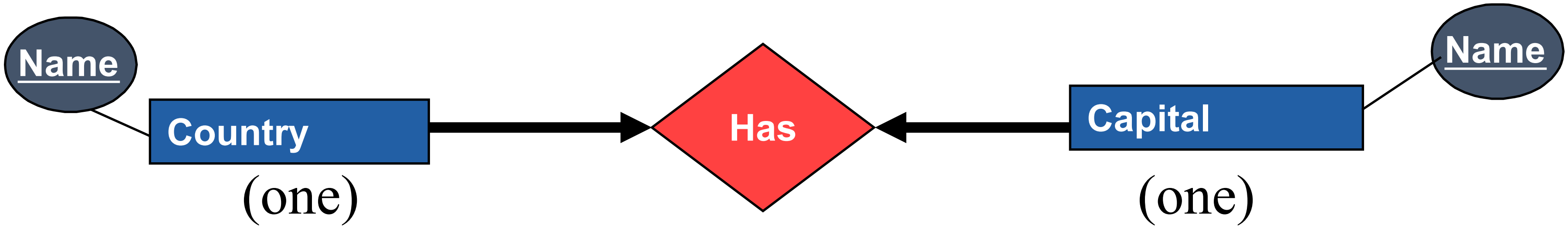
Administrative Notes- Jan 26, 2023

- Feb 3: Assignment 2 (individual) due
- Feb 17: Assignment 3 (pairs if you want) due
  - Make sure you sign up in pairs beforehand as we need time to configure Canvas to allow both partners to view the submission and subsequent feedback
- Feb 20 – 24: Reading Break! (yay!)
  - No lectures, tutorials, or office hours during this week

Foreign Keys and UNIQUE columns

- Textbook: "The foreign key in the reference relation… must match the primary key of the referenced relation."
- Can foreign keys refer to some combination of columns that have a UNIQUE constraint?
  - It depends on the DBMS
  - Some DBMSs (e.g., InnoDB which powers MySQL) will even allow a foreign key constraint to reference a non-unique column
- Even the Oracle DB documentation is inconsistent (e.g., see documentation for Oracle 7)

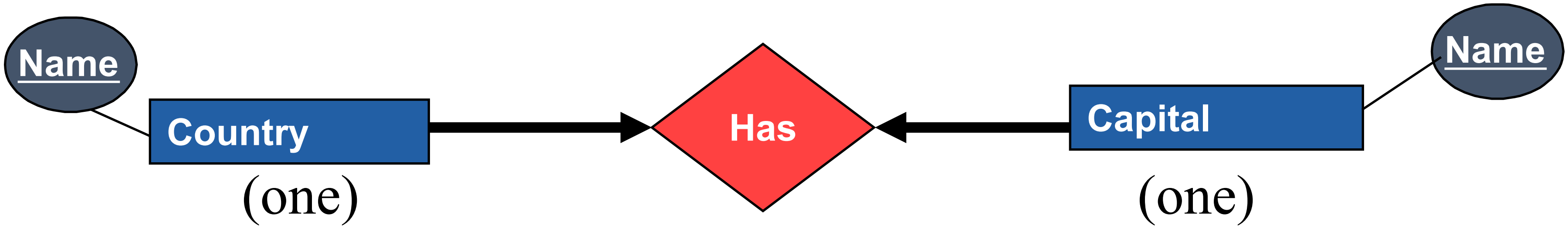
Details, details



Assume you went with Country(<u>coName</u>, caName) and all attributes have type Char(20) and we're not creating a separate relation for Capital. Write the SQL DDL that you would need for this relation.

```
CREATE TABLE Country(
        country-name CHAR(20)  PRIMARY KEY,
        capital-name CHAR(20),
    UNIQUE capital-name ← needed for one-to-one constraint
)
```

Details, Details



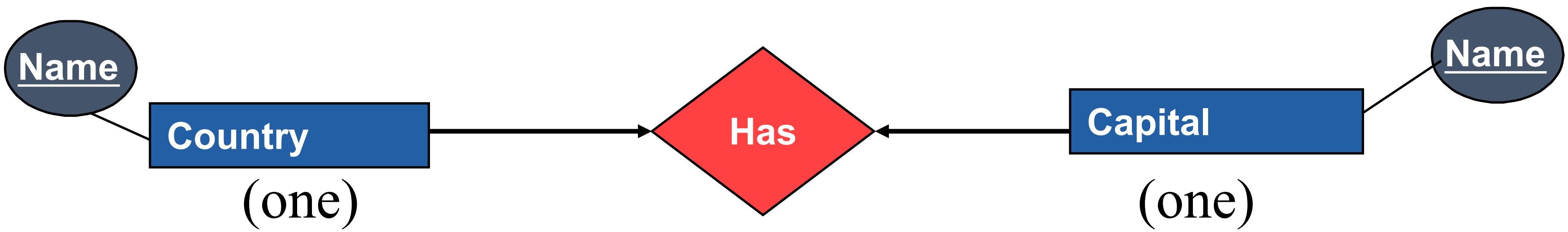
Assume Country(<u>coName</u>, caName) and all attributes of type Char(20) and we're not creating a separate relation for Capital.

```
CREATE TABLE Country(
        country-name CHAR(20)  PRIMARY KEY,
        capital-name CHAR(20),
        UNIQUE capital-name); ← needed for one-to-one constraint)
```

If we did have a separate table for Capital, would we still need the unique constraint?

A. Yes B. No

Clicker Exercise Explained



What happens when we create a separate table for capital? Do we still need UNIQUE?

Country(name, **capital**)

Name	Capital
Canada	
USA	
Mexico	

Capital(name)

Name
Ottawa
Washington, D.C.
Mexico City

What values are allowed in the Capital column?

Clicker Exercise Explained

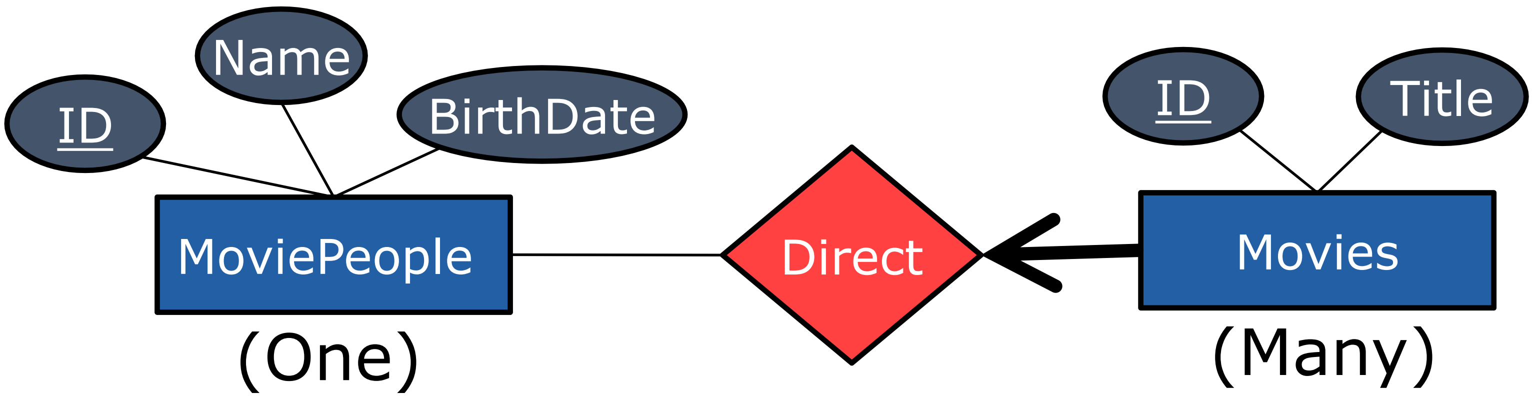
Country(name, **capital**)

Name	Capital
Canada	Ottawa
USA	Ottawa
Mexico	Ottawa

Capital(name)

Name
Ottawa
Washington, D.C.
Mexico City

Translating Participation Constraints



- Every movie must have a director.
  - Every tuple in the Movie table must appear with a non-null *MoviePeople ID* value
- How can we express that in SQL?

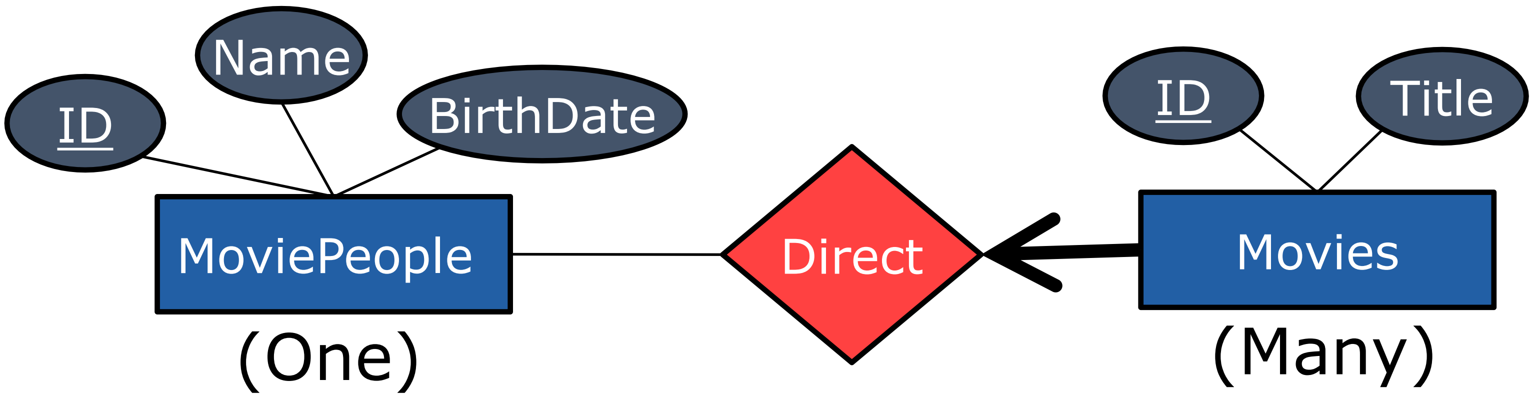
Participation Constraints in SQL

- Using method 2 (add Directs relation in the Movie table), we can capture participation constraints by
  - Ensuring that each **MID** is associated with a **MPID** that is not null
  - Not allowing deletion of a director before the director is replaced

```
CREATE TABLE  Directed_Movie(
  MID        INTEGER,
  title      CHAR(20),
  MPID       CHAR(11) NOT NULL,
  PRIMARY KEY  (MID),
  FOREIGN KEY (MPID) REFERENCES
MoviePeople
             ON DELETE NO ACTION
             ON UPDATE CASCADE)
```

- **Note: We cannot express this constraint if method 1 is used for Direct.**

Participation Constraints in SQL



MoviePeople

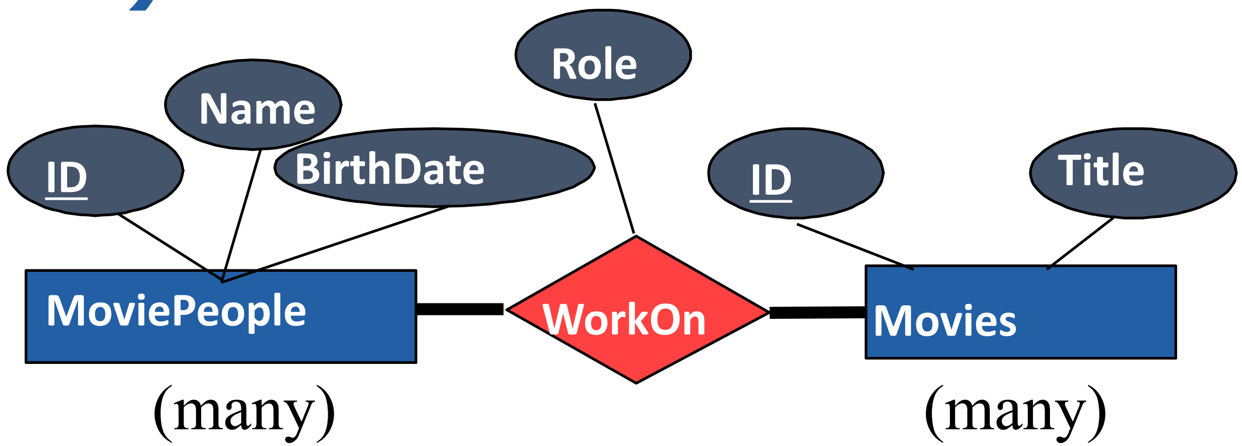
ID	Name	Birthdate
1	Christian Bale	1974/01/30
2	James Cameron	1954/08/16

DirectedMovie

Not legal!→

MID	Title	MPID is not null
1	Avatar	2
~~2~~	~~The Dark Knight~~	~~null~~

Participation Constraints in SQL (cont')



- How can we express that "every movie person works on a movie and every movie has some movie person in it"?
- Neither foreign-key nor not-null constraints in WorkOn can do that.
- We need assertions (later)

Let's see why we can't model this participation constraint using null restrictions

MoviePeople

ID	Name	Birthdate
1	Christian Bale	1974/01/30
2	James Cameron	1954/08/16

Movie

ID	Title
1	Gone With the Wind
2	Avatar

WorkOn

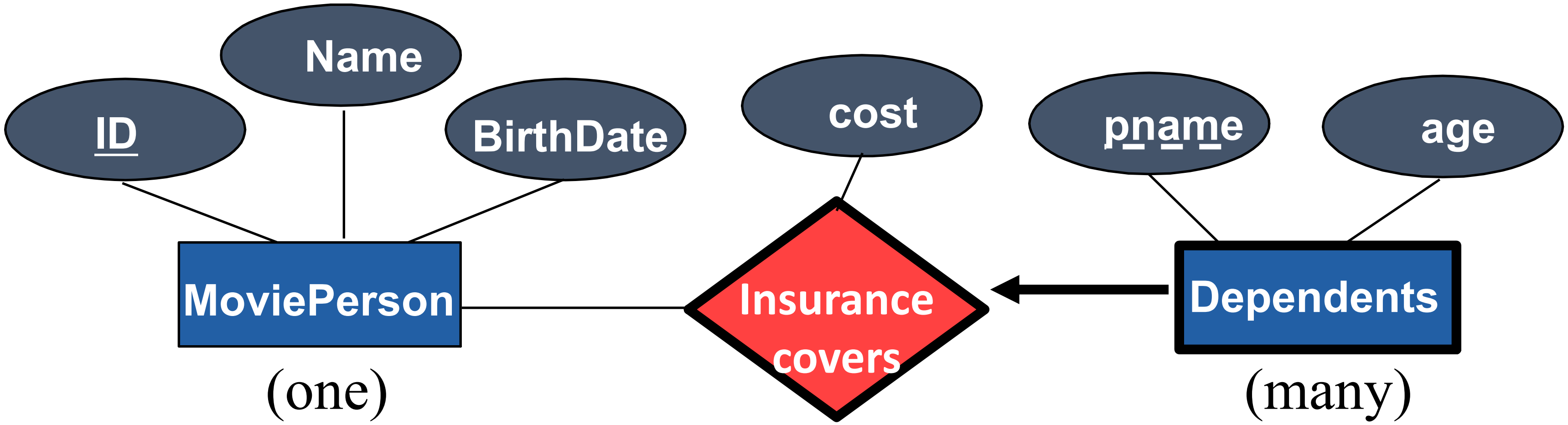
MPID	MID	Role
2	2	Director

No nulls, but Christian Bale does not work on a movie and Gone with the Wind has no one working on it

In-Class Exercise (Relational Model 1)

- See Canvas
- Same rules apply
- Back at 2:43

Translating Weak Entity Sets



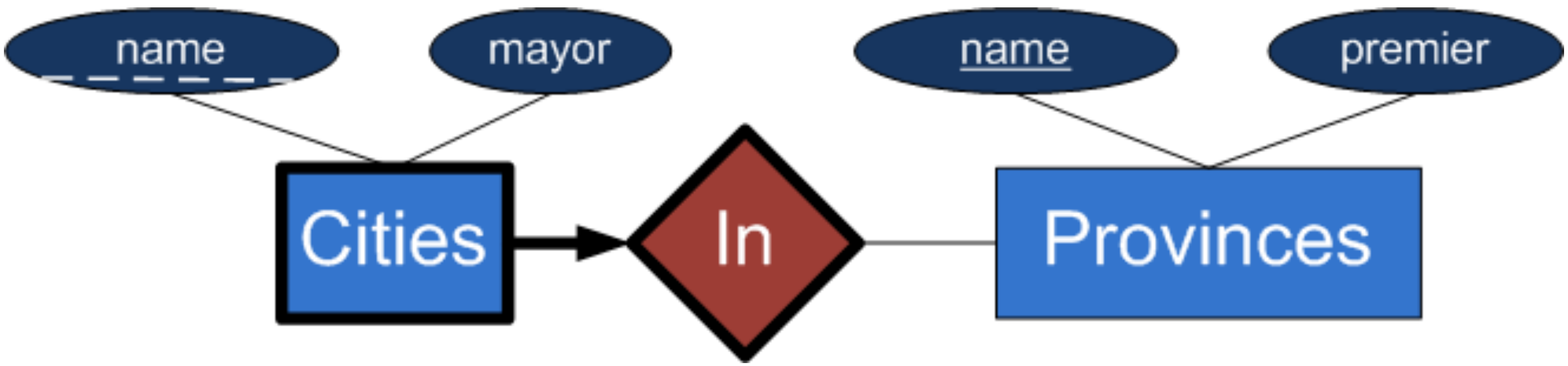
- A ***weak entity*** is identified by considering the primary key of the *owner* (strong) entity.
  - Owner entity set and weak entity set participate in a one-to-many identifying relationship set.
  - Weak entity set has total participation.
- What is the best way to translate it?

Translating Weak Entity Sets(cont')

- Weak entity set and its identifying relationship set are translated into a single table (like many to one anyway)
  - Primary key would consist of the owner's primary key and weak entity's partial key
  - When the owner entity is deleted, all owned weak entities must also be deleted.

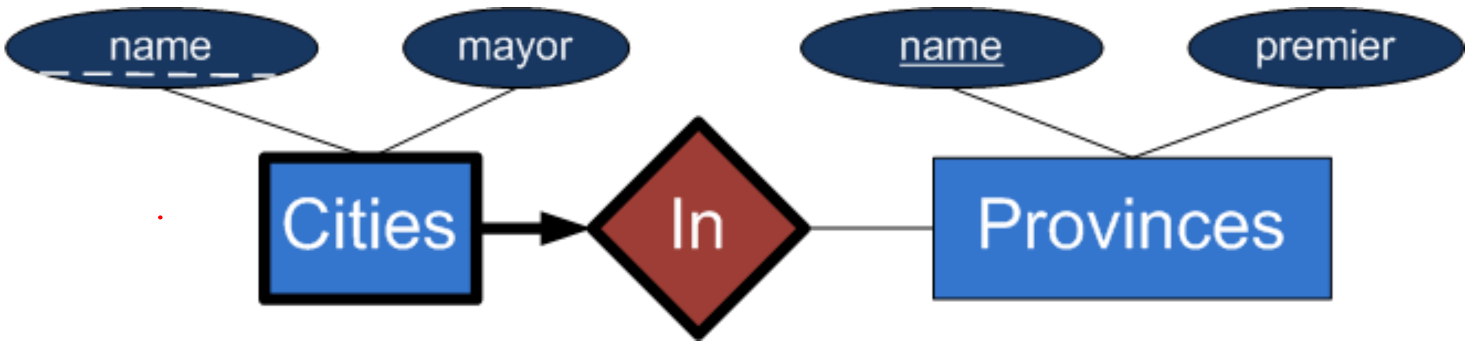
```
CREATE TABLE Dep_Insurance (
  pname  CHAR(20),
  age    INTEGER,
  cost   REAL,
  ID     CHAR(11)
  PRIMARY KEY (ID, pname),
  FOREIGN KEY (ID) REFERENCES MoviePeople,
    ON DELETE CASCADE)
```

In-Class Exercise (no need to hand it in)



Convert this E/R diagram to relations, resolving the dual use of "name" in some reasonable way.

Clicker Question



Convert this E/R diagram to relations, resolving "name" in some reasonable way. Foreign keys are bolded. Which schema below is the best translation from ER to relations?

A. Cities(name, mayor), Provinces(name, premier)

B. Cities(**cname**, **pname**, mayor), Provinces(pname, premier)

C. Cities(cname, **pname**, mayor), Provinces(pname, premier)

D. Cities(cname, **pname**, mayor), In(cname, pname), Provinces(name, premier)

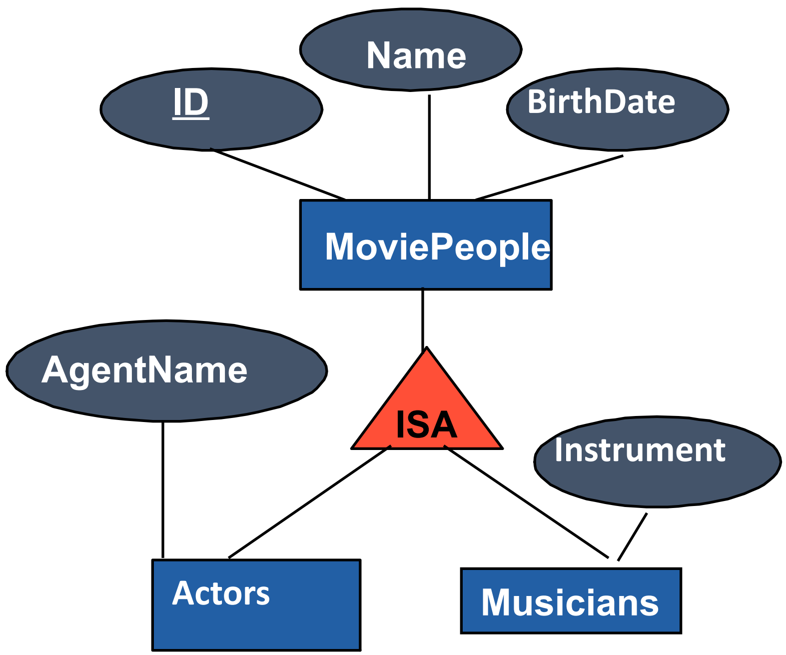
E. None of the above

Cities
Provinces
In

→

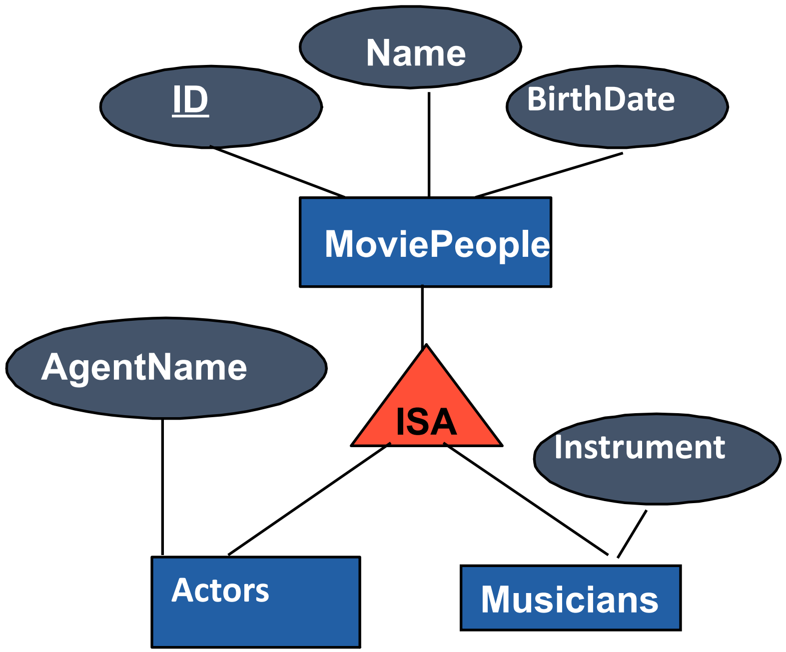
Cities(cname, **pname**, mayor)
Provinces(pname, premier)

Translating ISA Hierarchies to Relations



What is the best way to translate this into tables?

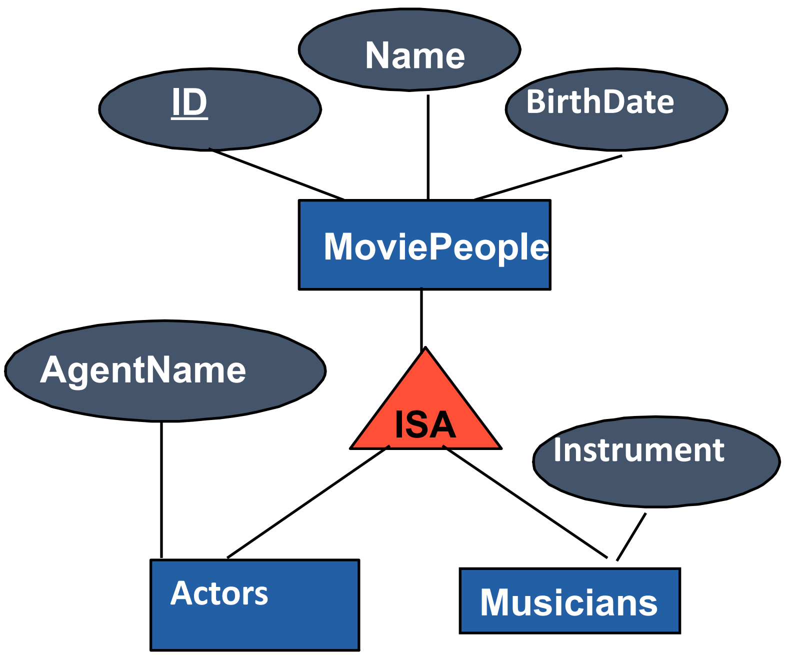
Totally unsatisfactory attempt: Safest but with lots of duplication (not in book)



One table per entity. Each has *all* attributes:

- MoviePeople(<u>ID</u>, Name, BirthDate, AgentName, Instrument)
- Actors(<u>ID</u>, Name, BirthDate, AgentName, Instrument)
- Musicians(<u>ID</u>, Name, BirthDate, AgentName, Instrument)

Method 1:have only one table with *all* attributes (not in book)



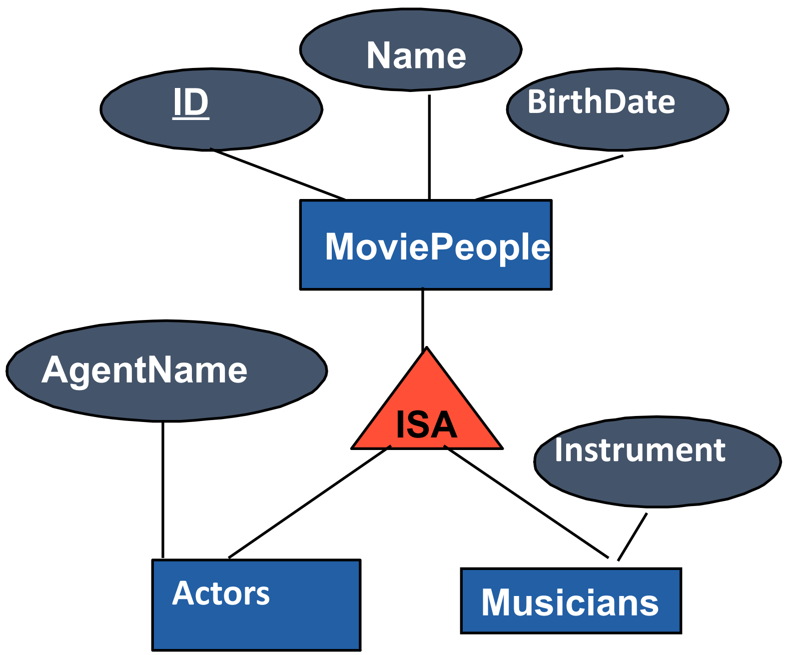
MoviePeople(<u>ID</u>, Name, BirthDate, AgentName, Instrument)

~~Actors(ID, Name, BirthDate, AgentName, Instrument)~~

~~Musicians(ID, Name, BirthDate, AgentName, Instrument)~~

✯Lots of space needed for nulls

Method 2: 3 tables, remove excess attributes



- Superclass table contains all superclass attributes
- Subclass table contains primary key of superclass (as foreign key) and the subclass attributes

MoviePeople(ID, Name, BirthDate, ~~AgentName, Instrument~~)

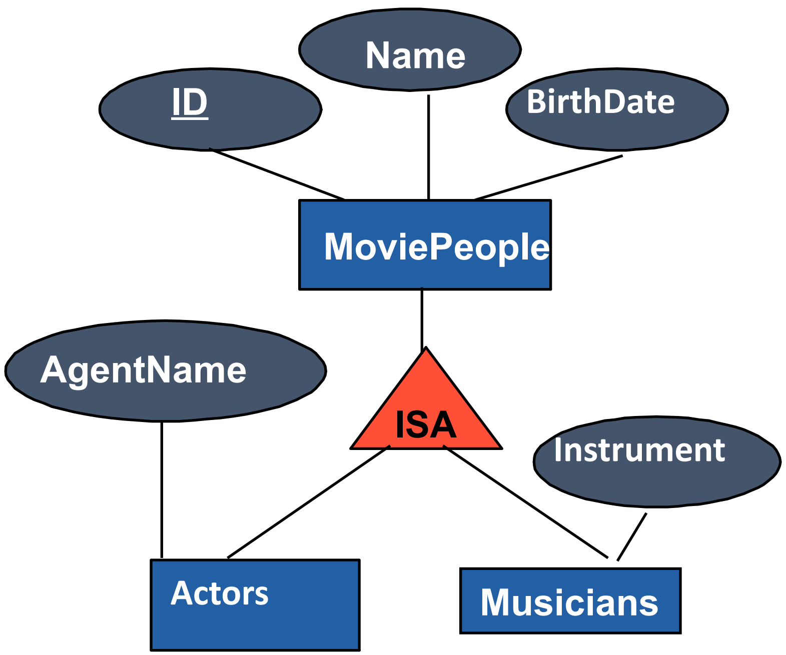
Actors(**ID**, ~~Name, BirthDate~~, AgentName, ~~Instrument~~)

Musicians(**ID**, ~~Name, BirthDate, AgentName~~, Instrument)

✪Works well for concentrating on superclass.

✯Have to combine two tables to get all attributes for a subclass

Method 3: 2 tables, none for superclass



- No table for superclass
- One table per subclass
- Subclass tables have:
  - *All* superclass attributes
  - Subclass attributes

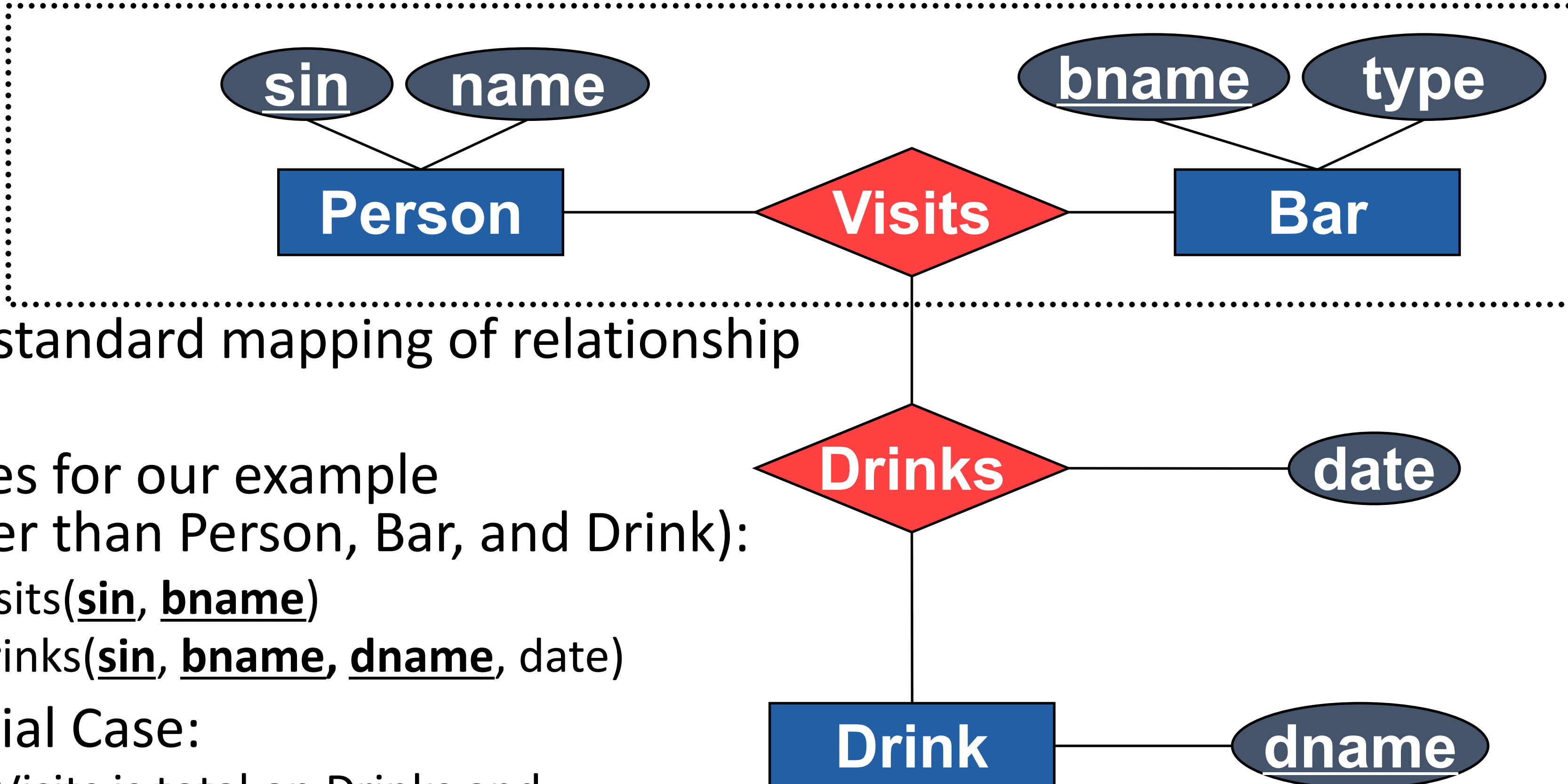
~~MoviePeople(ID, Name, BirthDate, AgentName, Instrument)~~

Actors(<u>ID</u>, Name, BirthDate, AgentName, ~~Instrument~~)

Musicians(<u>ID</u>, Name, BirthDate, ~~AgentName,~~ Instrument)

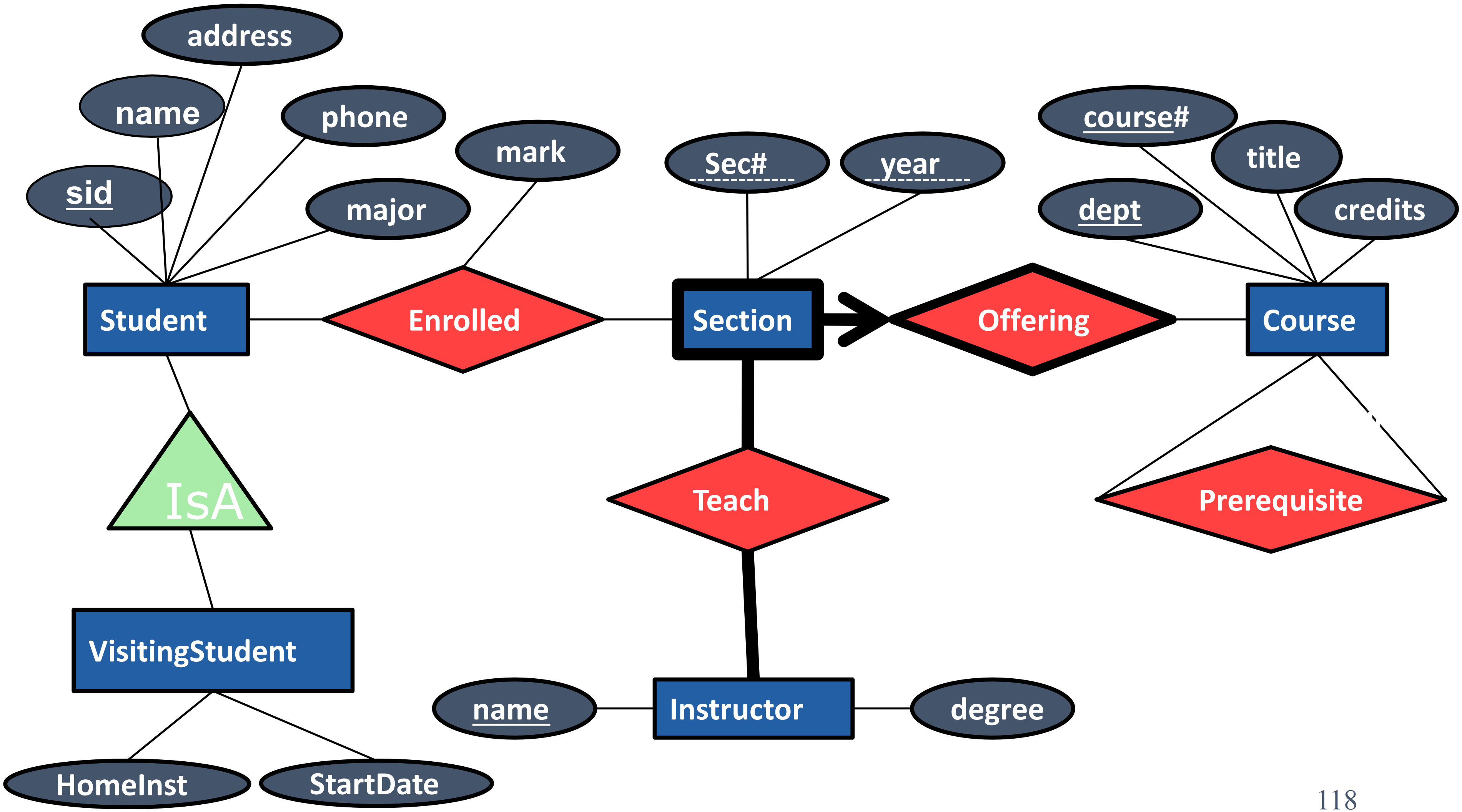
- Works poorly with relationships to superclass
- If ISA-relation is partial, it cannot be applied (loose entities)
- If ISA-relation is not disjoint, it duplicates info

Translating Aggregation



- Use standard mapping of relationship sets
- Tables for our example (other than Person, Bar, and Drink):
  - Visits(<u>**sin**</u>, <u>**bname**</u>)
  - Drinks(<u>**sin**</u>, <u>**bname,**</u> <u>**dname**</u>, date)
- Special Case:
  - If Visits is total on Drinks and Visits has no descriptive attributes we could keep only the Drinks table (discard Visits).

In-Class Exercise (Relational Model 2): Consider the following diagram for a university. List the tables, keys, and foreign keys when converted to relational. Do not write SQL DDL.



Relational Model: Summary

- A tabular representation of data.
- Simple and intuitive, currently the most widely used.
- Integrity constraints can be specified, based on application semantics. DBMS checks for violations.
  - Important ICs: primary and foreign keys
  - Additional constraints can be defined with assertions (but are expensive to check)
- Powerful and natural query languages exist.
- Rules to translate ER to relational model

Learning Goals Revisited

- Compare and contrast *logical* and *physical data independence*.
- Define the components (and synonyms) of the relational model: tables, rows, columns, keys, associations, etc.
- Create tables, including the attributes, keys, and field lengths, using Data Definition Language (DDL)
- Explain and differentiate the kinds of integrity constraints in a database
- Explain the purpose of referential integrity.
- Enforce referential integrity in a database using DML. Determine which delete, insert, or update policy to use when coding rules/defaults for referential integrity. Analyze the impact that a poor choice has.
- Map ER diagrams to the relational model (i.e., DDL), including constraints, weak entity sets, etc.