Fpt University

Chapter 4. High-Level Database Model

Objectives

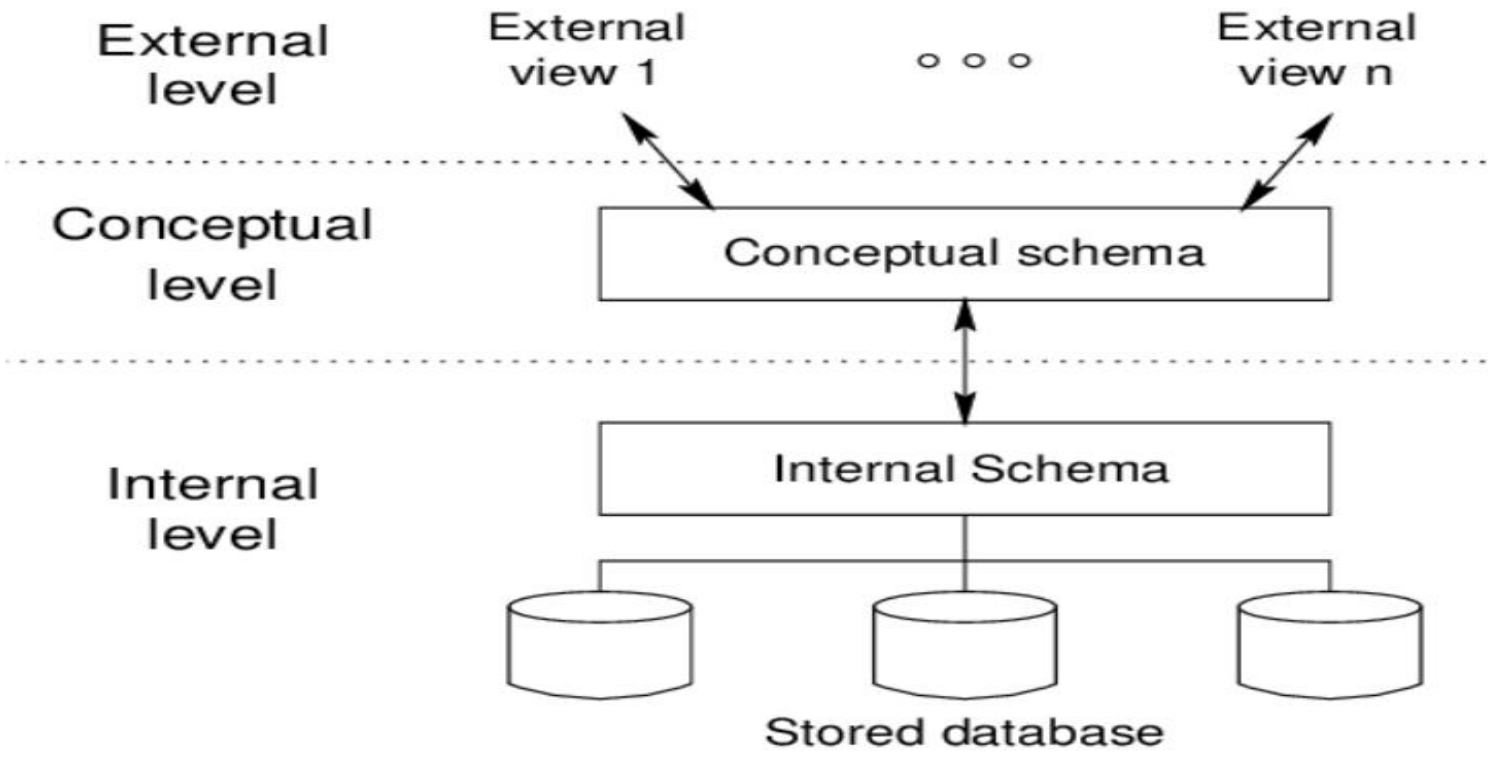
- Understand the Database Design Process
- Understand data modeling basing on entity relationship
- Design a suitable database adapted business requirements in reality

Fpt University

Contents

- Database design process
- Entity relationship model
- What are entity, entity set, attribute, relationship?
- Entity Relationship Diagram (ERD)
- Attributes on Relationships
- Weak Entities
- Sub-class
- From ER Relationship to Relations

Data model - Overview



Database modeling and implementation process

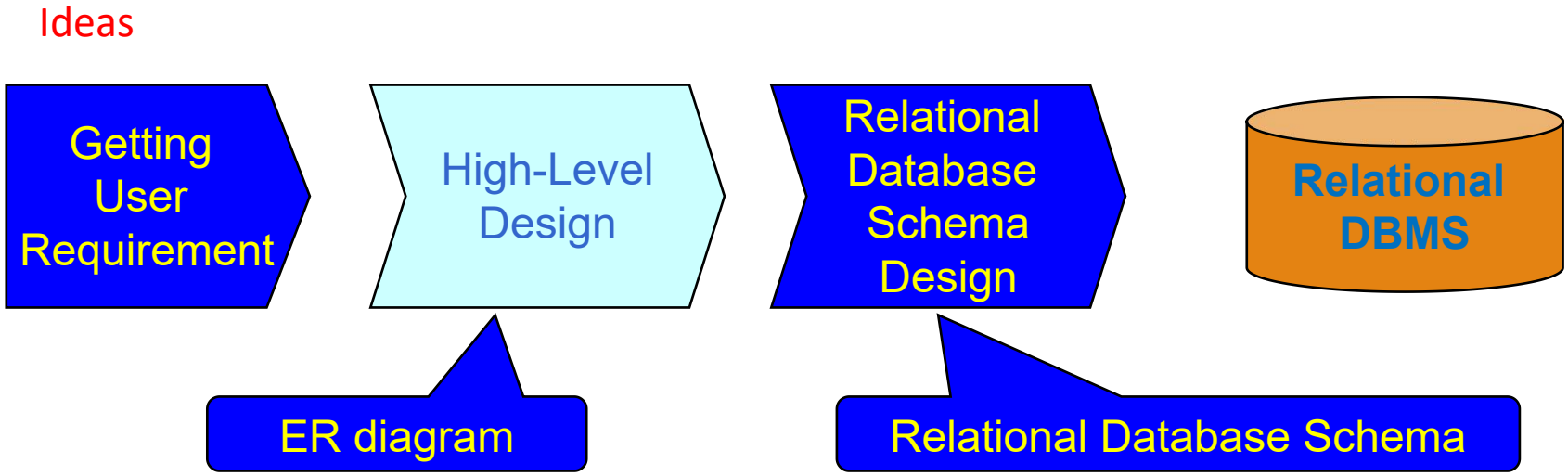


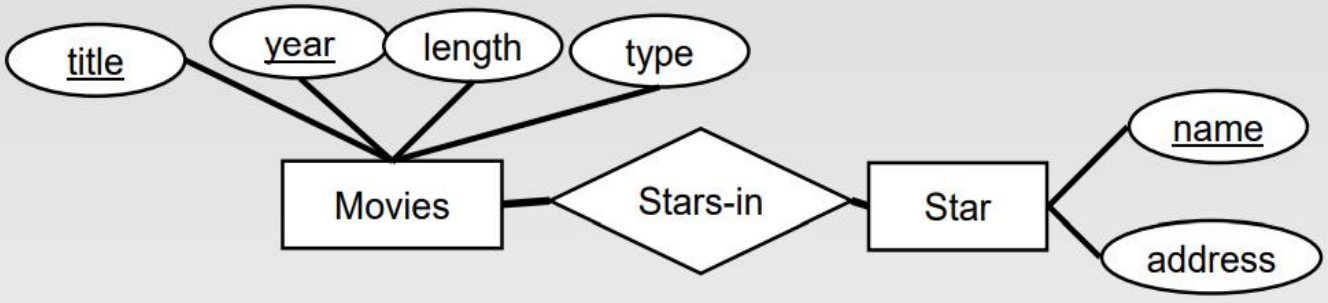
Figure 4.1: The database modeling and implementation process

Steps in Database Design

1. **Requirements Analysis**
   - user needs; what must database do?
2. **Conceptual Design**
   - high level description (Entity Relationship diagram-ERD)
3. **Logical Design**
   - translate ERD into DBMS data model
4. **Schema Refinement (1NF, 2NF, 3NF, BCNF→3.5NF)**
   - consistency, normalization
5. **Physical Design**
   - indexes, disk layout
6. **Security Design**
   - who accesses what, and how

Steps in Database Design

Ex: From Entity Sets to Relations



Movies (title, year, length, genre) **Stars (name, address)**

Steps in Database Design

FRAMEWORK FOR ER

DB design is a serious and complex business

- Clients want to have a database→ but they don't know what they want in it or how it should look
- ER diagrams are the first step to creating a database
- Sketching the main DB components --> Efficient way to develop a database

--> It is much better to start with a good design, rather than try to repair a poor design

Fpt University

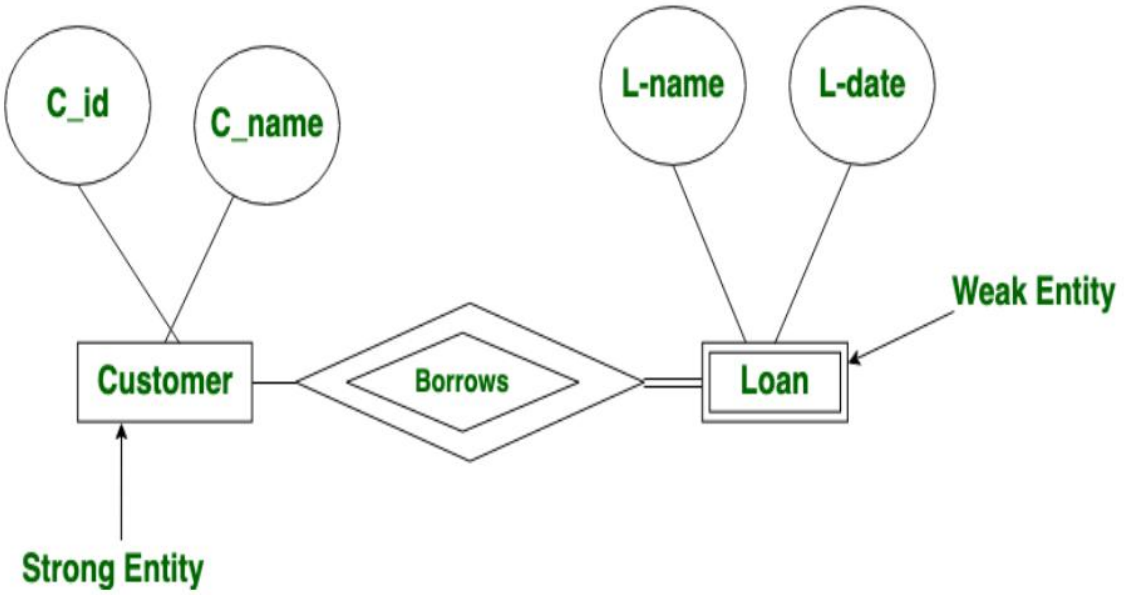
Steps in Database Design

PURPOSE OF ER MODEL

- Shows the logical structure of the database
- Some constraints
- Basically, through the entity model:
    1. things --> called: entity sets
    2. attributes --> properties of entities,
    3. and relationships between entities

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.



Fpt University

What is an ER diagram?

- An Entity Relationship (ER) Diagram is a type of flowchart that illustrates how “entities” such as people, objects or concepts relate to each other within a system
- ERDs or ER Models, they use a defined set of symbols such as rectangles, diamonds, ovals and connecting lines to depict the interconnectedness of entities, relationships and their attributes
- They mirror grammatical structure: entities as nouns and relationships as verbs

Fpt University

Limitations of ER diagrams and models

- Only for relational data: Understand that the purpose is to show relationships. ER diagrams show only that relational structure.
- Not for unstructured data
- Difficulty integrating with an existing database: Using ER Models to integrate with an existing database can be a challenge because of the different architectures

Fpt University

How to draw a basic ER diagram

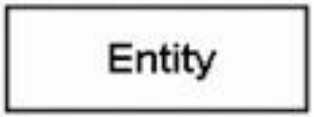
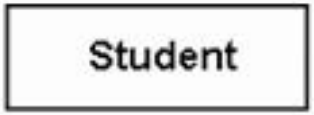
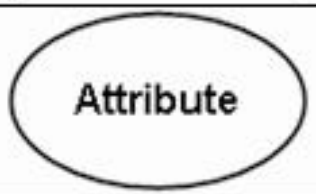
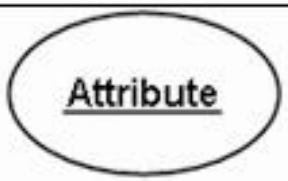
- Purpose and scope: Define the purpose and scope of what you're analyzing or modeling.
- Entities: Identify the entities that are involved. They are nouns
- Relationships:

  + Determine how the entities are all related

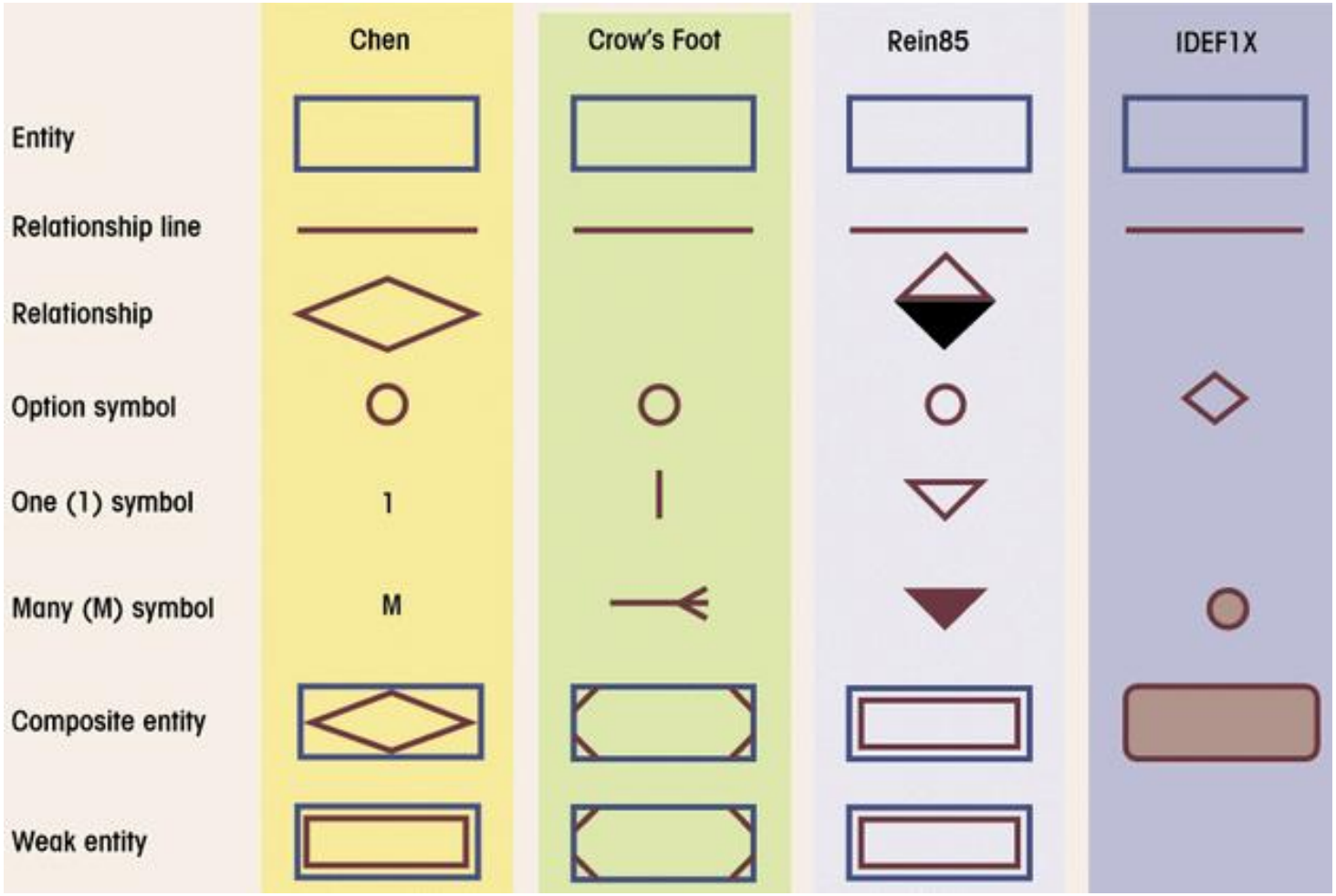
  + Draw lines between them to signify the relationships and label them

  + Some entities may not be related, and that's fine
- Attributes: Layer in more detail by adding key attributes of entities.
- Cardinality: Show whether the relationship is 1-1, 1-many or many-to-many

Entity Relationship Diagram - Notations

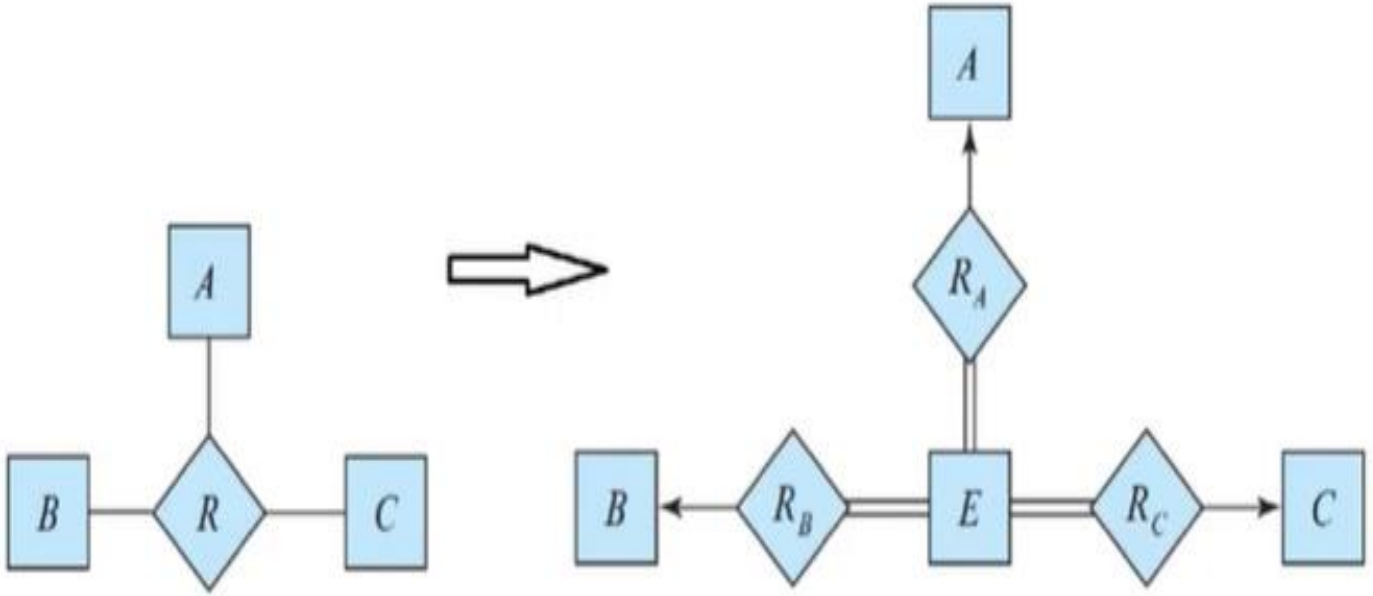
Component	Symbol	Example
Entity		
Weak Entity		
Attribute		
Relationship		
Key Attribute		

Comparison of ER Modeling notations



Comparison of E-R Modeling notations

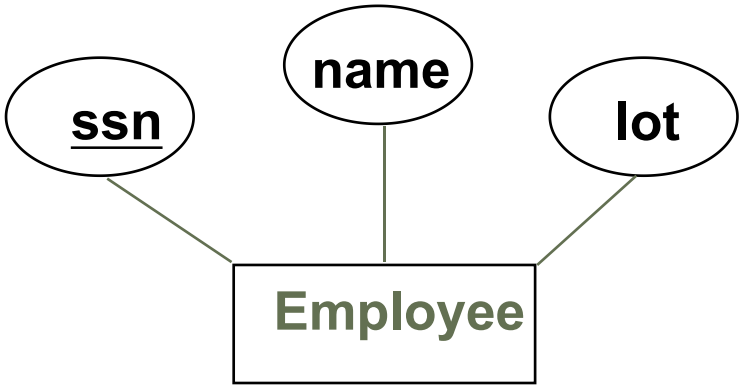
Convert link 3 to 2



Chen Diagram Symbols

- ***Entity***
  - Real-world thing, distinguishable from other objects.
  - Noun phrase
  - Entity described by set of *attributes*
  - Each entity must have a unique identifier, known as a primary key, which allows it to be distinguished from other entities (the attributes with an underline beneath the name)
  - Entities may also have additional attributes, which are used to describe its properties.

EX: name, lot,…

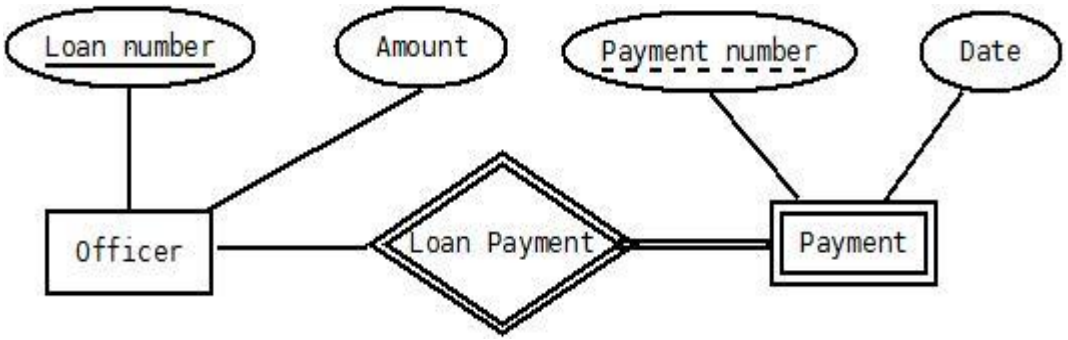


ERD – How to construct

- **Strong Entity**
  + A **strong entity** is **not dependent** on **any other entity** in the schema
  + A **strong entity** will always have a **primary key**
  + **Strong entities** are represented by a **single rectangle**

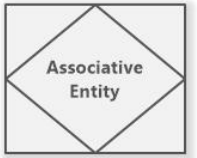
Chen Diagram Symbols

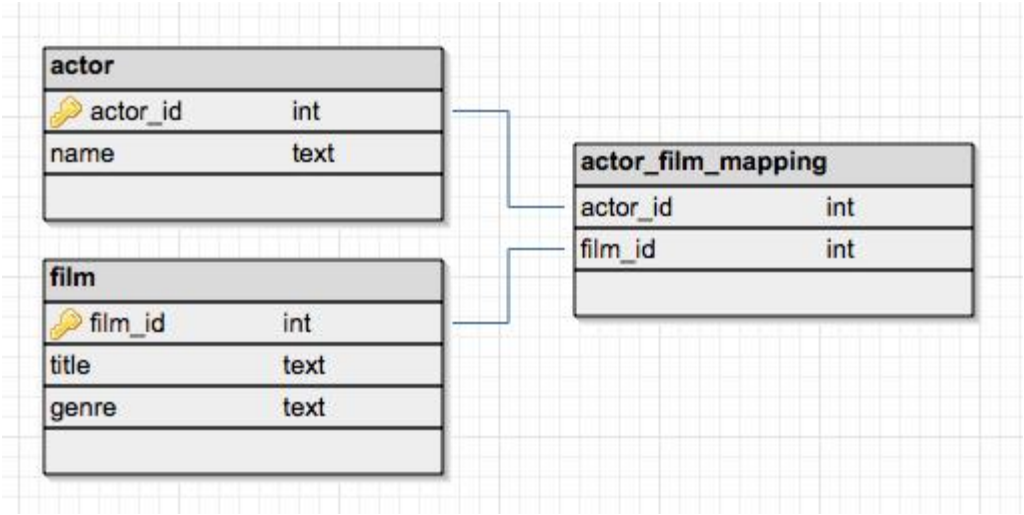
- ***Weak Entity***
  - A weak entity is a type of entity that does not have any unique key for the attribute tuples.
  - A weak entity depends on another entity
  - A weak entity is an entity that cannot exist without an entity it depends
  - It is depicted as a rectangle with a double border



Chen Diagram Symbols

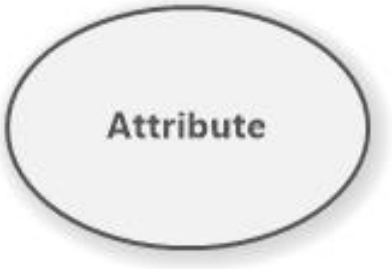
- ***Associative entity***
  - An associative entity represents a many-to-many relationship that is expressed by another entity (or a table in a database)
  - An associative entity is a specific type of a weak entity
  - Depicted as a rectangle with a diamond inside.





Chen Diagram Symbols

- ***Attribute***
  - Attribute contains a piece of information that describes an entity
  - Attribute provides more detail about the entity and expresses its characteristics
  - An attribute is depicted as an ellipse



Chen Diagram Symbols

- ***Key Attribute***
  - The key attributes define a tuple within an entity
  - Key attribute used to establish relationships between entities
  - A key attribute is represented by an oval with underlined name

Chen Diagram Symbols

- ***Weak Key Attribute***
  - - A weak key attribute (or partial key) is an attribute that in combination with the owner's key creates a key for a weak entity
  - - A weak key attribute can be used to identify instances of an entity, but is not sufficient to uniquely identify each instance
  - - A weak key attribute is represented by an oval with dashed underlining of the name

Chen Diagram Symbols

- ***Multi-Value Attribute***

- A multi-value attribute can have multiple distinctive values for a single instance of an entity

- Represented by a double oval with the attribute name inside

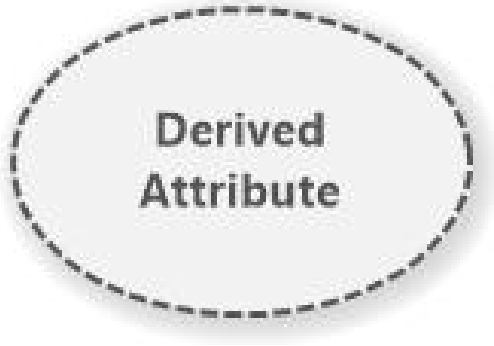
Chen Diagram Symbols

- ***Derived Attribute***

- A derived attribute represents an attribute that does not physically exist in a permanently stored form

- It is an attribute that is calculated from other attributes

- Represented in the diagram as an oval with a dashed line borderinside



Chen Diagram Symbols

- ***Relationship***

  - Defines the interconnection between two entities

  - Verb phrases

  - Depicted as a diamond

Chen Diagram Symbols

- ***Strong Relationship***

  - A strong relationship

  - Connects two entities in a diagram and signifies the mutual existential independence of the two entities. Entity A can exist without Entity B, and vice versa

  - Represented by a diamond shape with a solid border

Chen Diagram Symbols

- ***Participation***

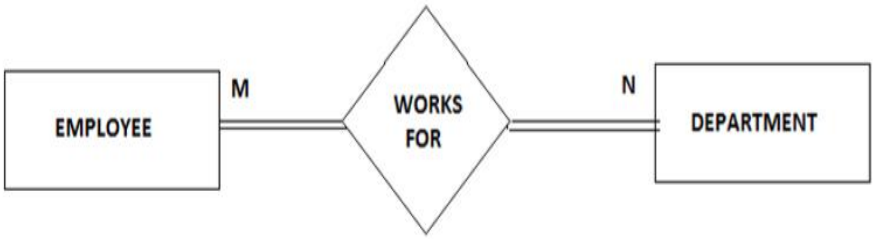
Define a type of entity participating in a relationship. It can be:

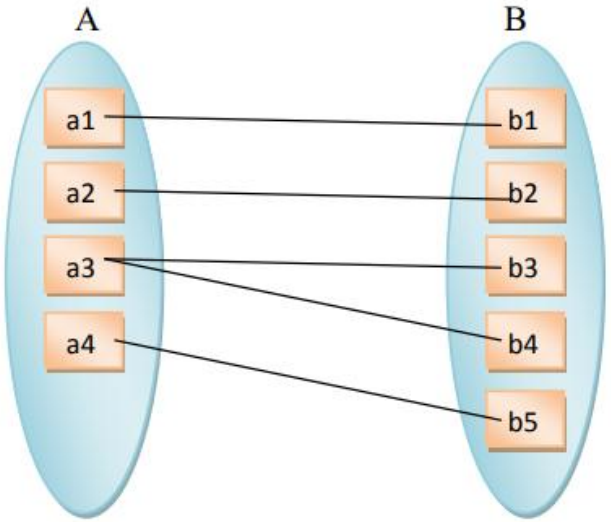
- Total participation - is depicted as a double line
- Partial participation - is depicted as a solid line
- Optional participation - is depicted as a dashed line

Chen Diagram Symbols

Total Participation

- Each entity in the entity set is involved in at least one relationship in a relations
- Represented by double line

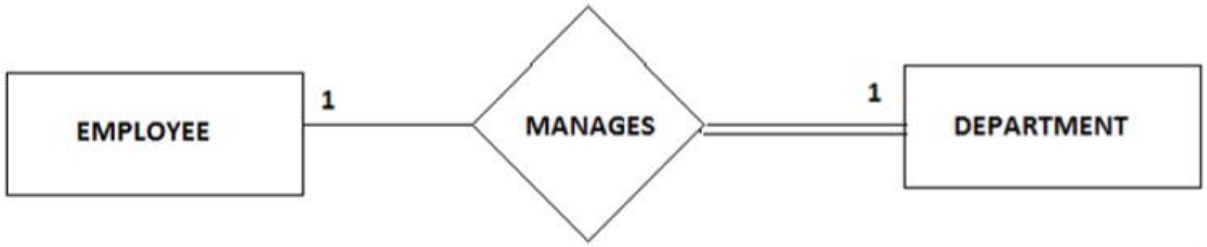


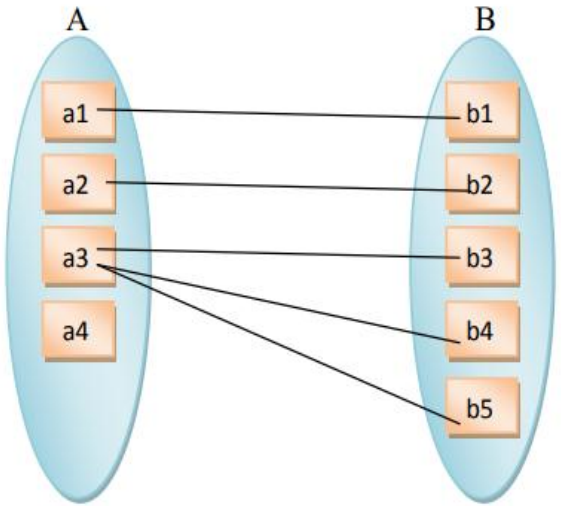


Chen Diagram Symbols

Partial Participation

- Each entity in entity set may or may not occur in at least one relationship in a relation set
- Represented by single line between entity set and relationship set.





Chen Diagram Symbols

- ***Cardinality***

- Represents the relationship between the data in two different tables

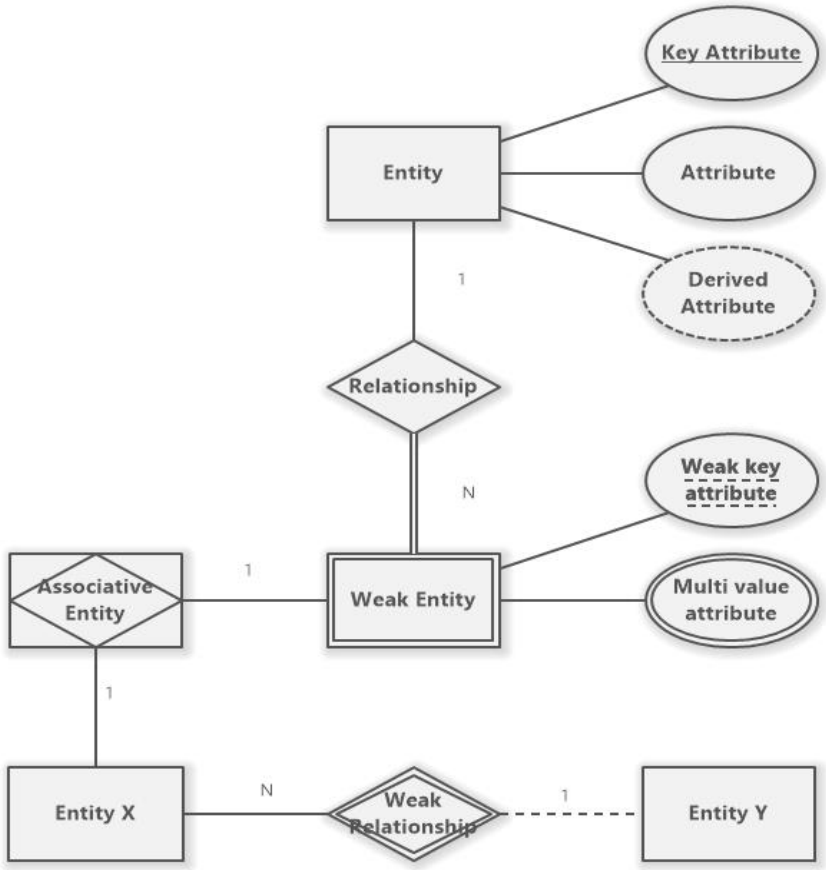
- Connect one or multiple tuples from one entity to one or multiple tuples of another entity

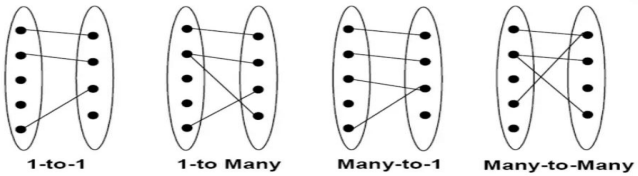
One-to-one (1:1)

One-to-many (1:N)

Many-to-one (N:1)

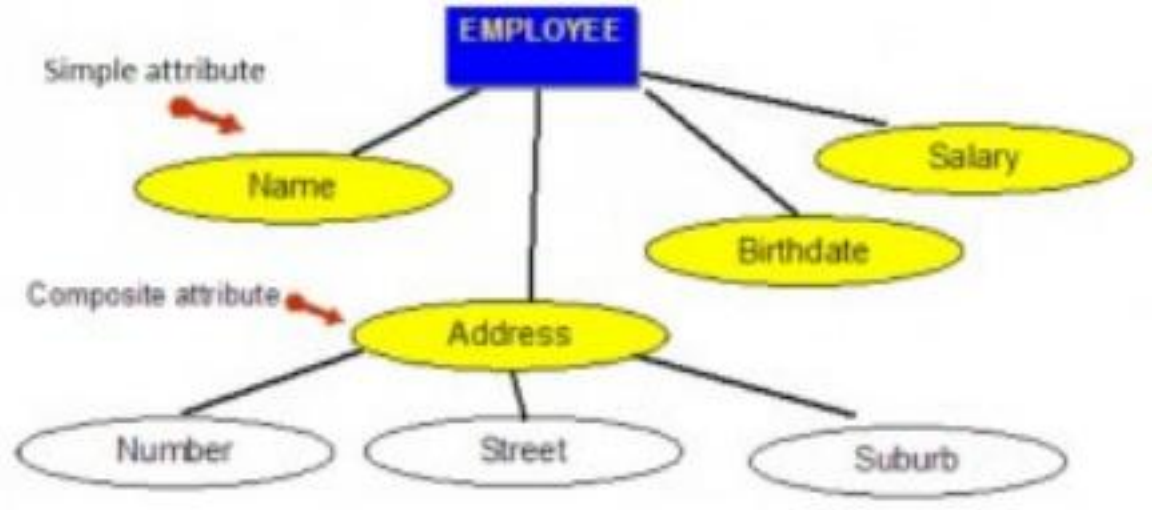
Many-to-many (M:N)





Composite attributes

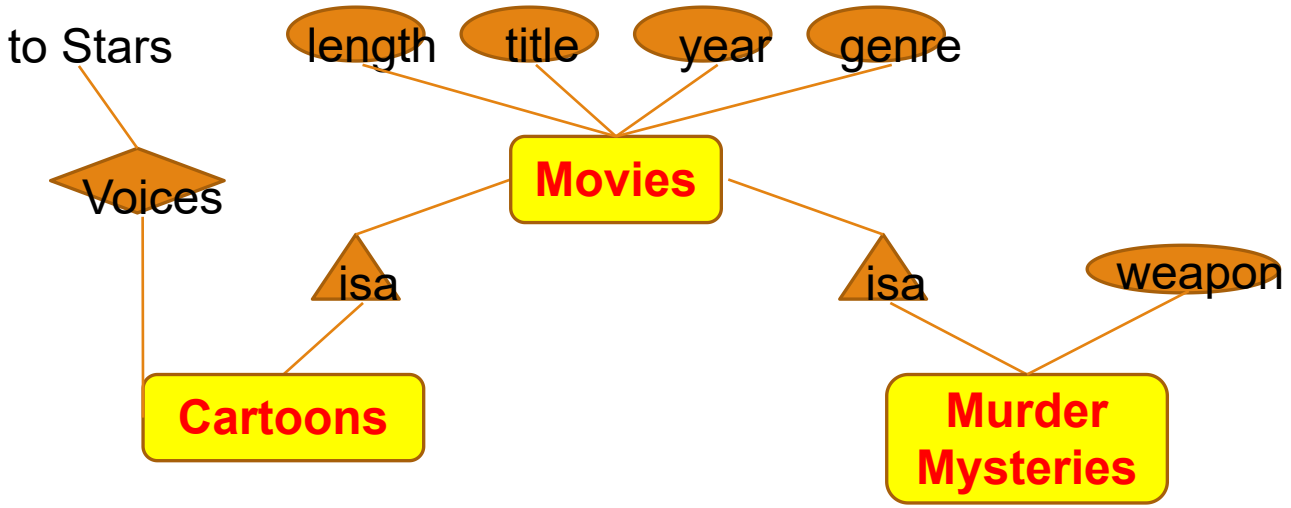
Composite attributes are those that consist of a hierarchy of attributes



Subclasses in E/R Model

A subclass is a class derived from the superclass. It inherits the properties of the superclass and also contains attributes of its own.

An example is:Cartoons and Murder Mysteries are the special kinds of movies, with some special properties



Example COMPANY Database – Construct ERD

Requirements of the Company (oversimplified for illustrative purposes)

- 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.

Fpt University

Example COMPANY Database (Cont.)

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 M-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

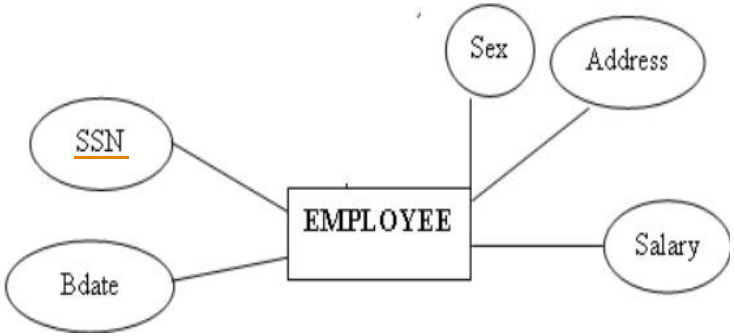
2 task

- Task 1:
  - Convert entity
  - Convert attribute
  - Convert relations
- Task 2:
  - Check the relations, and
  - Normalize to 3NF

From ER Diagram to Relational Model

- **Strong entity**

Entity	Relation
Name of entity	Name of relation
Single attribute of entity	Attibute of entity
Primary key of entity	Primary key of relation

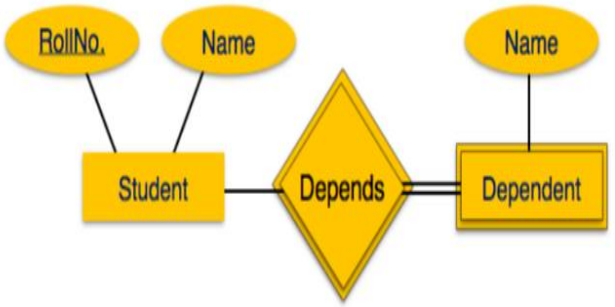


EMPLOYEE (SSN, Bdate, Sex, Address, Salary)

From ER Diagram to Relational Model

- **Weak entity**

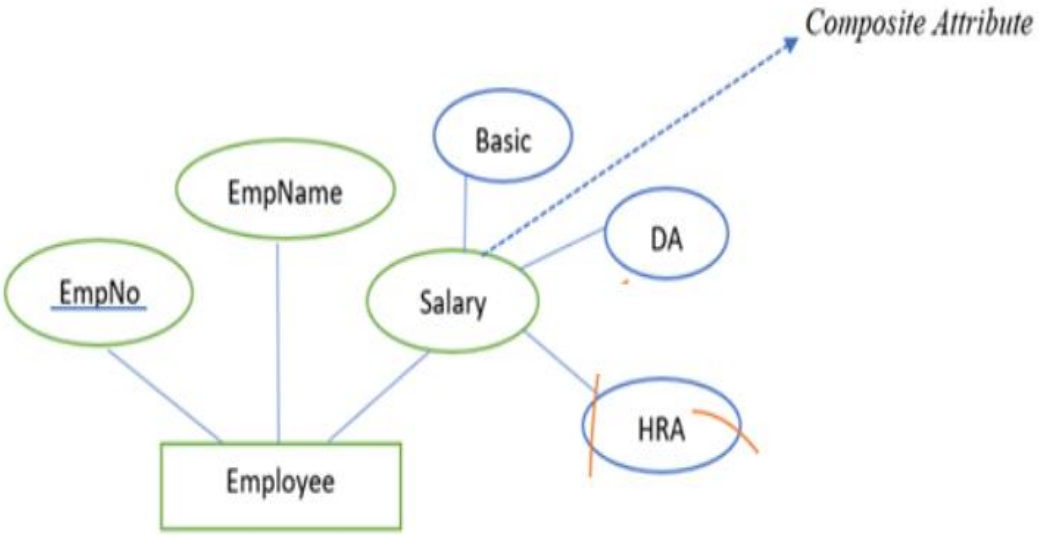
Entity	Relation
Name of weak entity	Name of relation
Single attribute of entity	Attibute of entity
partial key of entity (if have) + primary of strong entity	Primary key of relation



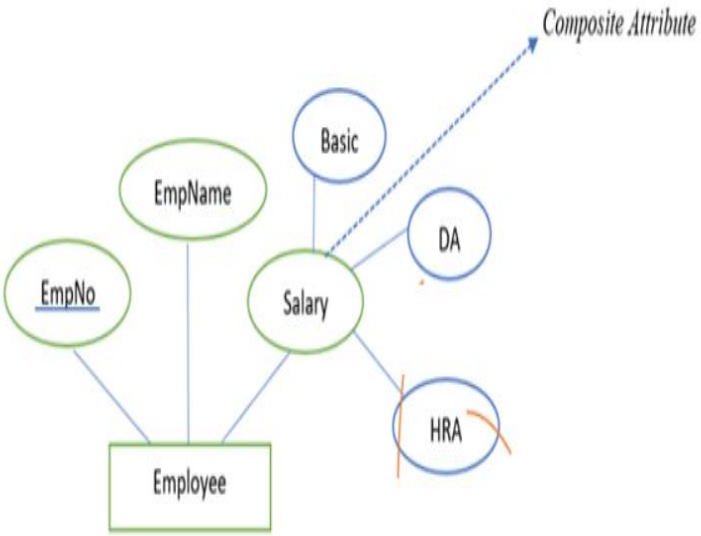
DEPENDENT (RollNo, Name)

From ER Diagram to Relational Model

- **Composite attribute**



From ER Diagram to Relational Model



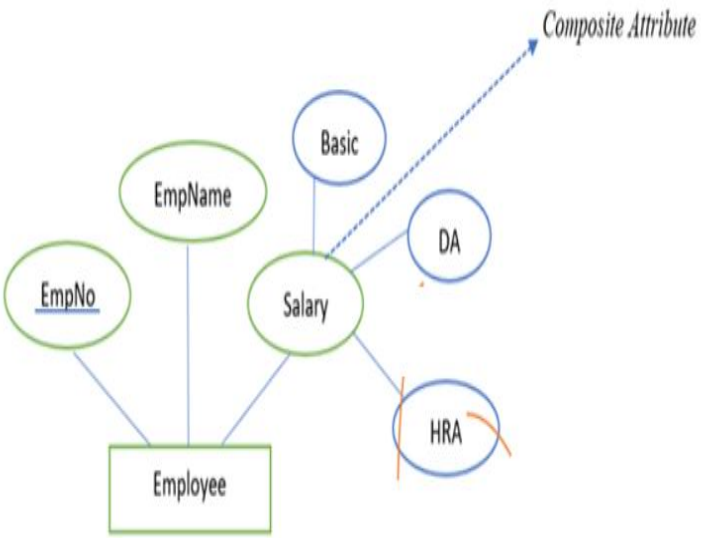
- Composite attribute set --> Single attribute

Employee(EmpNo,EmpName,Salary)

And then create one more relation Salary, we can create **ID_Salary** to connect 2 tables

Salary (basic, DA, HRA)

From ER Diagram to Relational Model

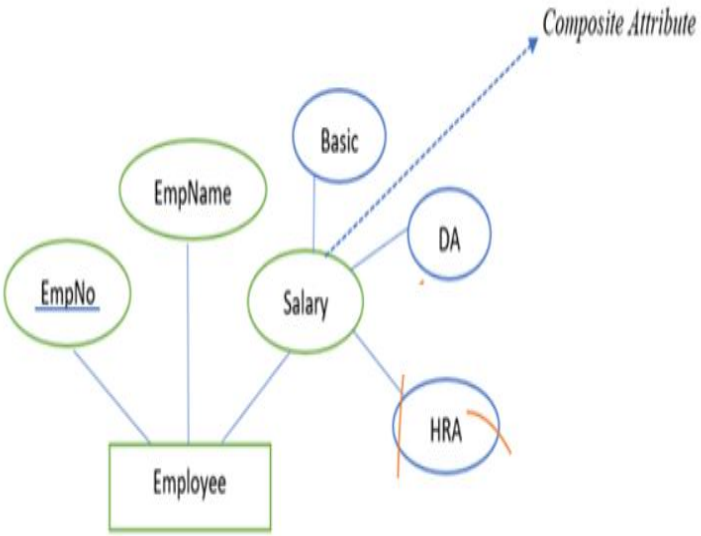


- Composite attribute set --> Single attribute

→ It's a pretty popular choice

Employee(EmpNo,EmpName,Basic, DA, HRA)

From ER Diagram to Relational Model



- Use both single and composite attributes

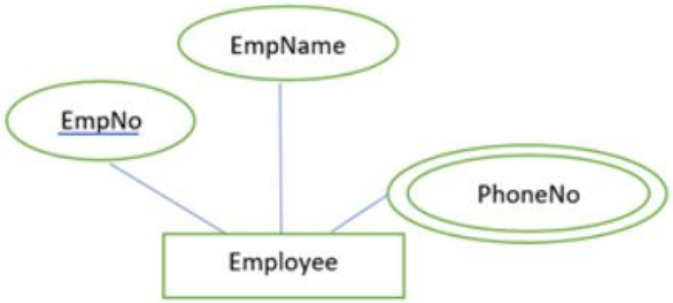
Employee(EmpNo,EmpName,Salary, Basic, DA, HRA)

From ER Diagram to Relational Model

- **Multivalued attribute**

Multivalued attribute --> Relation

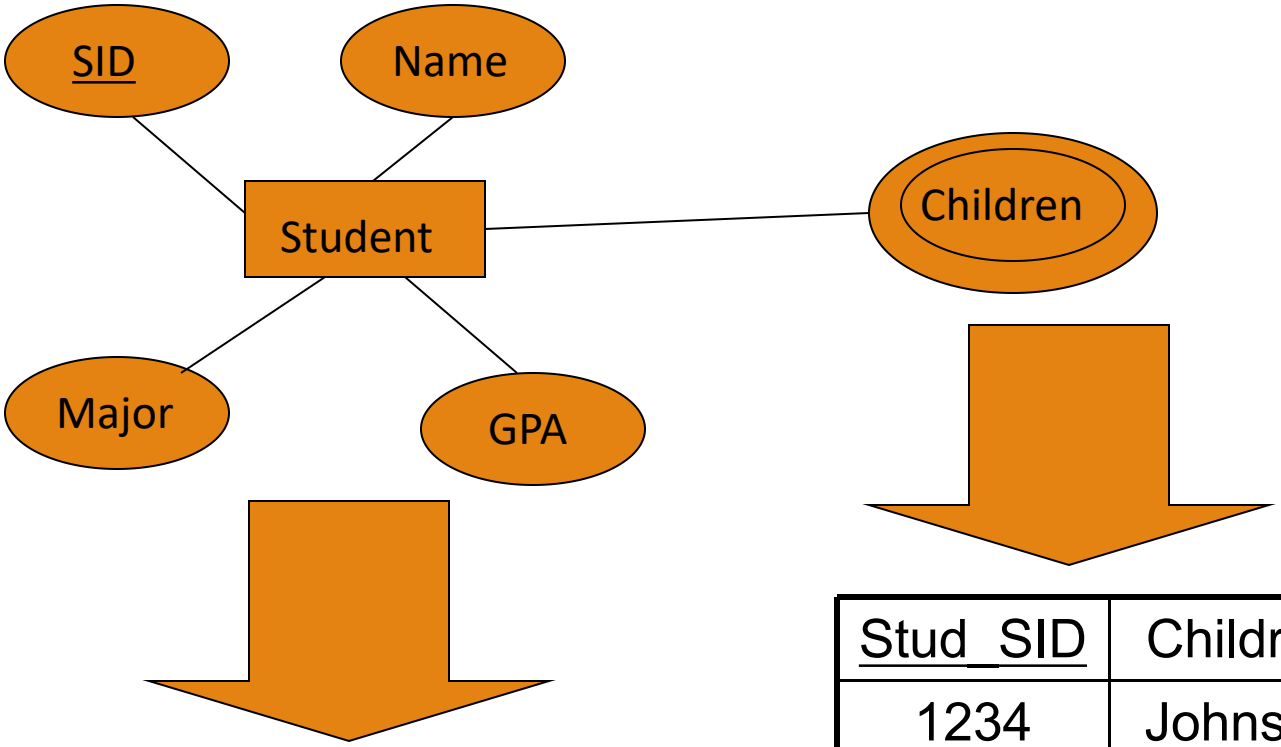
Relation has key: Entity primary key + multivalued attribute



Employee(EmpNo, EmpName)

P_Employee(EmpNo,PhoneNo)

Example – Multivalue attribute

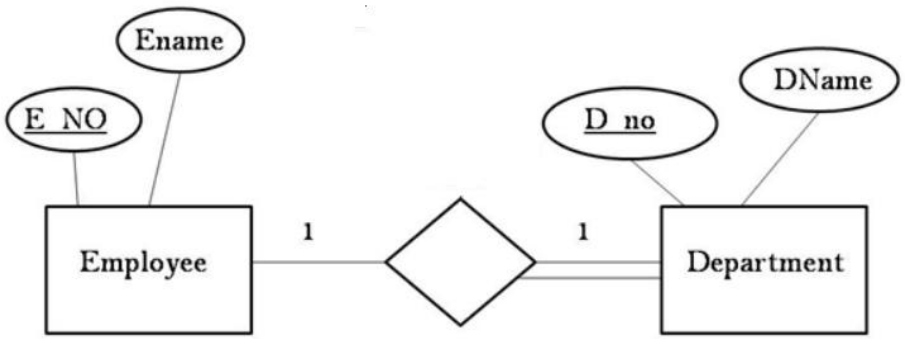


SID	Name	Major	GPA
1234	John	CS	2.8
5678	Homer	EE	3.6

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

From ER Diagram to Relational Model

- **Convert relations**
  - **For one-to-one relations (1-1):**
    - Total participation of entity: S
    - Part participation of entity: T
    - We **convert** the **primary key** of **T into S**



Department(D_no, E_No, DName)
Employee(E_NO, Ename)

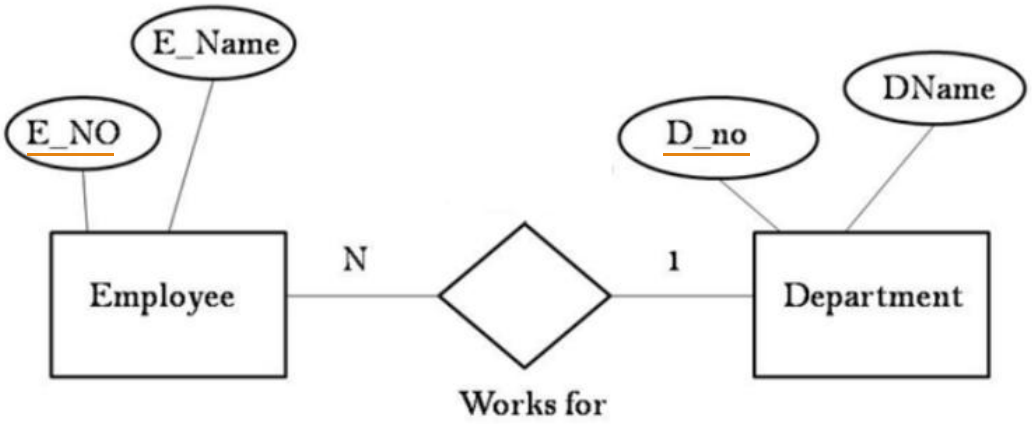
From ER Diagram to Relational Model

- **Convert relations**
  - **For one-to-many relations (1-M):**
    - Composite with attributes
    - Composite no attributes

From ER Diagram to Relational Model

- **Composite no attributes**
  - Add the key attribute set of the "**many**" side into entity have "**one**" side

Department(D_No, E_No, DName)



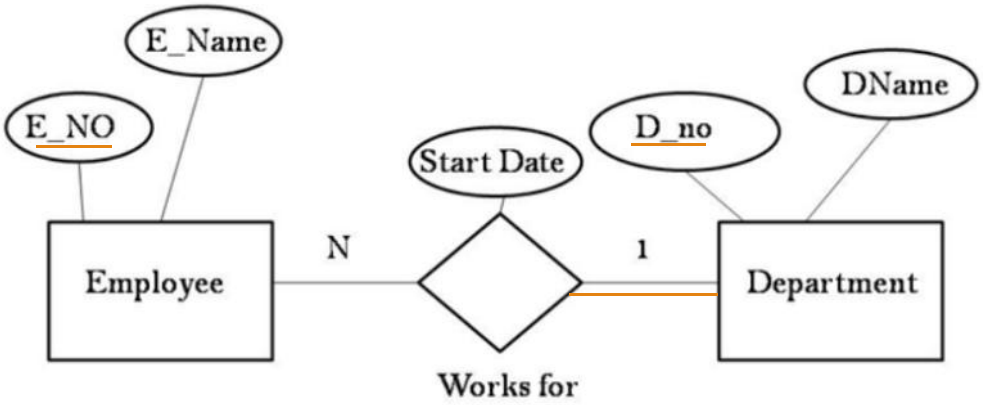
From ER Diagram to Relational Model

- **Composite with attributes**

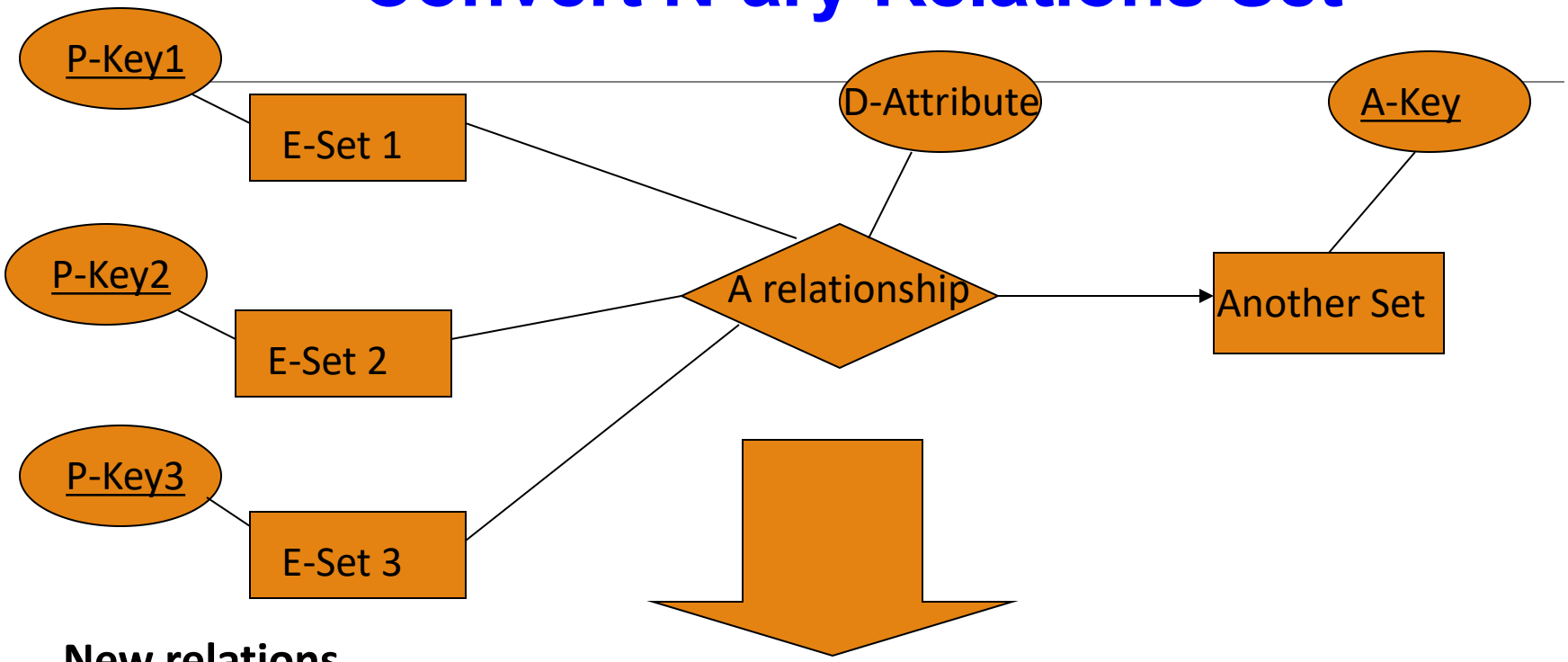
- Add primary key attribute set of **“Many”** side to **"one"** side
- Let S be the entity that fully participates in the relations (ex: Department)

  Add the composite attribute to the S entity

  Department(D_No, E_No, **Start_Date**, DName)



Convert N-ary Relations Set

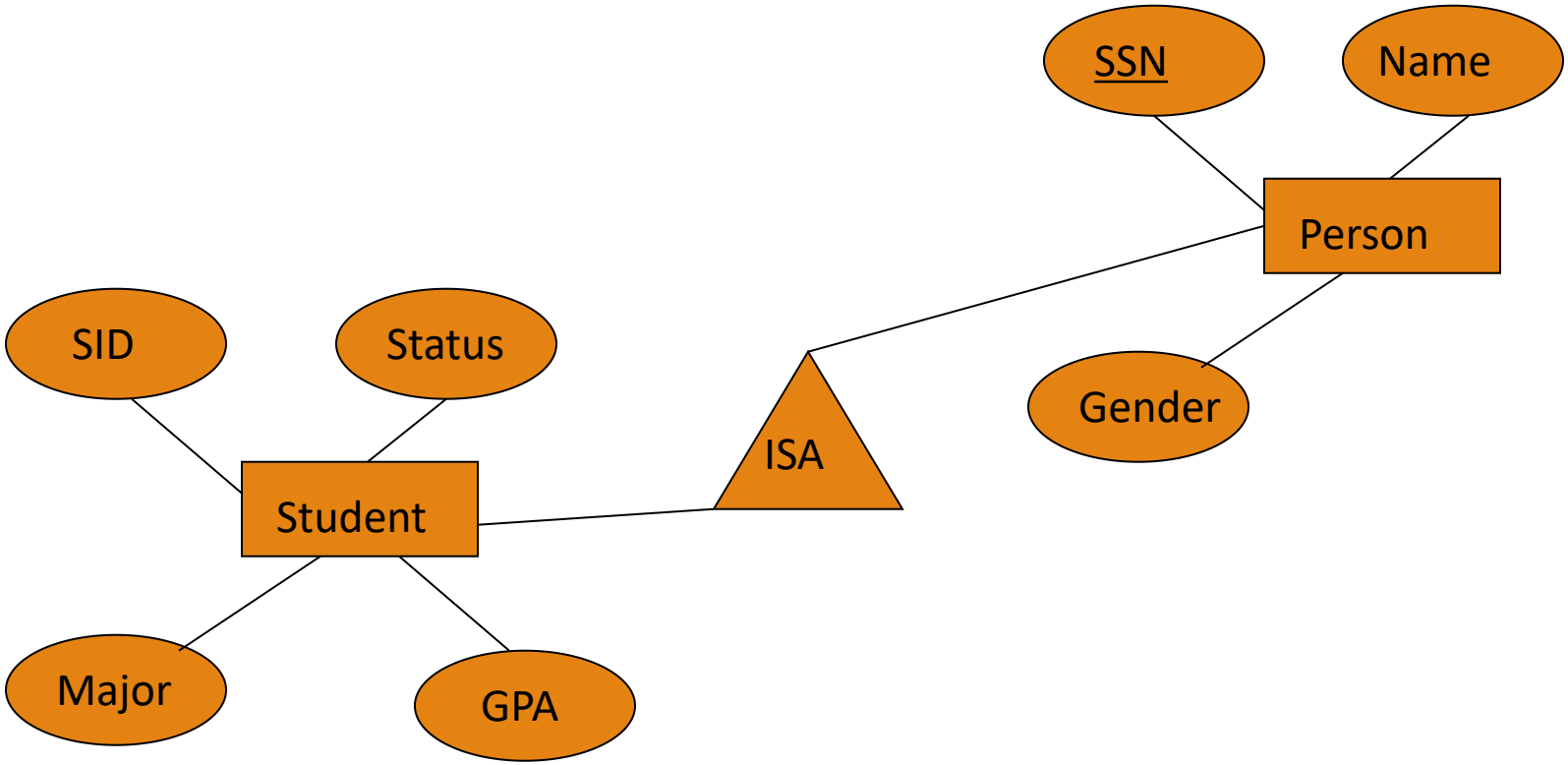


New relations

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 Class Hierarchy



Fpt University

Representing Class Hierarchy

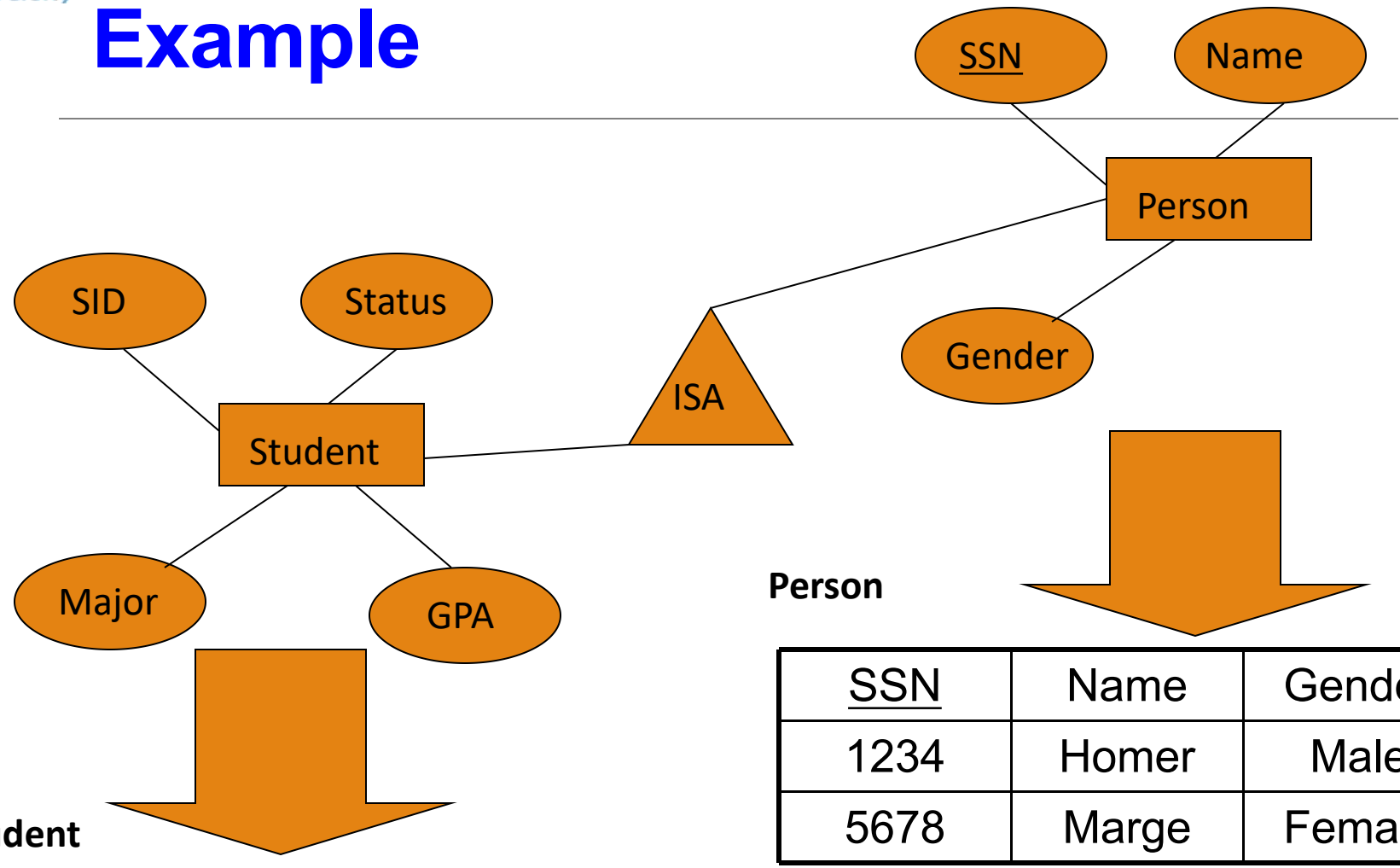
- **For non-disjoint and/or non-complete class hierarchy**
  - **Create a table** for each **super class** entity
  - **Create a table** for each **subclass entity** set

    - With **a column** for each of the **attributes** of **subclass entity** set

    - **Add attributes** of the **primary key** of the **super class** entity set

    - **This primary key** of **super class entity** is also **used** as the **primary key for this new table**

Example



Person

SSN	Name	Gender
1234	Homer	Male
5678	Marge	Female

Student

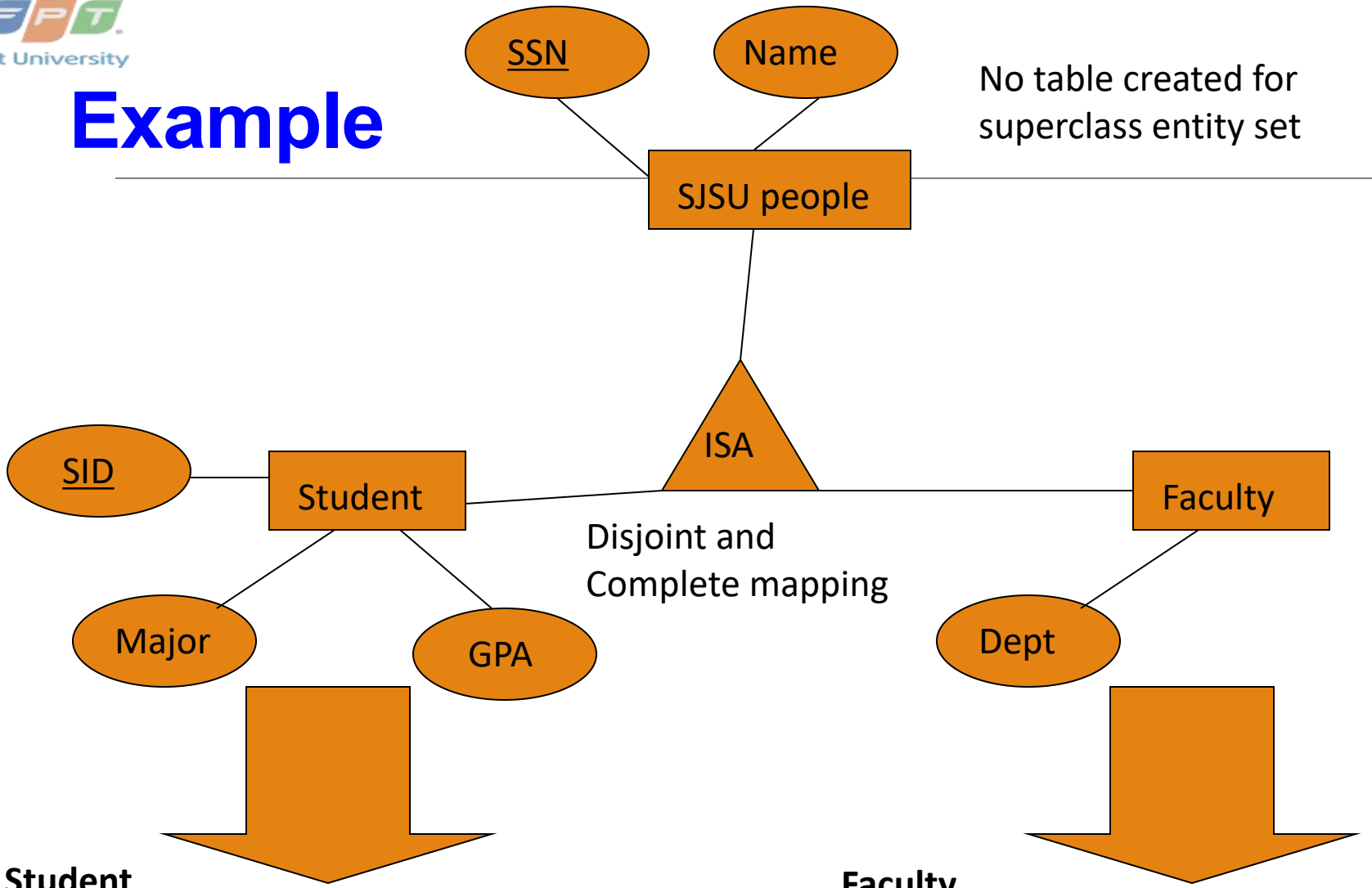
SSN	SID	Status	Major	GPA
1234	9999	Full	CS	2.8
5678	8888	Part	EE	3.6

Fpt University

Representing Class Hierarchy

- 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:**
    - **All attributes of that subclass entity set**
    - **Attributes** of the **superclass entity set**

Example



Student

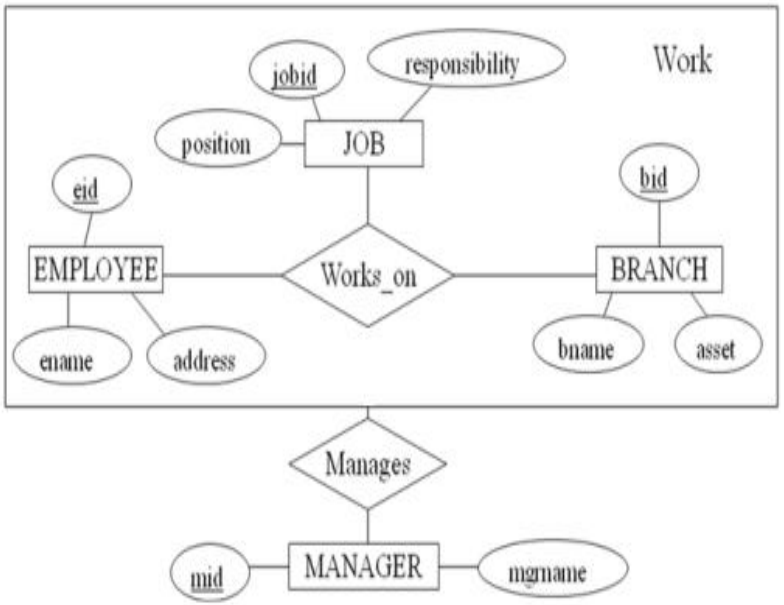
SSN	Name	SID	Major	GPA
1234	John	9999	CS	2.8
5678	Mary	8888	EE	3.6

Faculty

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

Representing Aggregation (general)

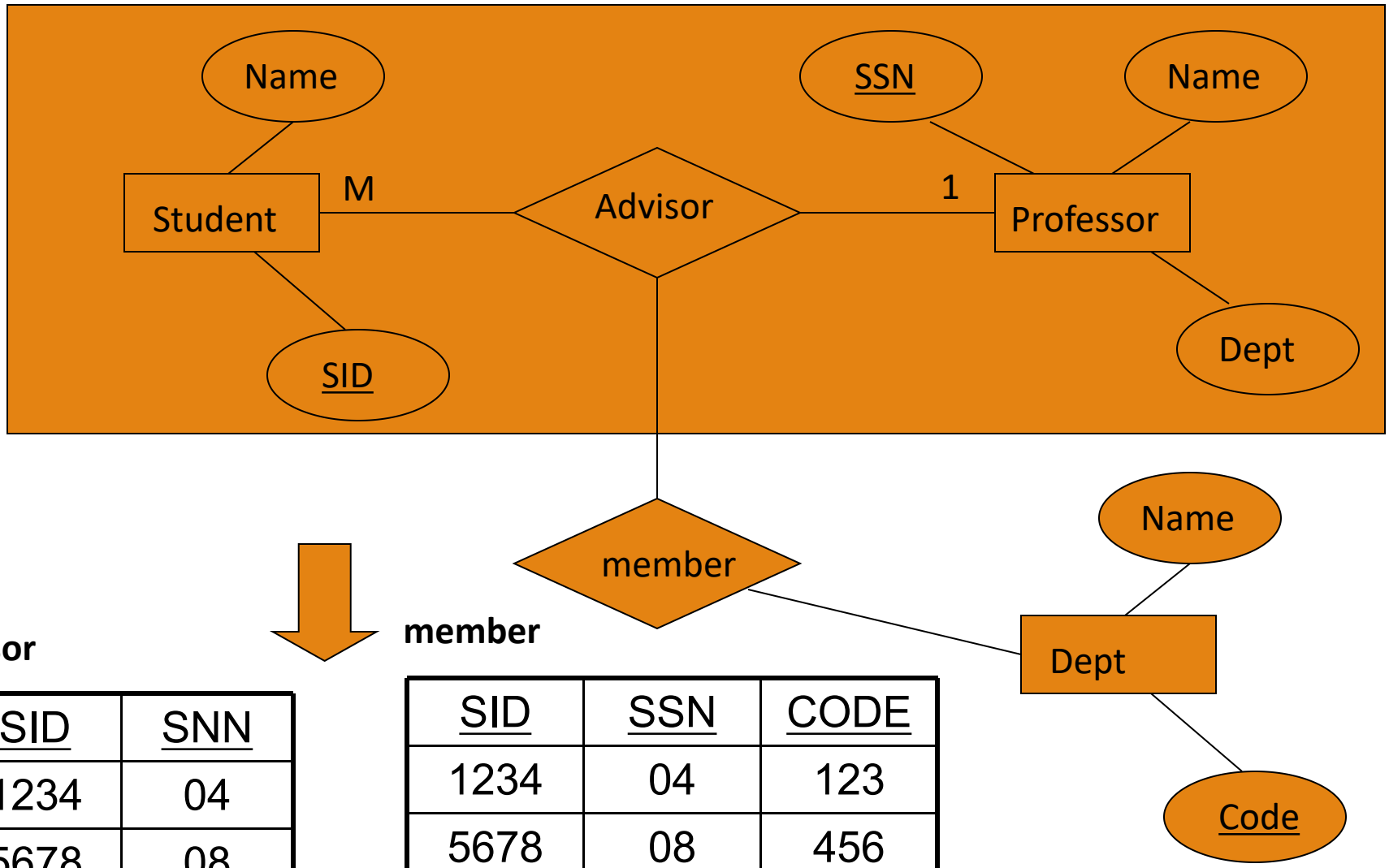
- **Create a table** containing the **primary key of the aggregated relations**
- **Primary key** of the **associated entity set** and **descriptive attributes** (if any)



Employee = (eid, name, address)
Branch = (bid, bname, asset)
Job = (jobid, position, responsibility)
Manager = (mid, mgrname)

Works_on = (eia, bia, jobia)
Manages = (eia, bia, jobia, mia)

Representing Aggregation



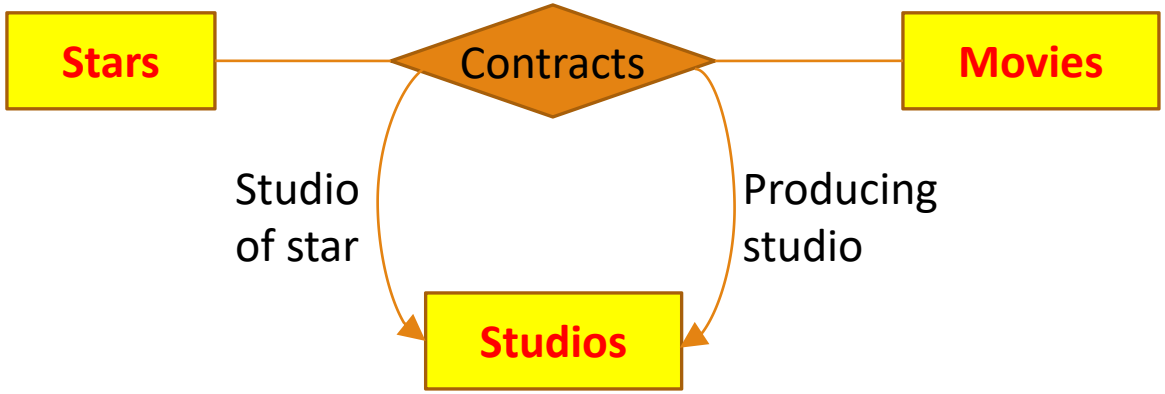
Advisor

SID	SNN
1234	04
5678	08

member

SID	SSN	CODE
1234	04	123
5678	08	456

Recursive relations



- Create another copy of the key in the relation.

Contracts (Contracts_ID, Contracts_Part_ID (fk), Studios, Description)

OR

Contracts (Contracts_ID, Studios_ID(fk), Description)
Studios (Studios_ID, Description)

Fpt University

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.

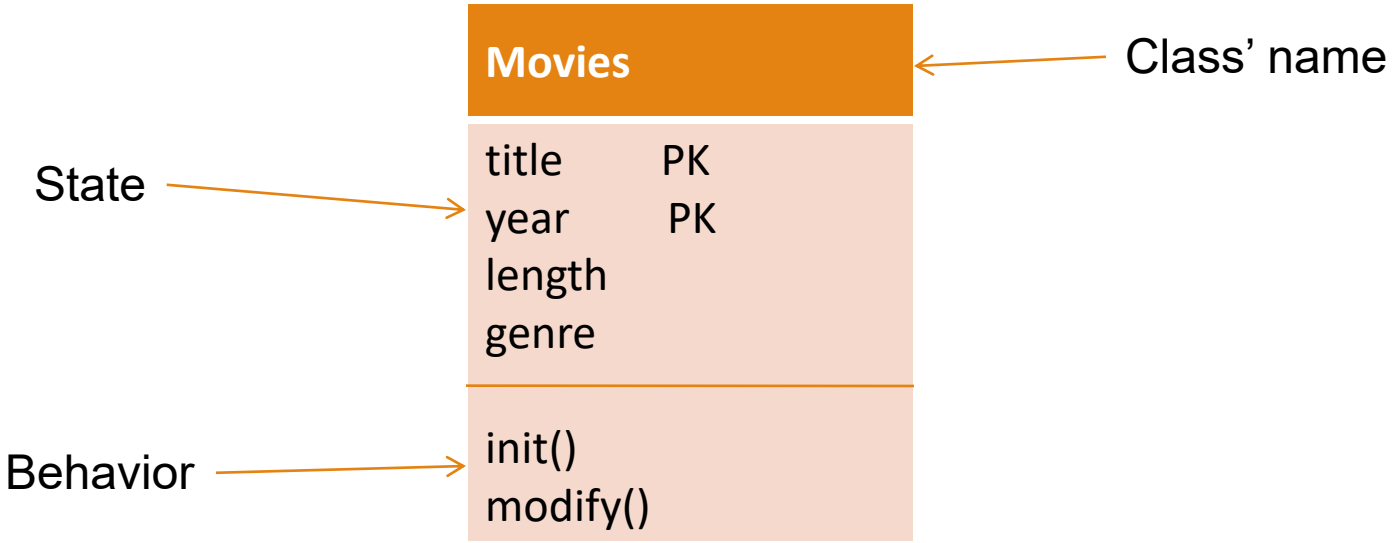
Fpt University

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

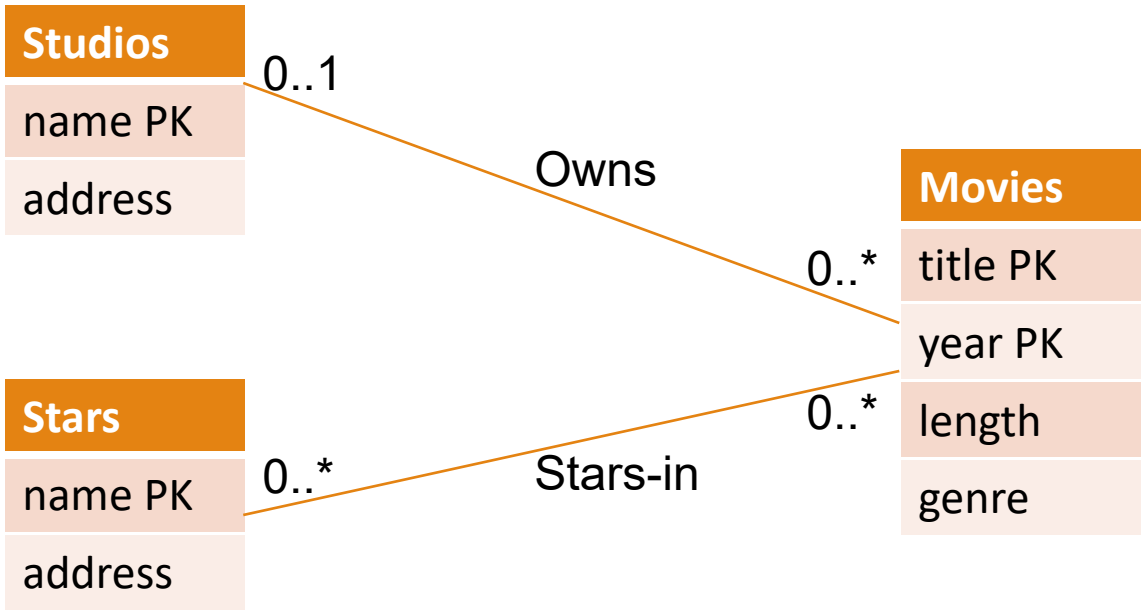
Figure 4.34: Comparison between UML and E/R terminology

UML Classes


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

Associations

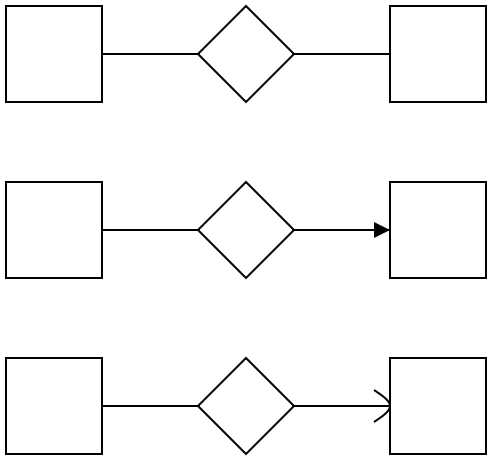
Consider an associations between Movies, Stars, and Studios in UML



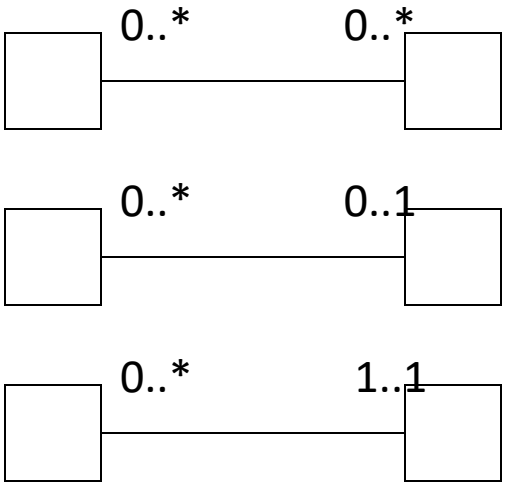
Associations

Comparison with E/R Multiplicities

E/R



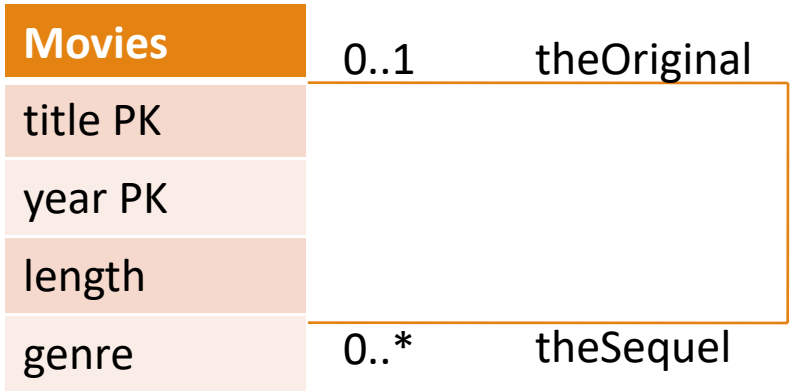
UML



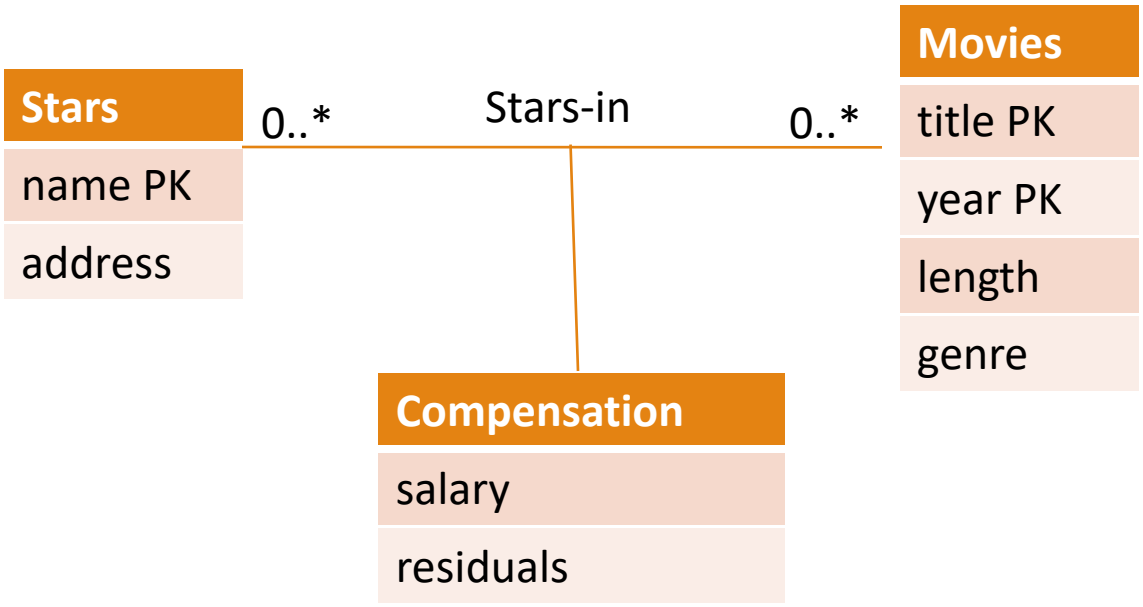
Self-Associations

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

Example



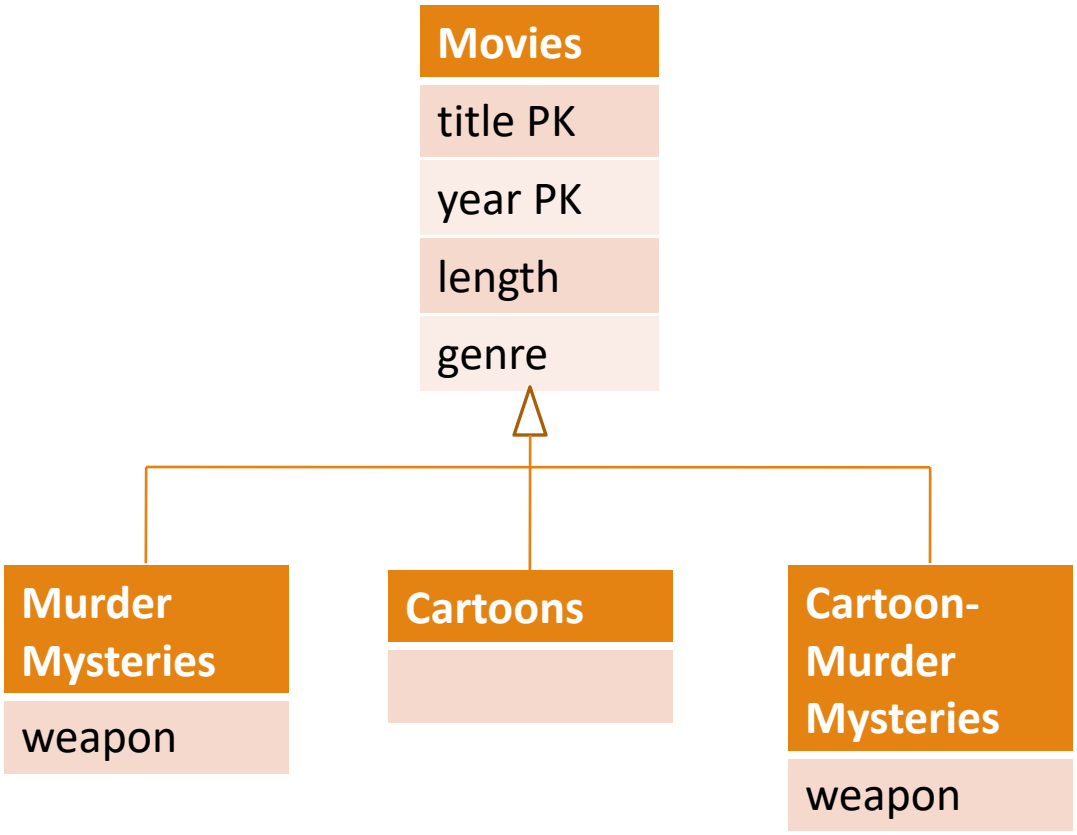
Association Classes



Subclasses in UML

Consider Movies and its three subclasses

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



Aggregations and Compositions

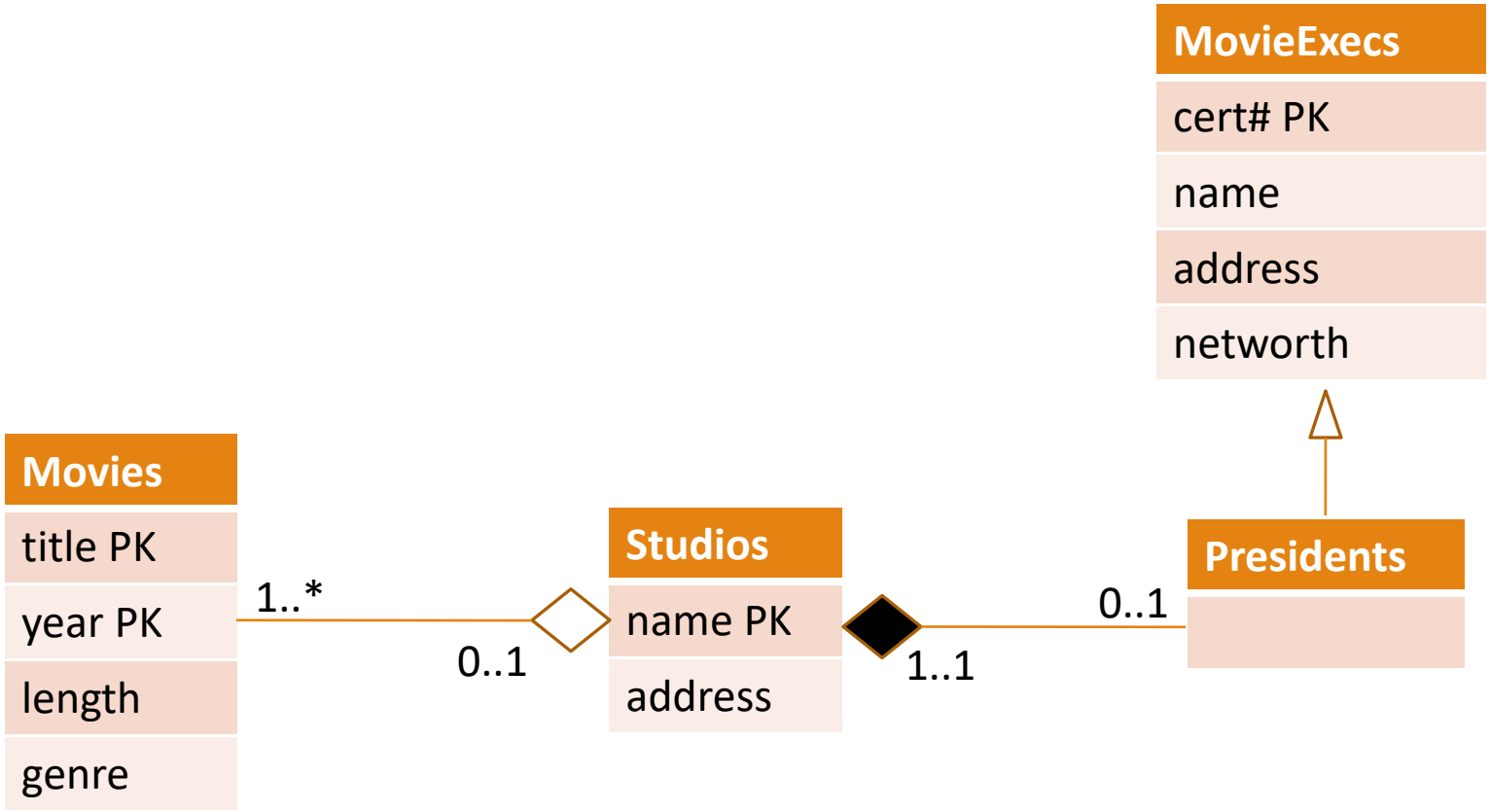


Figure 4.41: An aggregation from Movies to Studios and a composition from Presidents to Studios

Fpt University

UML-to-Relations Basics

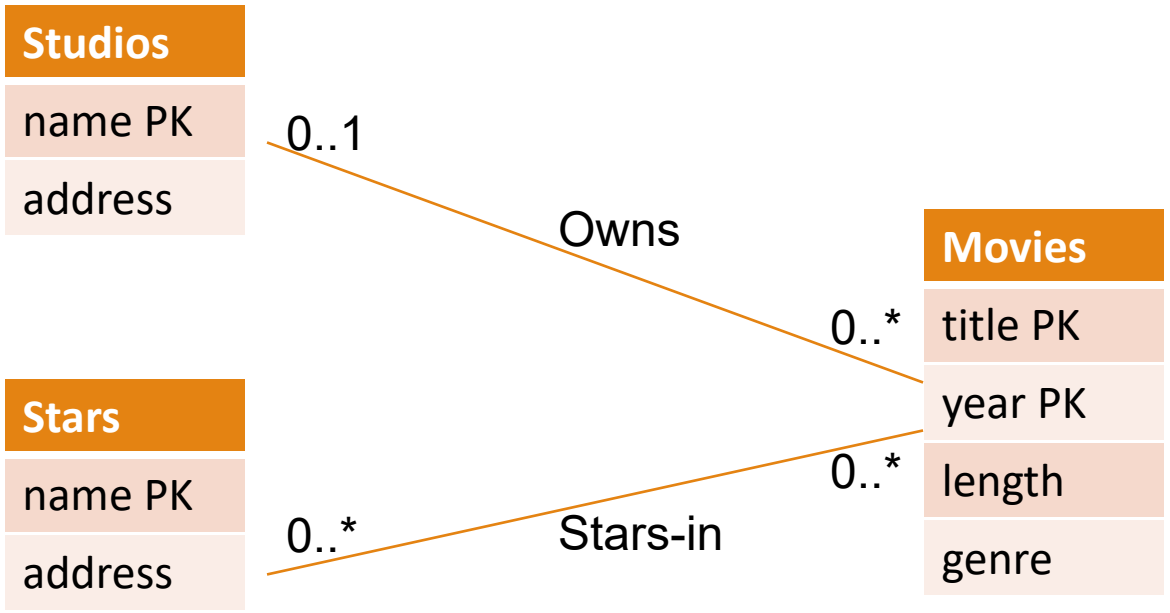
Classes to Relations

- 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(title,year,length,genre)
Stars(name,address)
Studios(name,address)

Stars-In(movieTitle,movieYear,starName)
Owns(movieTitle,movieYear,studioName)

Fpt University

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

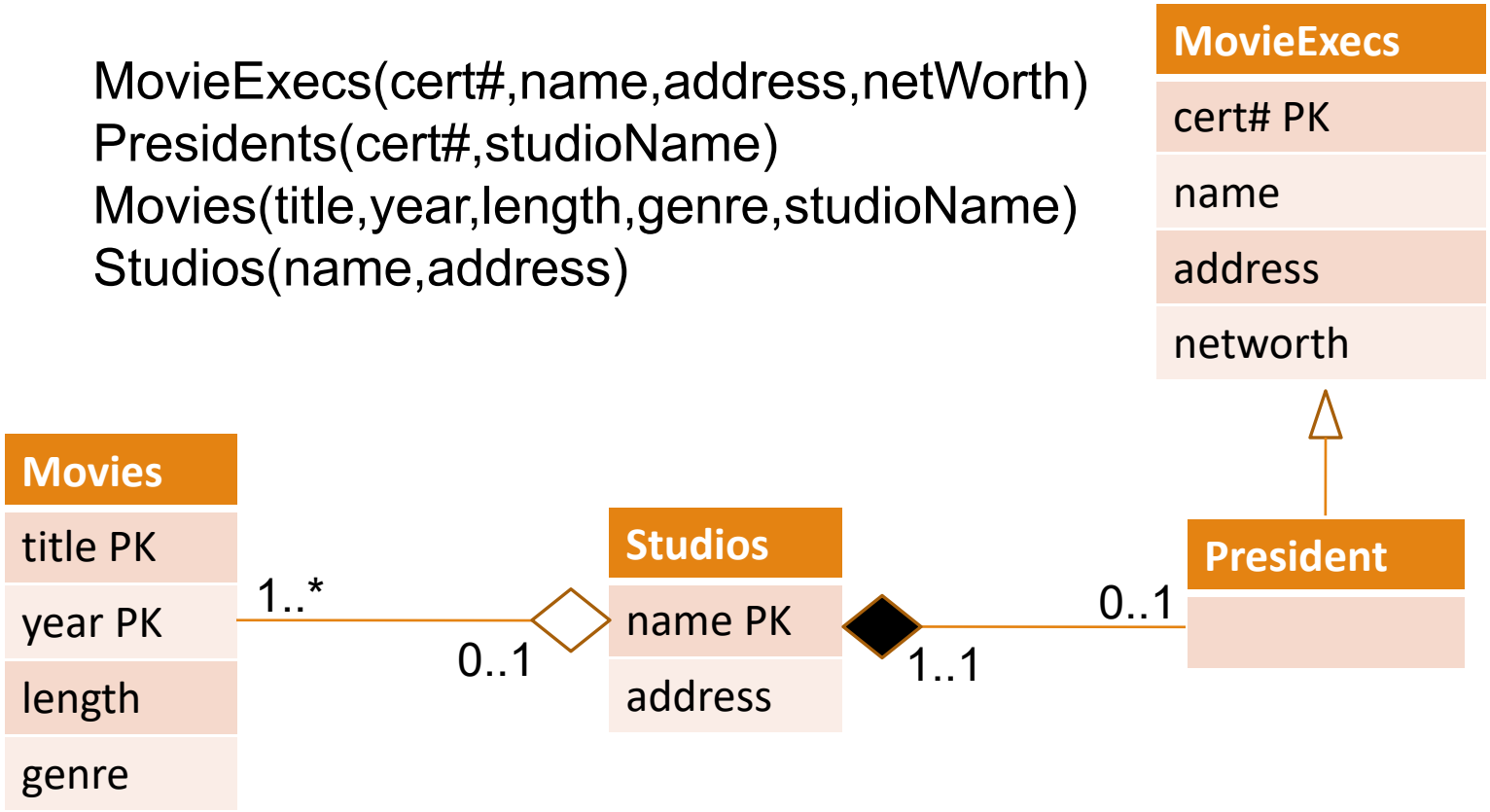
No relation for the aggregation or composition

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)