CPSC 304
Introduction to Database Systems

Conceptual Database Design
The Entity-Relationship Model

Textbook Reference
Database Management Systems: Chapter 2

Databases: the continuing saga…

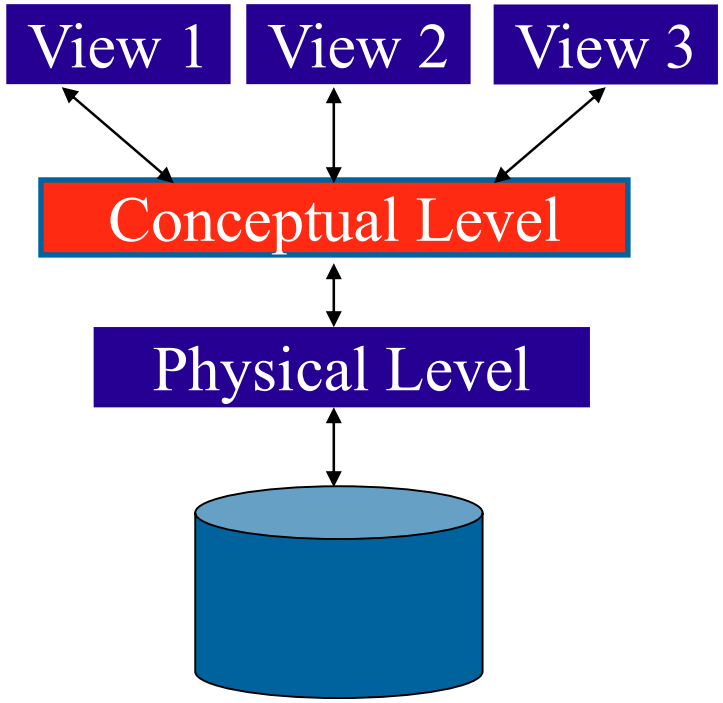
- We motivated that databases were great because they:
  - Store large amounts of data
  - Handle transactions
  - Allow efficient querying
  - *And many, many more classic favourites!*
- Before we can do all of these, we must design the database

Learning Goals

- Explain the purpose of an ER diagram, and list the major components.
- Given a problem description, create an ER diagram given a specification. Justify the decisions you make for entities, relationships, keys, key constraints, participation constraints, weak entities, is-a relationships, and aggregations.
- Given a problem description, identify alternative representations of the problem concepts and evaluate the choices
- Compare alternative ER models for the same domain and identify their strengths and weaknesses

Levels of Abstraction

- A major purpose of a DB system is to provide an abstract view of the data.
- Three abstraction levels:
  - **External (or View) level**: describes different part of the database to different users
    - convenience, security, etc.
    - Compare views of student, registrar, and database admin.
  - **Conceptual (or Logical) level**: how data is perceived by the users
  - **Physical level**: how data is actually stored (404) – covers things like indexes, bits on disk



Schema and Instances

- We create the conceptual **schema** – the logical structure of the database (e.g., students take courses)
- Later we'll populate **instances** – the content of the database at a particular point in time

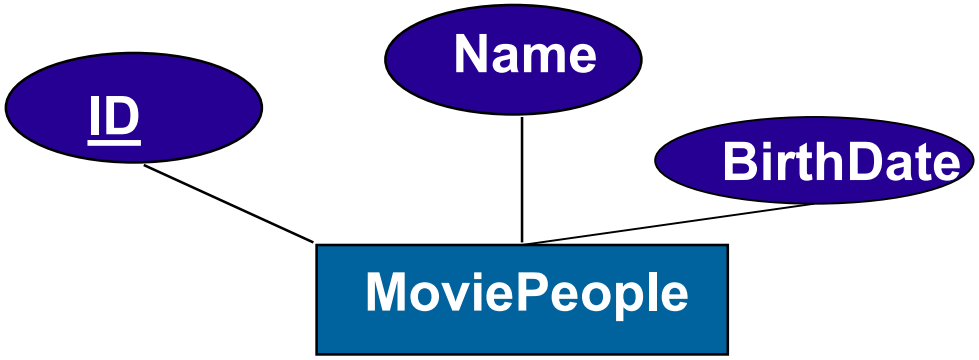
- E.g., Gradebook schema is set, but currently there are no grades for CPSC 304

Conceptual Database Design

- What are the entities and relationships in the enterprise?
  - Entities are usually nouns
    - e.g., Students and Courses
  - Relationships are statements about 2 or more objects. Often, verbs.
    - e.g., an instructor Teaches a course
- What information about these entities and relationships should we store in the database?
- What integrity constraints or other rules hold?
- In relational databases, this is generally encoded in an **Entity-Relationship (ER) Diagram**

ER Model Basics: Entities

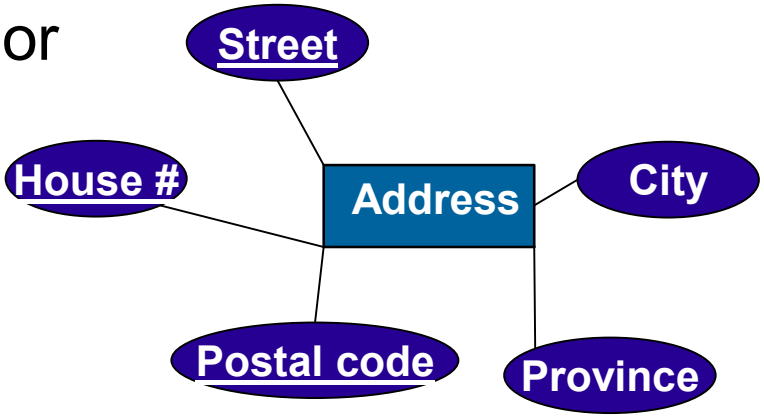
- ***Entity:*** Real-world object distinguishable from other objects.
- An entity is described using a set of ***attributes***.



- ***Entity Set:*** A collection of similar entities. E.g., all Movie People.
  - All entities in an entity set have the same set of attributes. (Until we consider ISA hierarchies, anyway!)
  - Each attribute has a ***domain***. *(e.g.., float, date, int)*
  - Each entity set has a ***key***. The key is composed of all underlined attributes

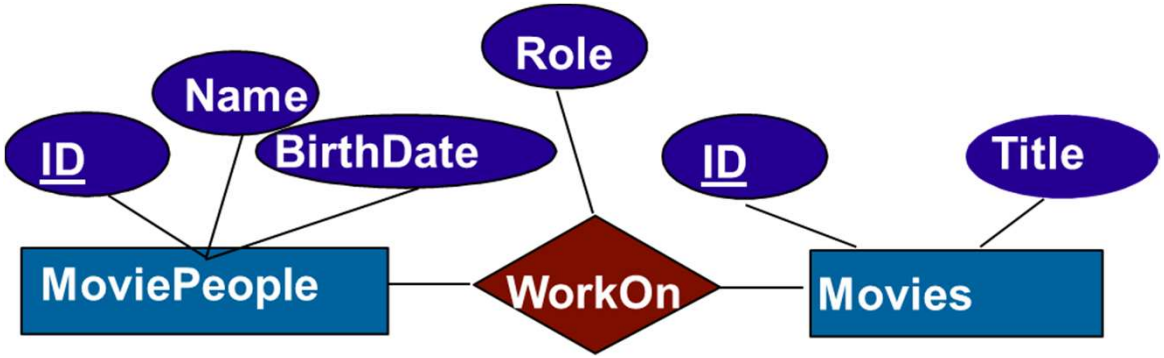
Keys

- Distinguish entities
- A *key* is the **minimal** set of one or more attributes which, taken collectively, identify uniquely an entity in an entity set.
  - In Canada, ~50 addresses share the same postal code



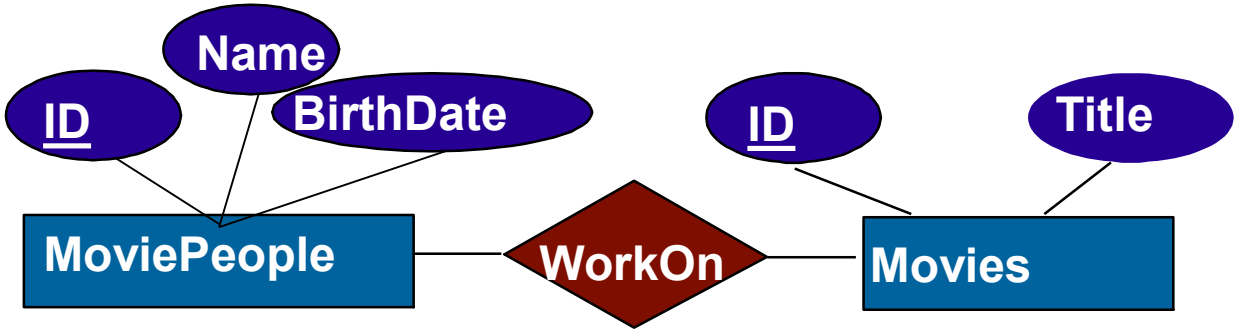
- A *primary key* is the key chosen as the principal means to identify entities in an entity set
- The only keys shown in ER diagrams are primary keys (do not worry about this for now)
- We'll discuss superkeys when we consider normal forms (for now, don't worry about them)

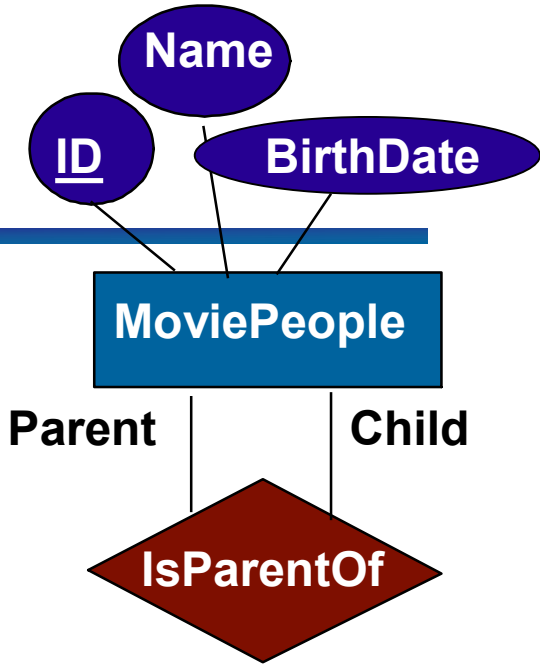
ER Model Basics (Cont.)



- ***Relationship:*** Association among two or more entities.
  - E.g., Michelle Yeoh worked on Everything Everywhere All at Once.

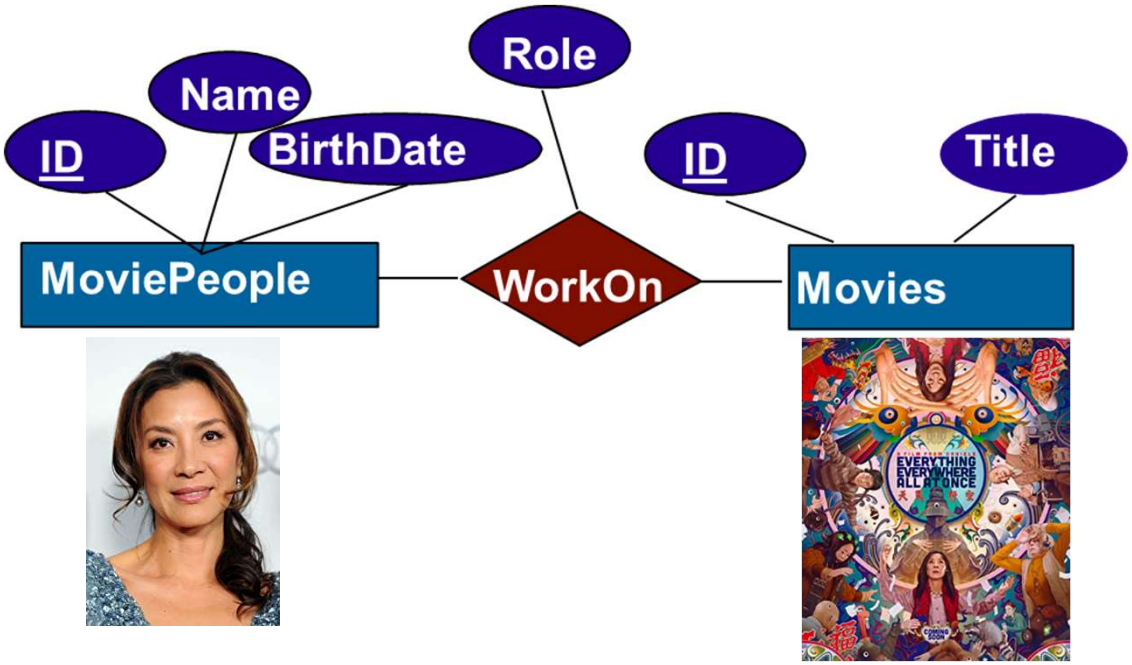
ER Model Basics (Cont.)





- ***Relationship Set:*** Collection of similar relationships.
  - Collection of all MoviePeople that have worked in Movies.
- Same entity set could participate in different relationship sets, or in different "roles" in same set. (Kirk Douglas isParentOf Michael Douglas)

ER Model Basics (Cont.)



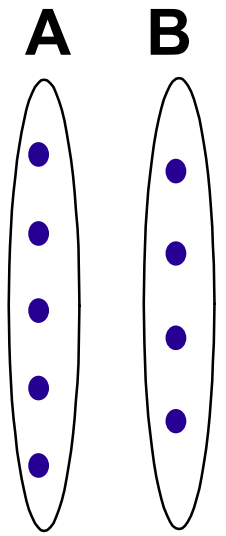
- A relationship must be uniquely defined by the entities involved (it may not be a key because it may not be minimal, as we'll see shortly)
- A relationship set may have ***descriptive attributes*** (like Role).
- An n-ary relationship set R relates n entity sets E_1 ... E_n; each relationship in R involves entities $e_1 \in E_1$, ..., $e_n \in E_n$
  - ***Degree*** or ***arity:*** # of entity sets in relationship (binary, ternary, etc.)

Cardinalities

- A **cardinality ratio** for a relationship set specifies the number of relationships in the set that an entity can participate in.

Let R be a relationship set between sets A and B. R can have 1 of 4 cardinalities:

1. ***one-to-one*** from A to B:
   - an entity in A is associated with **at most one** entity in B and vice versa
   - e.g. A: student, B: student ID #



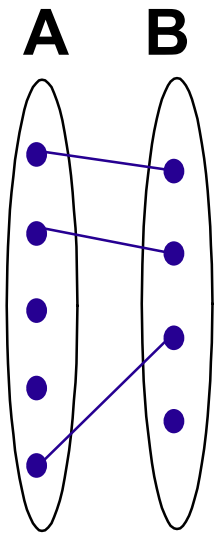
Entity sets A and B

Cardinalities

- A **cardinality ratio** for a relationship set specifies the number of relationships in the set that an entity can participate in.

Let R be a relationship set between sets A and B. R can have 1 of 4 cardinalities:

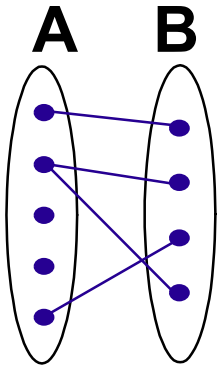
1. ***one-to-one*** from A to B:
   - an entity in A is associated with **at most one** entity in B and vice versa
   - e.g. A: student, B: student ID #



1-to-1

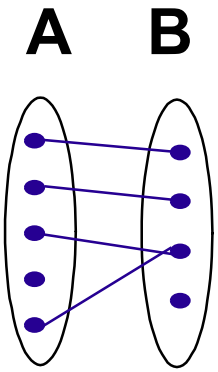
Cardinalities (cont')

2. ***one-to-many*** *from A to B*:
   - an entity in A is associated with any number of entities in B
   - an entity in B is associated with at most one entity in A
   - e.g. A: biological-mother, B: children



1-to Many

3. ***many-to-one*** *from A to B: (switch A and B above)*
   - an entity in B is associated with any number of entities in A
   - an entity in A is associated with at most one entity in B

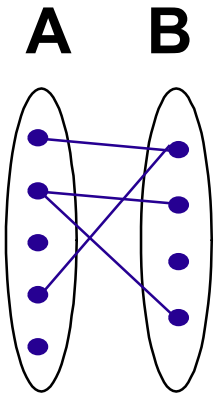


Many-to-1

Cardinalities (cont')

*4. **many-to-many** from A to B:*

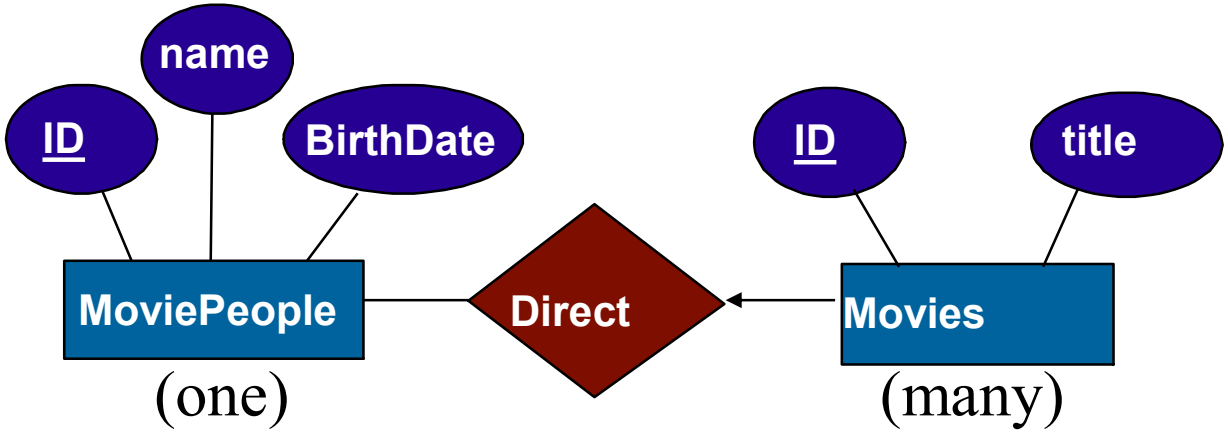
- *an entity in A is associated with any number of entities in B and vice versa*
- *e.g. A: students, B: courses*

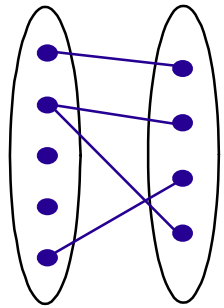


Many-to-Many

Key Constraints

- The restriction imposed by a 1-to-1 and 1-to-many ratios are examples of ***key constraints***.
- A key constraint is shown with an arrow in the ER diagram.
- Important on insertions



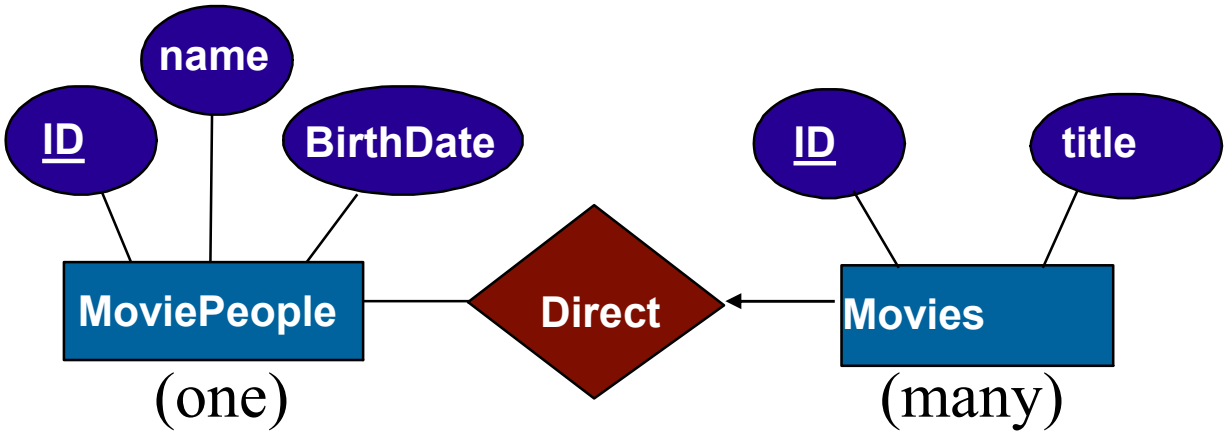


1-to Many

or

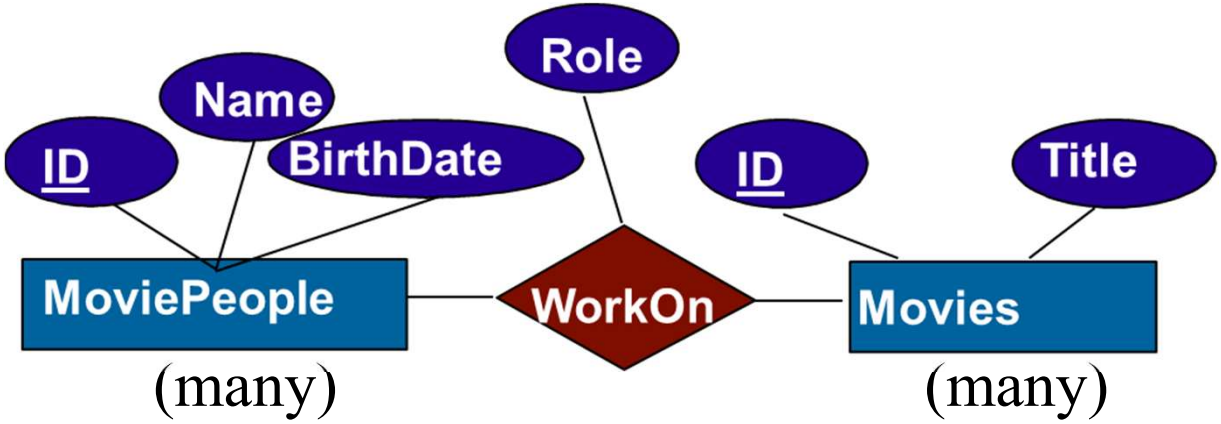
A brief digression on notation

The ER notation we use can be read: “if you know the entity with the arrow, then you know the relationship (and the other entities involved)” – the arrow points to the thing there is only one of



James Cameron

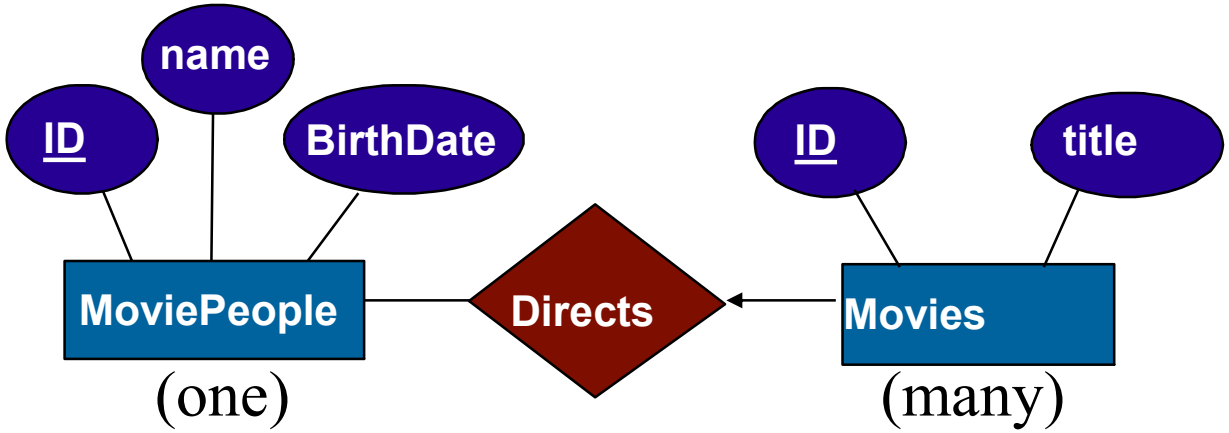
How can we uniquely identify a relationship?



How can we identify the role of a specific MoviePerson in a specific movie

Robert Pattinson
as
Bruce Wayne

How can we uniquely identify a relationship?



James Cameron

The key of a many to one relationship is the key of the entity on the many side.

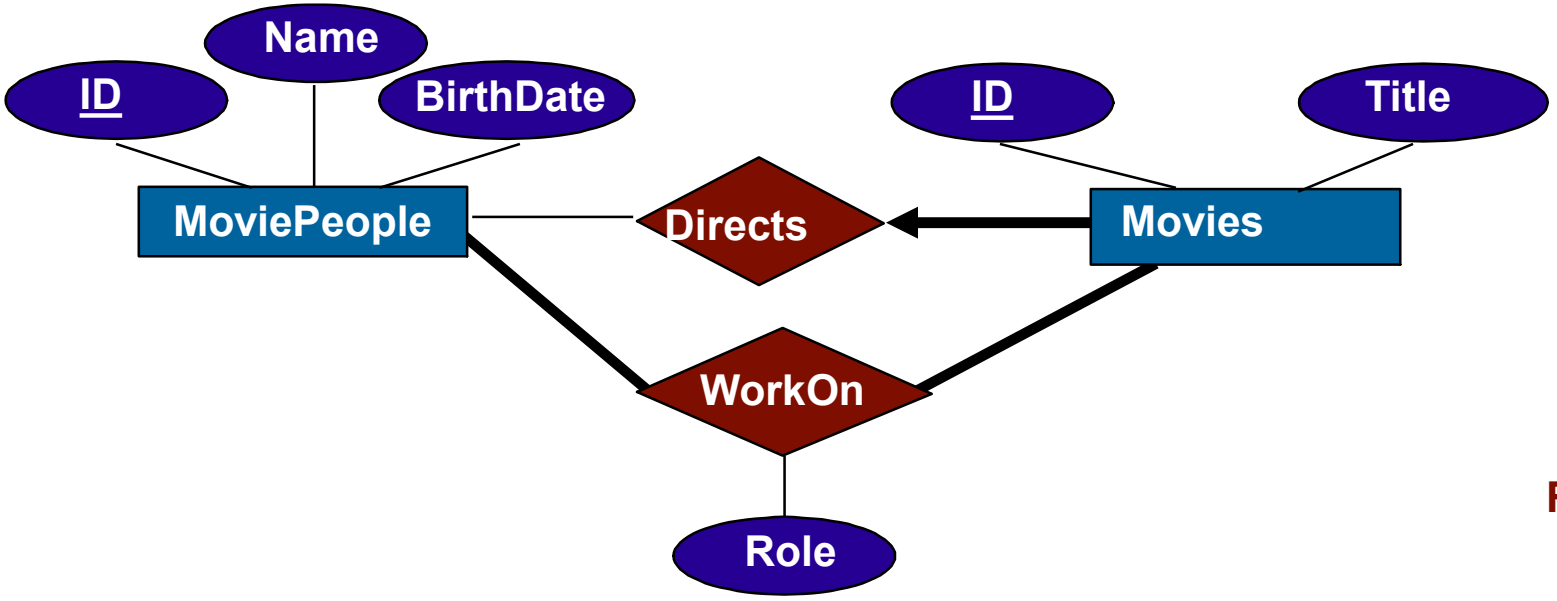
How can we uniquely identify a relationship?

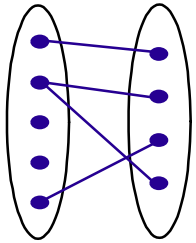
Ottawa

The key of a one-to-one relationship is the key of ONE of the entities.

Participation Constraints

- Participation : Indicates if all entities participate in the relationship.
- An entity's participation can be ***total*** or ***partial***.
- Requiring total participation is a ***participation constraint*** and it is shown with a thick line
  - Important on deletions
  - i.e., participation of Movie in Directs is total (thick line)
    - Every movie must appear in some relationship in the Directs set



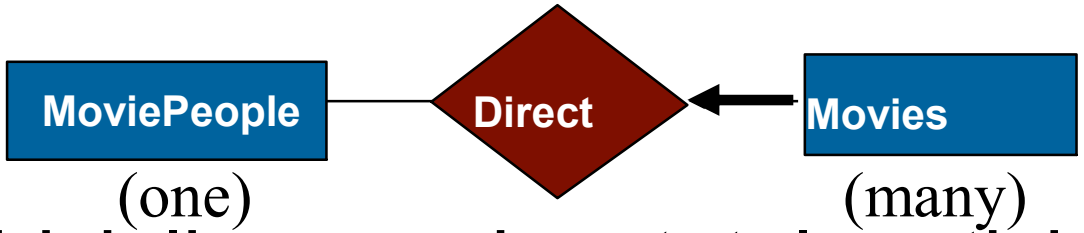


Remember?

Line types summarized

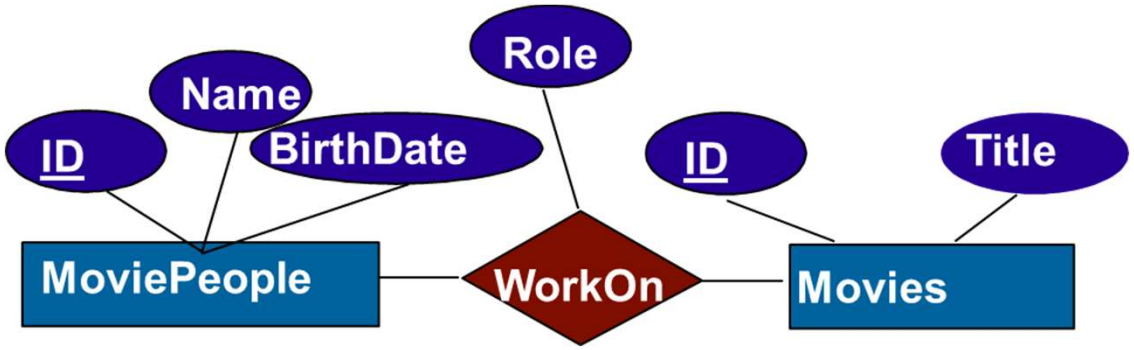
- Plain lines mean many to many:

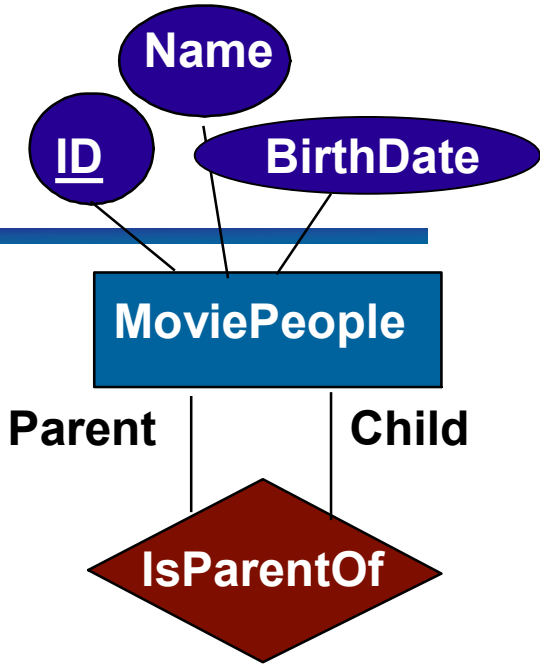
- Arrows mean the other side has a cardinality of one



- A thick line requires total participation and can be added to any line, arrow or not

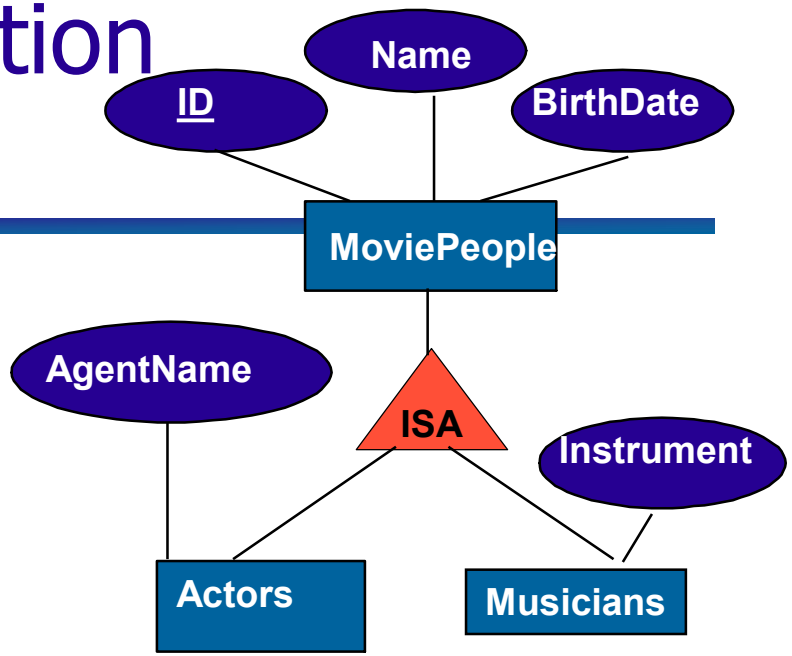
ER Model Basics (Cont.)





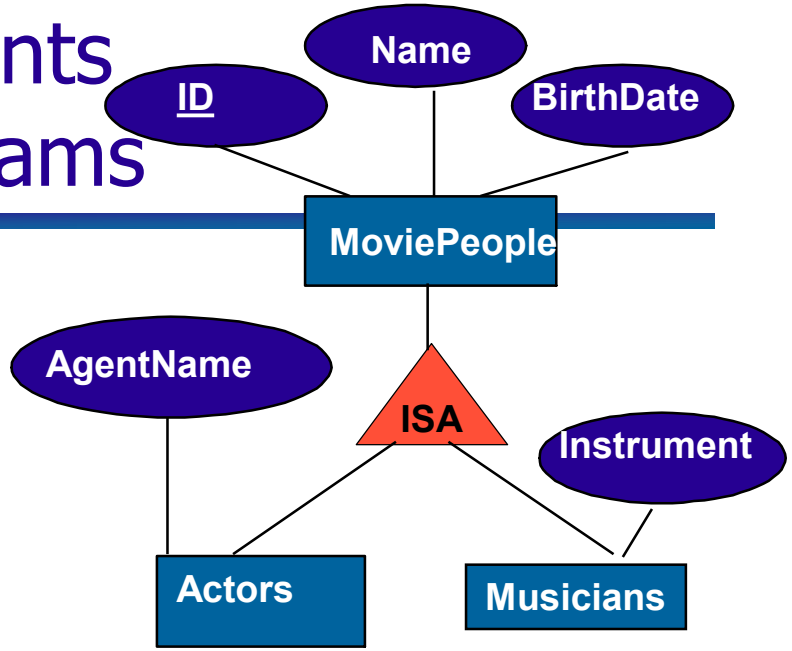
- ***Relationship Set:*** Collection of similar relationships.
  - Collection of all moviePeople that have worked in Movies.
- Same entity set could participate in different relationship sets, or in different "roles" in same set ("Parent"/"Child")
- **A relationship must be uniquely defined by the entities involved** (it may not be a key because it may not be minimal, as we'll see shortly)
- An n-ary relationship set R relates n entity sets E_1 ... E_n; each relationship in R involves entities $e_1 \in E_1$, ..., $e_n \in E_n$
  - ***Degree*** or ***arity:*** # of entity sets in relationship (binary, ternary, etc.)

Generalization/Specialization (ISA relationships)



- As in Java, or other PLs, attributes can be inherited.
- If we declare A ***ISA*** B, every A entity is a B entity.
- Reasons for using ISA:
  - To add descriptive attributes specific to a subclass.
  - To restrict entities that participate in a relationship.

There are some ISA constraints we can't express in ER diagrams

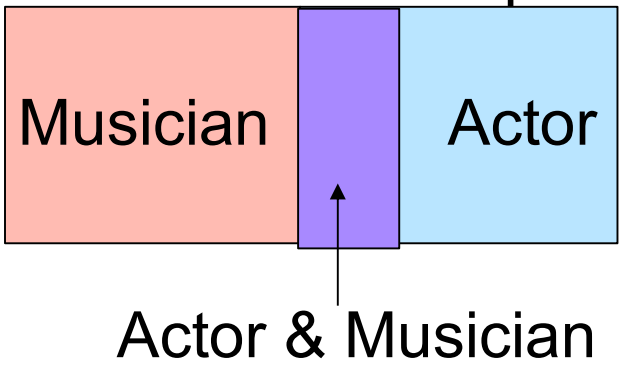


- ***Overlap constraints*** :
  Specializations can be:
  - ***Disjoint :*** a superclass entity belongs to no more than a single subclass
  - ***Overlapping :*** subclasses may overlap
- ***Covering constraints*** : Specializations can be:
  - ***Total :*** a superclass entity must belong to some subclass
  - ***Partial :*** some superclass entity may not be in any subclass

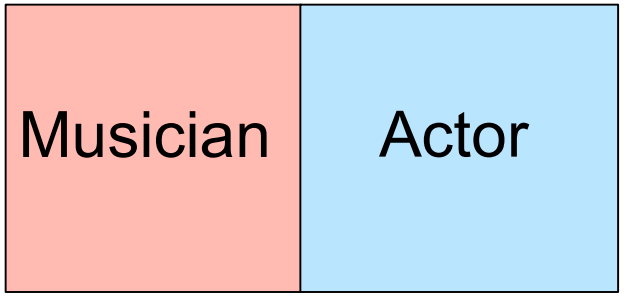
We can represent these constraints by just writing them in.

ISA constraints Illustrated

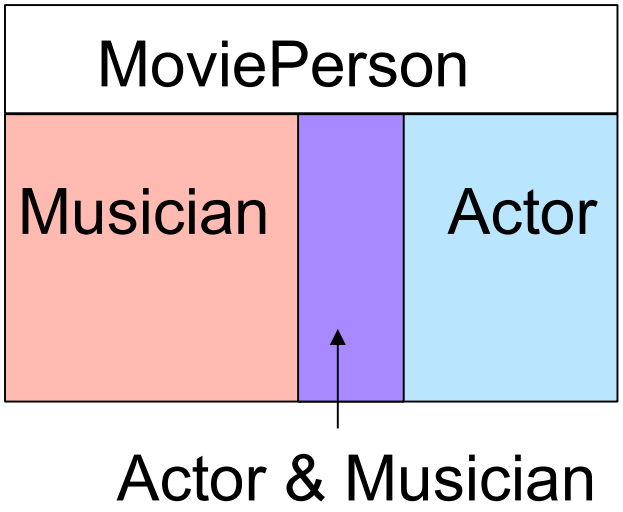
Total + Overlap



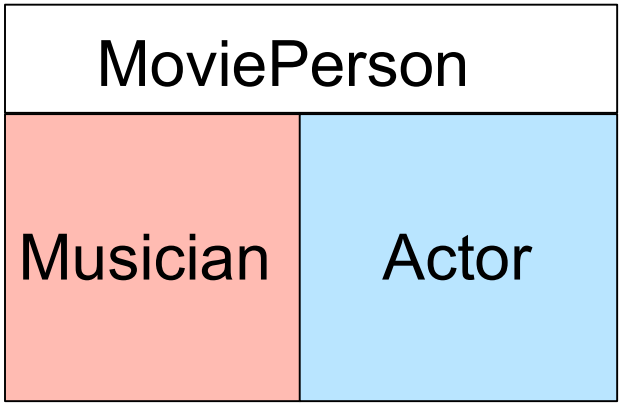
Total + Disjoint



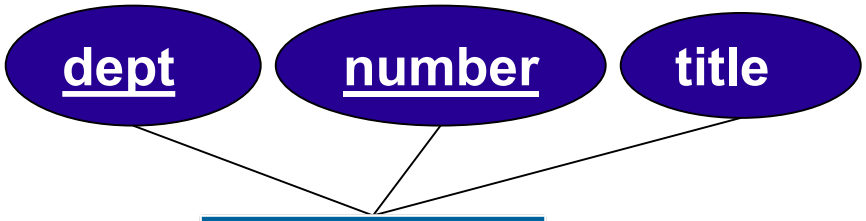
Partial + Overlap



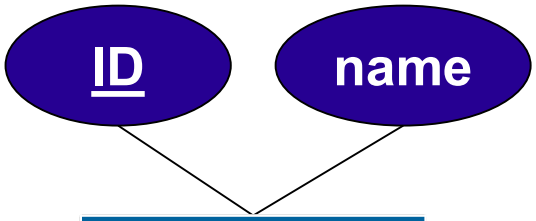
Partial + Disjoint



Review – Entity Sets

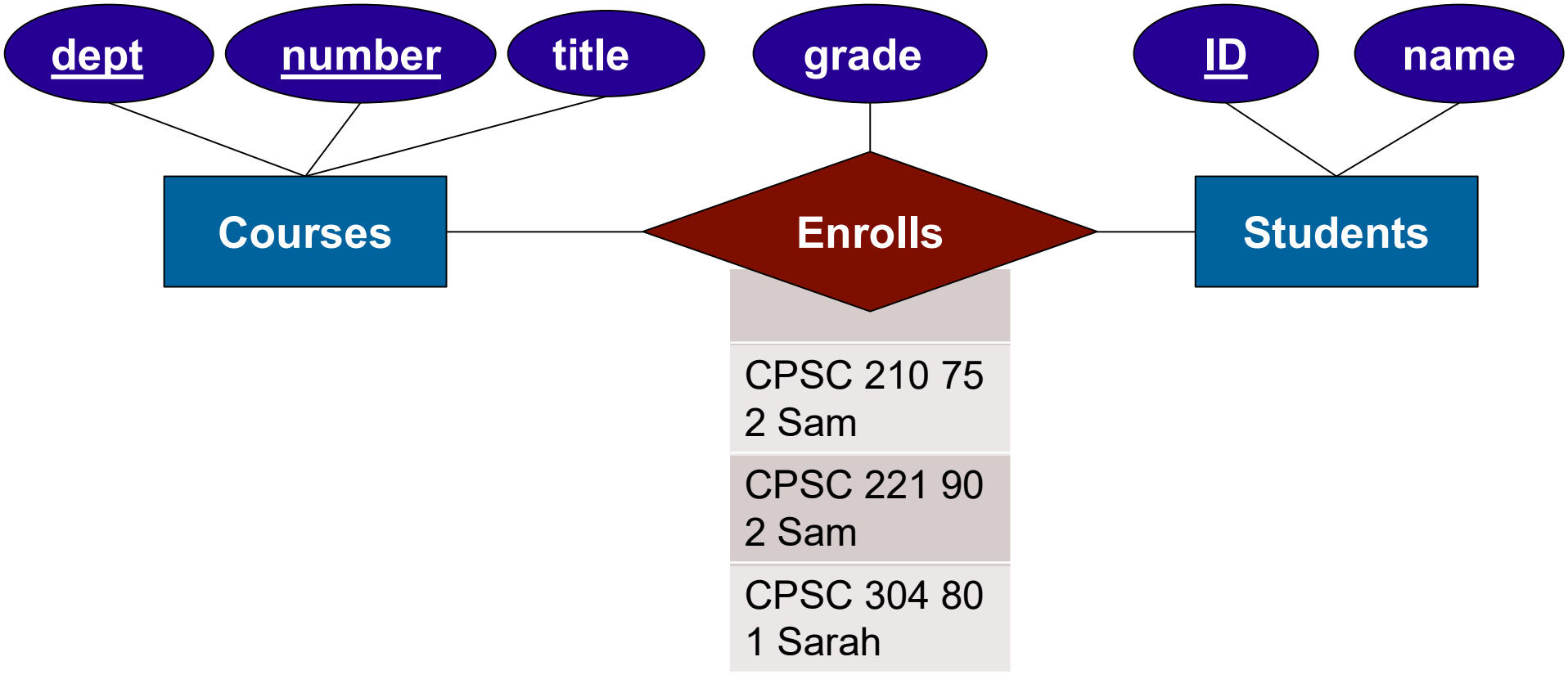


Courses
CPSC 110 …
CPSC 210 …
CPSC 221 …
CPSC 304 …



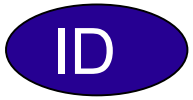
Students
1 Sarah
2 Sam
3 Sophia
4 Sebastian

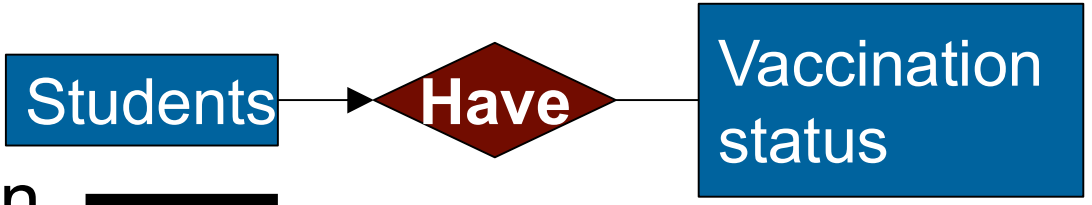
Review – Relationship Sets



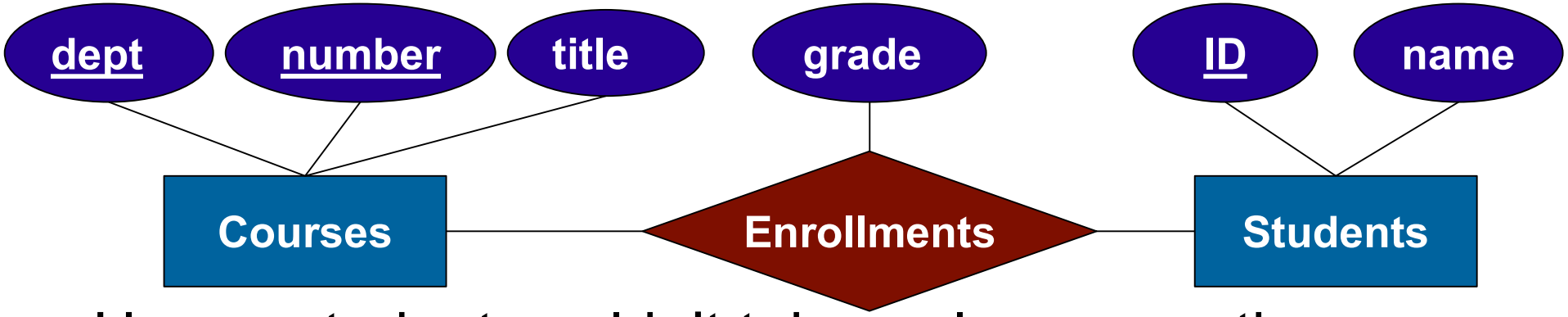
Now where were we...

- We'd just finished covering the basics of ER diagrams, including:
  - Entities
  - Attributes
  - Relationships
  - Key constraints
  - Total participation
  - Generalization/Specialization

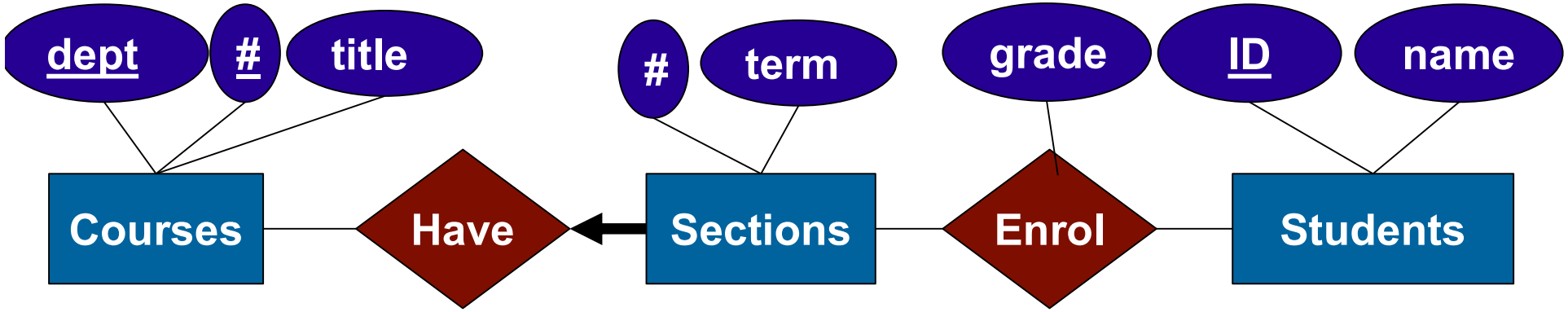




Can we improve the design?

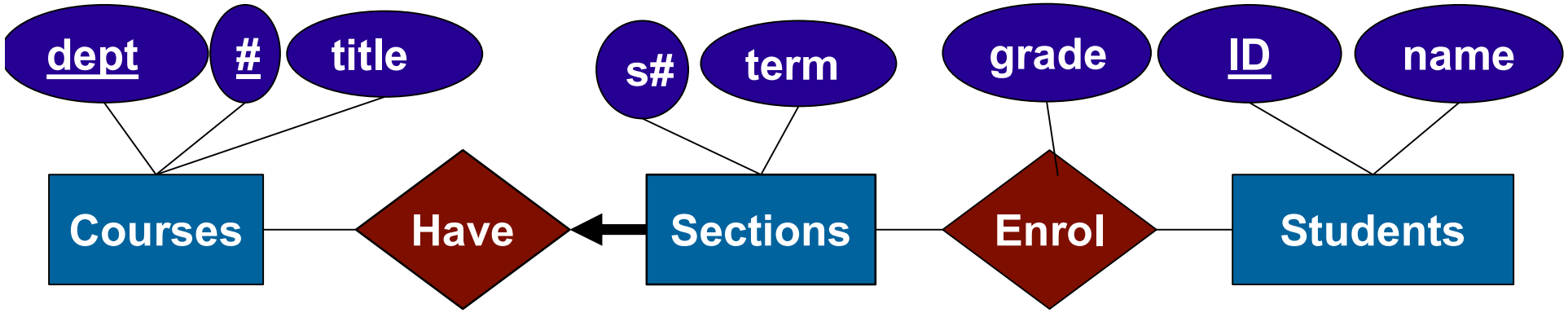


Here a student couldn't take a class more than once. Try:



But what should the key of Sections be?

A better solution would be if Sections could depend on Courses



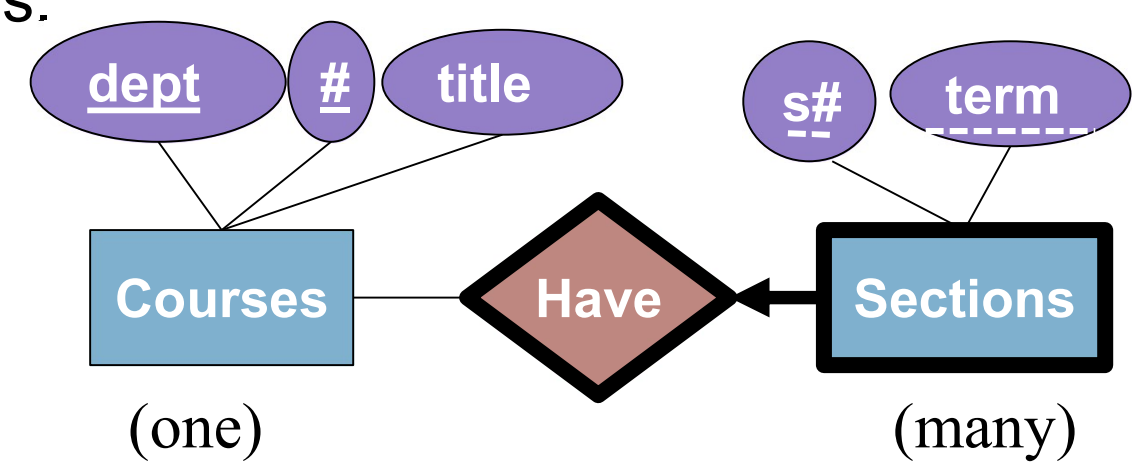
The key has to be unique, right?

- Looking only at Sections (s# and term):
  101 2021W1 is not unique
- Looking at Courses + Sections (dept # s# term):
  CPSC 304 101 2021W1 is unique

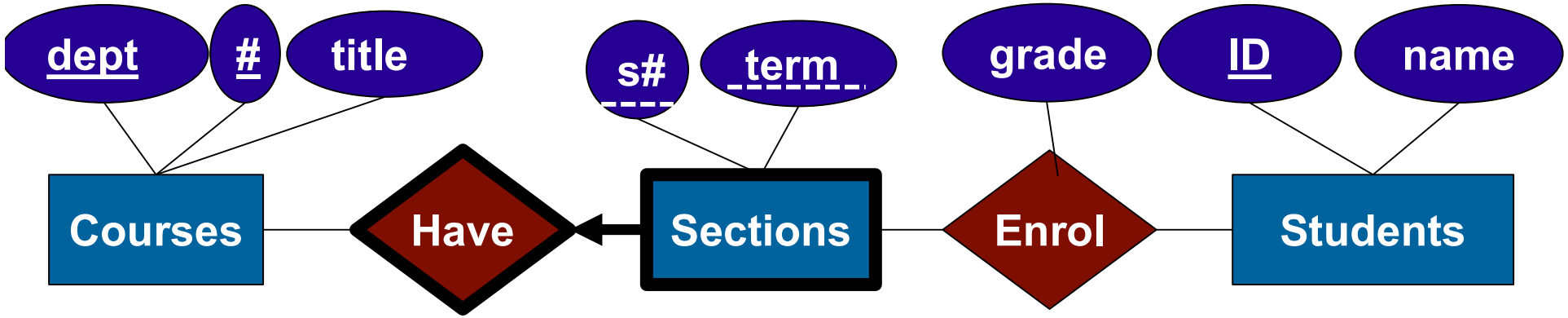
We can do this with a *weak entity*

Weak Entities

- A ***weak entity*** can be identified uniquely only by additionally considering the key of another (*owner*) entity.
  - Think of this as a "belongs to" relationship.
- Owner entity set and weak entity set must participate in a ***one-to-many*** relationship set (one owner, many weak entities).
- Weak entity set must have ***total*** participation in this identifying relationship set.
- Weak entity sets and their identifying relationship sets are shown with ***thick*** lines.



A better solution would be if Sections could depend on Courses



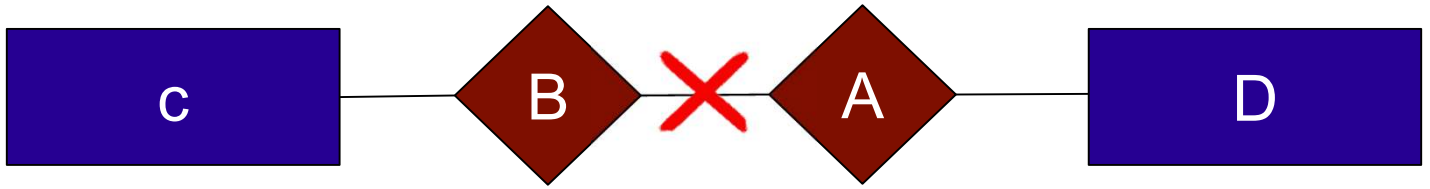
{dept, #, s#, term} is the ***key*** for Have.

{s#, term} is a ***partial key***, underlined with a dashed line.

Congratulations, you can now take CPSC 304 twice! (yay? ☺)

Aggregation

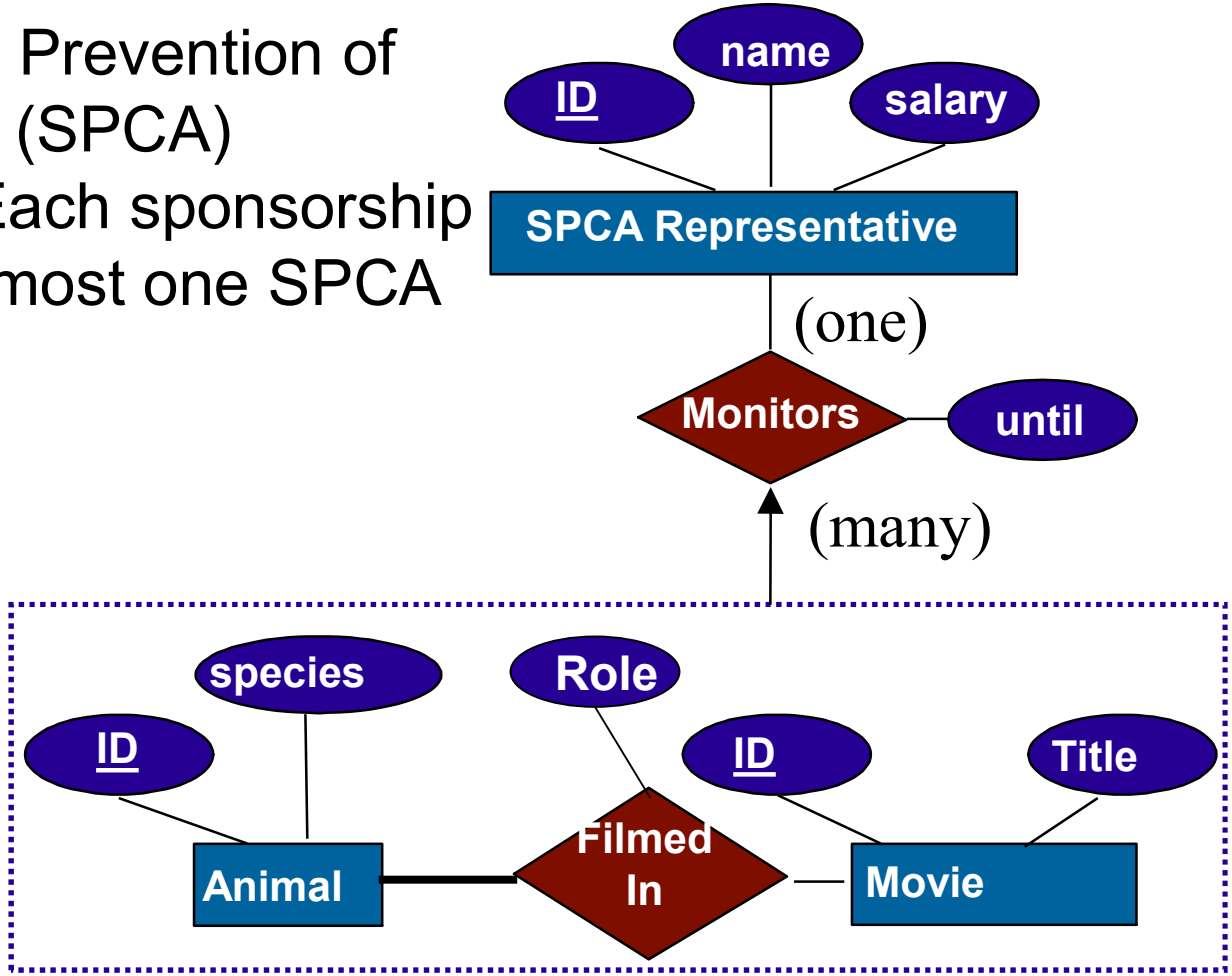
- Having a relationship between relationships is forbidden.



- *Aggregation* allows us to treat a relationship set as an entity set for purposes of participation in (other) relationships

Aggregation: getting around relationships between relationships

- The Society for the Prevention of Cruelty to Animals (SPCA) monitors movies. Each sponsorship is monitored by at most one SPCA representative

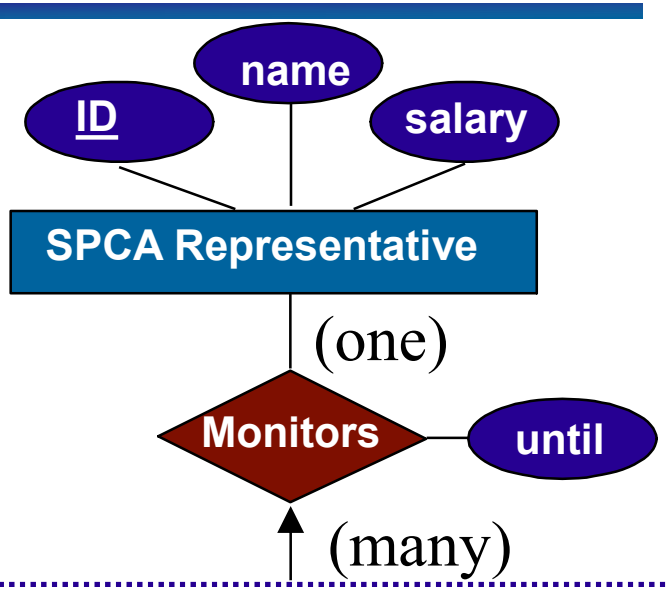


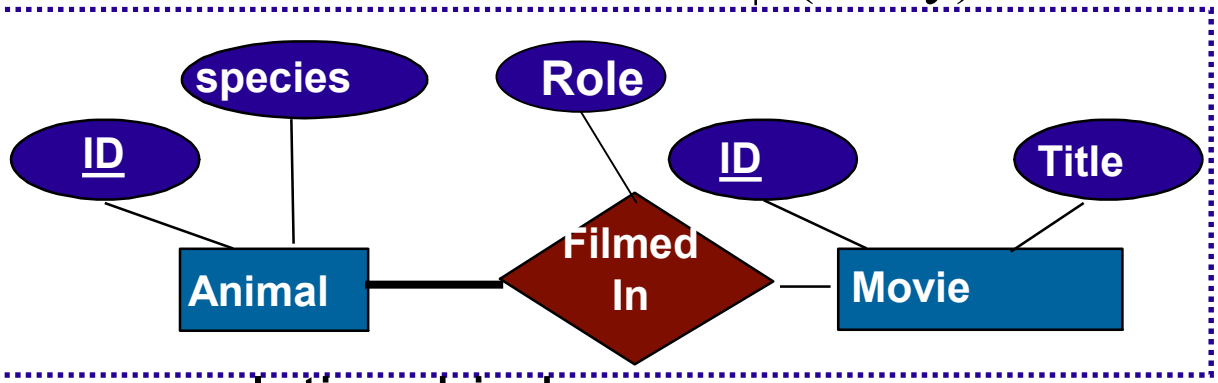
Aggregation: getting around relationships between relationships

- The Society for the Prevention of Cruelty to Animals (SPCA) monitors movies. Each sponsorship is monitored by at most one SPCA representative
- What is the key for FilmedIn?

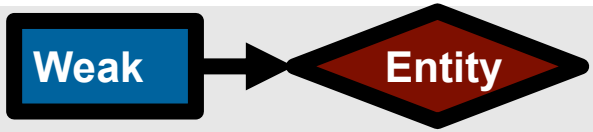
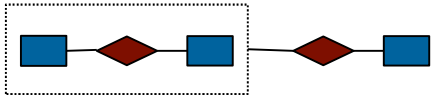
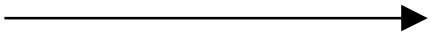
  Animal ID, movie ID
- What is the key for Monitors?

  Animal ID, movie ID
- This differs from a ternary relationship because monitors is its own relationship with a descriptive attribute

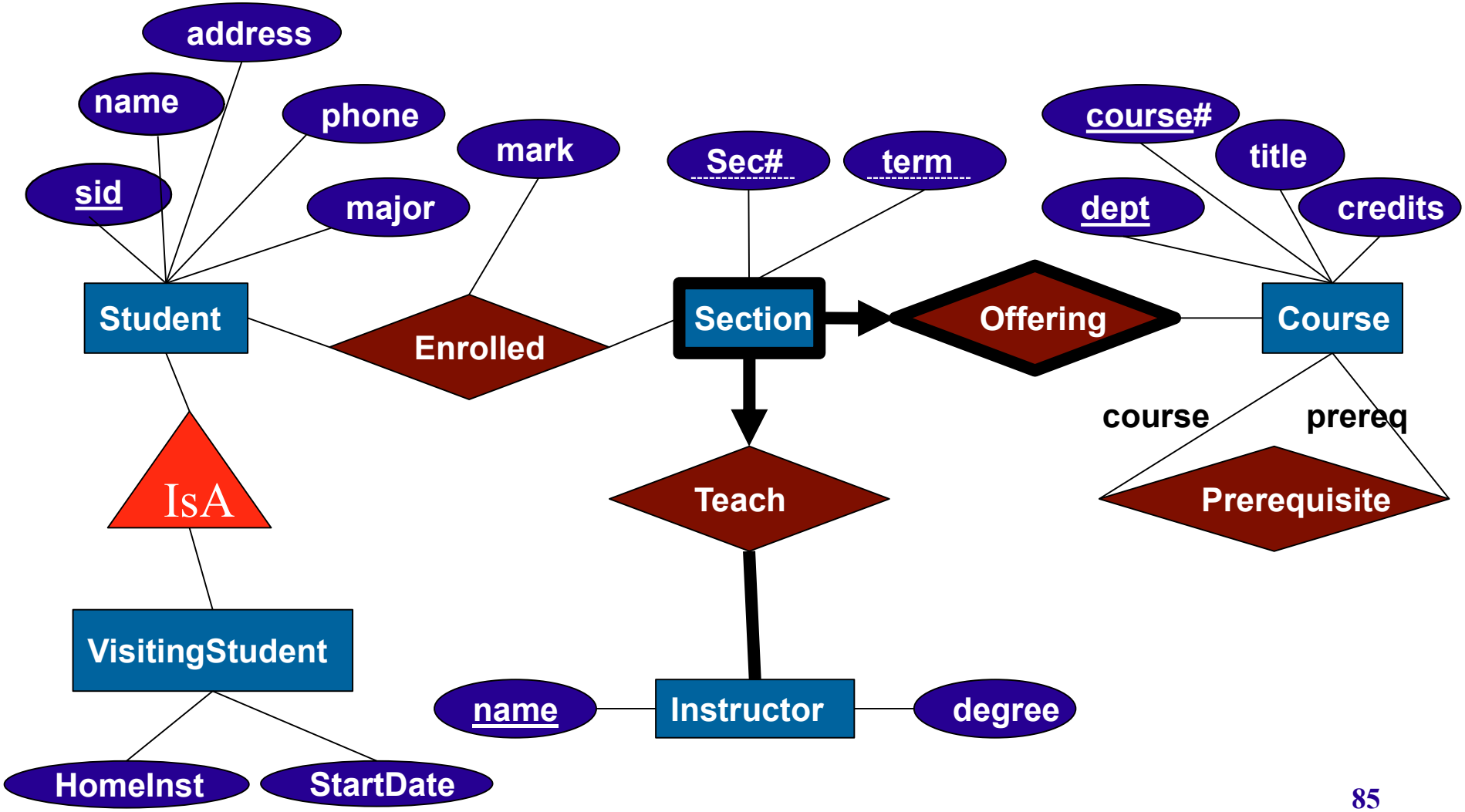




Summary

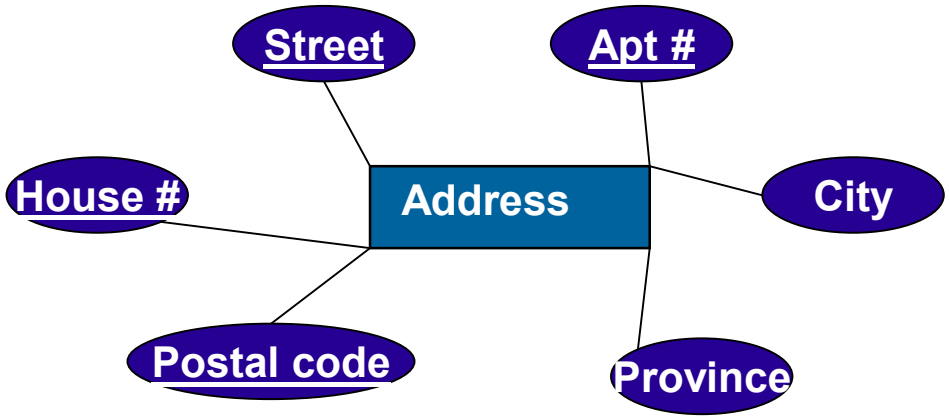
Name	Symbol
Entity	
Attribute	
Relationship	
Generalization/Specialization	
Weak Entity	
Aggregation	
Participation Constraints	
key constraints	

Sample solution



That's all there is to it

- Some ER models differ in expressiveness
- They model *most* concepts people want
- They don't model all of them, e.g.,
  - Functional dependencies – some attributes determine some other attributes, e.g., postal code determines (only) city and province



Conceptual Design Using the ER Model

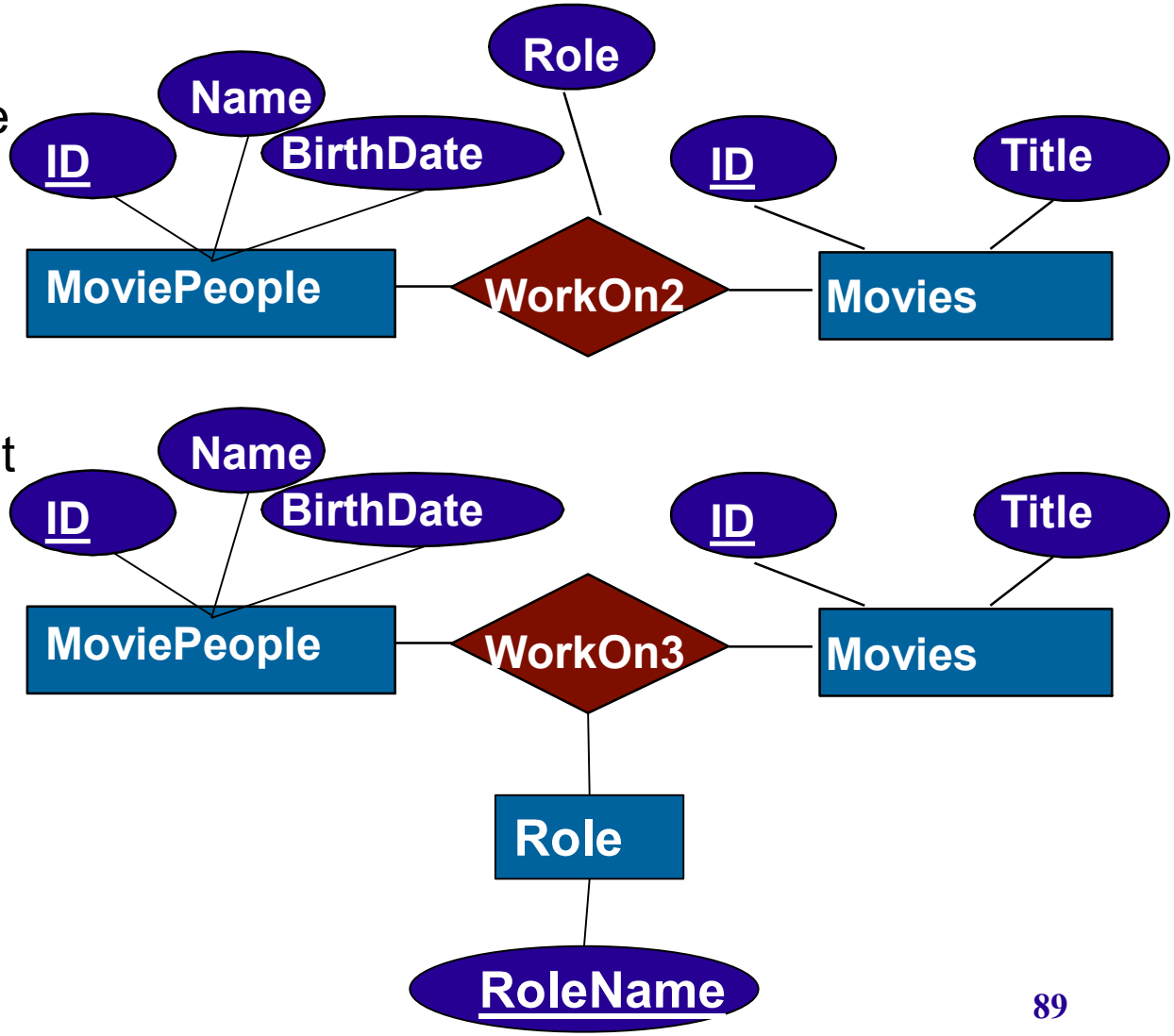
- Design choices:
  - Should a concept be modeled as an entity or an attribute?
  - Should a concept be modeled as an entity or a relationship?
  - Identifying relationships: Binary or ternary? Aggregation?
- Constraints in the ER Model:
  - A lot of data semantics can (and should) be captured.
  - But some constraints cannot be captured in ER diagrams.
    - i.e. domain constraints
    - dependencies

Entity vs. Attribute

- Should an *address* be an attribute of MoviePeople or an entity (connected to MoviePeople by a relationship)?
- Depends upon
  - the use we want to make of address information
  - the semantics of the data:
    - If we have several addresses per person, *address* must be an entity (since attributes cannot be set-valued).
    - If a person has only one street address, one city, one province, one postal code, etc. then these should simply be attributes.

Entity vs. Attribute (Cont.)

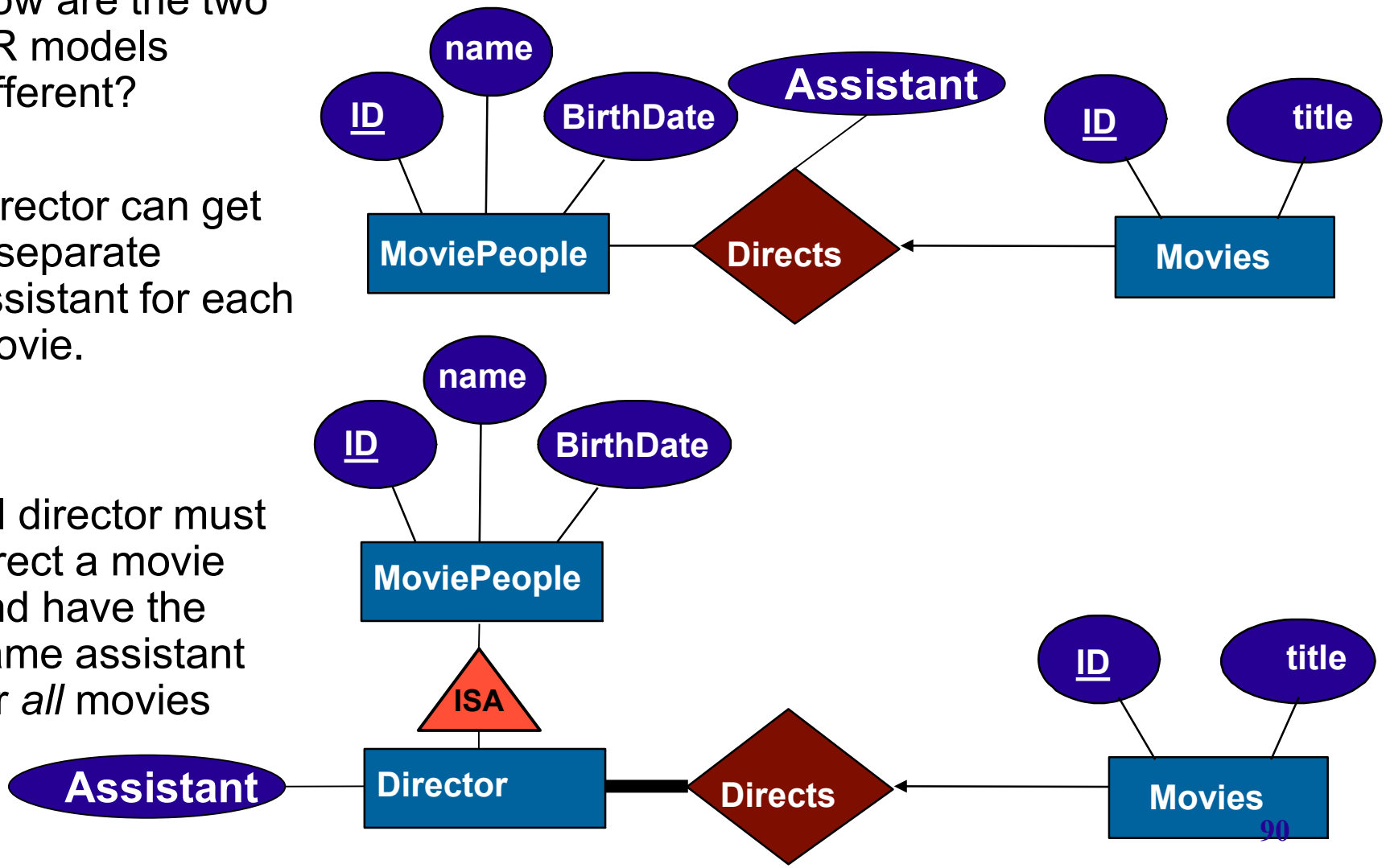
- WorkOn2 does not allow a person to have more than one role in the same movie.
- We want to associate the same pair (MoviePerson, Movie) with more than one set of values for the descriptive attributes?



- Solution: change descriptive attributes into entities.

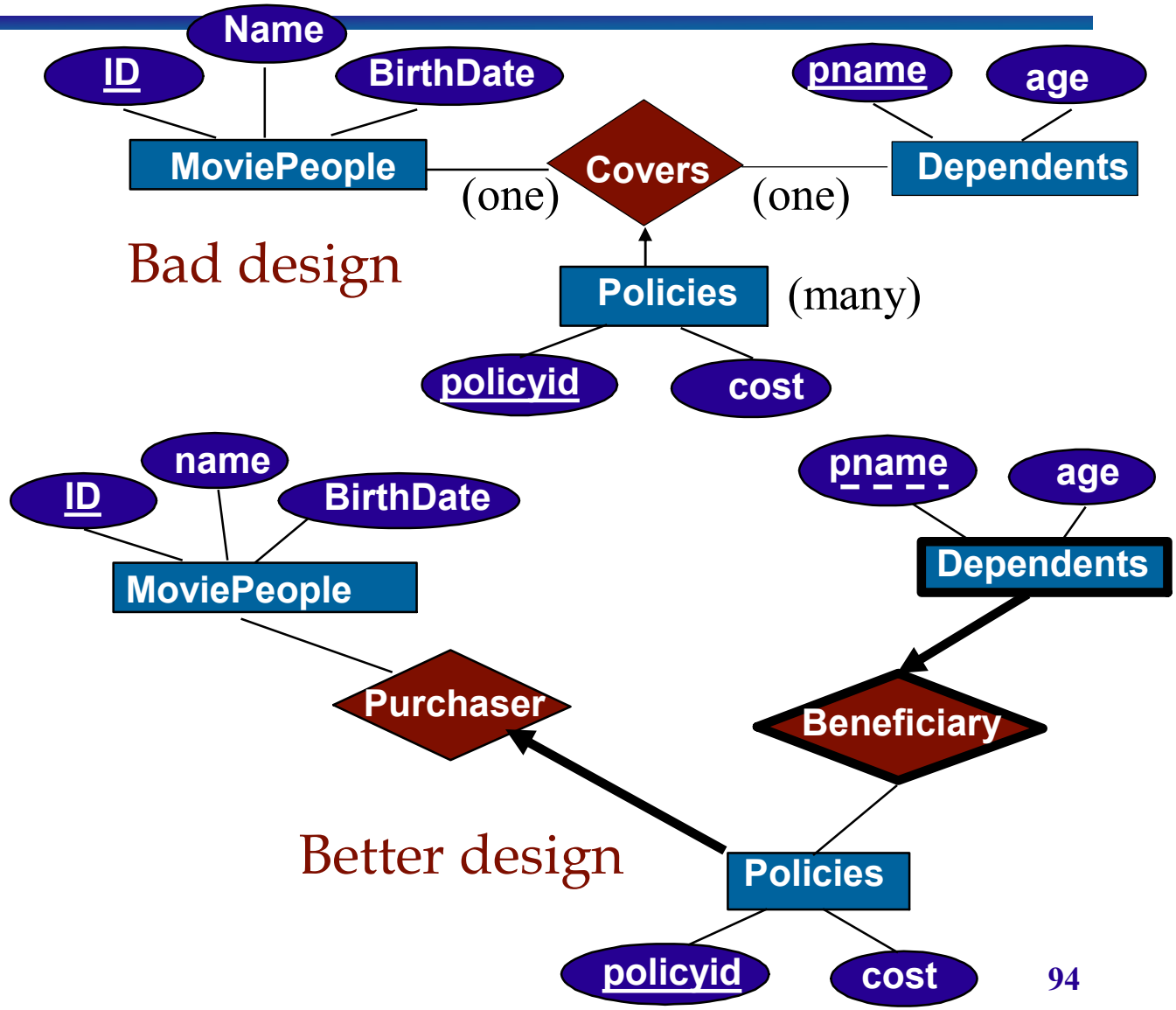
Entity vs. Relationship

- How are the two ER models different?
- Director can get a separate assistant for each movie.
- All director must direct a movie and have the same assistant for *all* movies



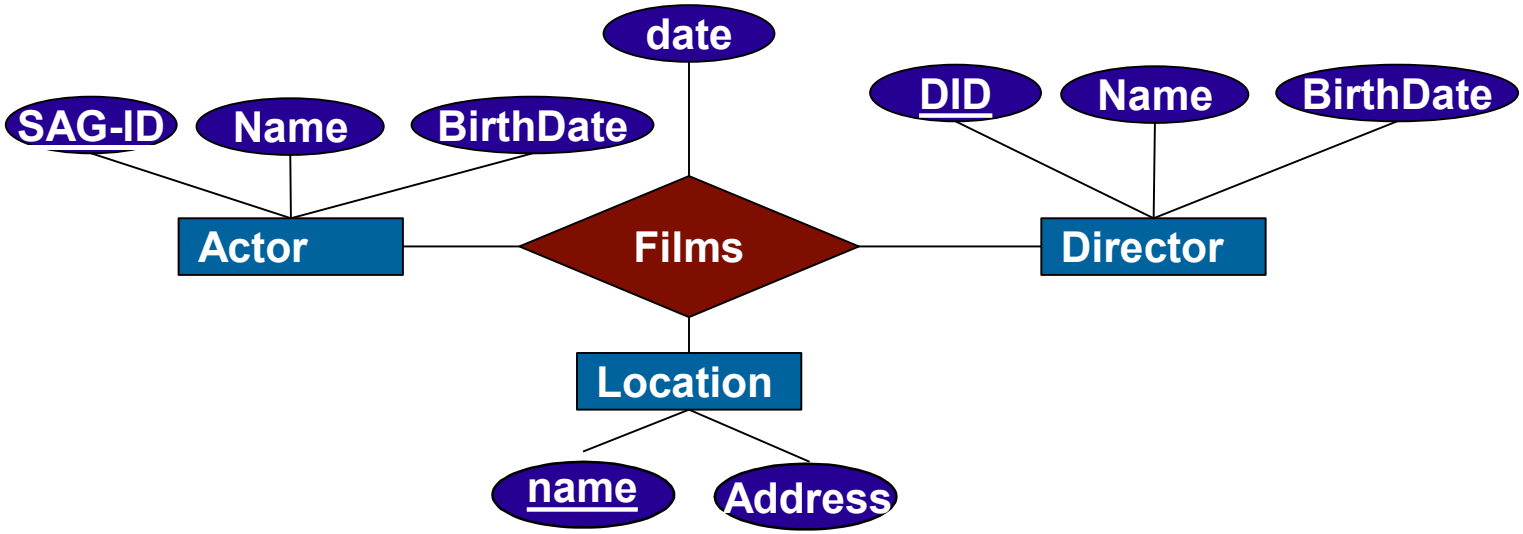
Binary vs. Ternary Relationships

- If each policy is owned by just 1 person:
  - Key constraint on Policies would mean policy can only cover 1 dependent!



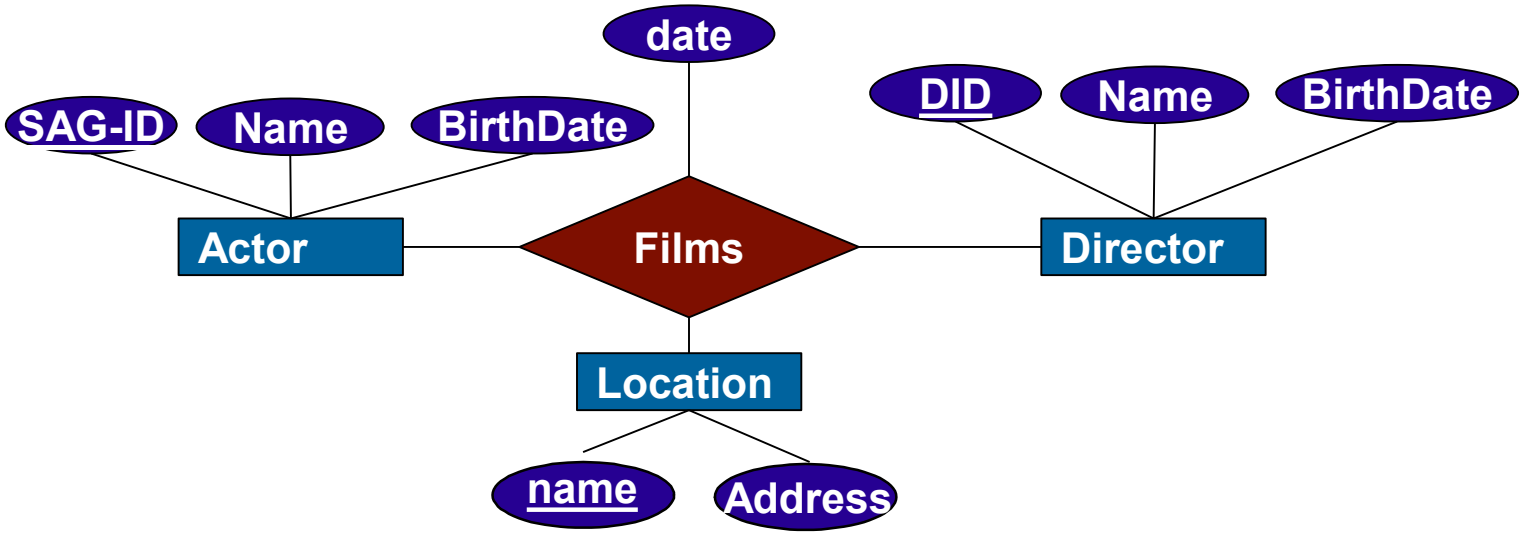
Binary vs. Ternary Relationships

- An example in the other direction: a ternary relationship Films relates entity sets Actor, Director and Location, and has descriptive attribute *date*.



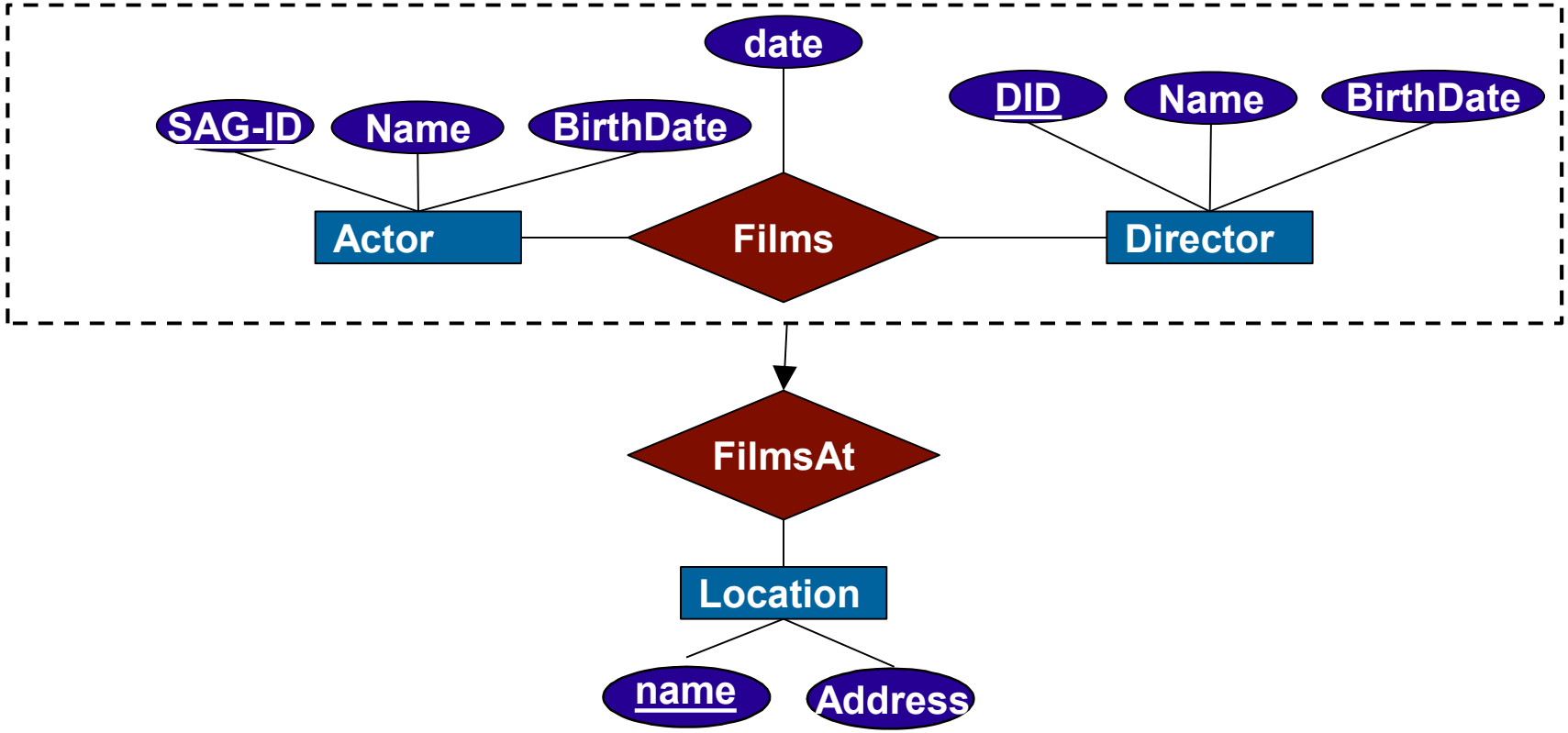
- Can we use two of binary relationships instead?

Binary vs. Ternary Relationships vs. Aggregation



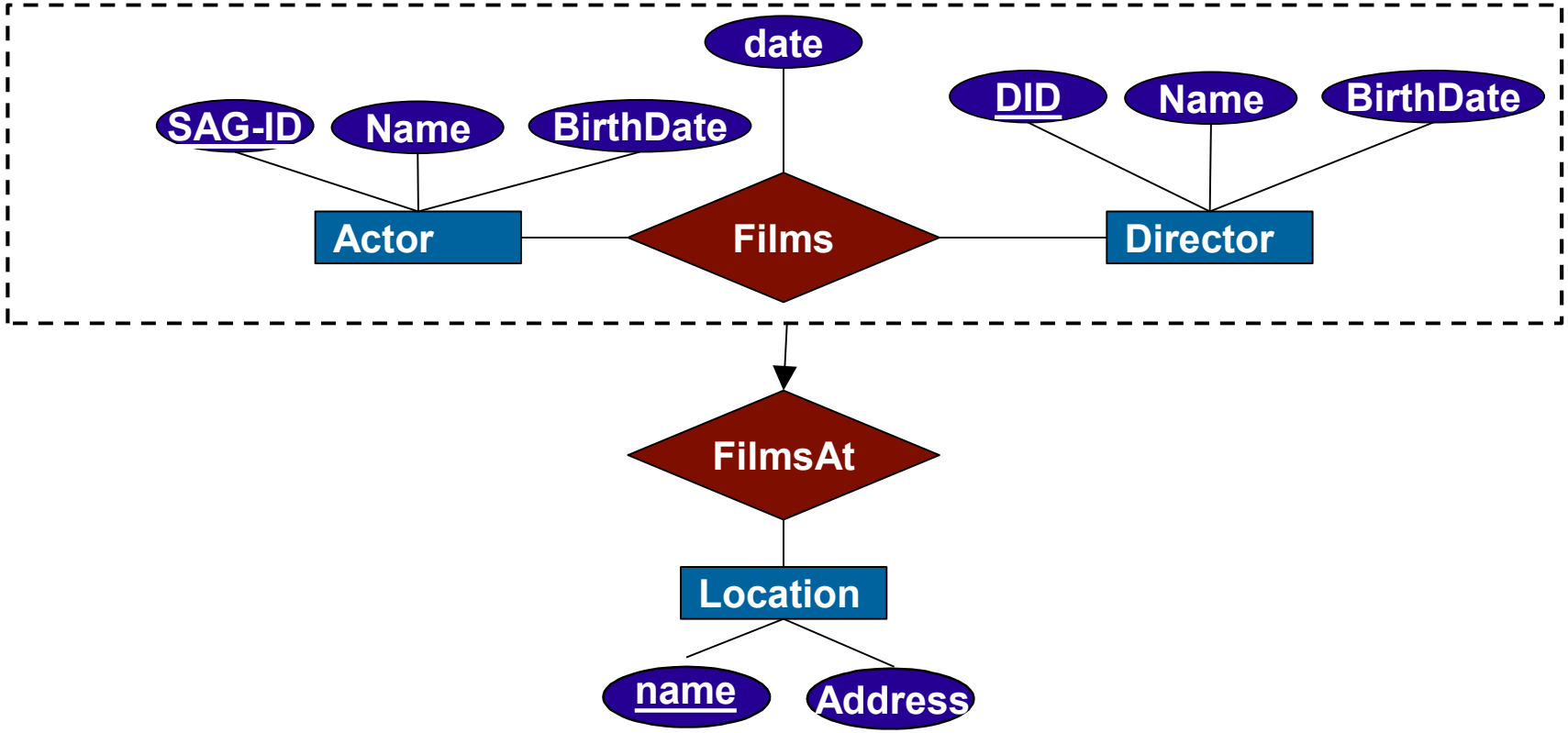
- No combination of binary relationships is an adequate substitute:
  - D "directs" A, D "is filmed at " L, and D "directs at" L does not imply that D directs A at L.
  - Also, how would we record *date*?

Binary vs. Ternary Relationships vs. Aggregation



- Aggregation can be used instead of a ternary relation if need to impose additional constraints:
  - I.e., If you know the actor and the director, you know the location they filmed at.

Binary vs. Ternary Relationships vs. Aggregation



- Aggregation can be used instead of a ternary relation if need to impose additional constraints:
  - I.e. A movie can only be filmed by one director. An actor can be directed by that director in any location.

Summary of Conceptual Design

- *Conceptual design* follows *requirements analysis*,
  - Yields a high-level description of data to be stored
- ER model popular for conceptual design
  - Constructs are expressive, close to the way people think about their applications.
- Basic constructs: *entities*, *relationships*, and *attributes* (of entities and relationships).
- Some additional constructs: *weak entities*, *ISA relationships*, and *aggregation*.
- Note: There are many variations on ER models.

Summary of ER (Cont.)

- Several kinds of integrity constraints can be expressed in the ER model: *key constraints*, *participation constraints*, and *overlap/covering constraints* for ISA relationships. Some *foreign key constraints* are also implicit in the definition of a relationship set.
  - Some constraints (notably, *functional dependencies*) cannot be expressed in the ER model.
  - Constraints play an important role in determining the best database design for an enterprise.

Summary of ER (Cont.)

- ER design is *subjective*. There are often many ways to model a given scenario! Analyzing alternatives can be tricky, especially for a large enterprise. Common choices include:
  - entity vs. attribute
  - entity vs. relationship
  - binary or n-ary relationship
  - whether or not to use ISA hierarchies
  - whether or not to use aggregation
- Ensuring good database design: resulting relational schema should be analyzed and refined further.

Learning Goals revisited

- Explain the purpose of an ER diagram, and list the major components.
- Given a problem description, create an ER diagram given a specification. Justify the decisions you make for entities, relationships, keys, key constraints, participation constraints, weak entities, is-a relationships, and aggregations.
- given a problem description, identify alternative representations of the problem concepts and evaluate the choices
- compare alternative ER models for the same domain and identify their strengths and weaknesses