Topic
Mapping a EER Schema to Relations

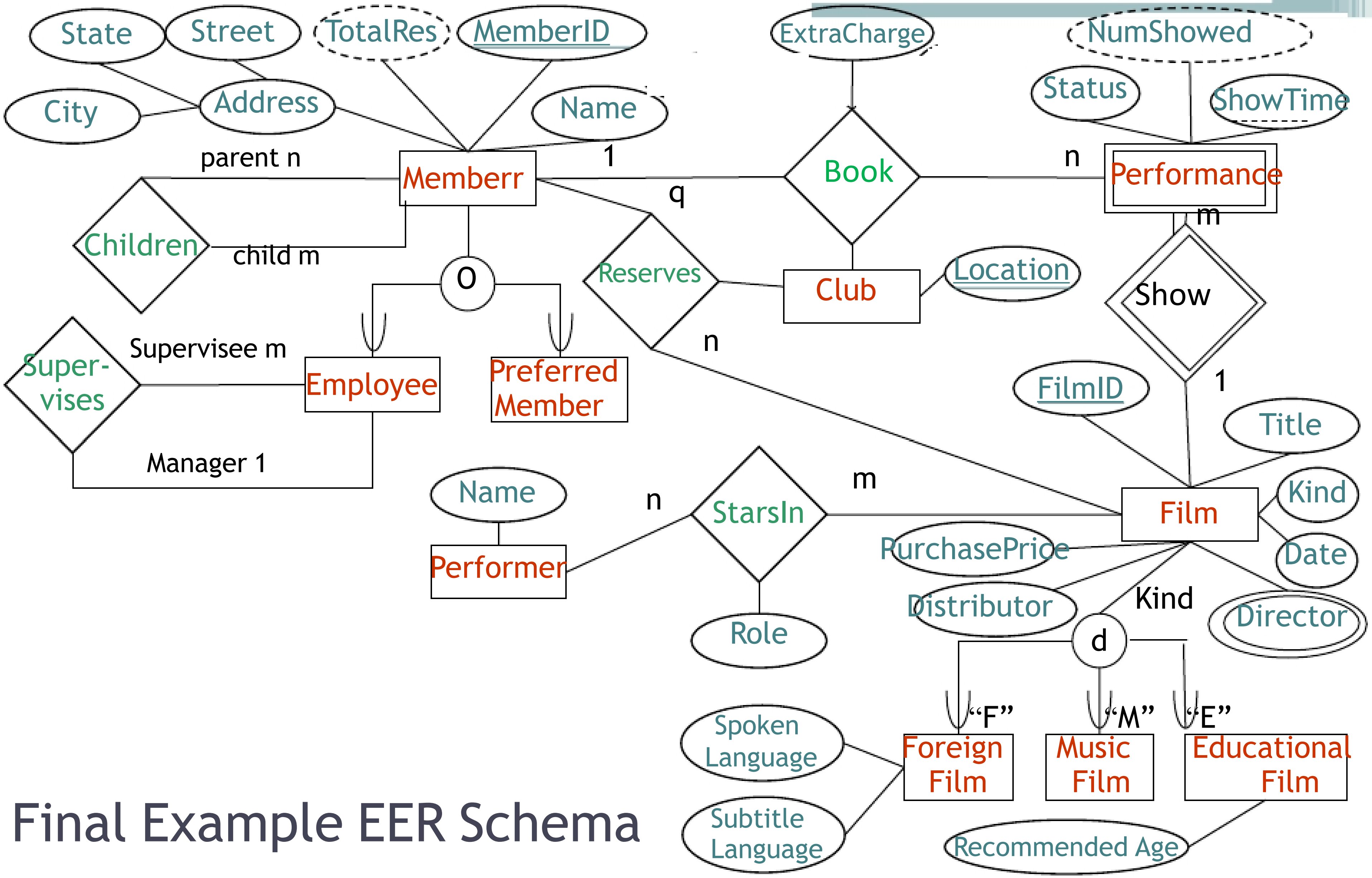
Overview

- Mapping entity types
- Mapping relationship types
  - One-to-one
  - One-to-many
  - Many-to-many

Mapping an EER Schema to Relations

- In a sequence of steps, a set of relations is created.
- Sometimes automated in CASE tools

1. Regular entity types
2. Weak entity types
3. Binary 1:1 relationship types
4. Binary 1:N relationship types
5. Binary M:N relationship types
6. *n*-ary relationship types
7. Multi-valued attributes



Final Example EER Schema

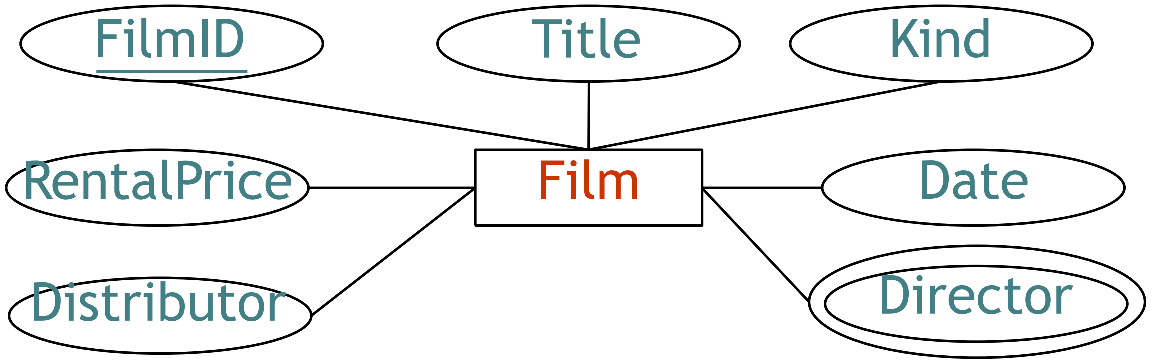
1. Entity Type Maps to a Table

- Create a table for each regular entity type.
  - One column in table for each *simple* attribute
  - Derived attributes may or may not appear (your choice)
  - Table‟s primary key is the primary key of the entity type

- *Optimization*: If there are no attributes other than the primary key, and if the entity participates totally in a relationship, then the table can be eliminated.

1. Entity Type Example

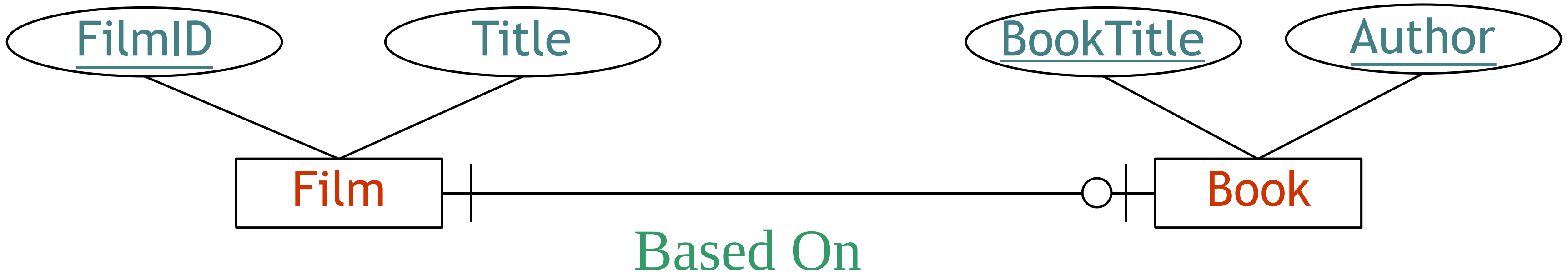
- Consider the Film entity type



- Maps to the following table (relational schema).

  Film (<u>FilmID</u>, Title, PubDate, RentalPrice, Distributor, Kind)
- Note, primary key of table is key of entity type.

- Assume each book is the basis for a film.



- Book table can be eliminated by putting Book information into Film table since Book participates totally and has only key attributes.
- Maps to the following table (relational schema).

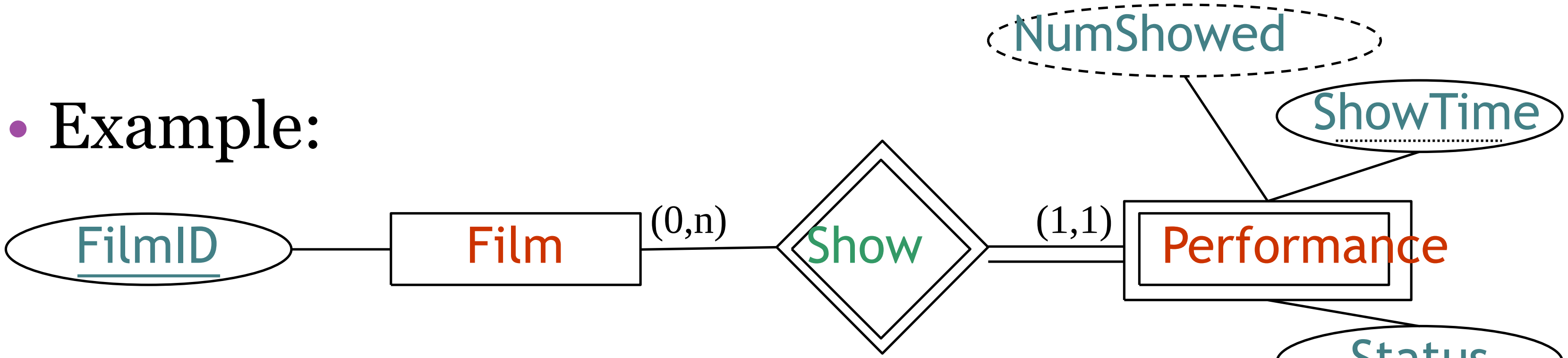
  Film (FilmID, Title, BookTitle, Author, Publisher)

2. Weak Entity Type Maps to a Table

- Create a table for each weak entity type
  - One column for each simple attribute
  - Include column(s) for the primary key of each *owner* entity type. These columns are *foreign keys*
  - The primary key is the combination of each owner primary key and the partial key.

2. Weak Entity Type Example

- Example:



- Performance weak entity type (and Show) maps to Performance (FilmID, ShowTime, Status)
    - Chose not to store derived attribute NumShowed
    - Film entity type maps to different table
    - Show relationship type is not mapped to a table

Overview

- Mapping entity types
- Mapping relationship types
  - One-to-one
  - One-to-many
  - Many-to-many

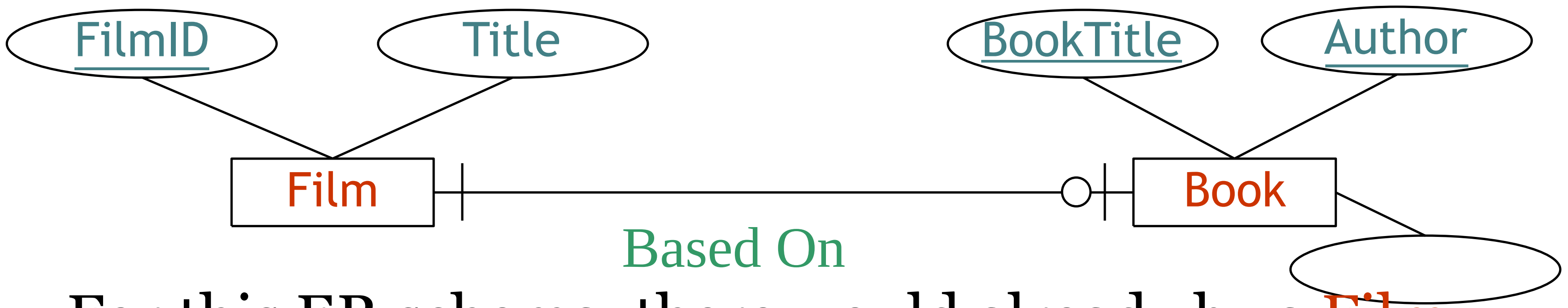
Mapping Relationship Types - General

- Each relationship type is mapped to a table
- Columns are
  - Attributes of relationship type
  - Key attributes of all the participating entity types
- Keys in table are
  - Primary key - combined key of all the "many" sides in relationship type
  - Foreign keys - Each "borrowed" key is a foreign key
- Optimization: Often the table can be eliminated by extending the table for one side of the relationship

3. Mapping 1-1 Relationship Types

- For each 1:1 binary relationship type, extend one of the tables for a participating entity type.
  - Primary key of the other entity type becomes a foreign key in this table
- It is best to extend a table of an entity type with total participation
- Add columns for each of the simple attributes of the relationship type
- *Optimization*: Perhaps remove the table corresponding to the other entity type

- Each book is the basis for some film

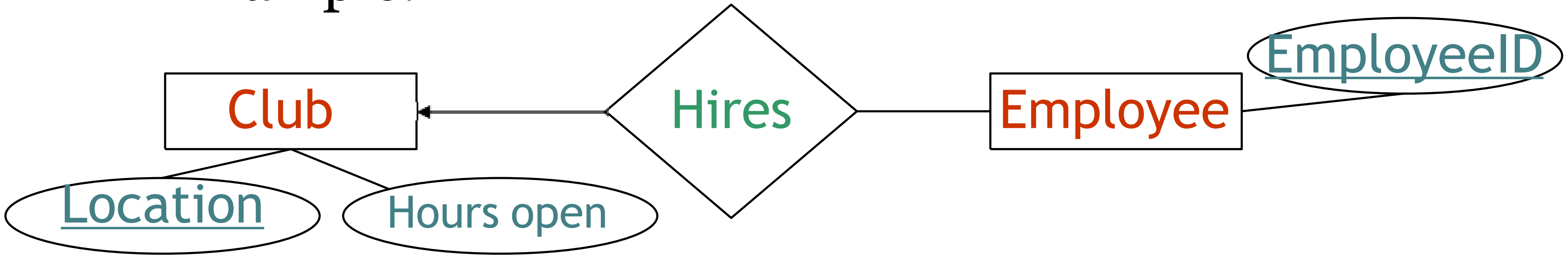


- For this ER schema, there would already by a Film table and a Book table from step 1, Extend the Film table to include the key of Book, which is BookTitle, Author.
- Film(FilmID, Title, BookTitle, Author)
- Optimize: remove Book table, add PubDate to Film

4. 1-to-Many Relationship Types

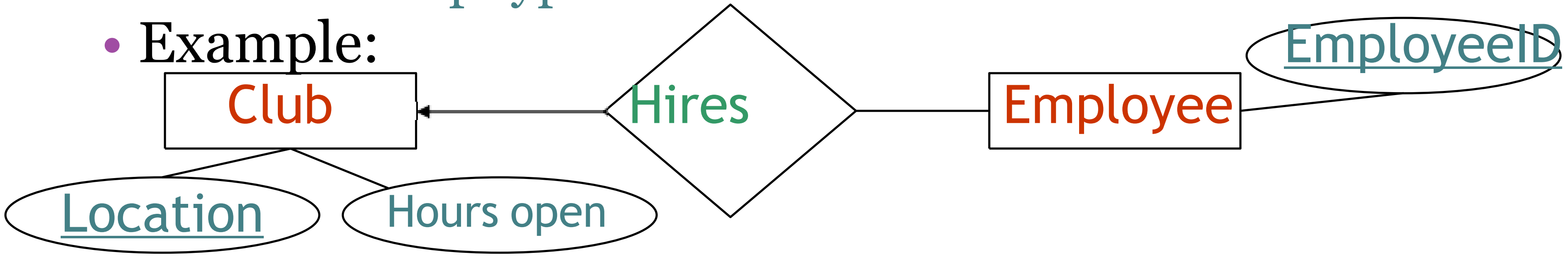
- For each regular 1:N binary relationship type, there are several approaches
  - Option 1: Create a separate table for the relationship type
    - Three tables result
    - Key of relationship table is key of "many" side
  - Option 2: If the relationship is total, then extend a table corresponding to the „many" entity type
    - Two tables result (optimization)
  - Option 3: If the relationship is not total, extend a table with nullable attributes (sometimes not allowed for foreign keys)
    - Two tables result (optimization)

- Create a table for the relationship type
  - Add columns for each of the simple attributes of the relationship type
  - Add columns for each of the keys of the participating entity types
  - The key of the table is the key of the „many“ side
- Example:



- Create a Hires table
  - Club(<u>Location</u>, HoursOpen)
  - Hires(Location, <u>EmployeeID</u>)
  - Employee(<u>EmployeeID</u>)

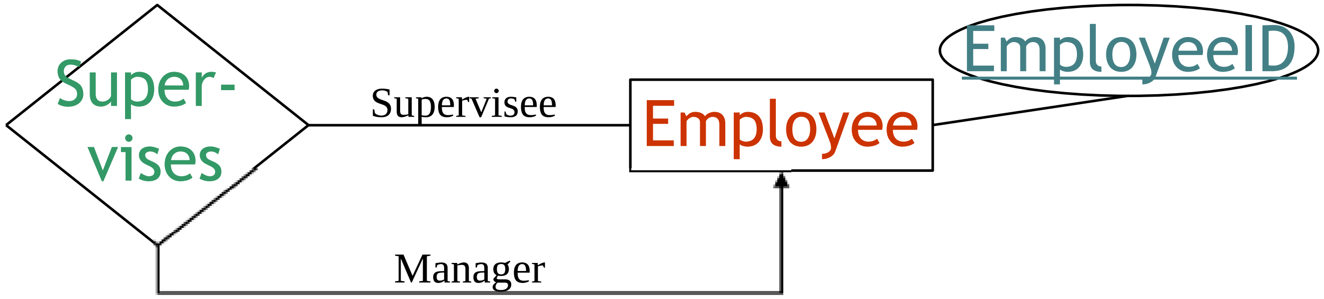
- Do not have a table for the relationship type
  - Extend the table"s „many" side with the primary key of the other participating entity type. This is a foreign key
  - Add columns for each of the simple attributes of the relationship type
- Example:



  - Extend the Employee table with a Location column. Employee(EmployeeID, Location) Club(Location, Hours Open)

4. Column Renaming

- Column names
  - Taken from attributes, usually unchanged
  - Two columns in a table cannot have the same name
    - Must rename columns to retain uniqueness
    - The renaming does not affect primary/foreign key status
- Example



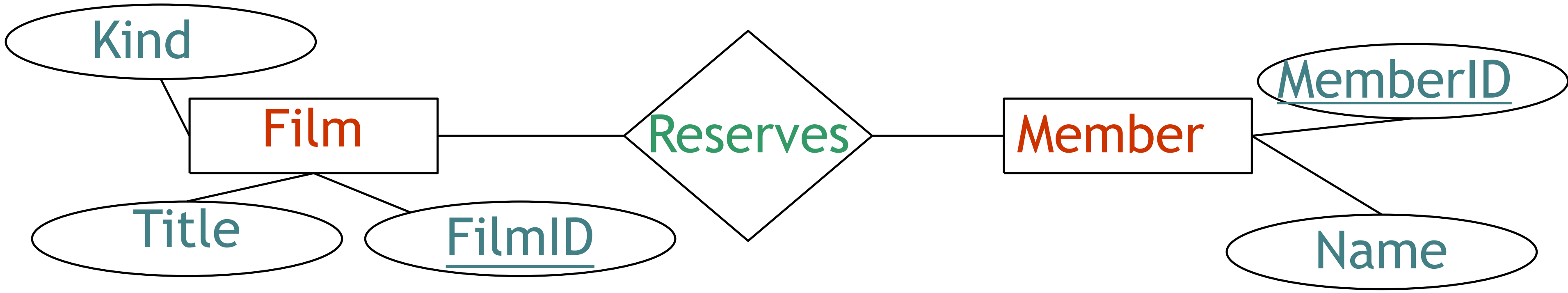
- Must rename EmployeeID columns to disambiguate

  Supervises(Manager, Supervisee)
  Employee(EmployeeID)

5. Many-to-Many Relationship Types

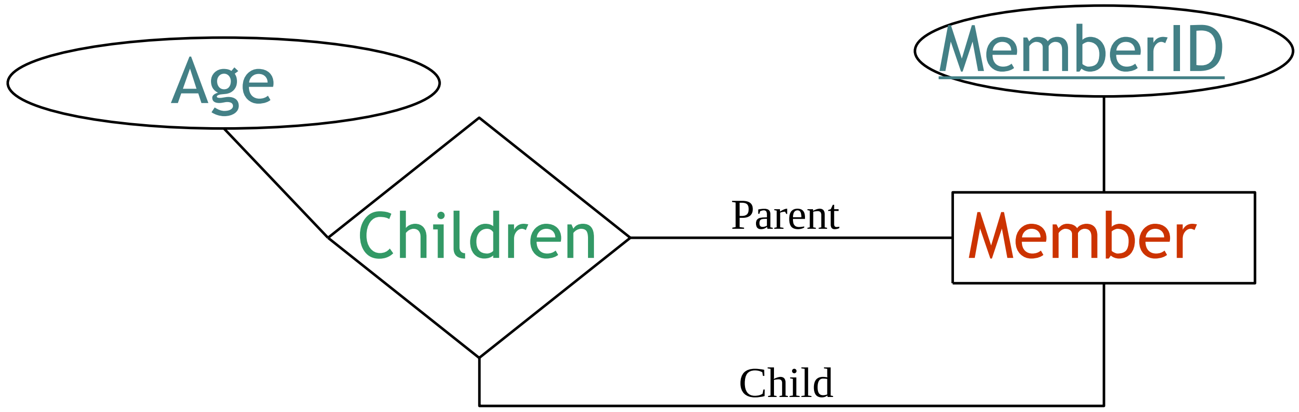
- Create a table for each binary M:N relationship type
- The table has columns for
  - A column for each primary key attribute in a participating entity type. These are foreign keys
  - A column for each of the simple attribute of the relationship type
- The primary key of the table is the union of the primary keys of the participating entity types

5. M:N Relationship Types Example



- Film (<u>FilmID</u>, Title, Kind)
- Reserves(<u>FilmID</u>, <u>MemberID</u>)
- Member (<u>MemberID</u>, Name)

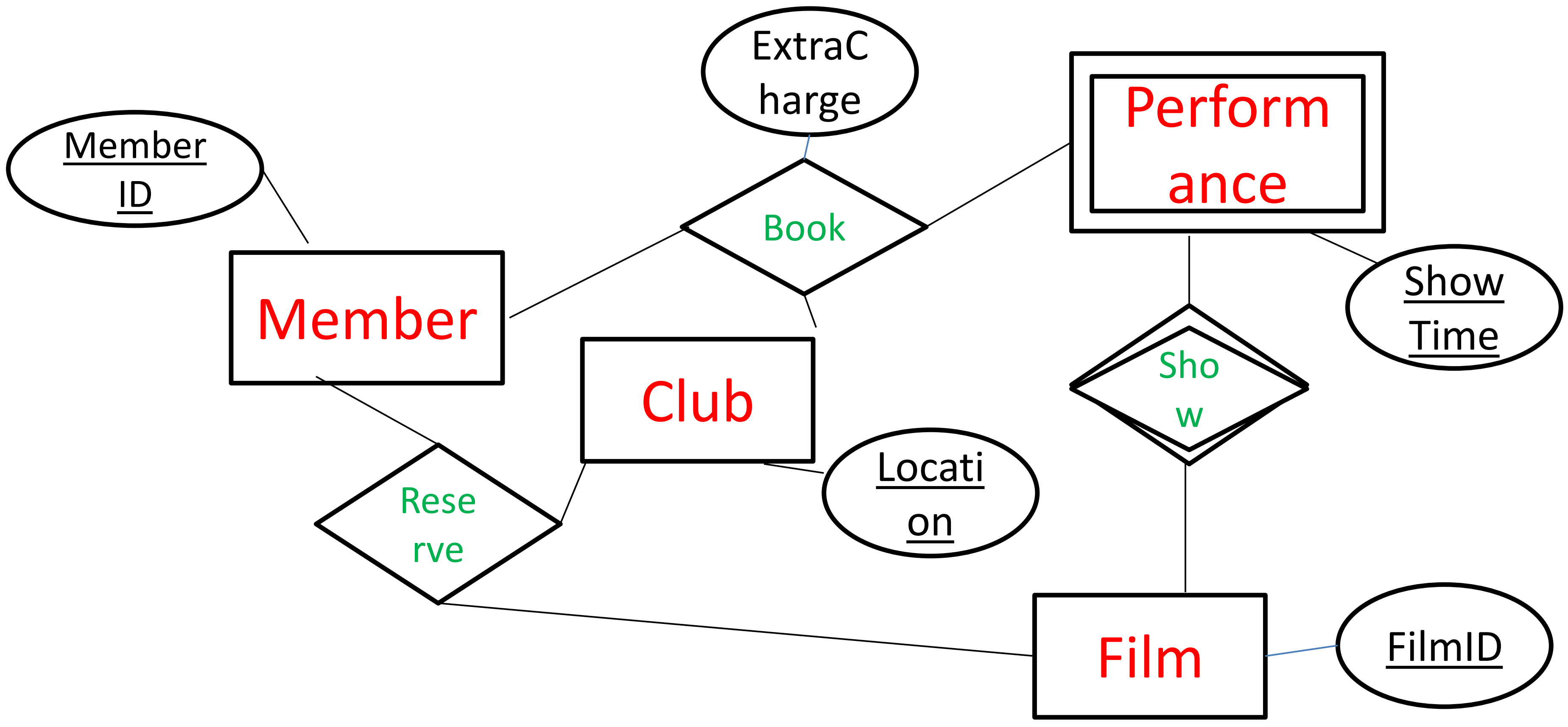
5. Reflexive M:N Rel Types Example



- Children (<u>Parent</u>, <u>Child</u>, Age)
- Member (<u>MemberID)</u>

6. N-ary Relationship Types

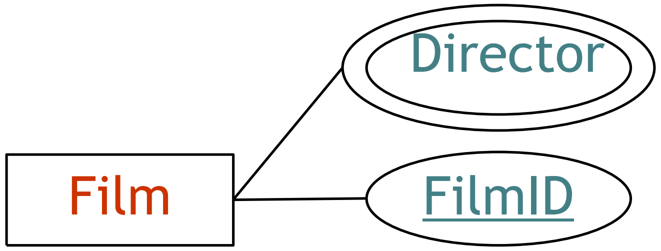
- Create a table for each *n*-ary (*n* > 2) relationship type
  - Columns in the table are the primary keys of the participating entity types. (These are foreign keys)
  - Also include columns for each simple attribute of the relationship type
- The primary key of the created table is the union of the primary keys of the participating entity types
- *Optimization*: If the relationship type is (1,1) on a side, it may be possible to remove an entity table, placing its attributes in the table associated with the relationship



- Reserves (MemberID, FilmID, Location)
- Book (MemberID, FilmID, ShowTime, Location, ExtraCharge)

7. Multivalued Attributes

- Create a table for each multivalued attribute
  - The table has a column for each simple attribute of the multivalued attribute
  - Add columns for the primary key of the entity or relationship type to which the attribute belongs. (This is a foreign key)
- The primary key is the combination of all the attributes
- Example:



- Director (<u>FilmID</u>, <u>Name</u>)

Result of Film Club Relational Schema

- Entity types
  - Member (MemberID, Name, Street, City, State)
  - Film (FilmID, Title, PubDate, PurchasePrice, Distributor, Kind,RecommendedAge, SpokenLanguage, SubtitleLanguage)
  - Performance (FilmID, ShowTime, Status)
  - Club (Location) The primary key is the combination of all the attributes
  - Performer (PerformerID,Name)

Result of Film Club Relational Schema, cont.

- Relationship types
  - ChildOf(Parent, Child)
  - Reserves (MemberID, FilmID, Location)
  - Book (MemberD, FilmID, ShowTime, Location, ExtraCharge)
  - StarsIn(PerformerID, FilmID, Role)
- Multi-valued attributes
  - Director (FilmID, Name)
- Subclasses
  - Employee (EmployeeID, MemberID, Manager)
  - Preferrred Member (MemberID, DiscountLevel)