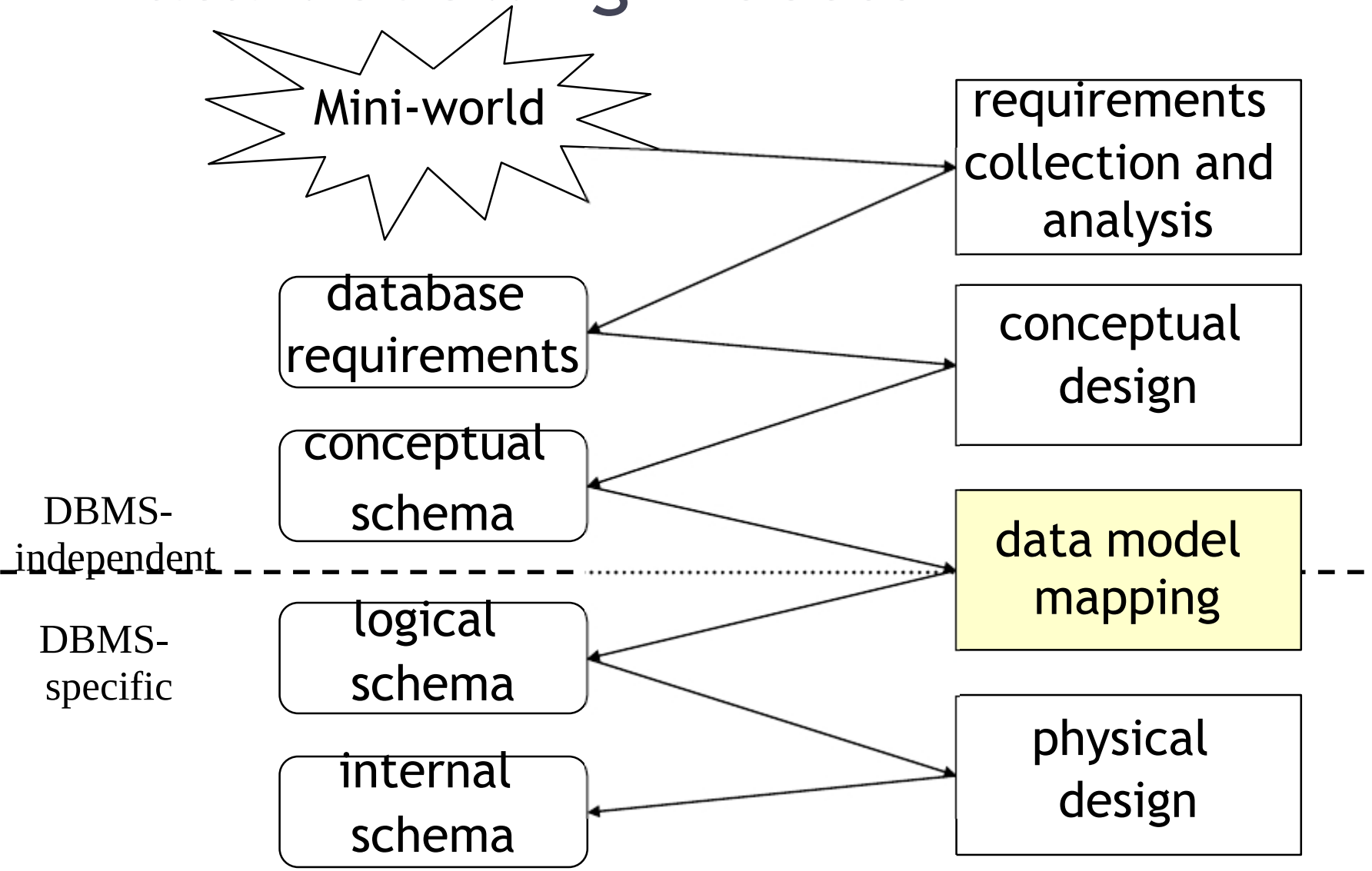
Topic
Database Modelling I

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

Data Modelling Process

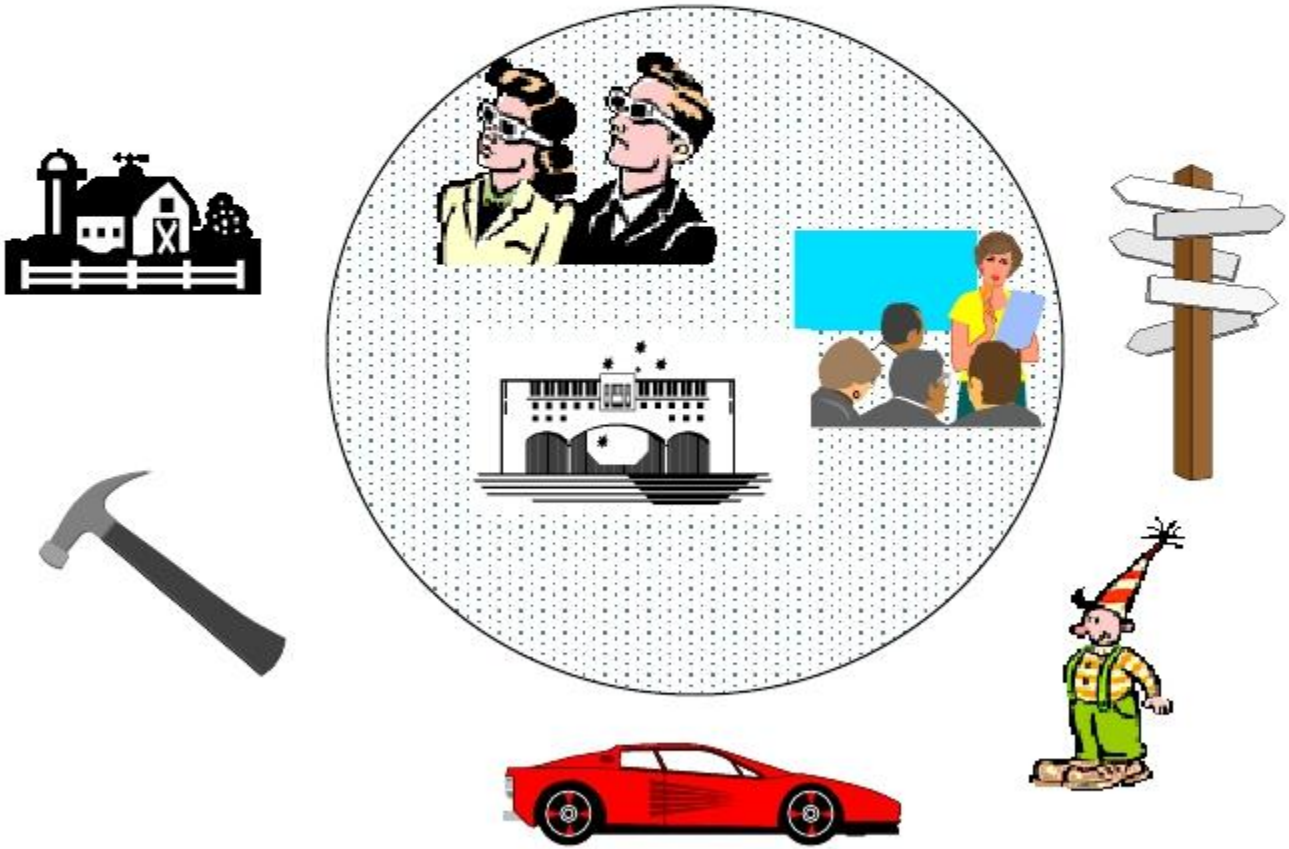


Entity-Relationship Model

- This model is used in conceptual design.
- Popular in CASE tools, e.g., Visual Paradigm, SDE
- A *database* can be modelled as
  - a collection of entity types, and
  - relationships among entity types.
- Result is an *ER Schema* or an *ER Diagram*.

MiniWorld

Definition: A *miniworld* is a part of the real world that we are interested in modelling.



Entity

Definition: An entity is an object that exists and is distinguishable from other objects.

- Object, exists, distinguishable
- Film Club mini-world example
  - Physically exist

    The MEMBER Jane Doe.

    The Drama CLUB.

    The EMPLOYEE Juan Gonzales.

Entity, cont.

- Abstract or organisational entities that do not exist physically
  - The Introduction to Databases COURSE.
  - The SCHOOL of Electrical Engineering and Computer Science.
- Events
  - The TUTORIAL on Friday, October 31, 20017.
  - The EXAMINATION on Tuesday December 15, 20017.

Attribute

Definition: An *attribute* is a property of an entity.

- Feature

  example: MEMBER

  Name

  Date of Birth (DOB)

  example: Performance

  Show Time
  Title

- MEMBER entity examples

  Name=Jane, DOB=1/1/70

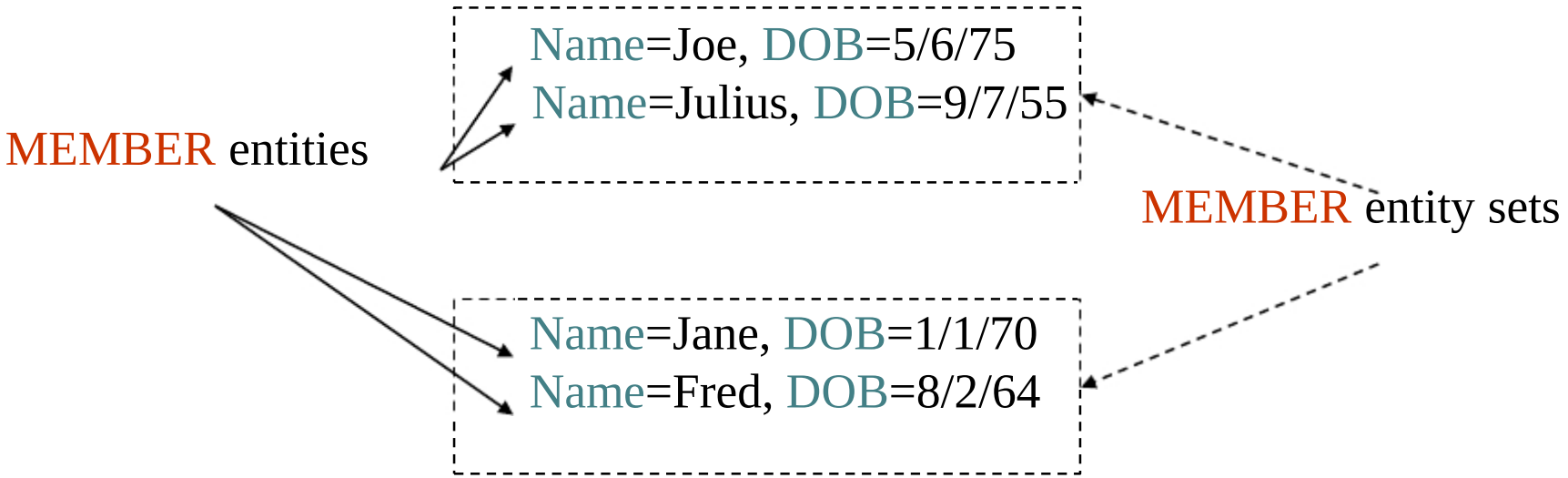
  Name=Joe, DOB=5/6/75

Entity Type

Definition: An *entity set* is a set of entities of the *same type*.

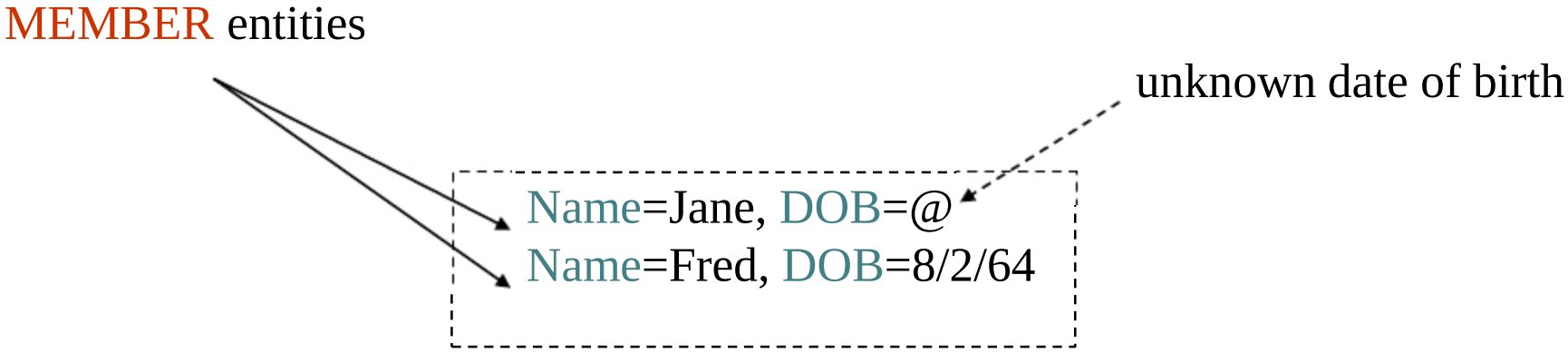
Definition: An *entity type* is a description of the attributes that a set of entities has in common.

- Entity *set*: a set of actual entities
- Entity *type*: the ideal description of a kind of entity
  - Example: A MEMBER entity type. A MEMBER has a name and a date of birth (DOB).



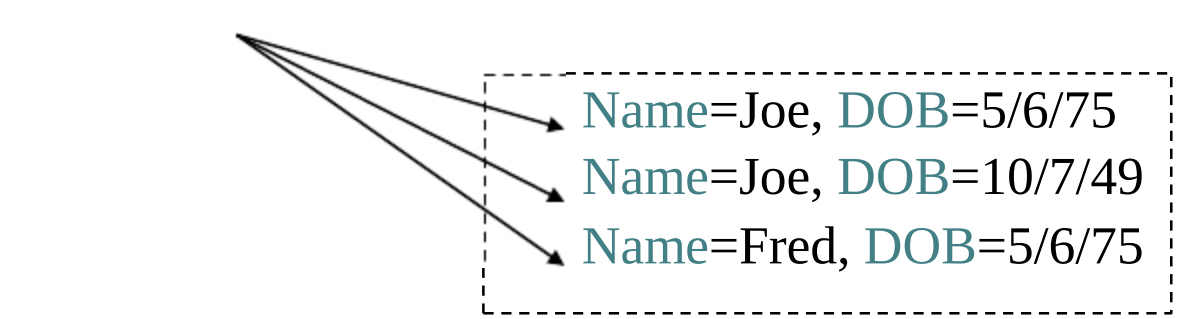
Entity Type, cont.

- Focus on attributes of an *ideal* entity
- An entity has a value for every attribute of the entity type
- Use a *null value* to represent the following.
  - An unknown attribute value
    A MemberAddress is unknown.
  - An inapplicable attribute value
    A MemberPhone number is inapplicable if that person does not own a phone.



Distinguishing Entities

- How can entities be distinguished from each other?
- Assumption: Each entity will have a unique combination of attribute values.
  - Example, assume the Member entity type has Name and DOB attributes. There may be several Member with a Name of „Joe“. Also, there may be several Member with a DOB of 5/6/75. But there can be *at most one* Member with the Name „Joe“ and a DOB of 5/6/75.



- Is there a *smaller* such set of attributes?

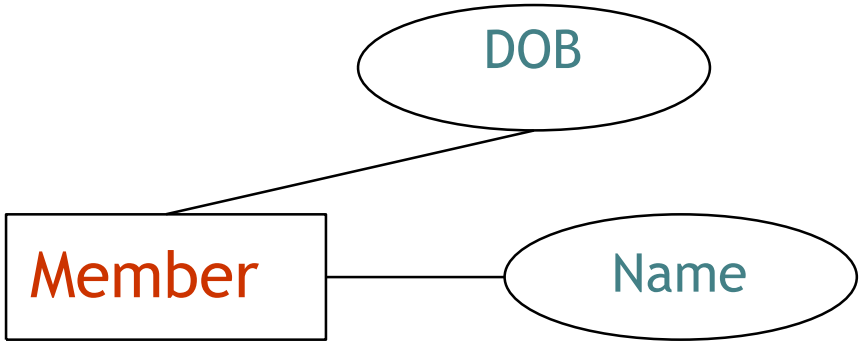
Keys

Definition: *A super key of an entity type is a set of one or more attributes whose values uniquely determine each entity.*

- Assume each Member has a unique ID. The following are super keys. {Name, DOB, ID}, {Name, ID}, {Name, DOB}, {DOB, ID}, {ID}
- Definition: A *candidate key* is a minimal super key. {Name, DOB} is a candidate key of Member. {ID} is a candidate key of Member.
- Definition: Generally, there are may be several candidate keys. One of the keys is selected to be the *primary key*.
- {ID} is the primary key of Member.

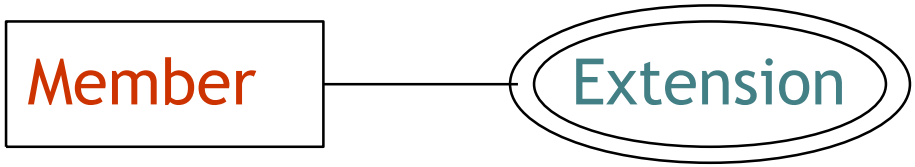
- An entity type is represented with a labelled rectangle.

- An attribute is represented with a labelled oval, connected to an entity type with a line.

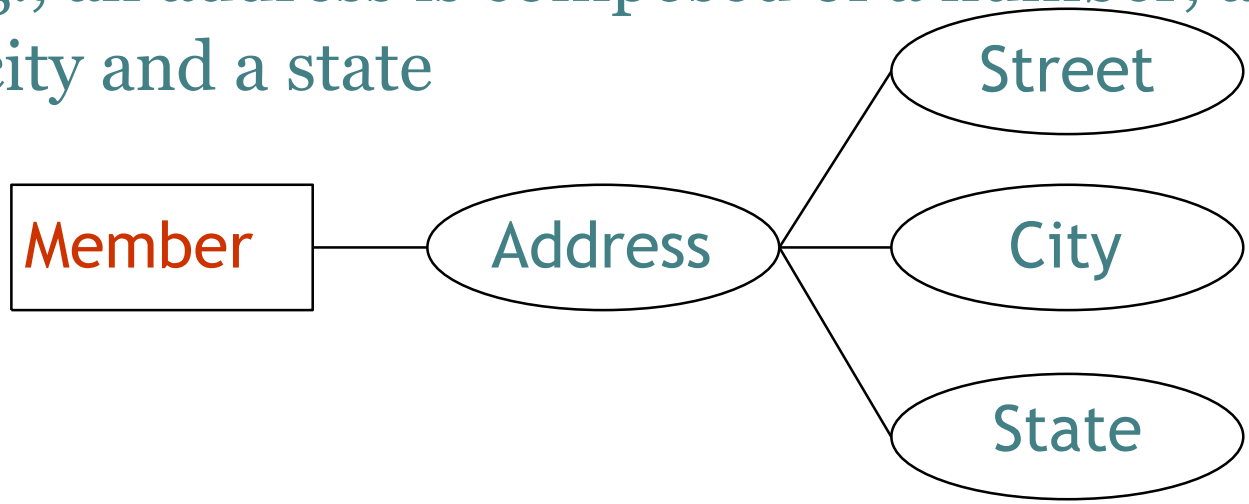


Attributes - Diamond Notation

- Single valued versus multi-valued
  - E.g., multiple telephone extensions

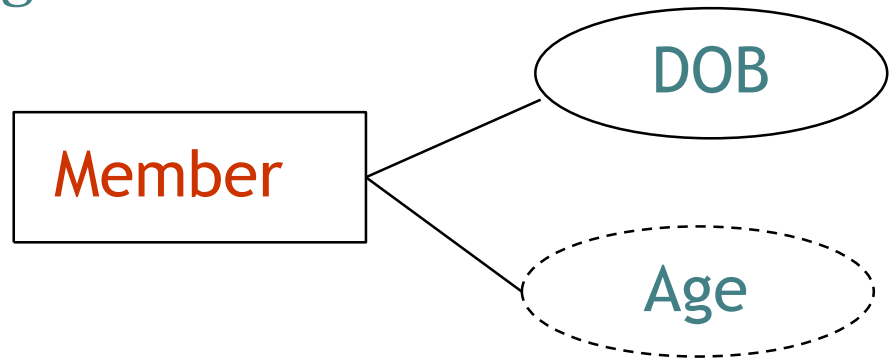


- Simple attributes versus structured attributes
  - E.g., an address is composed of a number, a street, a city and a state



Attributes, cont.

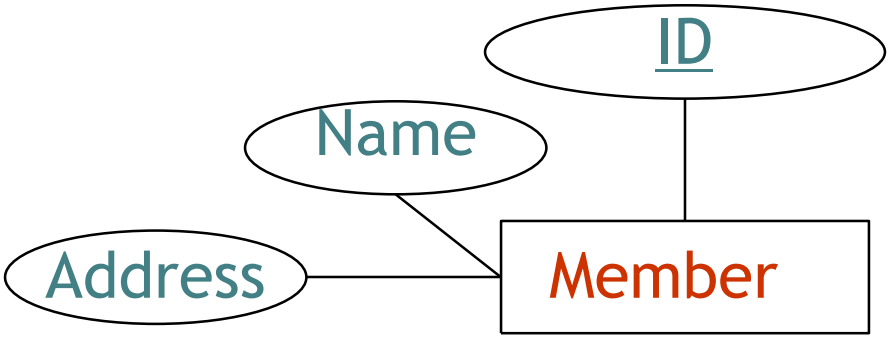
- Stored versus derived
  - E.g., an age can be derived from a stored birthdate



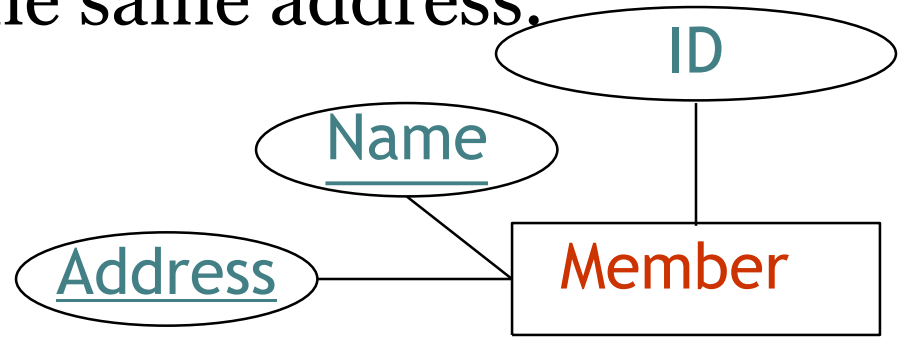
- Additional constraints - not represented in a diagram
  - Null versus non-null
  - Domain constrained
    - E.g., a student number must be exactly 9 decimal digits

Keys in ER Diagrams

- An attribute that is part of the (primary) key is underlined.
  - E.g., ID uniquely identifies customers

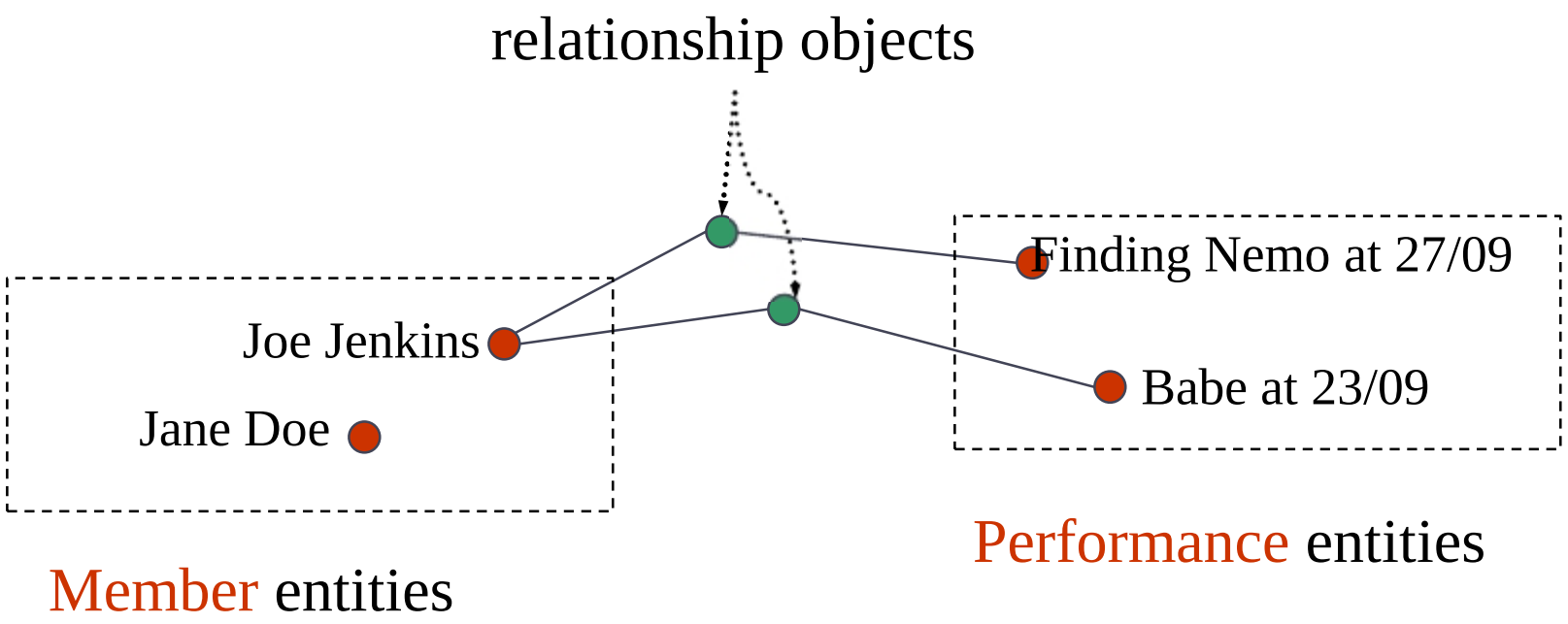


- Assume no two customers with the same name live at the same address.



Relationship

Definition: A *relationship* is an object that associates entities.



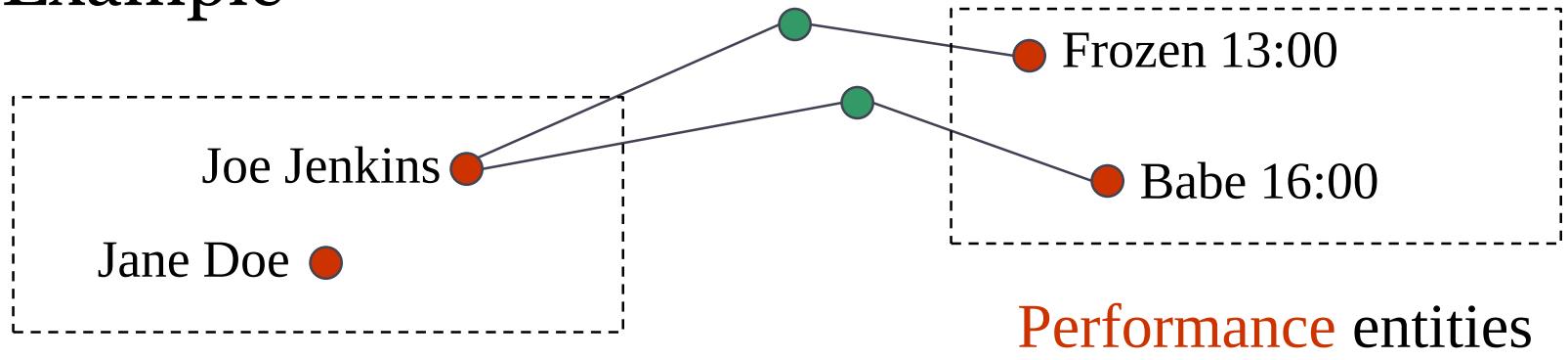
Relationship sets

Definition: A *relationship set* is a specific set of associated objects between entities of the same entity types.

$\{(e_1, e_2,\ldots,e_n) \mid e_1 \in E_1 \wedge e_2 \in E_2 \wedge \ldots. \wedge e_n \in E_n\}$

where the relationship is $(e_1, e_2,\ldots,e_n)$.

- Example

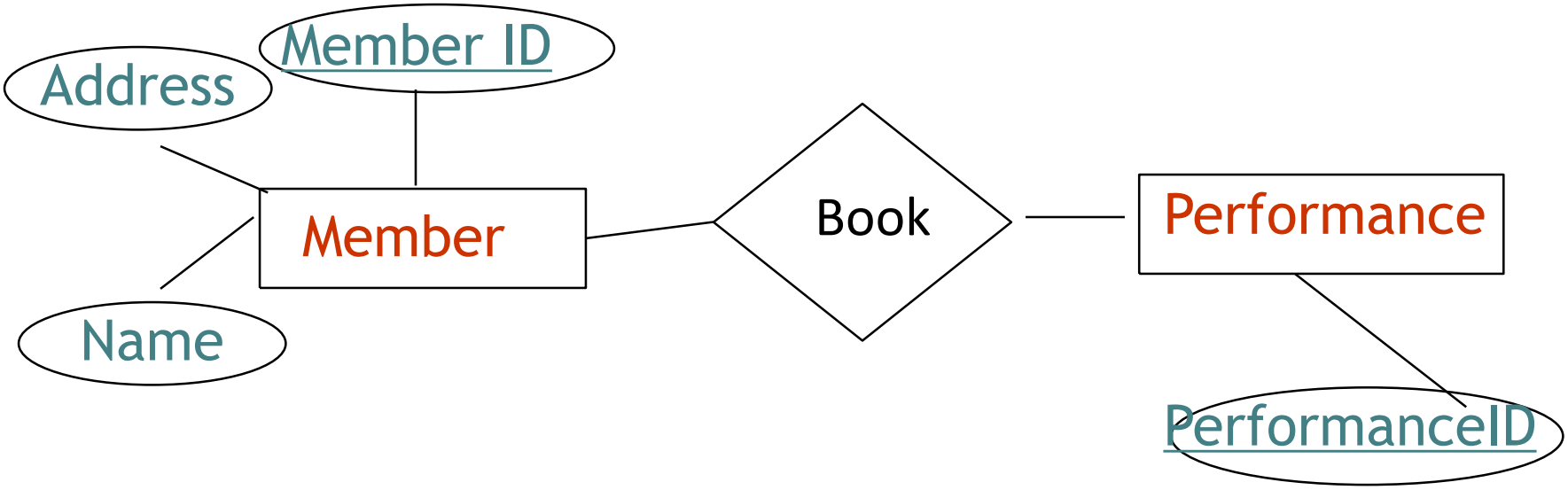


is the set {(Joe Jenkins, Frozen 13:00), (Joe Jenkins, Babe 16:00)}

Relationship types

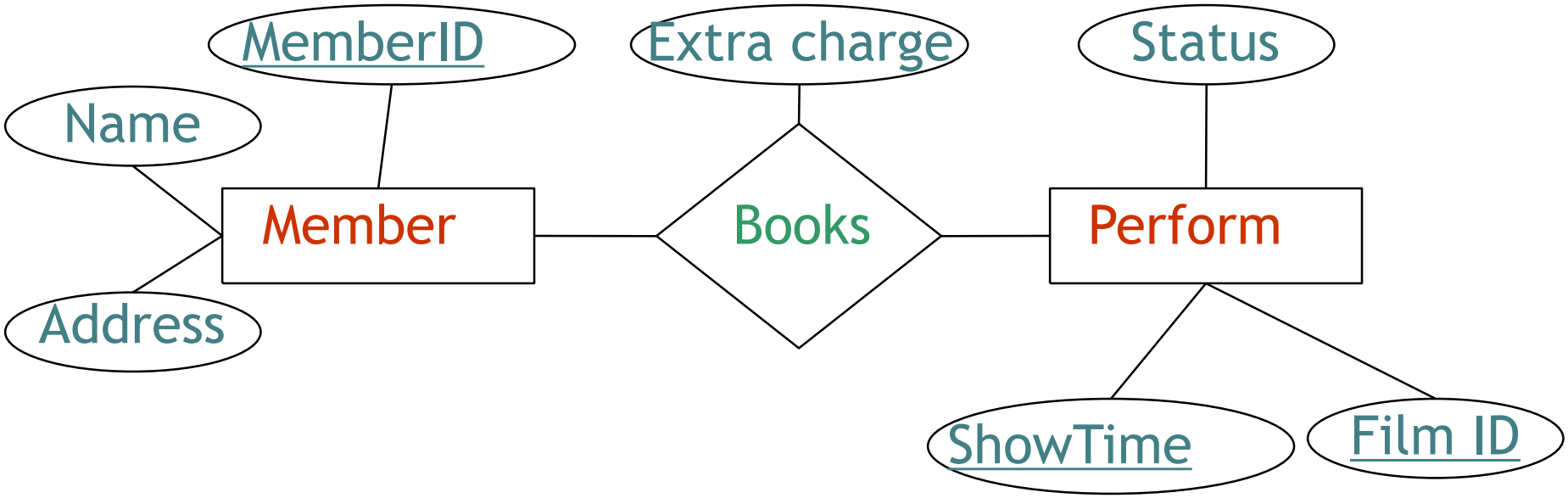
- Definition: A *relationship type* describes a relationship set.
- A relationship type is an ideal form (Plato), a kind, a type.
- Example: The Book relationship type relates the entity type Member with the entity type performance, capturing which member currently booked show.

- A relationship type is represented with a labelled diamond, and has lines to each of the entity types it relates.
- The relationship type could have attributes.



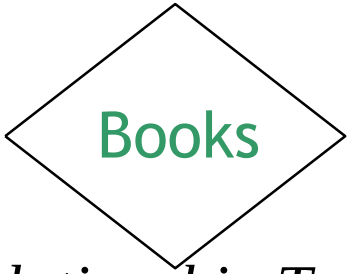
- A relationship type does not have *key* attribute(s) – the key comes from entity types that it relates.

- The entity-relationship diagram for the above example is:



Where:

Entity Type

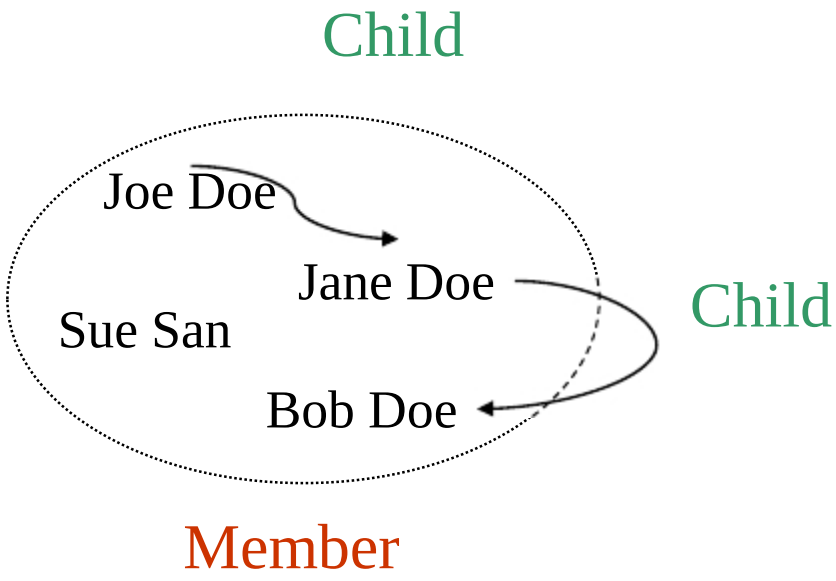


Relationship Type

Attribute

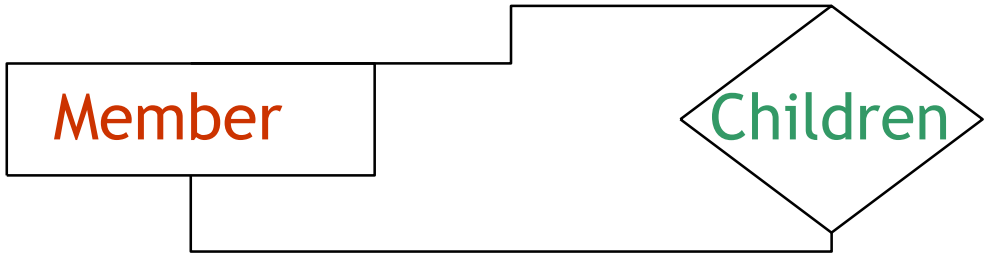
Reflexive Relationships

- An entity may be related to another entity of the same type.
  - Example: A Member could have Children, who are also Members. Joe Doe is Jane Doe‟s father. Bob Doe is Jane Doe‟s son.



Reflexive Relationship Types - Diagram

- Relationship type connects to self in an ER diagram.

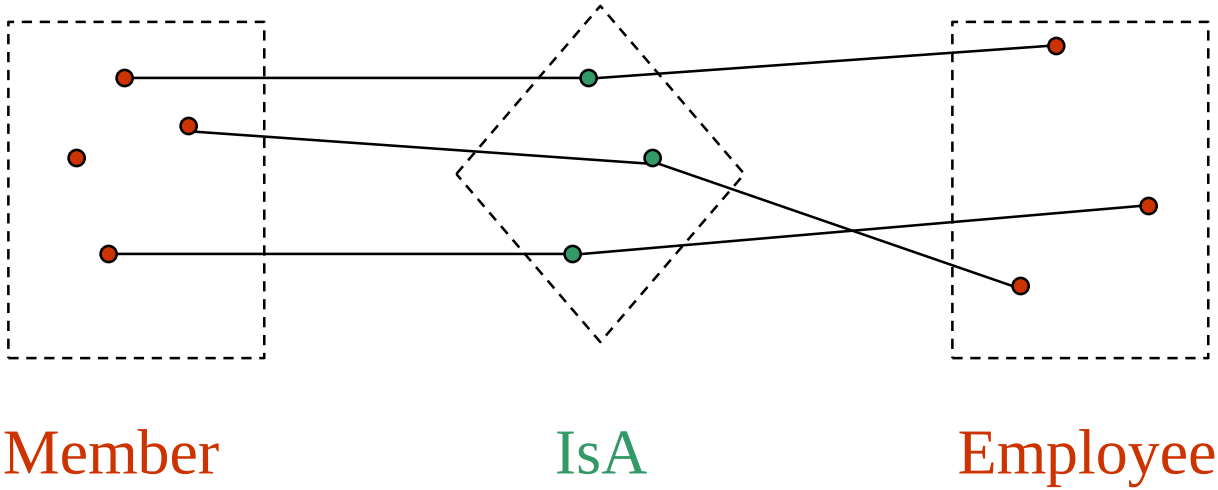


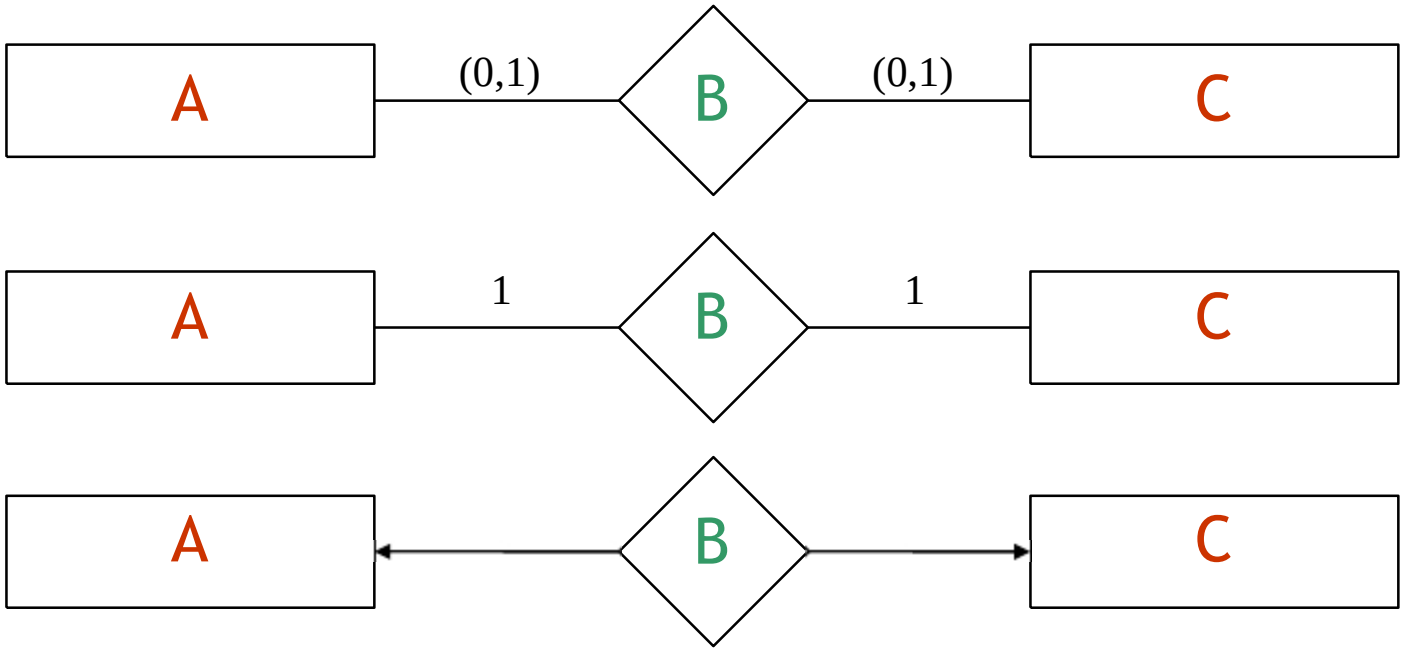
Kinds of Relationship Types

- How many of each entity type participate in a relationship type?
  - 1-1
  - 1-many
  - many-1
  - many-many
- Total vs. Partial participation
  - *Total* participation means that every entity participates in the relationship.
  - *Partial* means that some entities might not participate.
  - Bounds (min-max) on participation
- (At least) three different "diamond" notations
  - Arrow vs. straight line with or without participation bounds
  - DO NOT MIX NOTATIONS!

An Example One-to-One Relationship Type

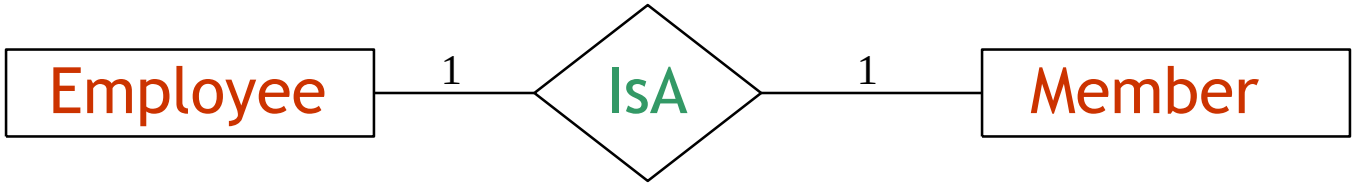
- A Member might be an Employee.
- An Employee IsA Member.
- Participation is *total* on the Employee side in this example.





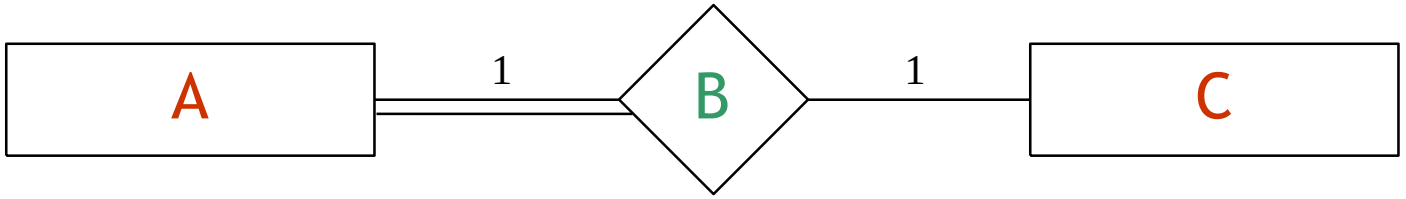
- An element in A is associated with at most one element in C via the relationship B. An element in C is associated with at most one element in A via B.

- An employee can also be a customer.

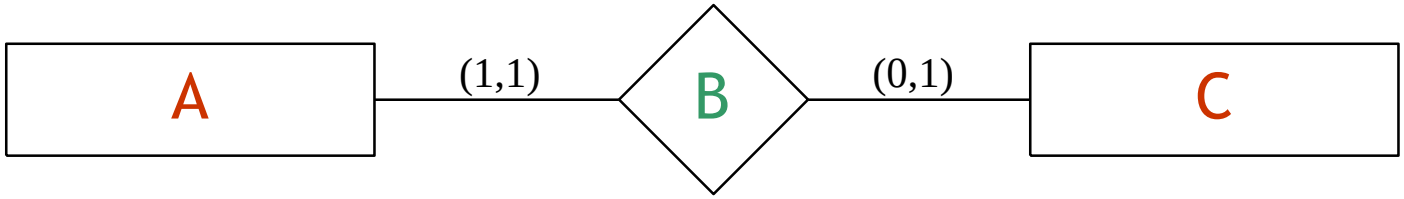


- Each Member entity can be matched with at most one Employee entity, and each Employee entity can be matched with at most one Member entity (both entities represent the same person).
- Often this can be better represented using additional attributes, or by using subclasses (wait a few slides!)

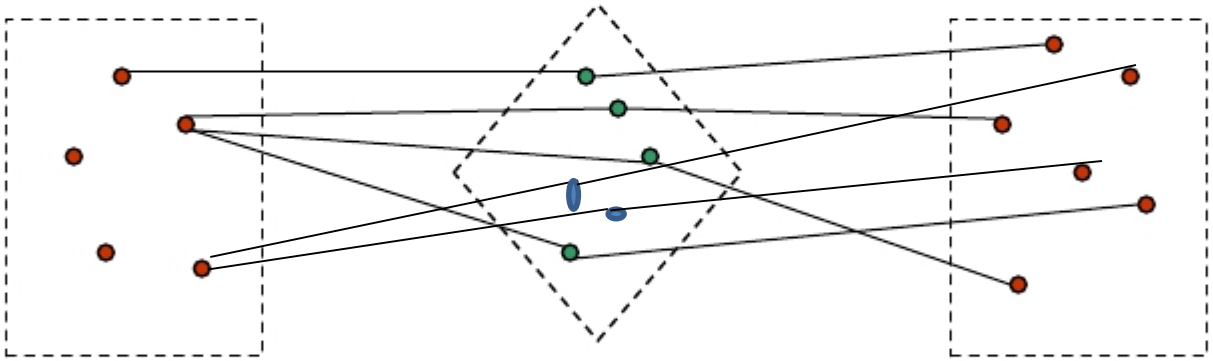
- Use a double line to indicate total participation.
- Example: Every A is in a B relationship with exactly one C, but some C‟s may be unrelated to an A.



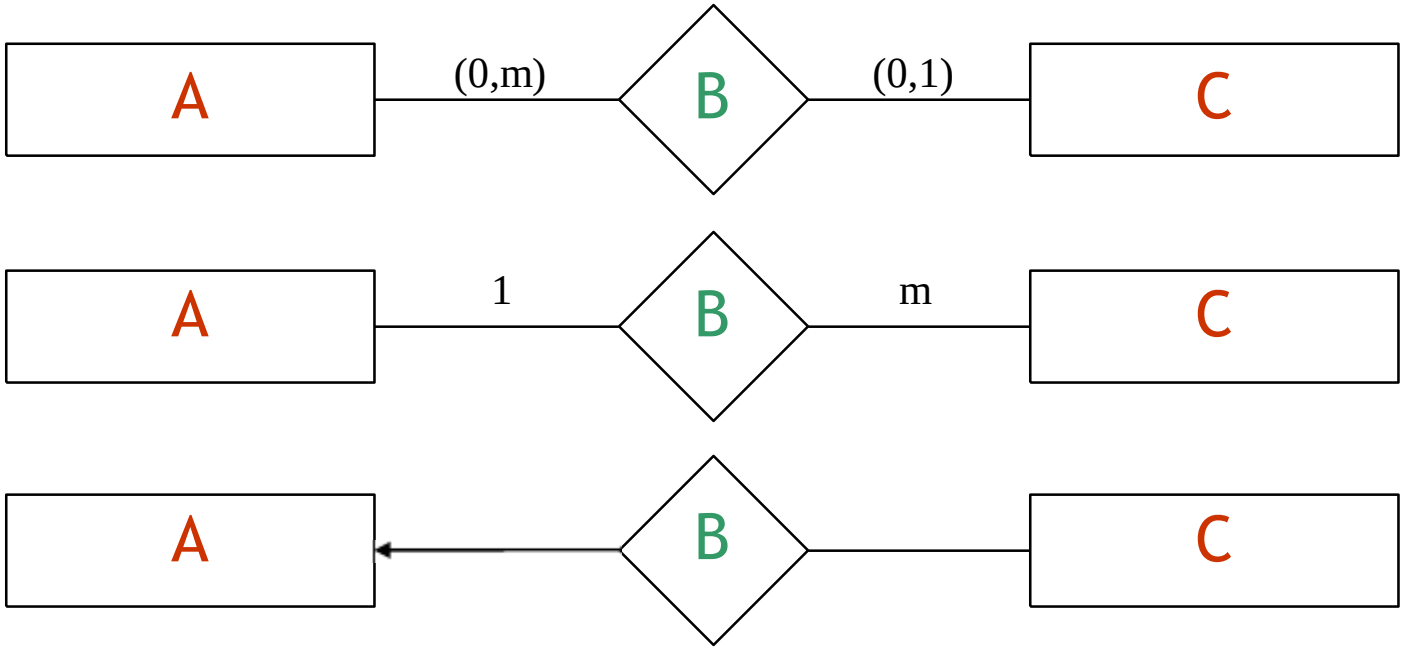
- Using participation constraints, total participation is a 1 on the minimum bound.



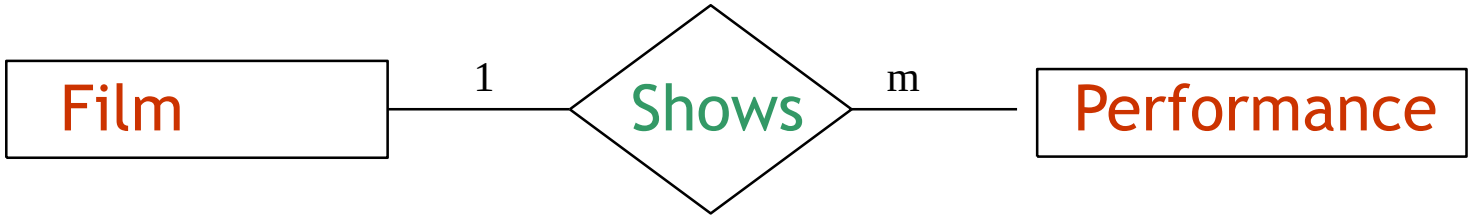
- A Film Shows zero to several Performance.
- A Performance can be showed by at one and most one Film.
- Participation is *partial* on one sides and *total on another side* in this example.



Film Shows Performance



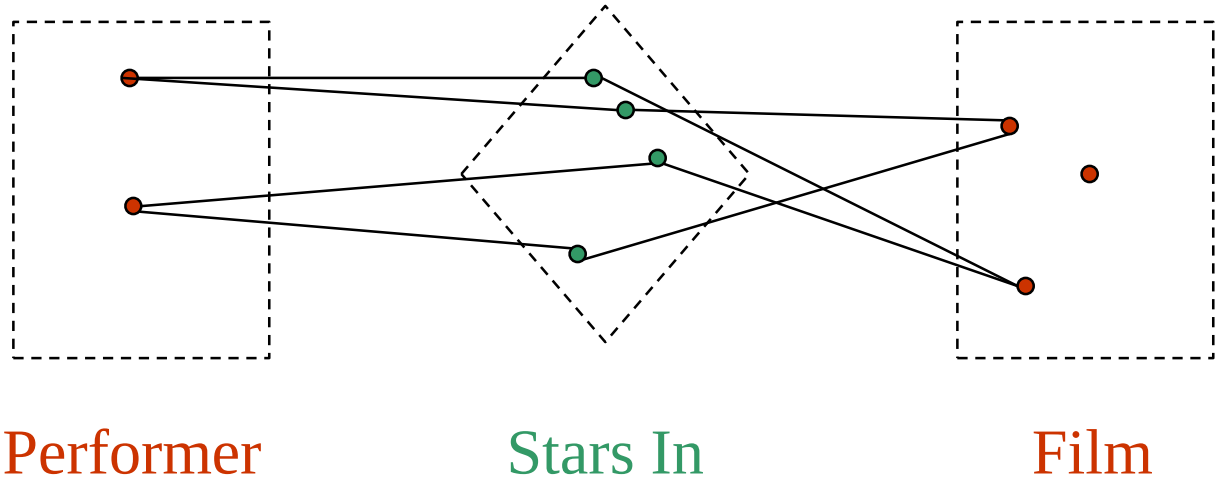
- An element in A is associated with several (including 0) elements in C via B. An element in C is associated with at most one element in A via B.



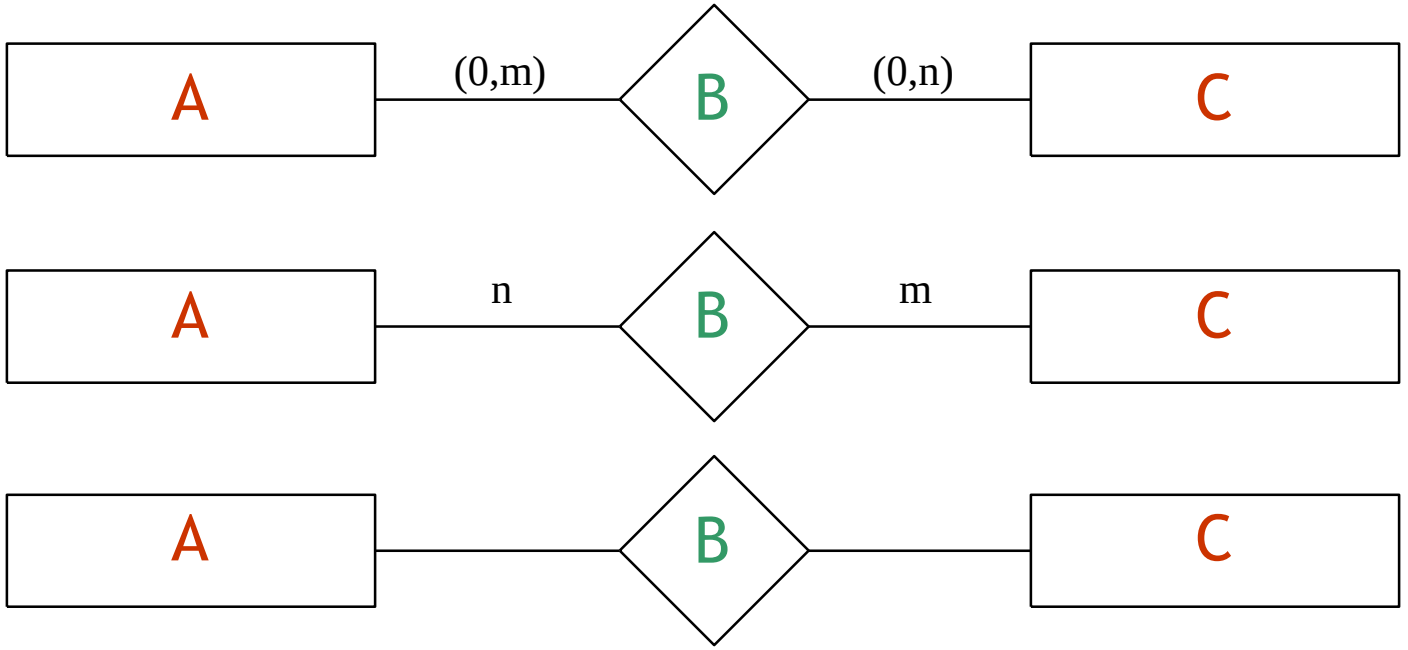
- A Film might not be related to a Performance entity.
- Each Performance entity is associated with at most one Film entity.

An Example Many-to-Many Rel. Type

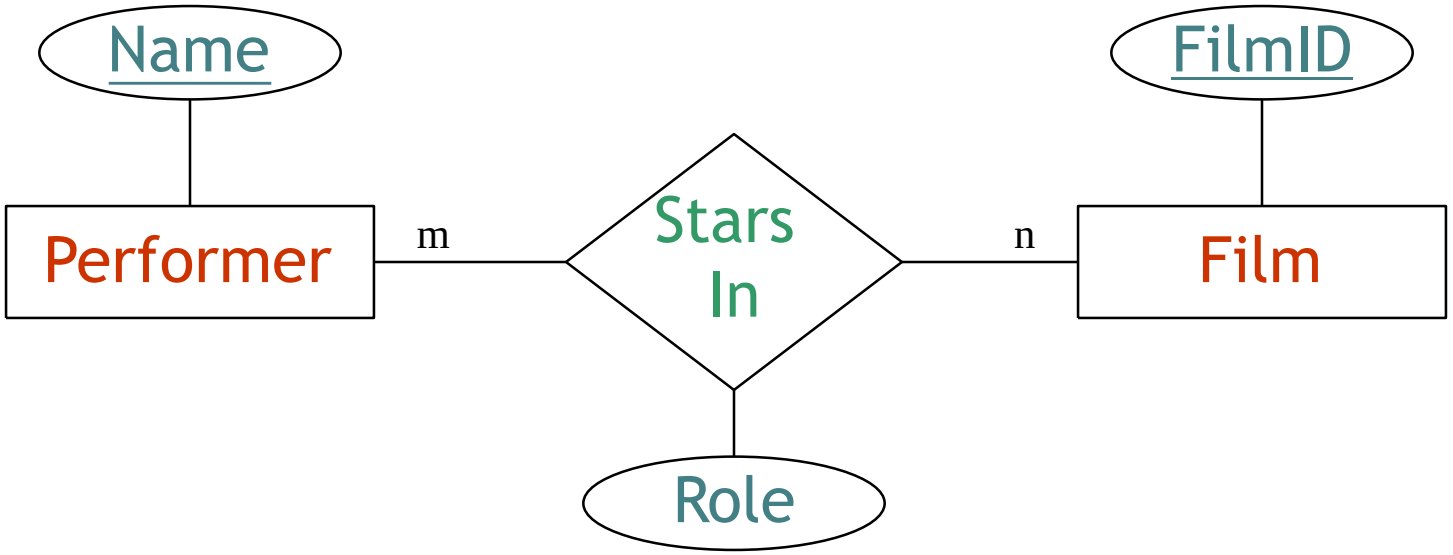
- A Performer Stars In one or many Films.
- A Film can Star zero to many Performers.
- Participation is *total* on the Performer in this example.



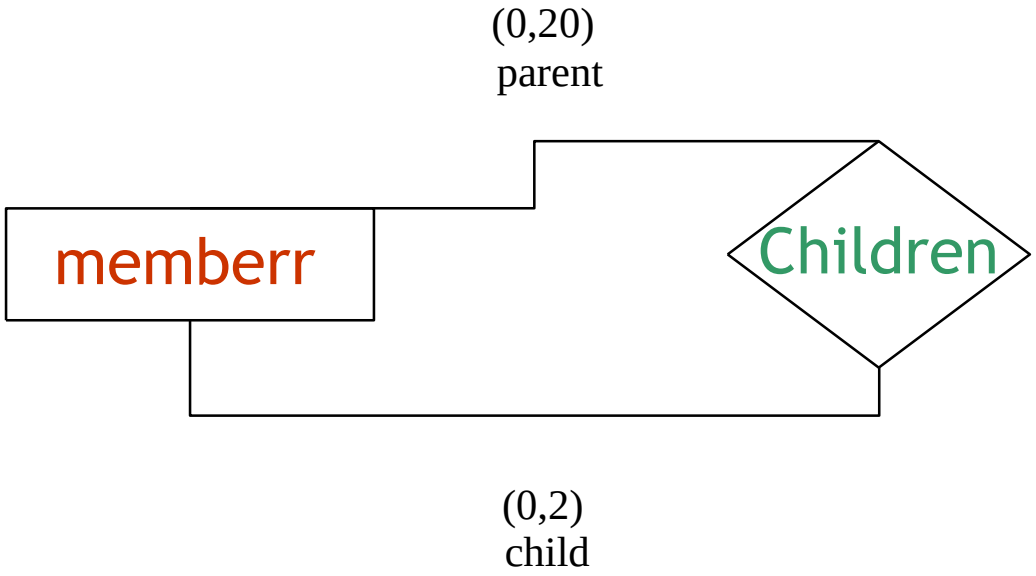
Many-To-Many Relationship



- An element in A is associated with several elements in C via B.
- An element in C is associated with several elements in A via B.


Name
FilmID
Performer
m
Stars
In
n
Film
Role

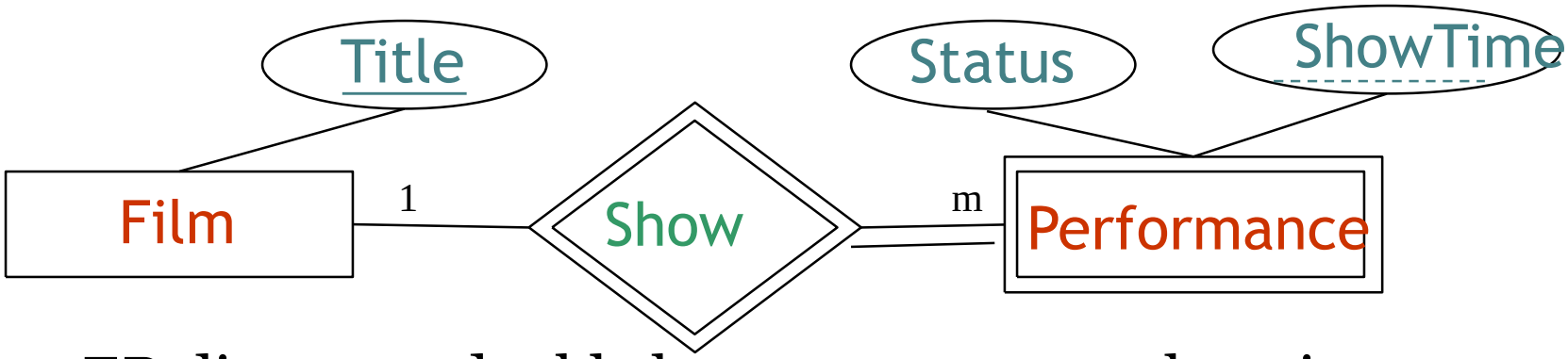
- Definition: A *role* is a label on a relationship type edge.
- Example: labels "parent" and "child" are roles.



- Roles are optional, and are used to clarify semantics of a relationship type.

Weak Entity Types

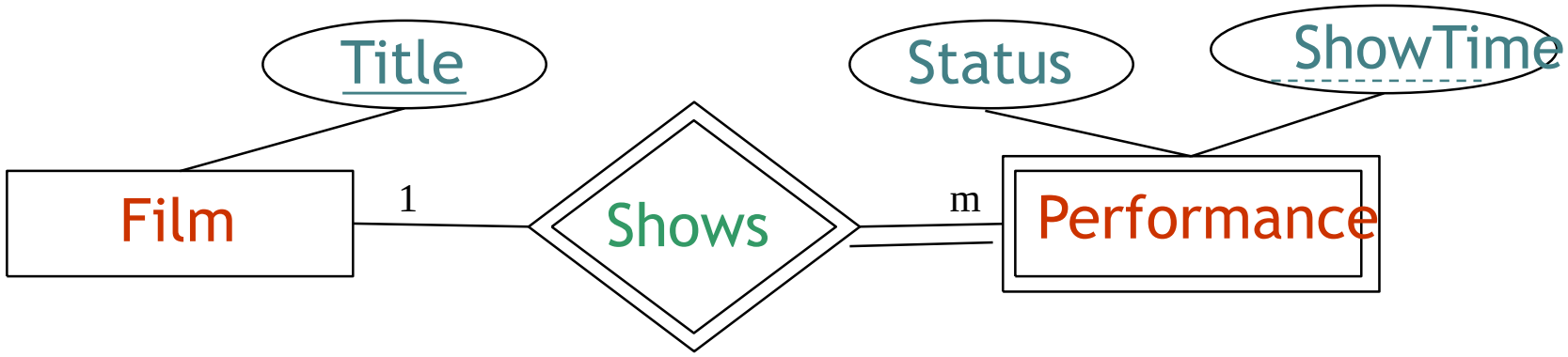
- Definition: A *weak entity type* borrows key attributes from another entity type (called the *owning* or *strong* entity type) to uniquely identify entities.
  - Example: A Performance has a ShowTime, relative to a Film title (e.g., `Babe 13:00", `Babe 19:00", `Finding Nemo 13:00". ShowTime is not a key, but combined with Title it is (for Performance).



- ER diagram - double box represents weak entity type.
- The existence of a Performance entity depends on the existence of a Film entity.

Weak Entity Type – Partial Keys

- A weak entity type has a *partial key* (key is completed by borrowing key attributes from owning entity type(s)).
  - Example: Key for Performance is (Title, ShowTime).



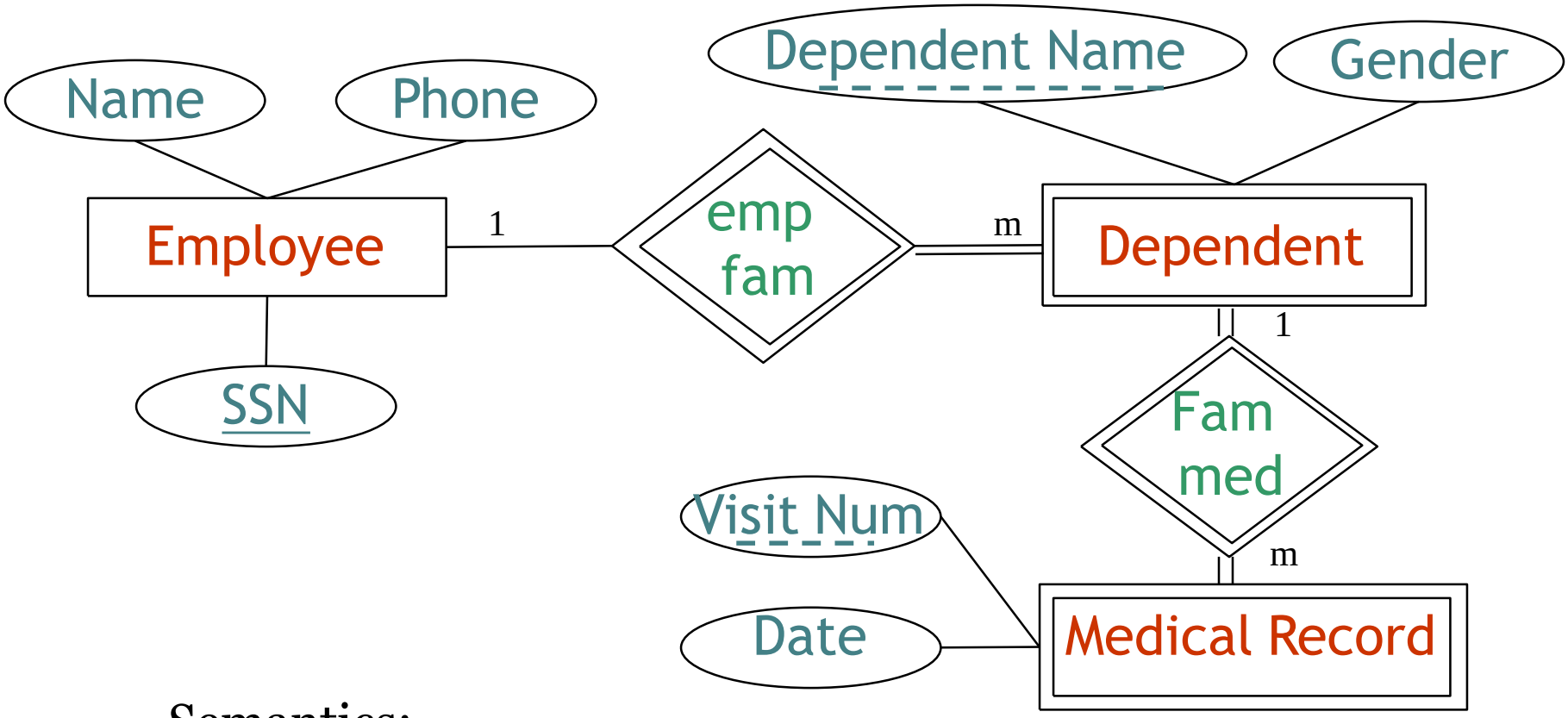
- ER diagram - partial key represented with dashed line.

Weak Entity Type, cont.

- Semantics:
  - Deletion of a Film entity requires deletion of that film's Performance entities.
- A weak entity is related to precisely one entity in the owning entity type, via a 1-1 or 1-many relationship.
- It is possible to introduce more attributes to the Performance entity type, so that a primary key will exist, but they may not be needed for database processing.

Cascaded Weak Entity Types

- Weak entity types can be cascaded:



- Semantics:
  - To delete an employee, the family's dependents and their medical records must also be deleted.

A Film Club ER Schema