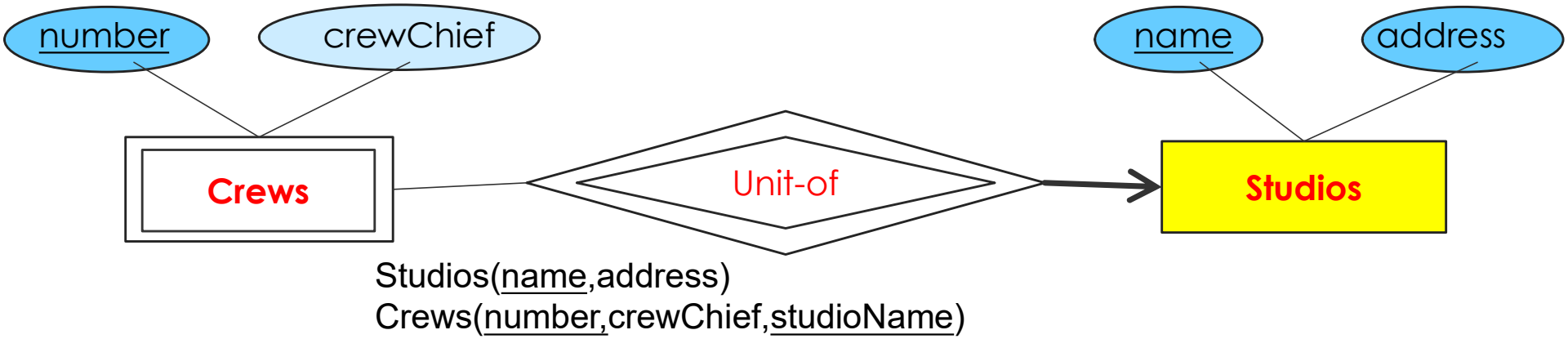
Combining Relations



Suppose an entity set E and a many-one relationship R from E to F. We can combine two relations E and R into one relation with a schema consisting of:

- All attributes of E,
- The key attributes of F, and all own attributes belonging to relationship R

Handling Weak Entity Sets



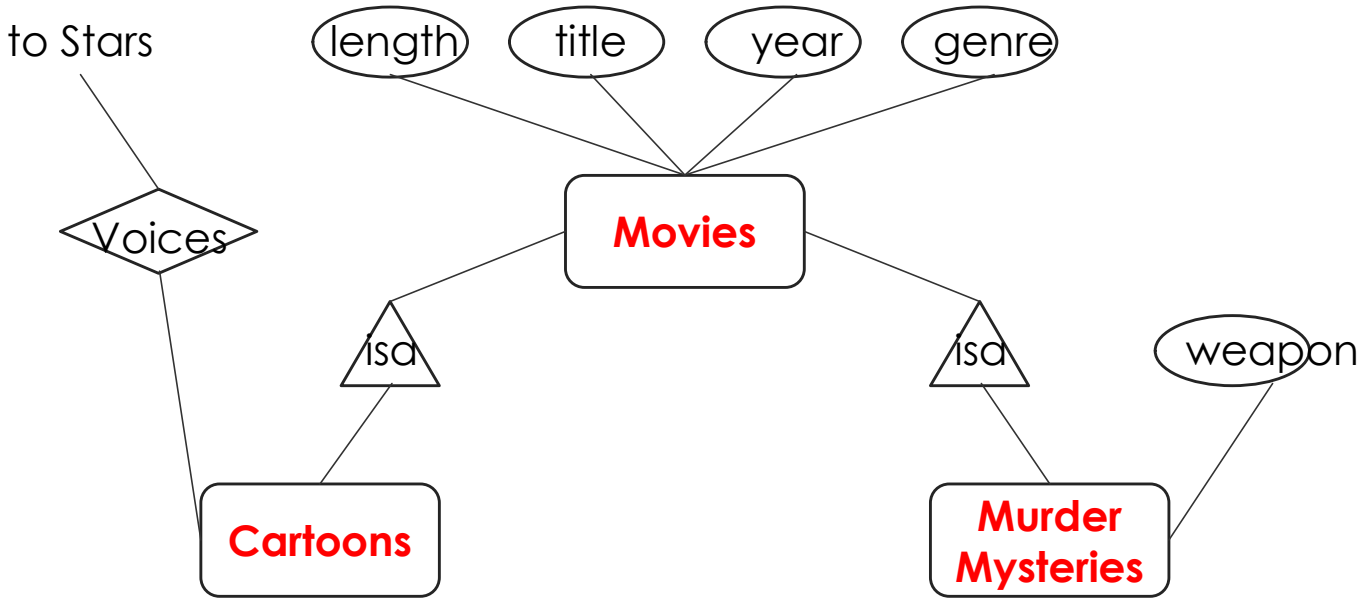
If W is a weak entity set, construct for W a relation whose schema consists of:

- All attributes of W
- All own attributes of supporting relationships for W
- For each supporting relationship for W, say a many-one relationship from W to entity set E, all the key attributes of E

- Rename attributes, if necessary, to avoid name conflicts
- Do not construct a relation for any supporting relationship for W

Converting Subclass Structures to Relations

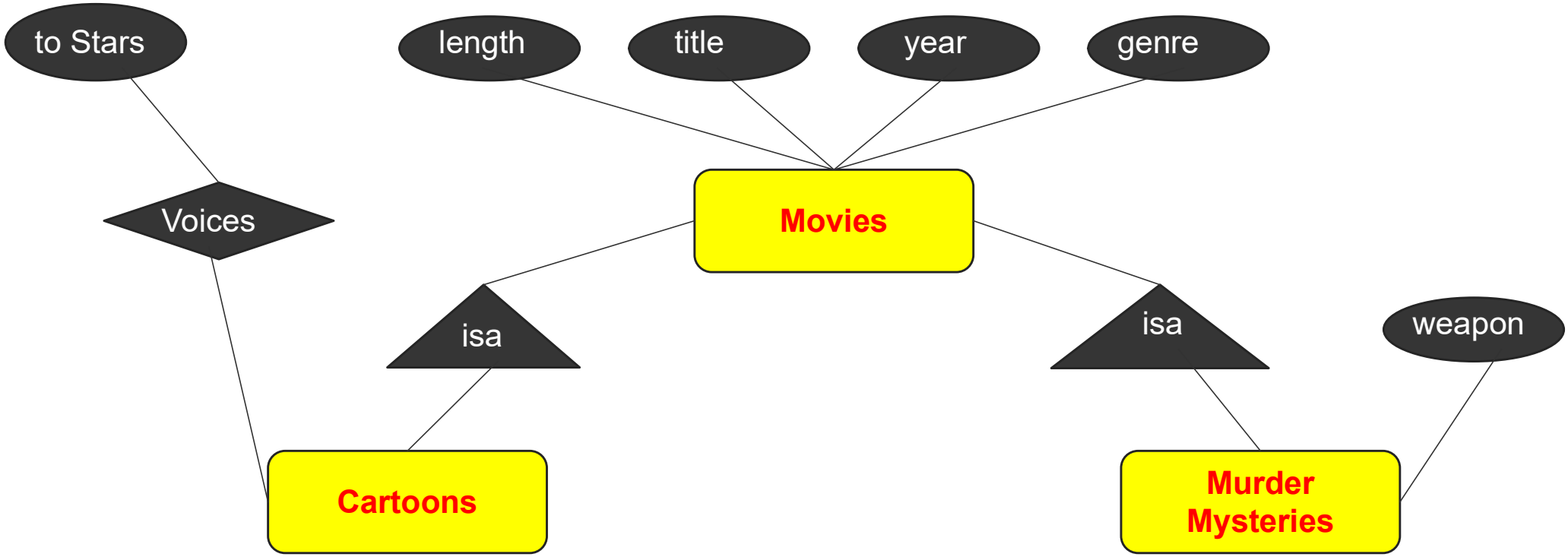
- How we convert this structure to relations?



Converting Subclass Structures to Relations

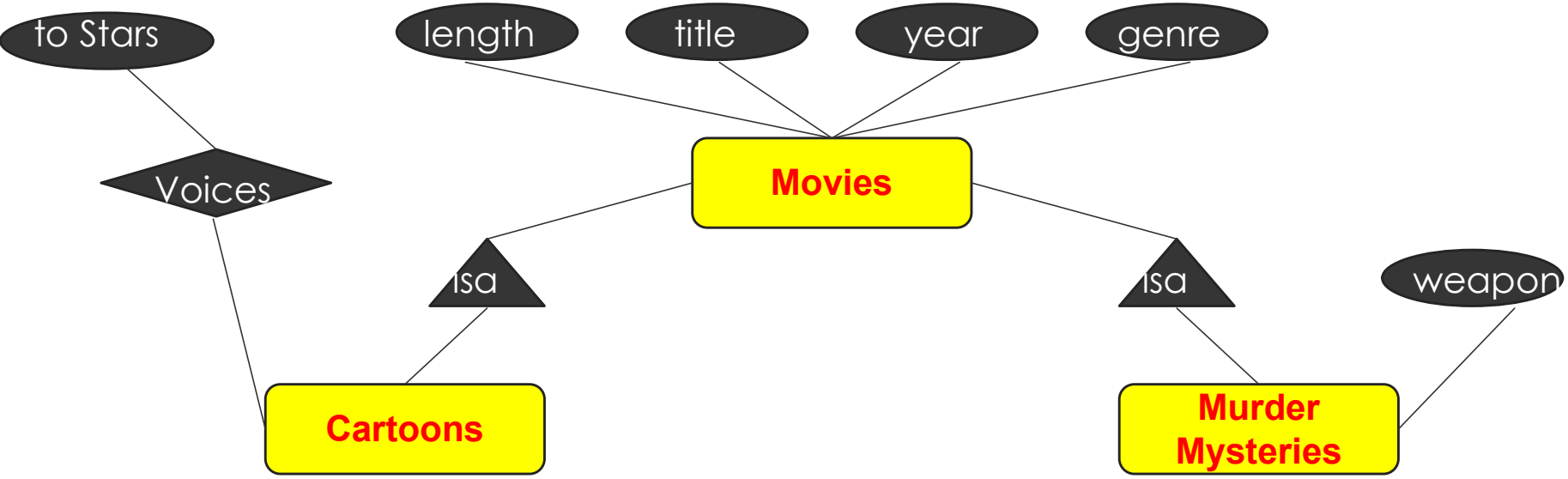
- The principal conversion strategies
  - Follow E/R viewpoint
    - For each entity set E in the hierarchy, create a relation that includes the key attributes from the root and any attributes belong to E
  - Treat entities as object-oriented
    - For each possible subtree that includes the root, create one relation, whose schema includes all the attributes of all the entity sets in the subtree
  - Use null values
    - Create only one relation with all attributes of all entity sets in the hierarchy. Each entity is represented by one tuple, and that tuple has a NULL value for whatever attributes the entity does not have

E/R Style Conversion



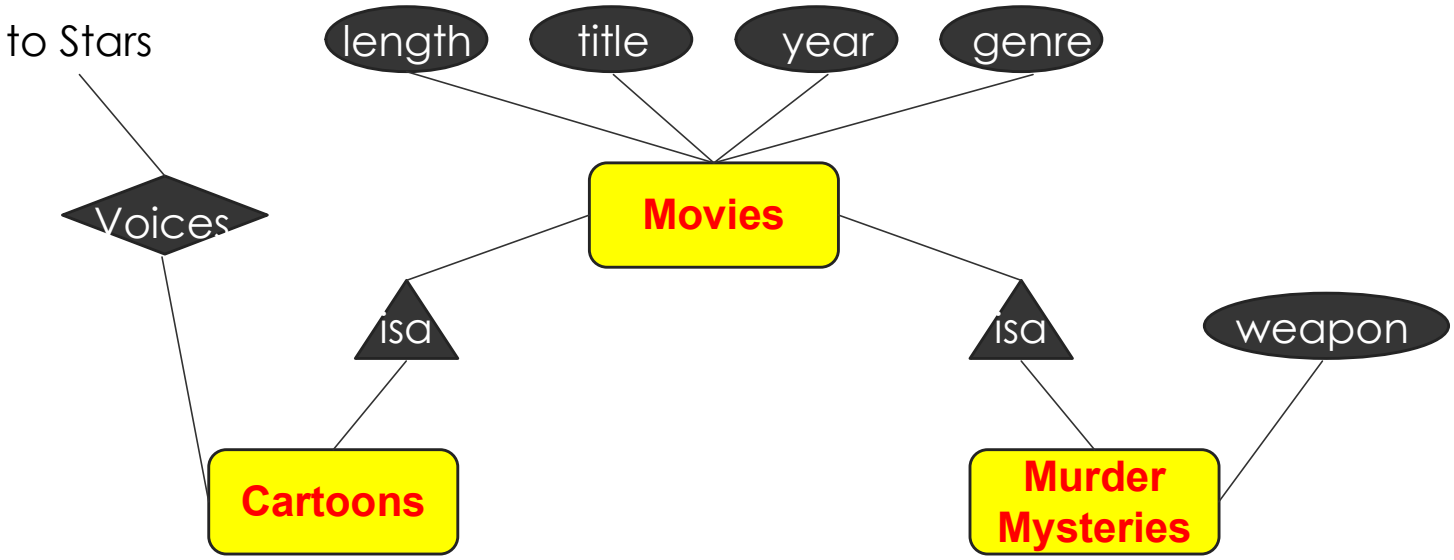
Movies(title,year,length,genre)
MurderMysteries(title,year,weapon)
~~Cartoons(title,year)~~ ← remove
Voices(title,year,starName)

An Object-Oriented Approach



Movies(title,year,length,genre)
MoviesC(title,year,length,genre)
MoviesMM(title,year,length,genre,weapon)
MoviesCMM(title,year,length,genre,weapon)

Using Null Values



Movie(title,year,length,genre,weapon)

Unified Modeling Language –self studying

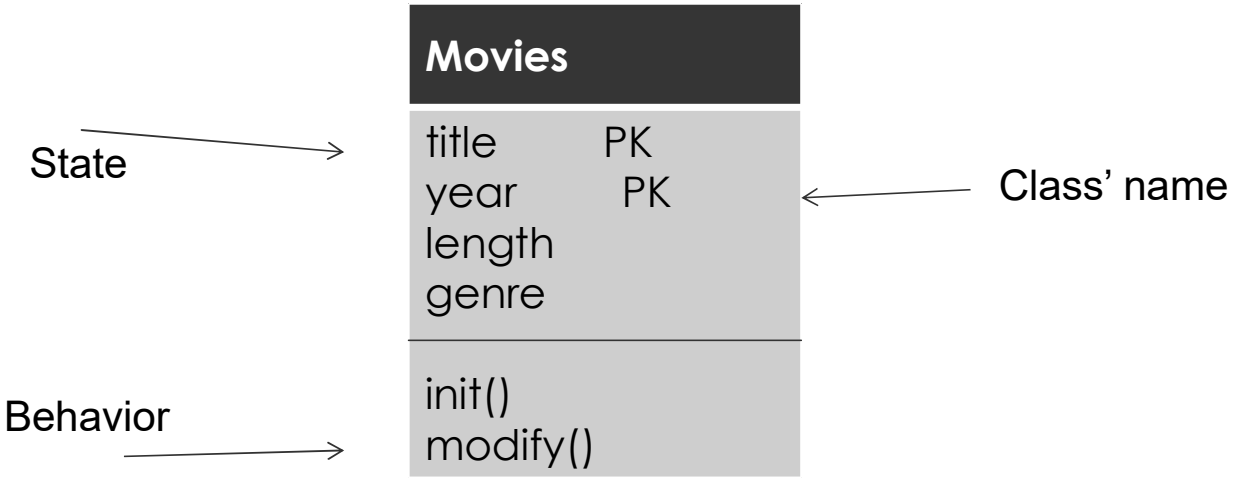
- Introduction
  - UML is designed to model software in an object-oriented style, but has been adapted as a database modeling language
  - UML offers much the same capabilities as the E/R model, with the exception of multi-way relationships, only binary relationships in UML.

UML vs. E/R Model

UML	E/R Model
Class	Entity Set
Association	Binary relationship
Association class	Attributes on a relationship
Subclass	is-a hierarchy
Aggregation	Many-one relationship
Composition	Many-one relationship with referential integrity

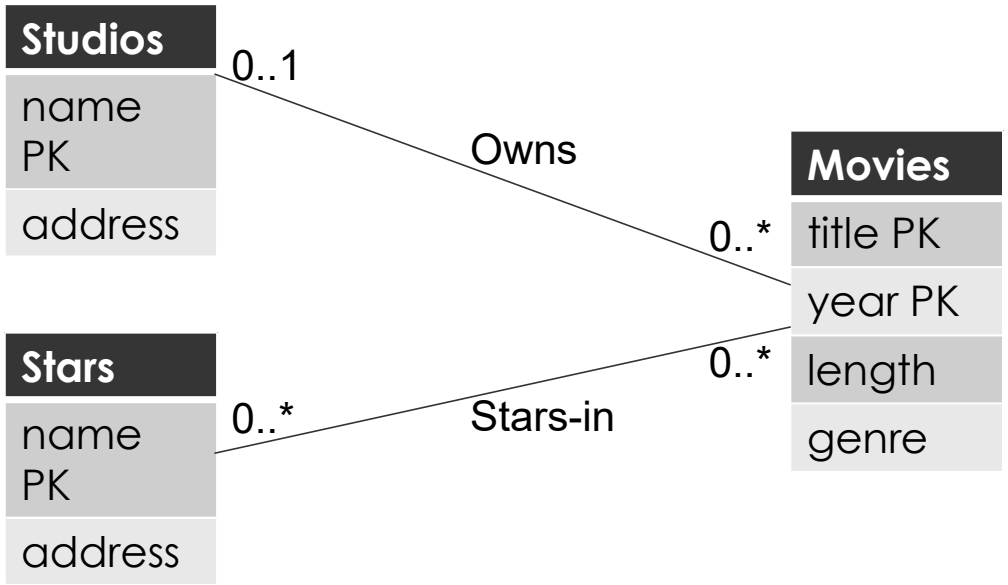
Figure 4.34: Comparison between UML and E/R terminology

UML Classes


Movies
title PK
year PK
length
genre
init()
modify()
State
Behavior
Class' name

Associations

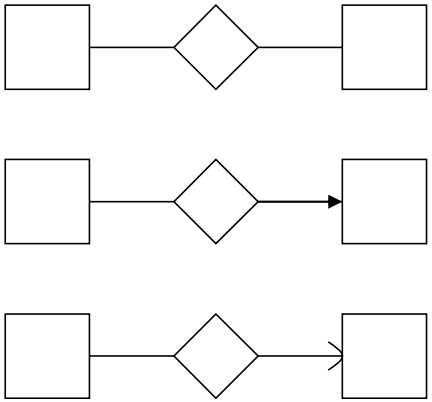
- Consider an associations between Movies, Stars, and Studios in UML



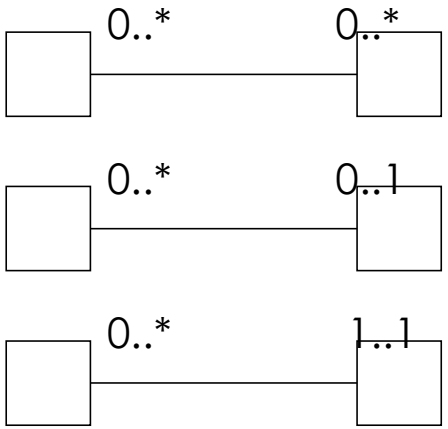
Associations

- Comparison with E/R Multiplicities

E/R

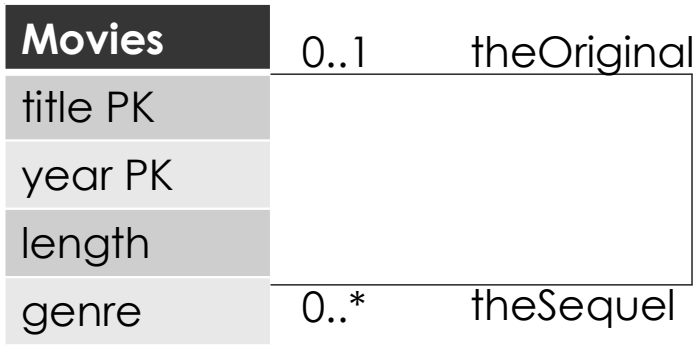


UML

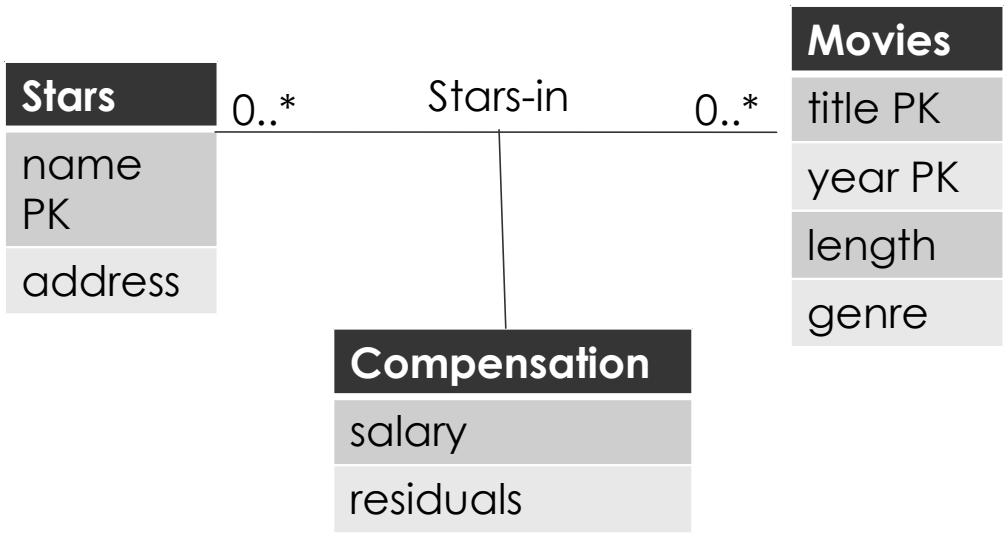


Self-Associations

- An association can have both ends at the same class; such an association is called a self-association
- Example



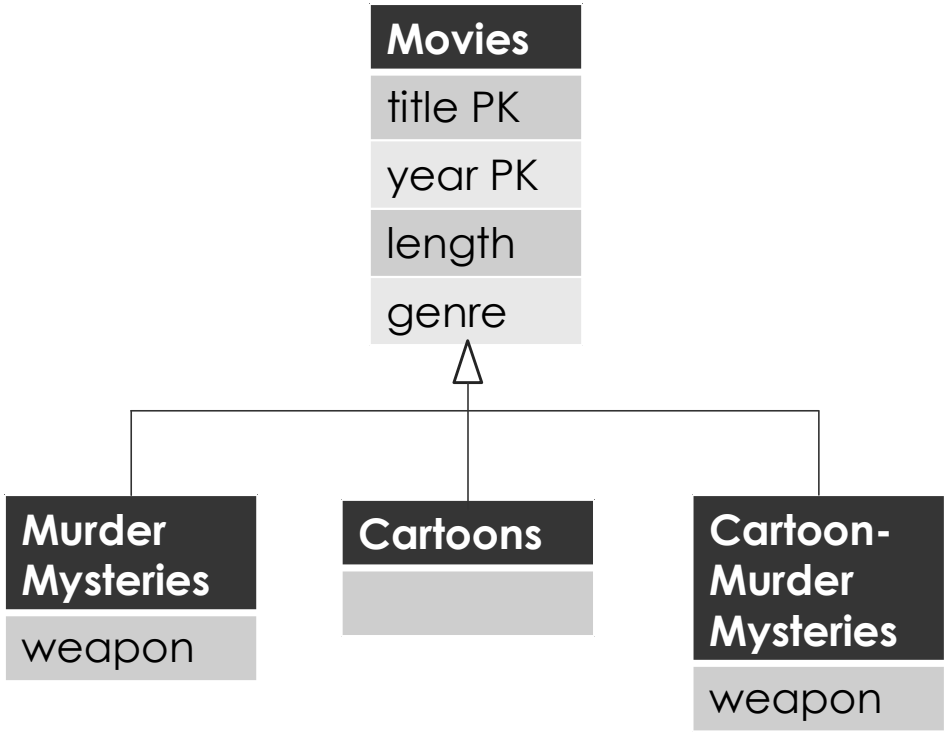
Association Classes



Subclasses in UML

- Consider Movies and its three subclasses

Figure 4.40: Cartoons and murder mysteries as disjoint subclasses of movies



Aggregations and Compositions

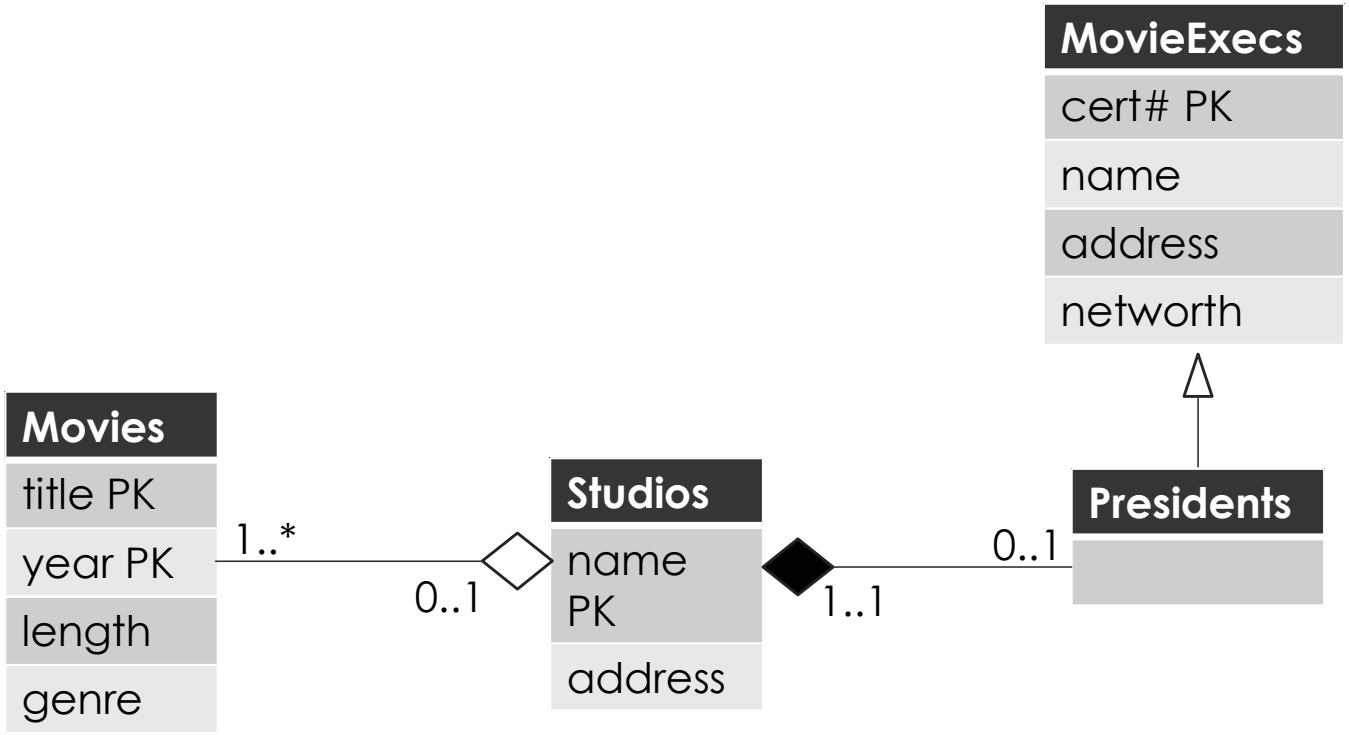
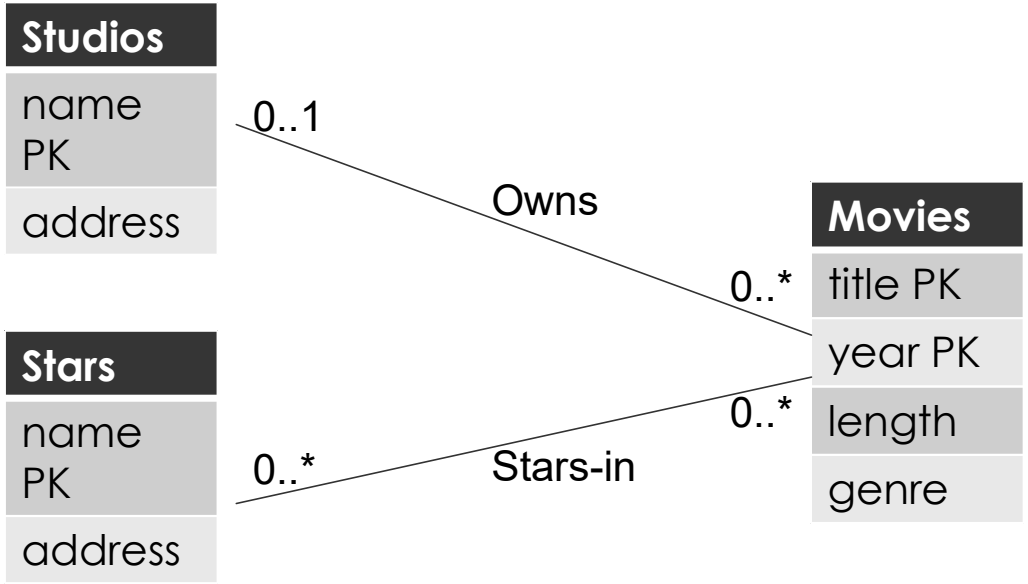


Figure 4.41: An aggregation from Movies to Studios and a composition from Presidents to Studios

UML-to-Relations Basics

- Classes to Relations
  - For each class, create a relation
    - name is the name of the class
    - attributes are the attributes of the class
- Associations to Relations
  - For each association, create a relation
    - name is the name of that association
    - attributes are the key attributes of the two connected classes

UML-to-Relations Basics



Movies(<u>title,year</u>,length,genre)
Stars(<u>name</u>,address)
Studios(<u>name</u>,address)

Stars-In(<u>movieTitle,movieYear,starName</u>)
Owns(<u>movieTitle,movieYear,</u>studioName)

From UML Subclasses to Relations

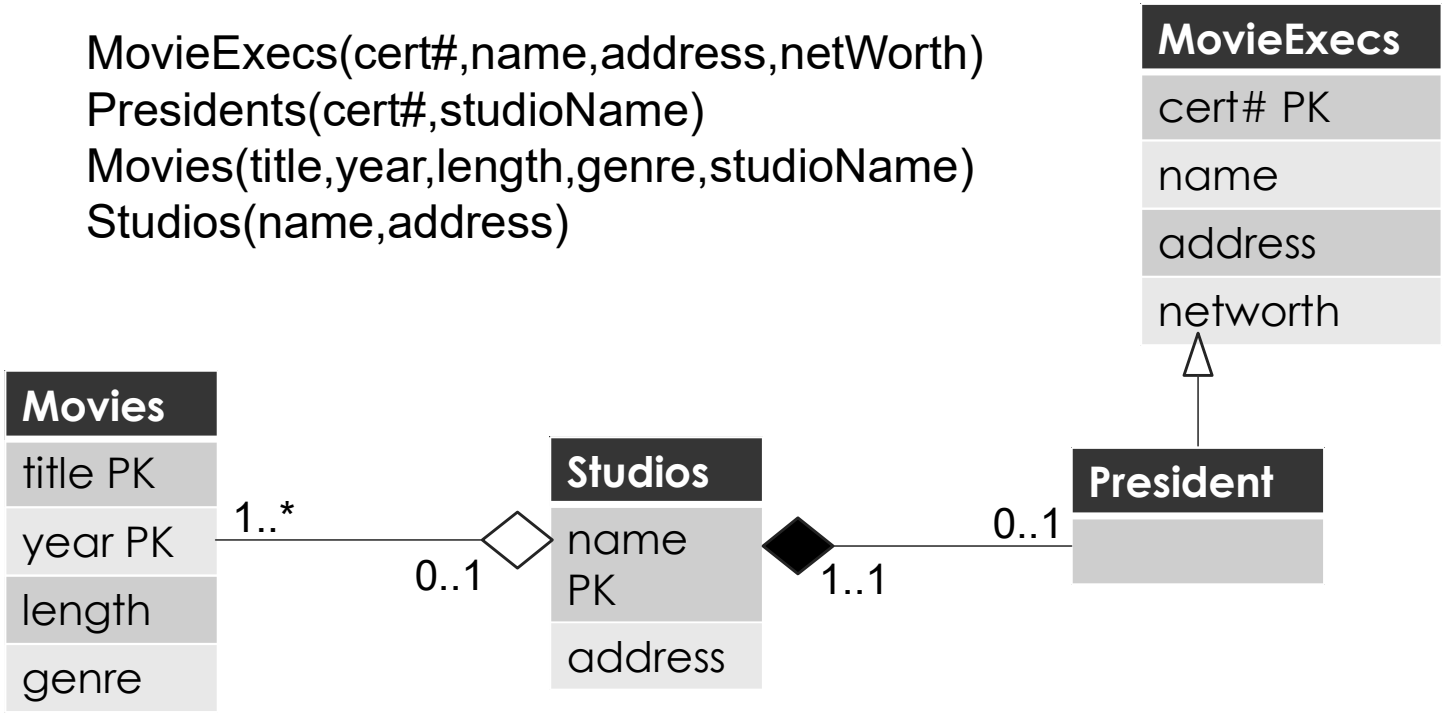
- We can use any of the three strategies outlined for E/R to convert a class and its subclasses to relations
  - E/R-style: each subclass' relation stores only its own attributes, plus key
  - OO-style: relations store attributes of subclass and all super-classes
  - Nulls: One relation, with NULL's as needed

From Aggregations and Composition to Relation

- No relation for the aggregation or composition
- Add to the relation for the class at the non-diamond end the key attribute(s) of the class at the diamond end
  - In the case of an aggregation, it is possible that these attributes can be null

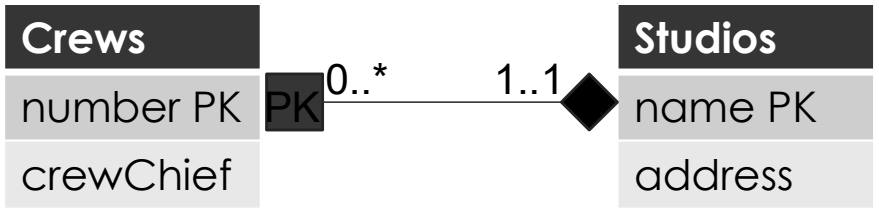
From Aggregations and Composition to Relation

MovieExecs(cert#,name,address,netWorth)
Presidents(cert#,studioName)
Movies(title,year,length,genre,studioName)
Studios(name,address)



The UML Analog of Weak Entity Sets

- We use the composition, which goes from the weak class to the supporting class, for a weak entity set
- Example: Studios(name,address)
  Crews(number,crewChief,studioName)



EXERCISE 1

Exercise 4.1.1: Design a database for a bank, including information about customers and their accounts. Information about a customer includes their name, address, phone, and Social Security number. Accounts have numbers, types (e.g., savings, checking) and balances. Also record the customer(s) who own an account. Draw the E/R diagram for this database. Be sure to include arrows where appropriate, to indicate the multiplicity of a relationship.

EXERCISE 2 - Convert ERD to RD

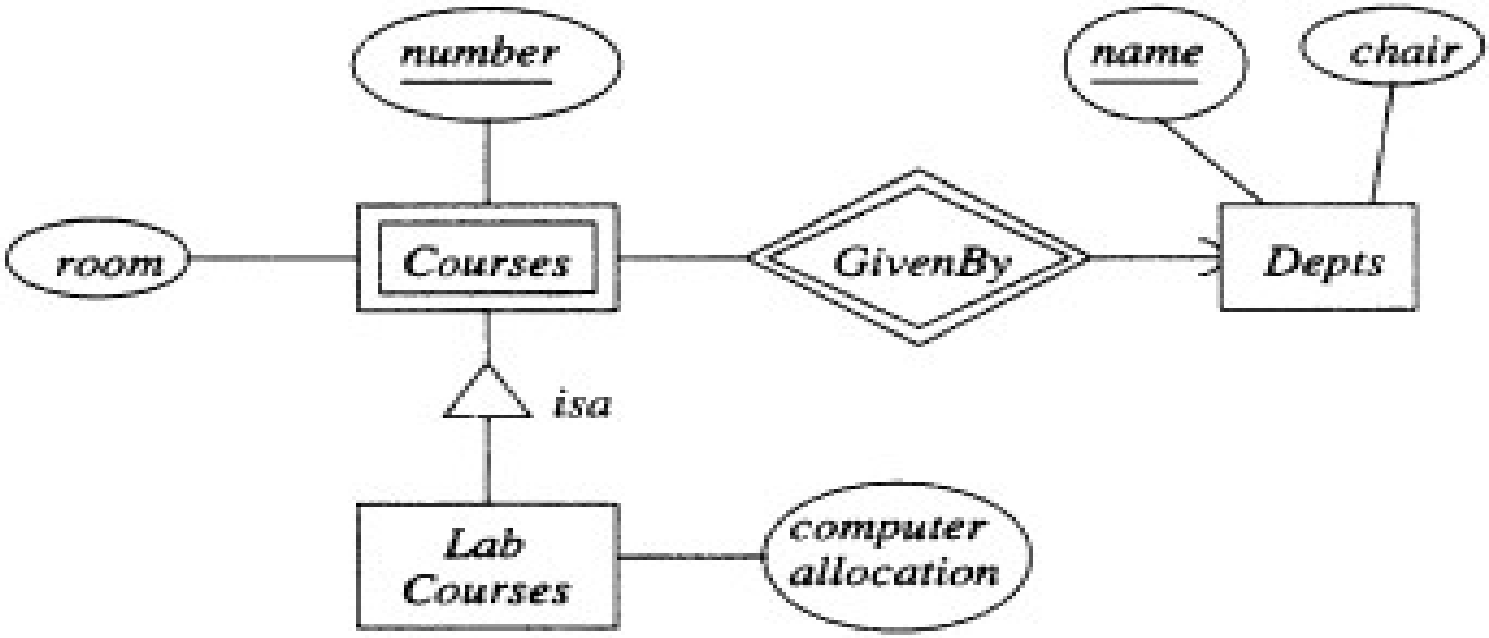


Figure 4.32: E/R diagram for Exercise 4.6.1

Steps to Convert ERD to Relational Schema

1. **Identify Entities**
   - Convert each strong entity into a table.
   - Attributes of the entity become columns in the table.
   - Choose the primary key (PK) from the entity's key attributes.
2. **Handle Weak Entities**
   - Create a table for each weak entity.
   - Include attributes of the weak entity and add a foreign key (FK) referencing the strong entity it depends on.
   - The combination of the FK and the weak entity's discriminator (if any) becomes the PK.

3. Map Relationships

- 1:1 Relationship: Add FK to one of the participating entity tables (preferably on the side with total participation).
- 1:N Relationship: Add FK to the "many" side referencing the "one" side.
- M:N Relationship: Create a new table. Include FKs referencing the primary keys of the participating entities and any attributes of the relationship. Combine the FKs to form the PK.

4. Handle Multivalued Attributes

- Create a new table.
- Include the multivalued attribute and the PK of the entity it belongs to as FKs.
- Combine the PK of the entity and the multivalued attribute to form the PK of the new table.