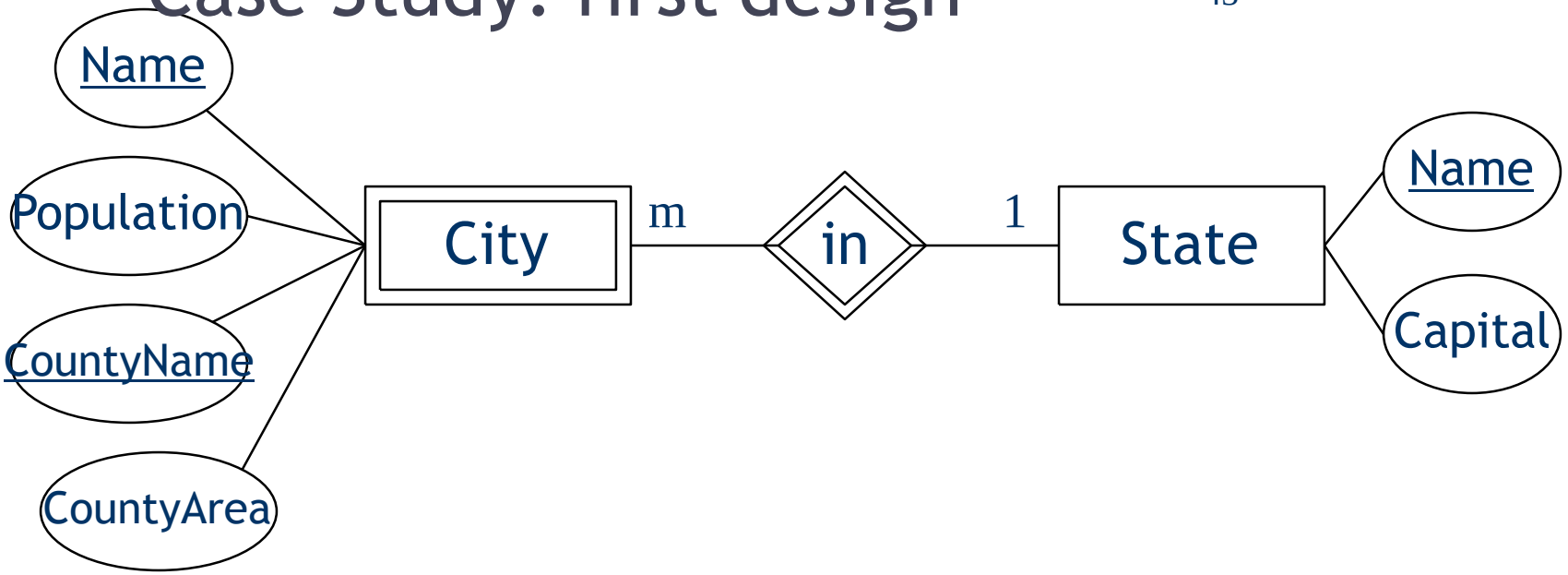
Topic
Enhanced Database Modelling and Crow's Feet Notation

Case Study

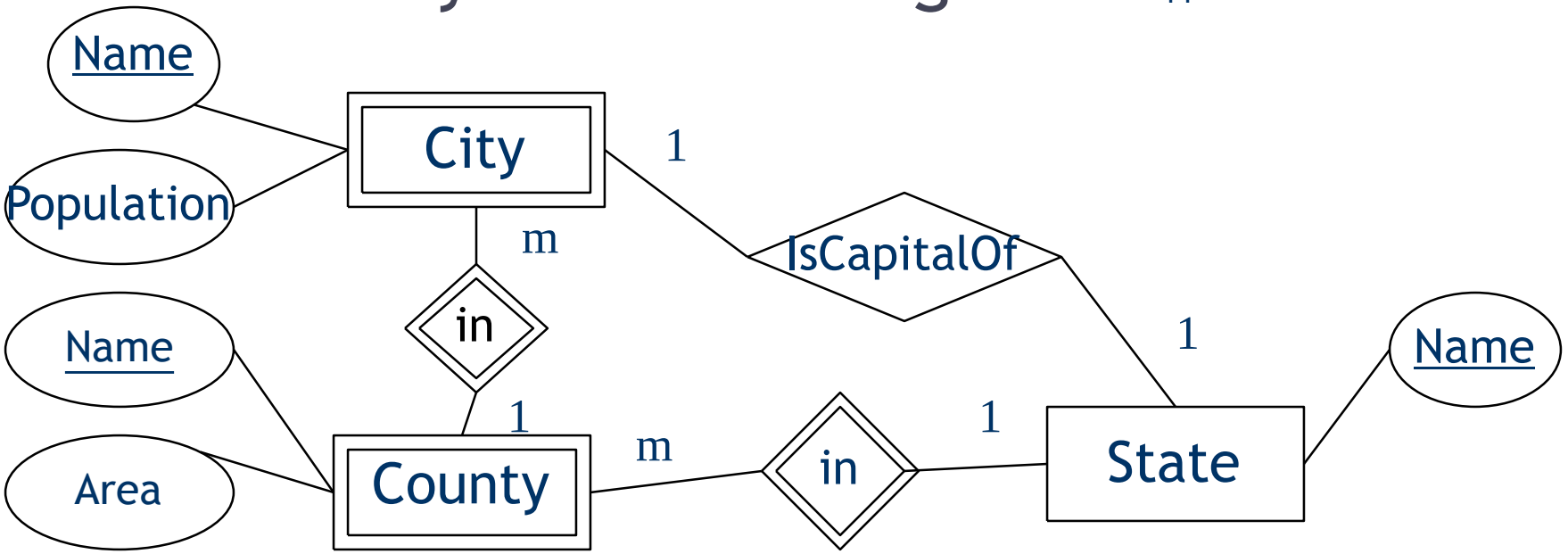
- Design a database representing cities, counties, and states
  - For states, record name and capital (city)
  - For counties, record name, area, and location (state)
  - For cities, record name, population, and location (county and state)
- Assume the following:
  - Names of states are unique
  - Names of counties are only unique within a state
  - Names of cities are only unique within a county
  - A city is always located in a single county
  - A county is always located in a single state

Case Study: first design



- County area information is repeated for every city in the county
  - ☞Redundancy is bad (why?)
- State capital should really be a city
  - ☞"Reference" entities through explicit relationships

Case Study: second design



- Technically, nothing in this design could prevent a city in state *X* from being the capital of anotherstate *Y*, but oh well…

Relationships between entities

When two entities are "related".. There should be a word to describe this relationship

e.g.	**Entities**	**Relationship**
	staff, branches	Staff *manage* bank branches
	student, modules	student *attends* classes
	newspaper, rentalProperties	newspapers *advertise* rentalProperties

You need to understand this "word" in order to figure out the information needed to define the relationship for the ERD

On an ERD, need to know how many of each entity takes part in the relationship – question *both* entities about "how many"?

e.g. For the Staff manages bank branches relationship

Staff **can** manage how many branches at most?? A staff person can manage one branch
Staff **must** manage how many branches at min? A staff person can have no branch to manage

Relationship cardinality

- **Cardinality** means "count," and is expressed as a number.
- **Maximum cardinality** is the maximum number of entity instances that **can** participate in a relationship.
- **Minimum cardinality** is the minimum number of entity instances that **must** participate in a relationship.

"eh…What's the cardinality of your mobile phone spend in euro each week?

Relationship cardinality is about assigning the 1:1, 1:n etc etc

"Can" and "Must" get you there..

Maximum Cardinality

- **Maximum cardinality** is the maximum number of entity instances that **can** participate in a relationship.
- There are three types of maximum cardinality:
  - One-to-One [1:1]
  - One-to-Many [1:N]
  - Many-to-Many [N:M]

Can….

The 3 types of maximum cardinality

One to one 1:1	An employee *can be assigned* one computer
One to many 1:n	A mother *can have* many children
Many to many m:n	An order *can have* many product *and reverse*

How can you tell the difference between 1:m and m:n
e.g. Student and module entities. What's the maximum cardinality?

Minimum cardinality

- **Minimum cardinality** is the minimum number of entity instances that **must** participate in a relationship.
- Minimums are generally stated as either zero or one:
  - IF **zero [0]** THEN participation in the relationship by the entity is **optional**, and **no** entity instance must participate in the relationship.
  - IF **one [1]** THEN participation in the relationship by the entity is **mandatory**, and **at least one** entity instance must participate in the relationship.

Must…

Minimal cardinality

What's the minimum cardinality* for our examples?

An employee *must be assigned* **?** computer

A mother *must have* ? children

An order *must have* ? product

* zero (optional) or one (must have at least one)

Minimal cardinality

What's the minimum cardinality* for our examples?

An employee *must be assigned* **?** Computer

Ans: Depends on the rule of the individual company. Probably zero

A mother *must have* ? children

ans: Definitely zero (optional!)

An order *can have* ? Products

ans: must have at least one

* zero (optional) or one (must have at least one)

Maximum and minimum cardinality

Now we know how to get the maximum (can) and minimum –

But it has to be done BOTH ways to complete the relationship.
We only did ONE direction

Maximum cardinality

A computer can be assigned to one employee

A child can have one mother

A product can be on many orders

Minimim cardinality

A computer must be assigned to one employee -No

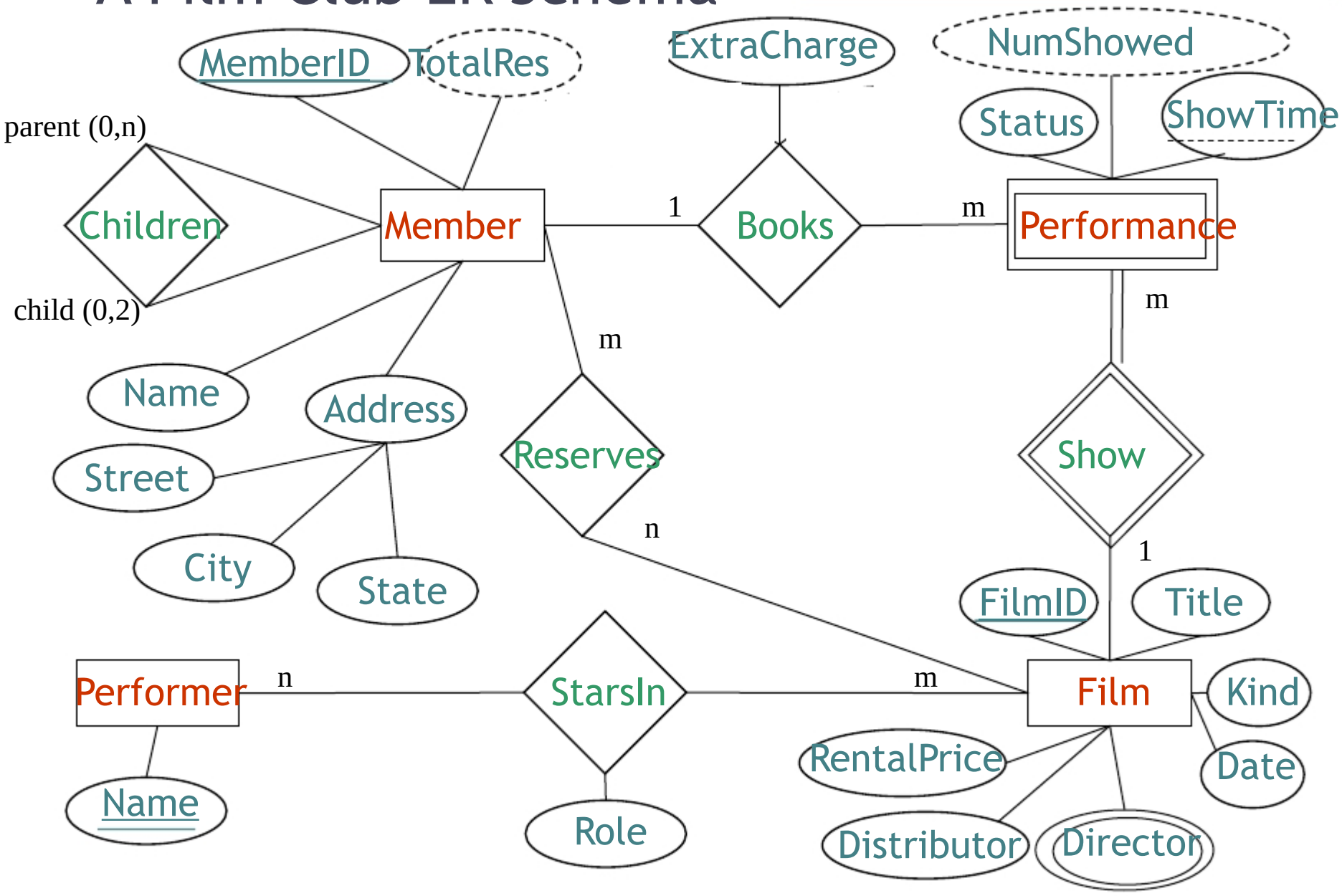
A child must have a mother? YES

A product must be on an order? NO

Outline

- ER model
  - Overview
  - Entity types
    - Attributes, keys
  - Relationship types
  - Weak entity types
- EER model
  - Subclasses
  - Specialization/Generalization
- Schema Design
  - Single DB
  - View integration in IS
- uses Integration DEFinition for Information Modeling (IDEF1X) notation in ERwin

A Film Club ER Schema

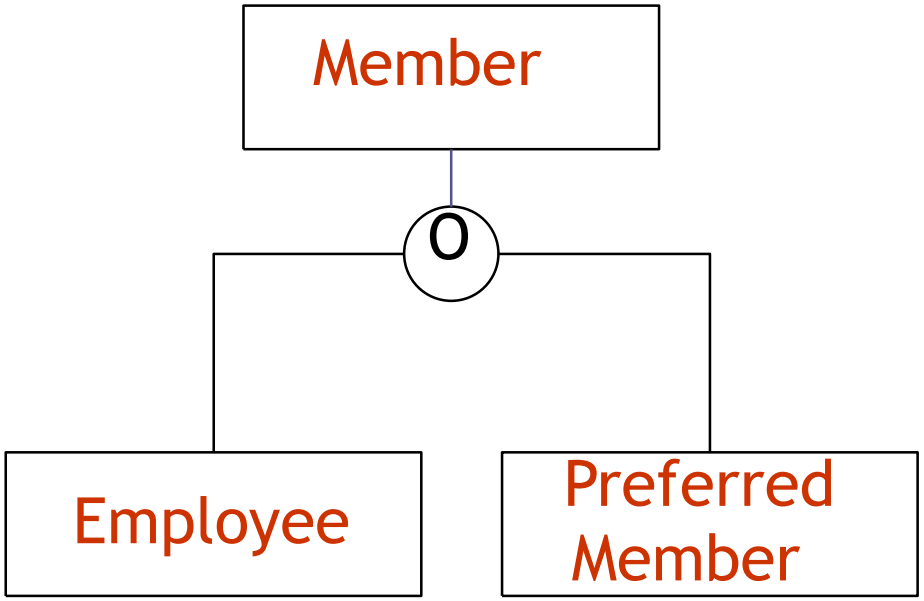


Subclasses And Superclasses

- Grouping of the entities of an entity type into subgroups (forming an "IS-A" relationship).
- Entities in a *superclass* are grouped into one or more *subclasses*.
- An arbitrary number of levels is permitted in a class hierarchy.
- An entity in a subclass exists in the superclass also (recursively).

Subclasses/Superclasses Example

- The Member entity type has the subclasses PreferredMember and Employee.
- All employees are members.

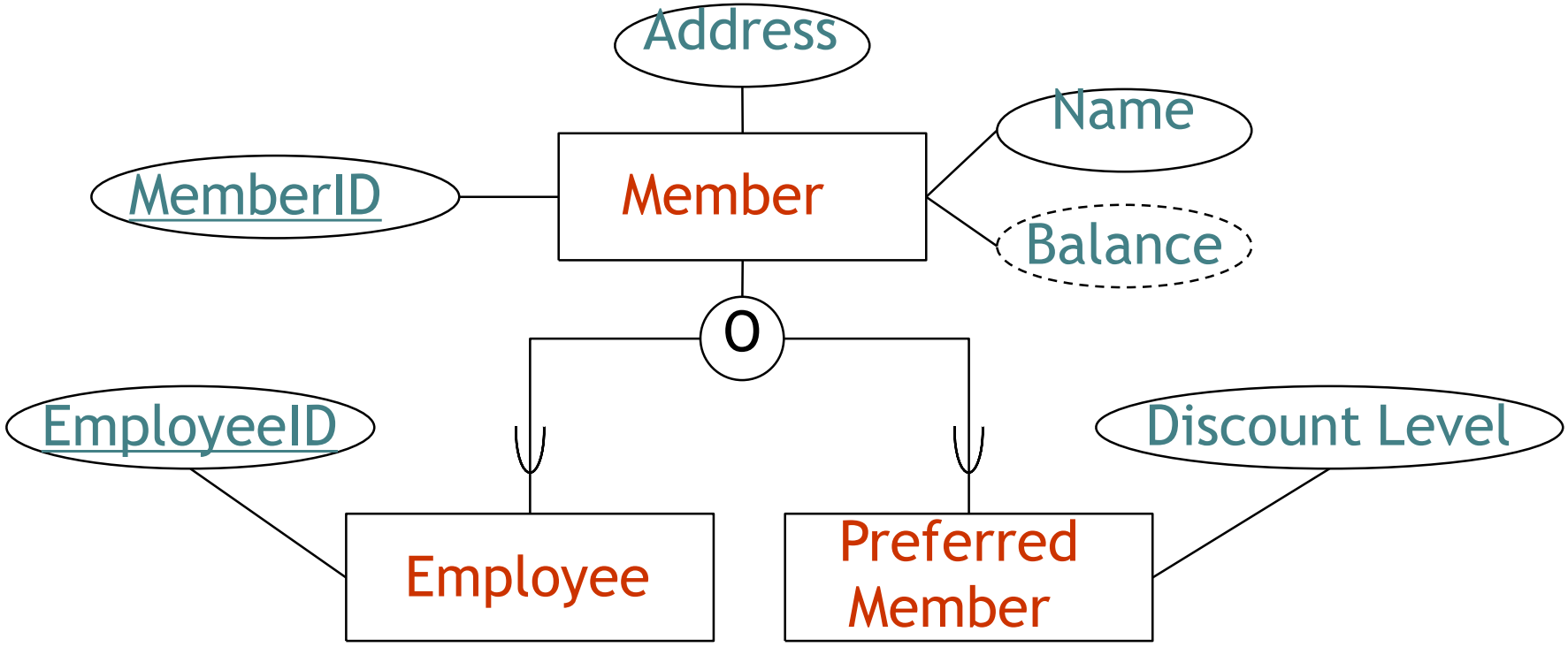


Inheritance Among Classes

- Entities in a subclass *inherit* the attributes of the superclass.
- A subclass may have its own attributes (termed the *specific attributes*).
- Entities in a subclass may participate in relationships directly (termed *specific relationships*), and they may participate in relationships via their superclass(es).

Inheritance Example

- A Preferredmember entity inherits the attributes Name, Address, MemberID and Balance from Member.
- The PreferredMember subclass also has the attribute DiscountLevel.

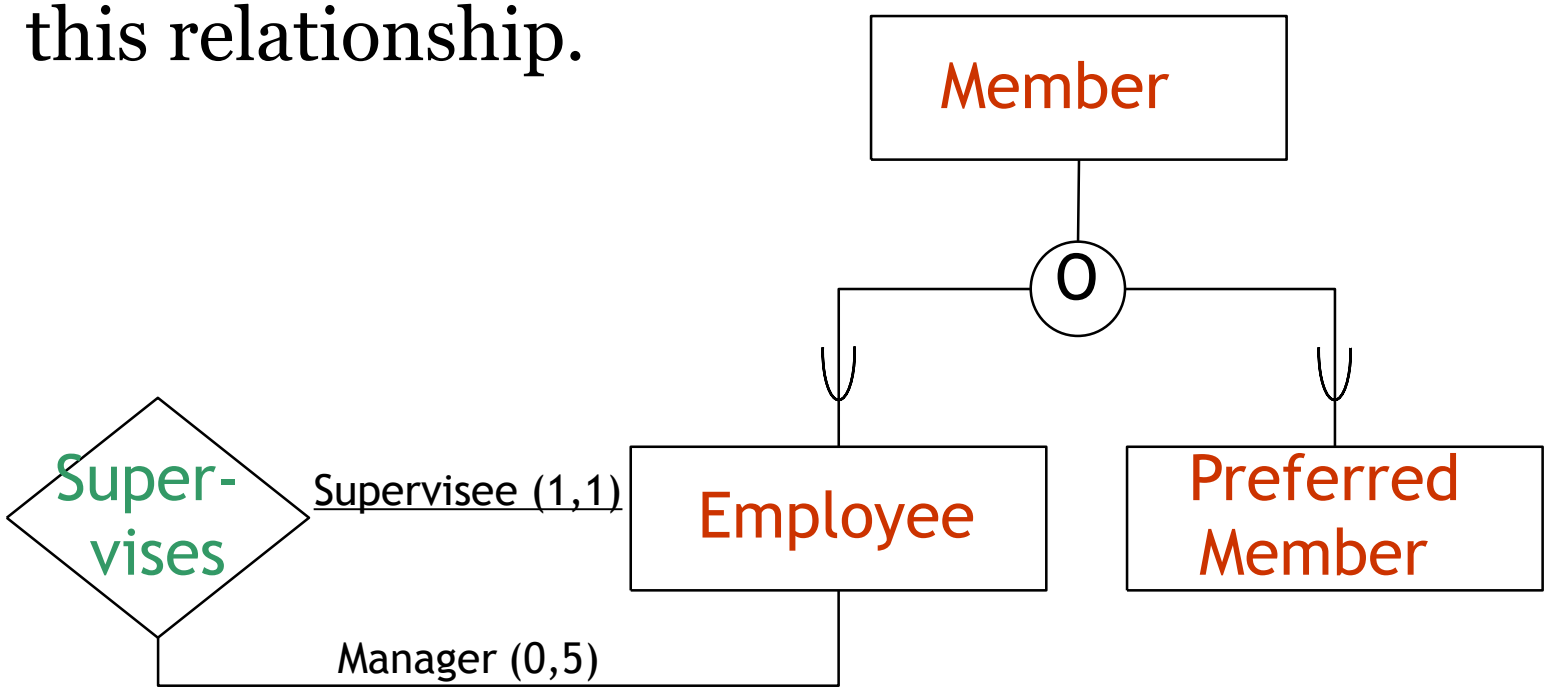


Abstraction

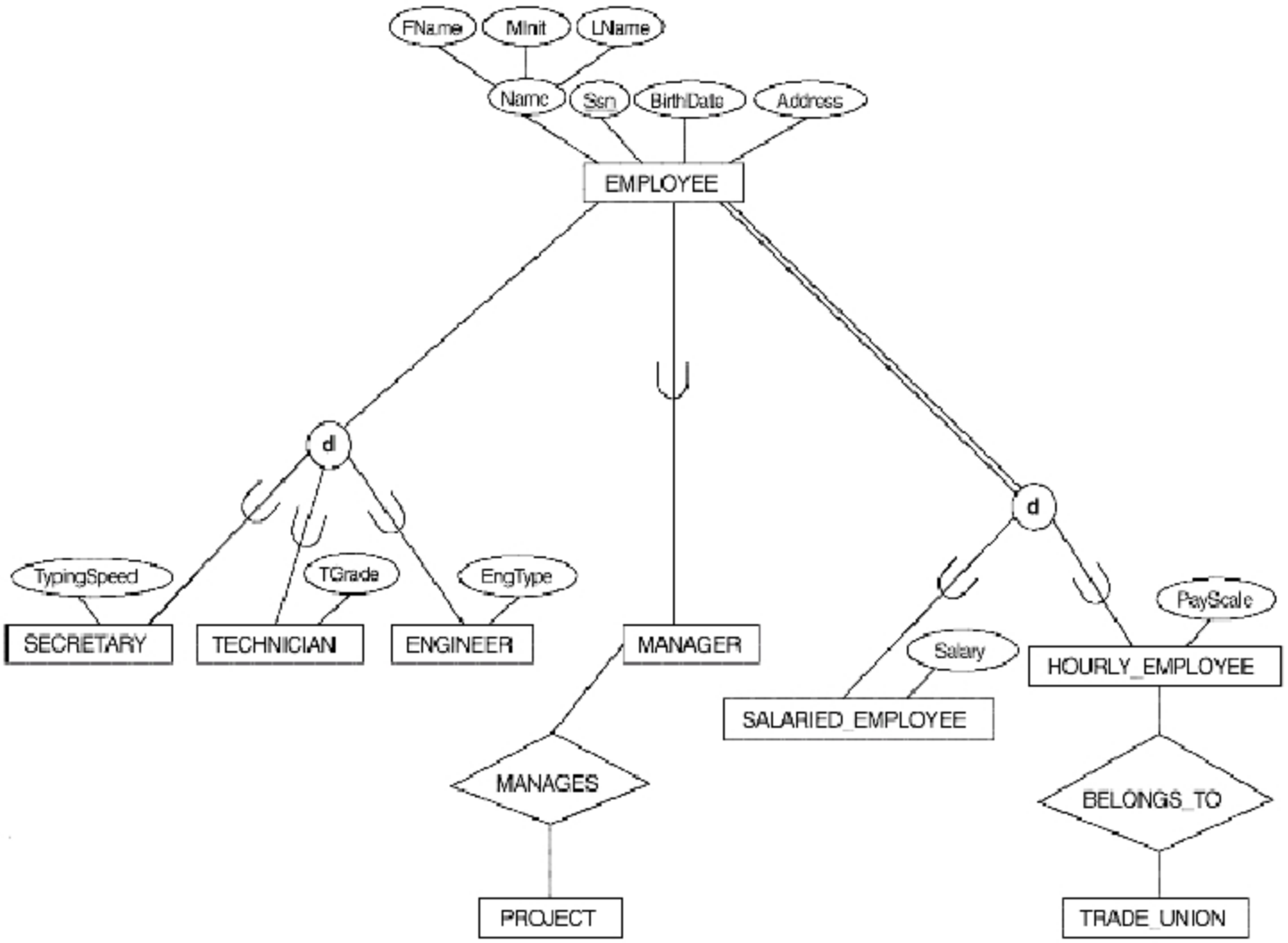
- *Specialization* and *generalization* are reverse processes of abstraction.
  - Specialization - process of defining a set of subclasses of an entity type.
  - Generalization - process of forming a single superclass for a set of entity types.
- When to abstract
  - Attribute(s) relevant only to a subclass
  - Relationship type(s) relevant only to a subclass

Abstraction Example - Subclass Rel. Type

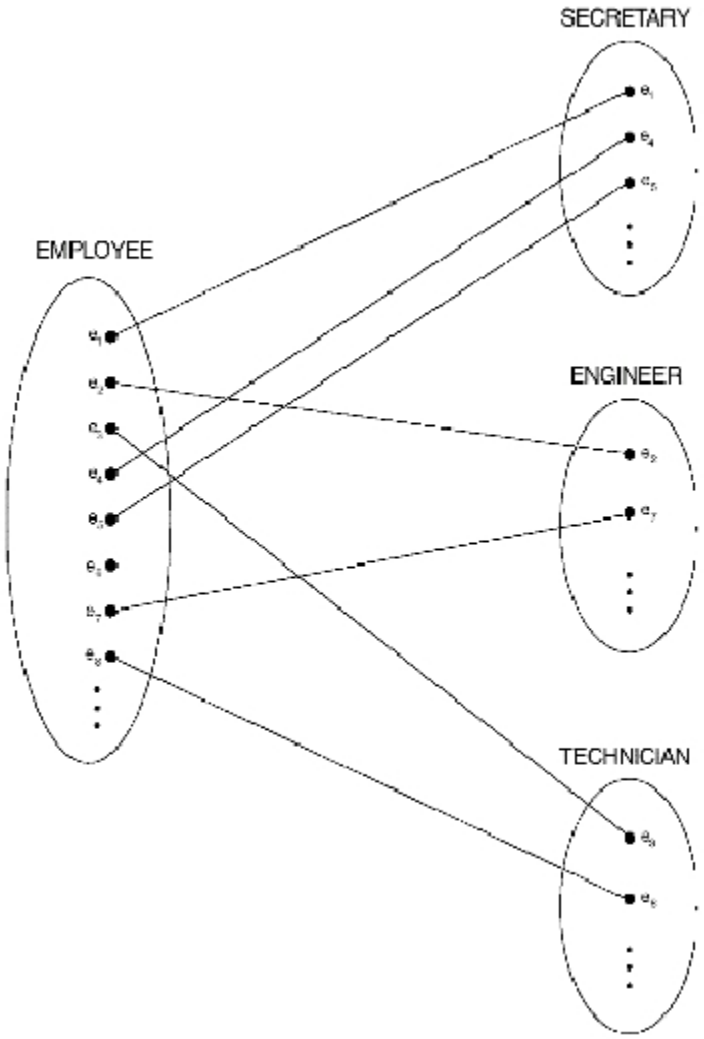
- Entities in the subclass Employee participate in the Supervises relationship.
- PreferredMember entities do not participate in this relationship.



A Example of Specialization



A Example of Specialization

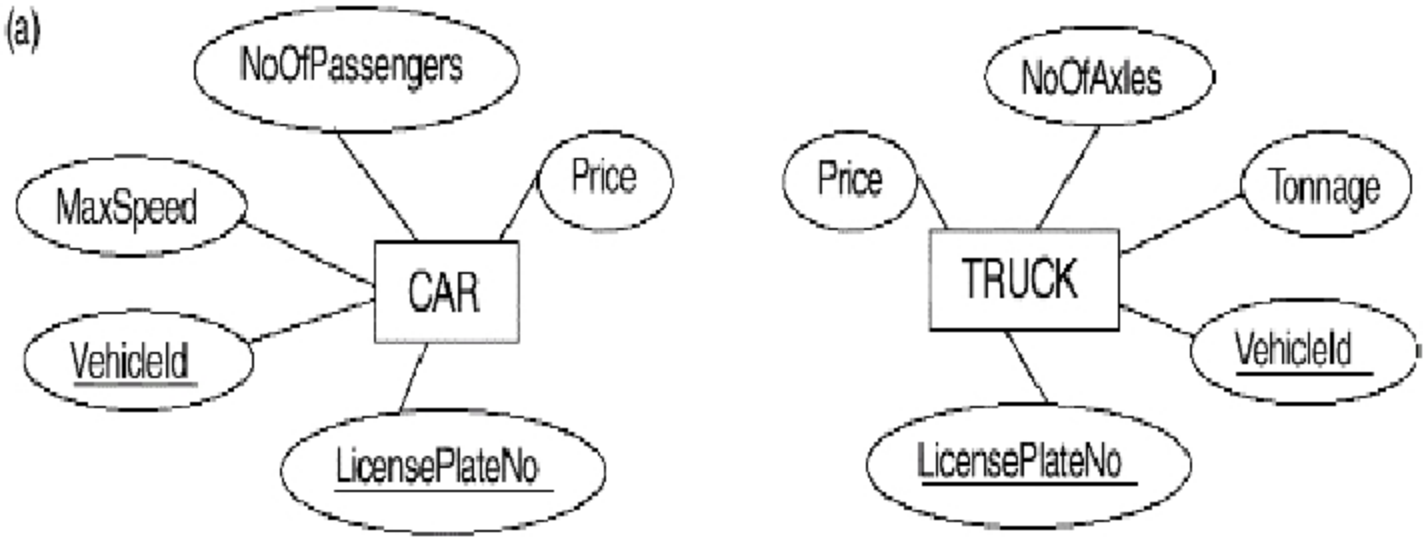


Generalization

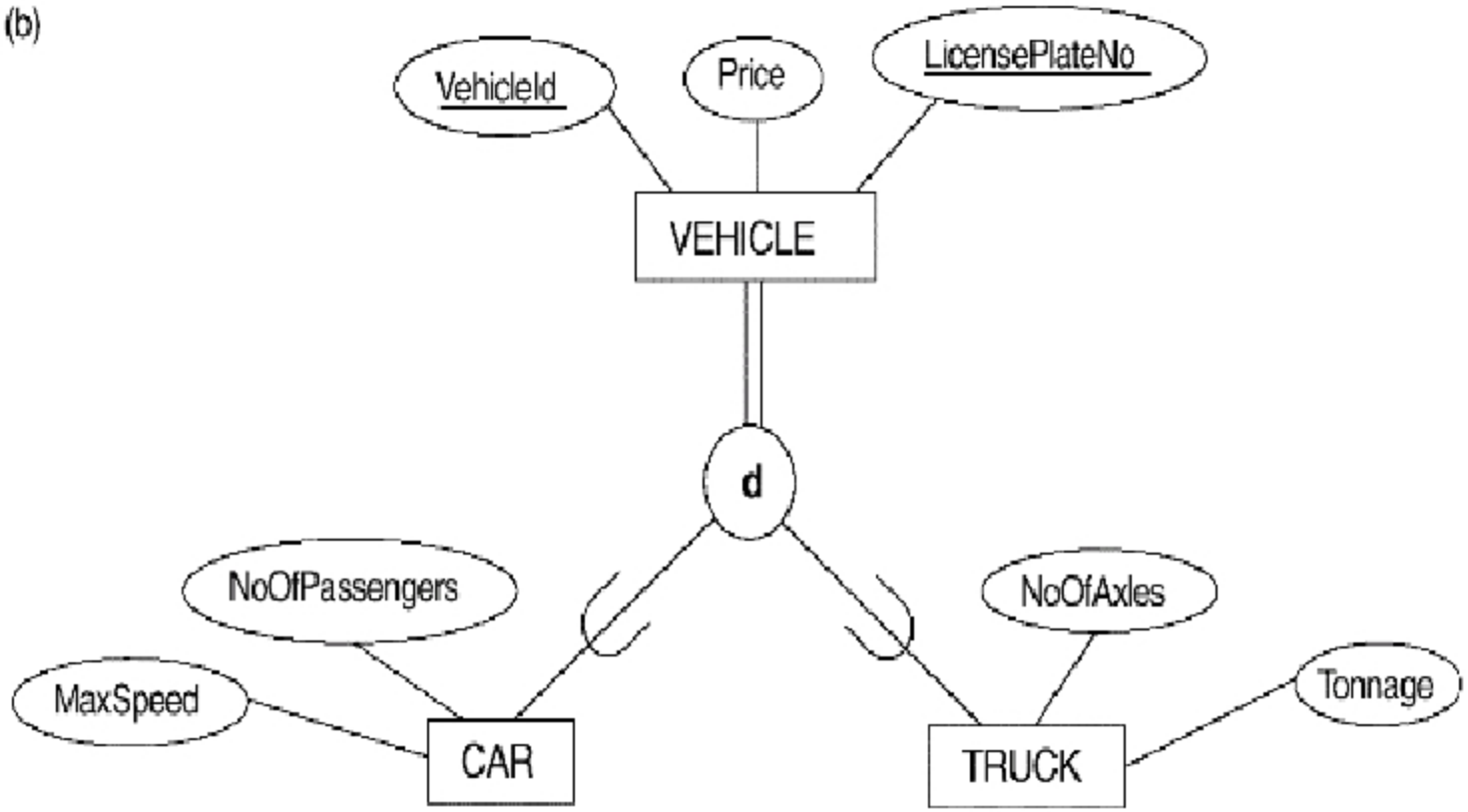
A REVERSE PROCESS abstraction

- Suppress the differences among several entity types, identify their common features.
- Generalize them into a single superclass of which the original entity types are special subclasses.

An Example of Generalization



An Example of Generalization



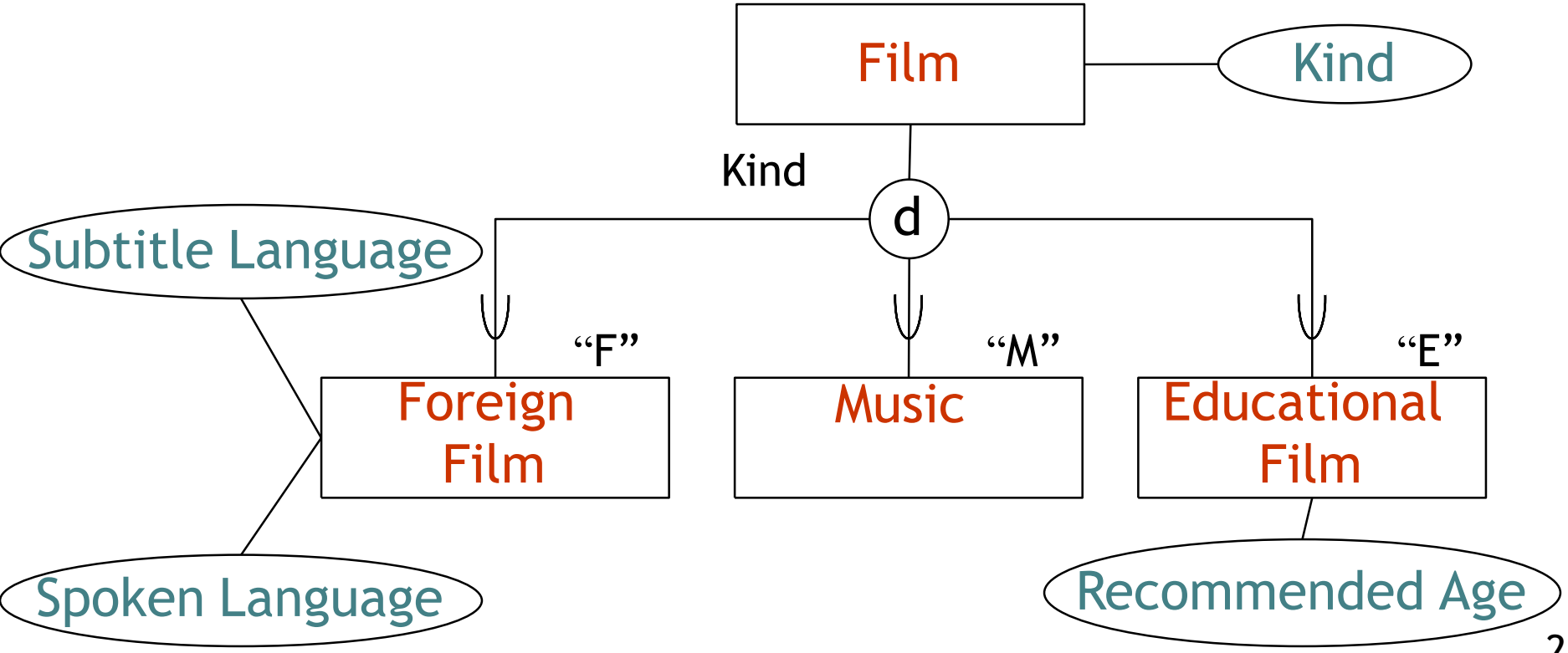
Constraints on Spec. and Gen.

- When subclass membership is *user-defined*, class membership is determined individually for each entity.

- Subclass membership can be defined by a predicate (termed *predicate-defined* or *condition-defined*). Class membership is predefined.
  - Attribute-defined specialization

Constraints Example

- The Film class (with a Kind attribute) can be specialized into a Music (Kind = “M”), Educational Film (Kind = “E”) or Foreign film (Kind = “F”).



Subclass Participation

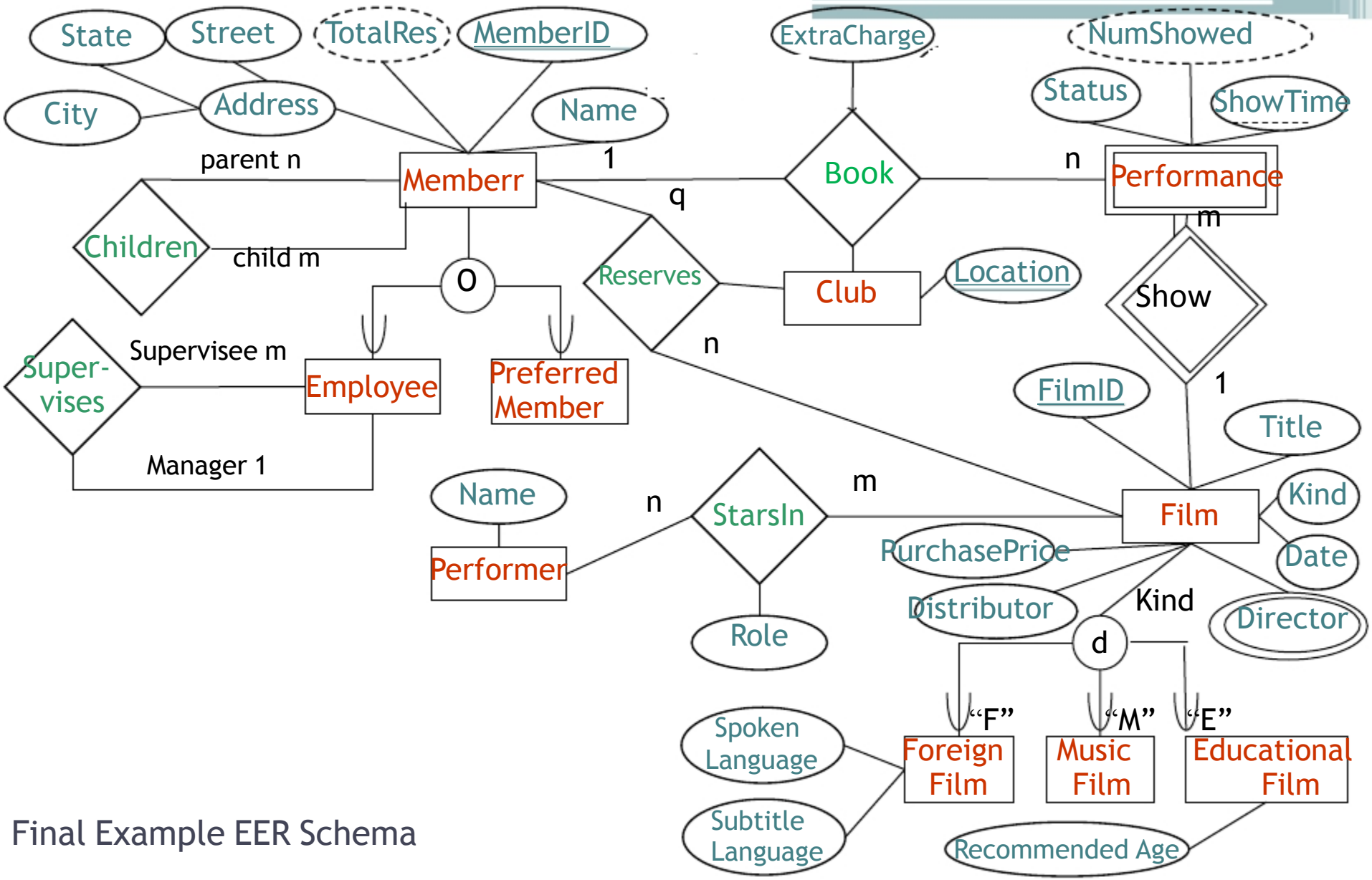
- Subclass groupings
  - disjoint - no entity can be in more than one subclass
    - diamond notation – represent with an (d)
    - crow's feet notation : represent wrepresent with an (x)
  - overlapping – entities may be in several subclasses
    - diamond notation - represent with an (o)
    - crow's feet notation - default representation

Subclass Participation

- Completeness of subclass
  - total - all entities in the superclass are in *some* subclass
    - diamond notation - represent with a double line
    - crow's feet - ??
  - partial - an entity in a superclass need not be in a subclass
    - diamond and crow's feet notation - default representation
- Example:
  - The specialization of the Film class into MusicFilm, EducationalFilm, and ForeignFilm subclasses is disjoint and partial.

Update for Specialization and Generalization

- Deletion *cascades* to subclasses.
  - If a Member entity that is also an employee is deleted from the Member class, it is also deleted from the Employee class.
- Insertions are cascaded automatically to subclasses when entity membership is predicate-defined.
  - Example: The insertion of an entity with a Kind of "E" into the Film class will result in that entity being inserted into the EducationalFilm
- For total specializations, entities are cascaded to at least one subclass.
- Other rules may be derived.



Final Example EER Schema

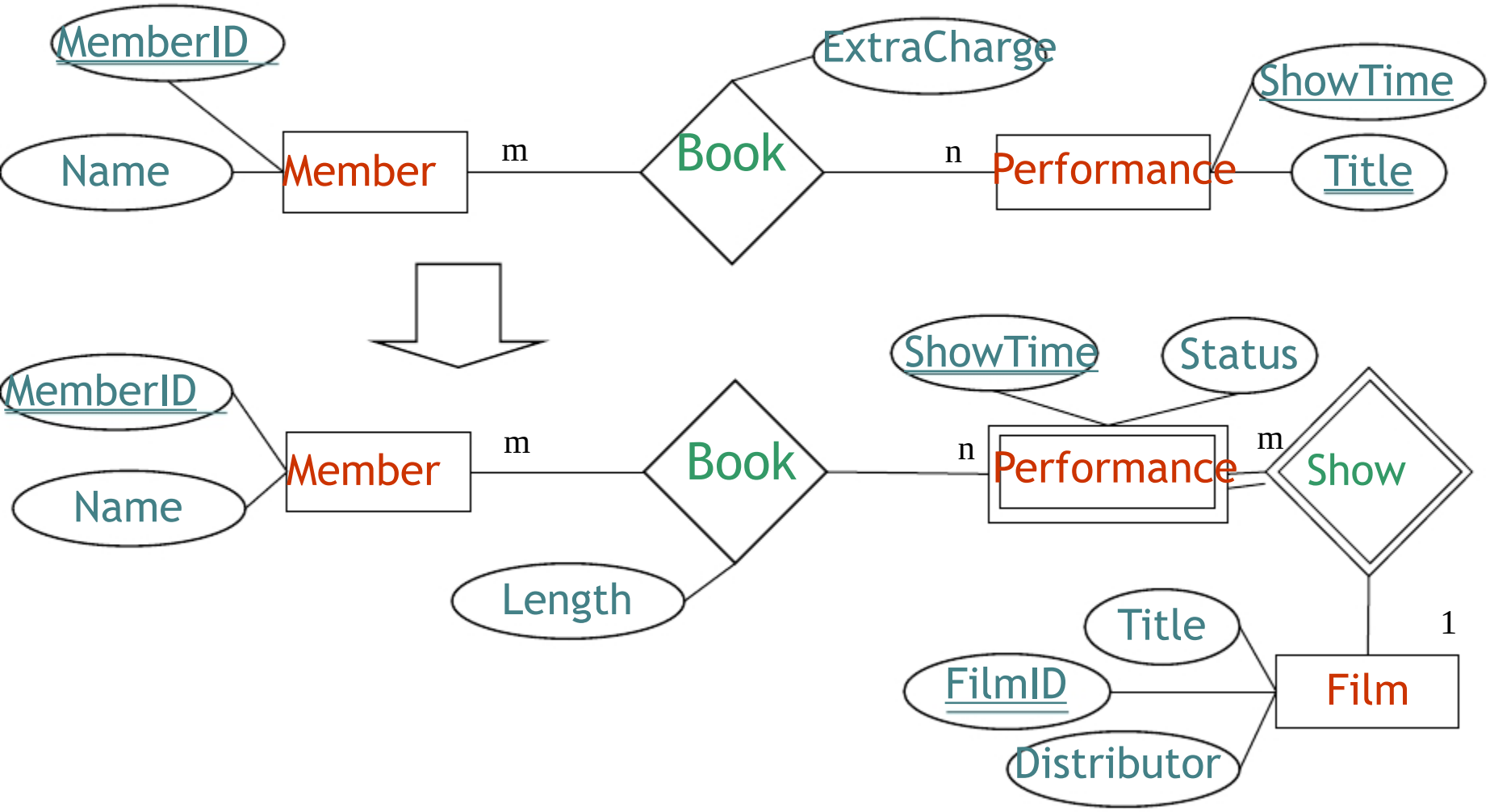
Outline

- ER model
  - Overview
  - Entity types
    - Attributes, keys
  - Relationship types
  - Weak entity types
- EER model
  - Subclasses
  - Specialization/Generalization
- Schema Design
  - Single DB
  - View integration in IS
- uses Integration DEFinition for Information Modeling (IDEF1X) notation in ERwin

Schema Design Strategies

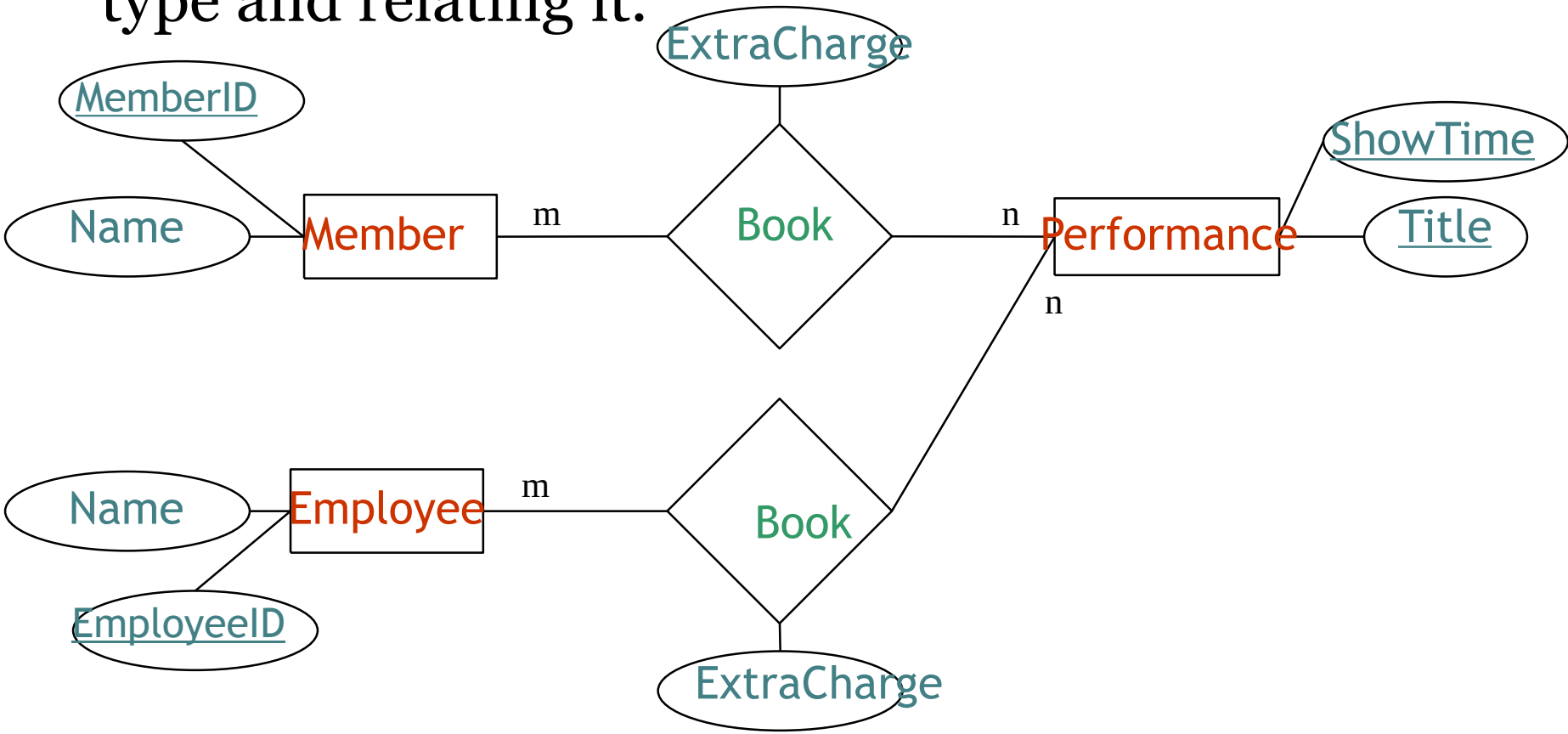
- *Top-down strategy*
  - Start out with high-level, abstract concepts and apply step-wise refinements (e.g., specialization) to add "detail."
- *Bottom-up strategy*
  - Start out with basic concepts and apply refinements (e.g., generalization).
- *Inside-out strategy*
  - Start with a few central concepts and successively include additional concepts.
- *Mixed strategy*
  - Combine the above strategies.

Top-down Strategy

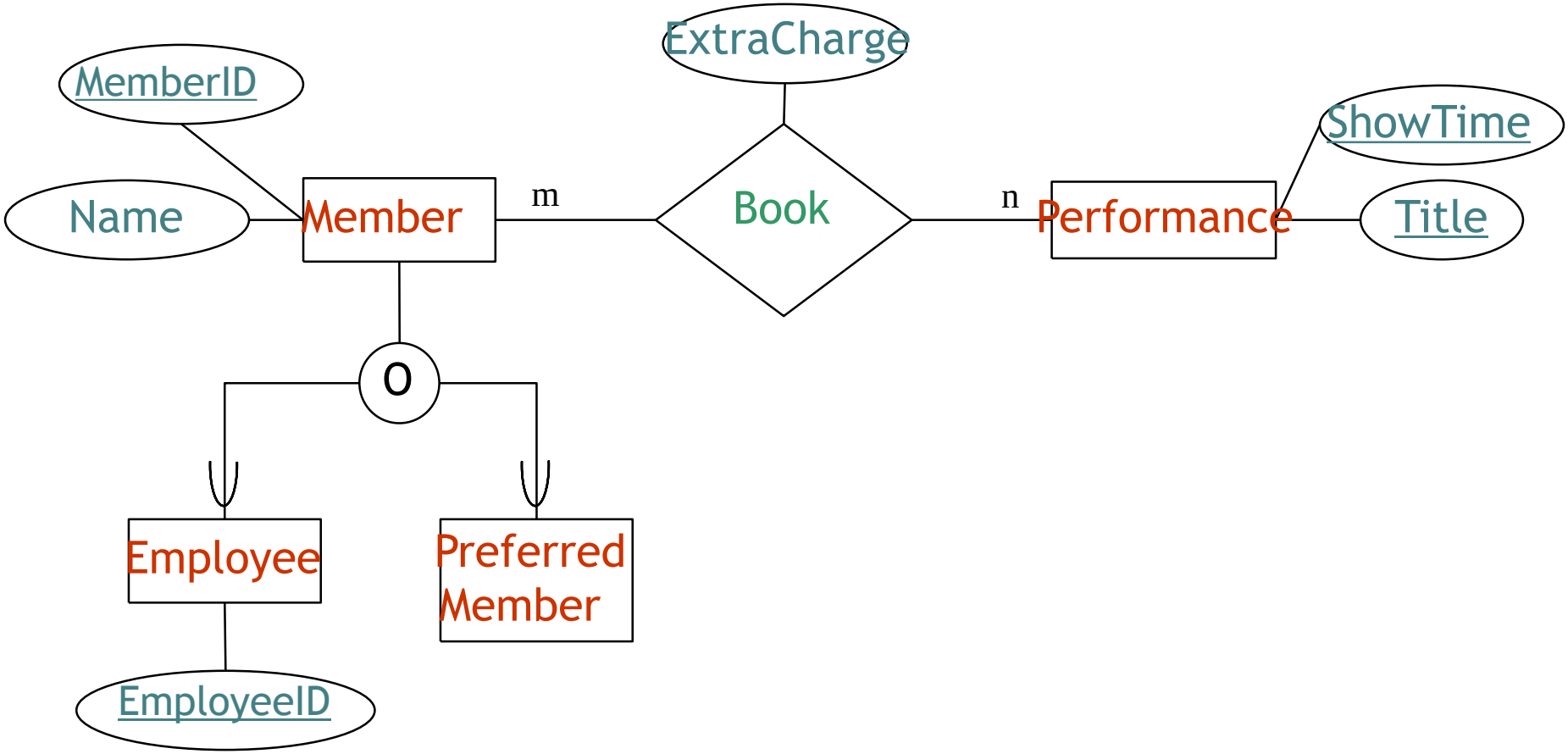


Bottom-up Strategy

- Example: discovering a new generalized entity type and relating it.



- This is converted to:

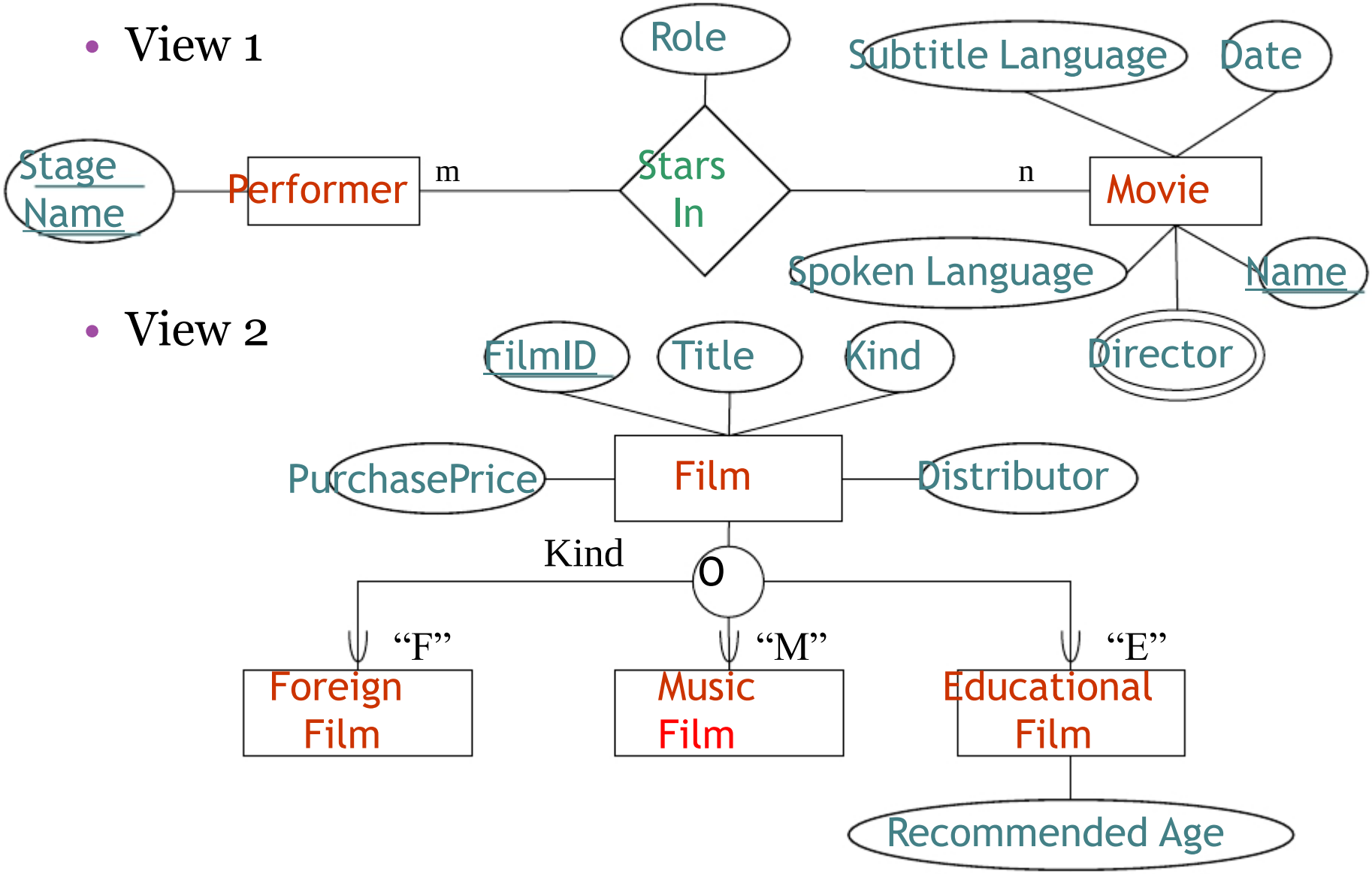


Conceptual Database Design Approaches

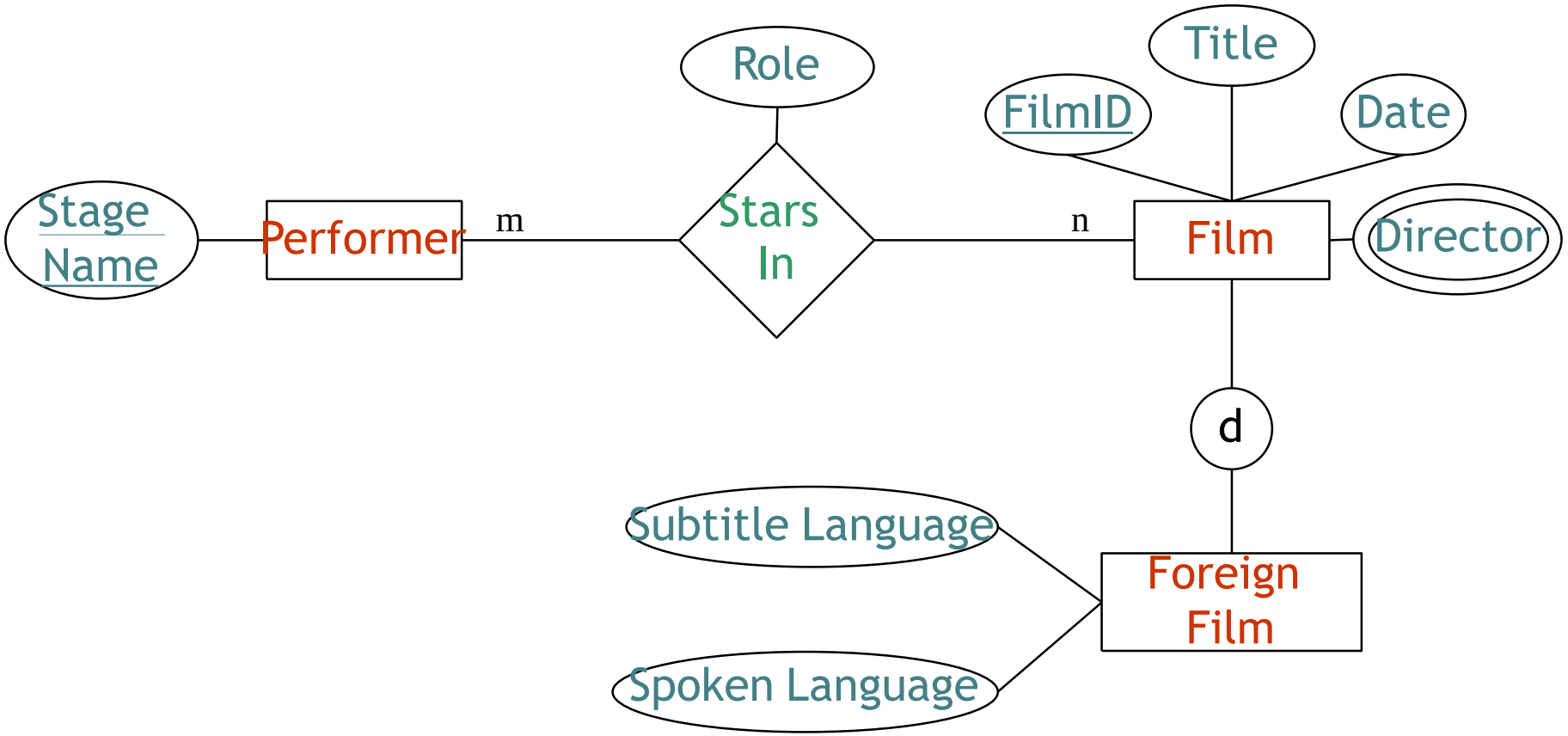
- Centralized design approach
  - Integrate first the requirements for all applications and then design a single schema.
  - Assumes a centralized organization.
  - The DBA merges the multiple sets of requirements.
  - The DBA designs the schema.
- View integration approach
  - Design first a schema for each application in isolation, then integrate the schemas into a single global schema.
  - Each user group can design its own schema.
  - The DBA designs the global schema.

View Integration Example

- View 1
- View 2

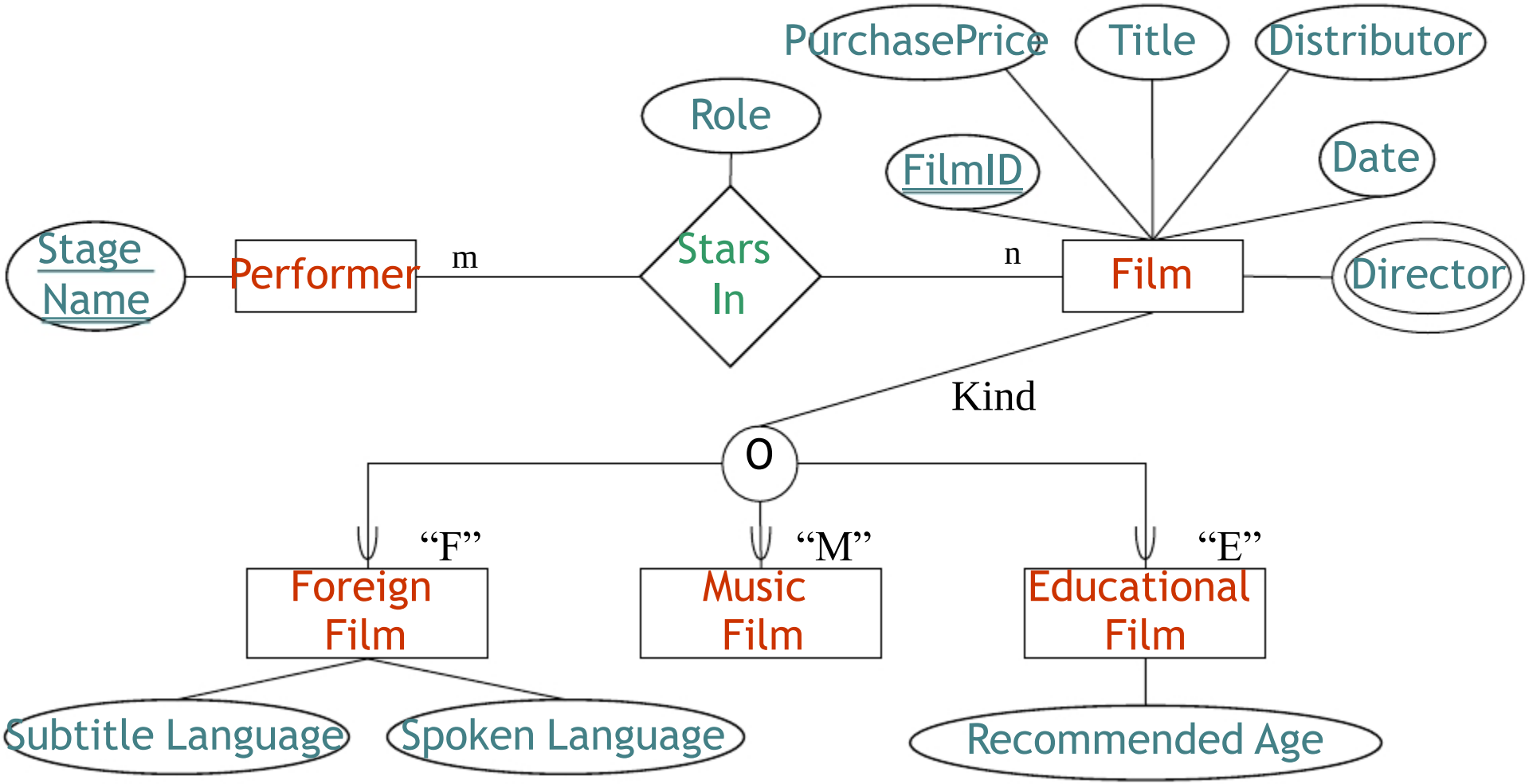


- View 1 is modified to better fit View 2 as shown below.



View Integration Example, cont.

- The two views are then integrated as shown below.



View Integration Methodology

- Identify correspondences and conflicts among views.
  - Naming conflicts (synonyms, homonyms).
  - Type conflicts (e.g., entity vs. attribute vs. relationship).
  - Domain conflicts (e.g., integer vs. string).
  - Constraint conflicts.
- Homogenize views.
  - Resolve (some) conflicts.
- Merge views.
  - Create the global schema.
- Restructure the global schema.
  - Remove redundancy and unnecessary complexity.
- Different integration orders are possible using binary and n-ary integration.

Outline

- ER model
  - Overview
  - Entity types
    - Attributes, keys
  - Relationship types
  - Weak entity types
- EER model
  - Subclasses
  - Specialization/Generalization
- Schema Design
  - Single DB
  - View integration in IS
- uses Integration DEFinition for Information Modeling (IDEF1X) notation in ERwin