CPSC 304 – September 25-26, 2024 Administrative Notes

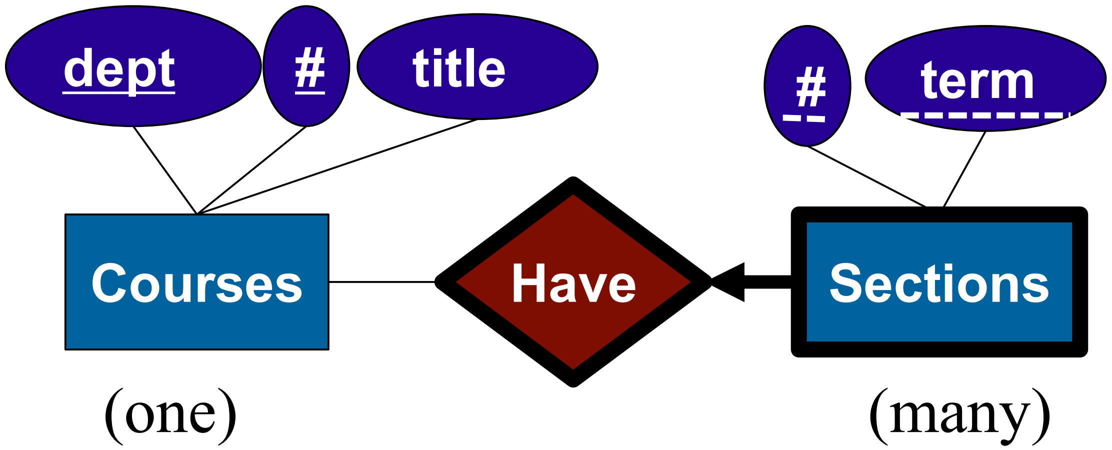
- Project
  - Groups are all formed
  - Milestone 1 due on October 1
- Tutorial
  - week of the 23rd: project work time (get feedback!)
  - Week of September 30: Relational model
    - If you have tutorial on Monday, go to another section

Now where were we…

- We’d been covering translating ER diagrams to the relational model
- We’d handled the basics, including
  - Entities
  - Many-to-many relationships
  - Many-to-one and one-to-one relationships
  - Total participation constraints

Let’s do the rest of the concepts.

Translating Weak Entity Sets



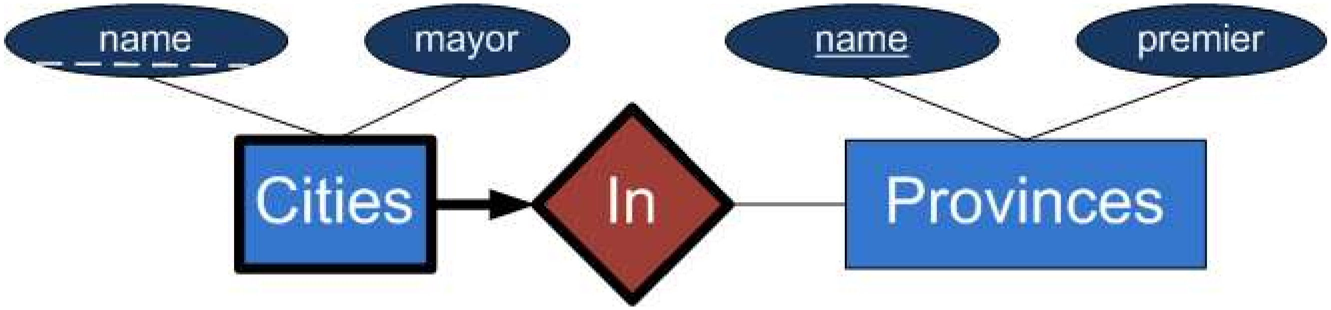
- A ***weak entity*** is identified by considering the primary key of the *owner* (strong) entity.
  - Owner entity set and weak entity set participate in a one-to-many identifying relationship set.
  - Weak entity set has total participation.
- What is the best way to translate it?

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.

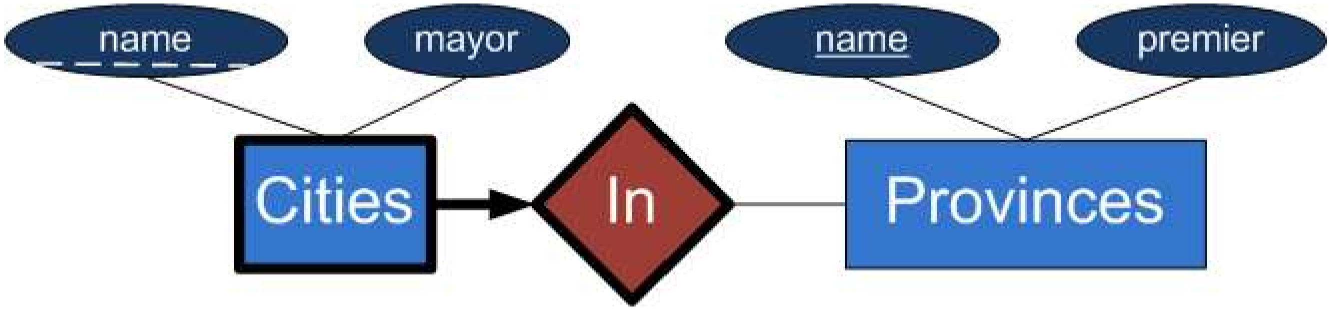
```
CREATE TABLE  Course_Section (
   dept               CHAR(4),
   course_num INTEGER,
   section_num INTEGER,
   term                CHAR(6)
   PRIMARY KEY  (dept, course_num, section_num, term),
   FOREIGN KEY  (dept, course_num) REFERENCES
                         Courses(dept, num),
                         ON DELETE CASCADE)
```

Clicker exercise



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

Clicker exercise



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)

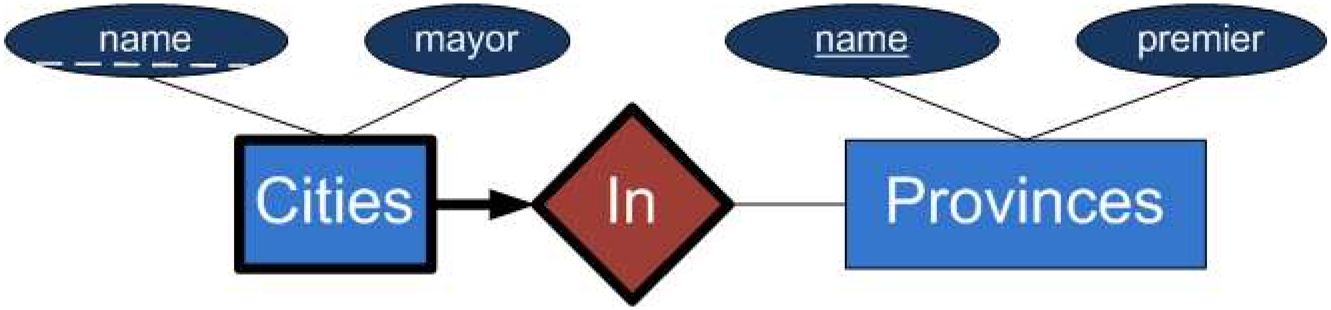
B. Cities(**cname**, **pname**, mayor), Provinces(pname, premier)

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

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

E. None of the above

Clicker exercise



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)

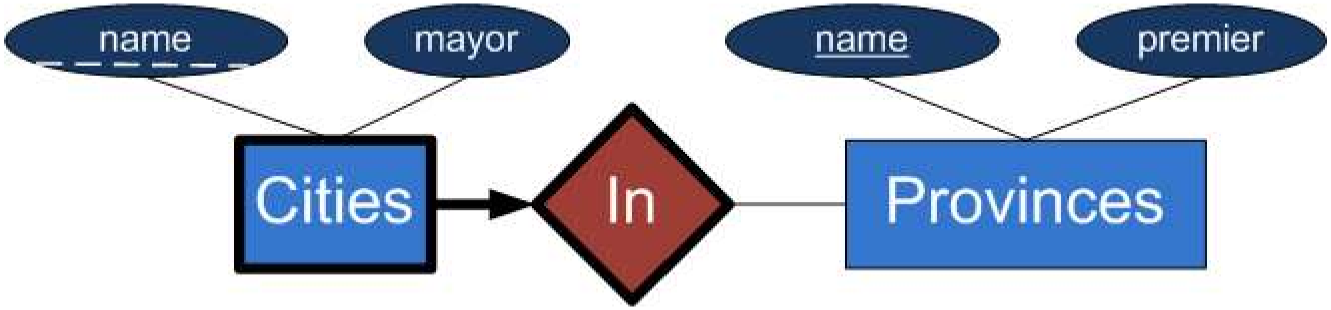
[✓ C. Cities(cname, **pname**, mayor), Provinces(pname, premier)]

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)

Clicker exercise



Given the solution in the previous clicker question:

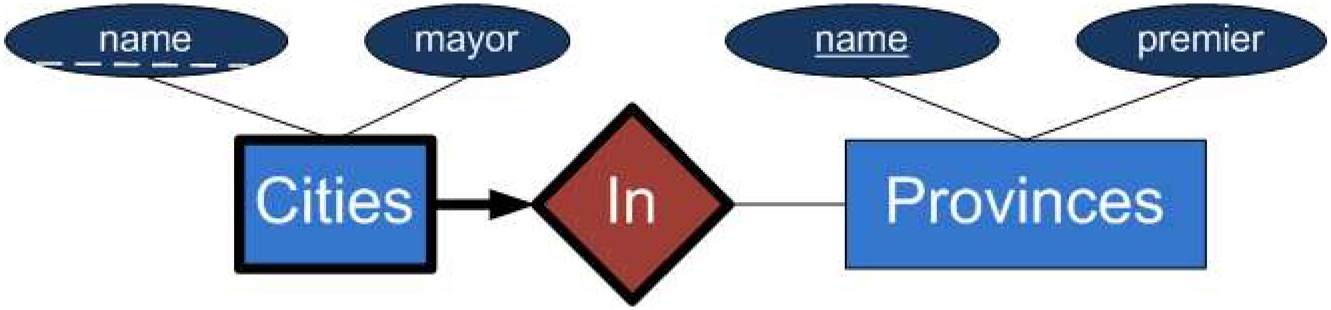
Cities(<u>cname</u>, **<u>pname</u>**, mayor), Provinces(<u>pname</u>, premier)

Do we need to have a “not null” constraint on pname due to the total participation constraint?

A. Yes

B. No

Clicker exercise



Given the solution in the previous clicker question:

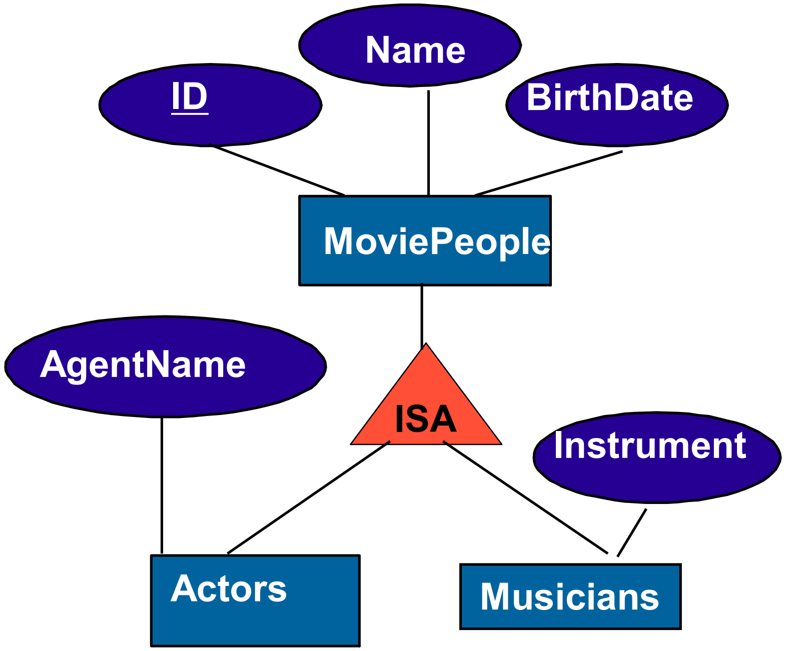
Cities(cname, **pname**, mayor), Provinces(pname, premier)

Do we need to have a “not null” constraint on pname due to the total participation constraint?

A. Yes

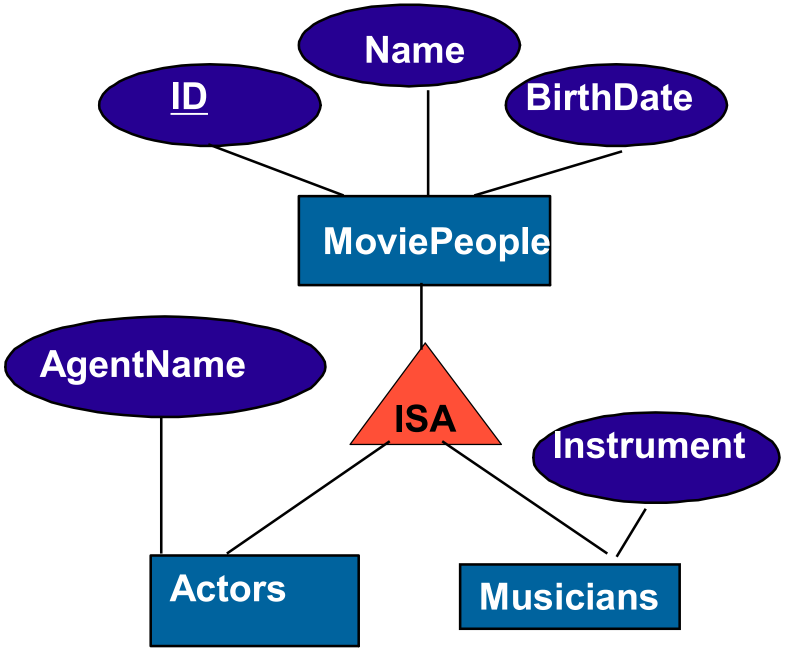
[✓ B. No]

Translating ISA Hierarchies to Relations



What is the best way to translate this into tables?

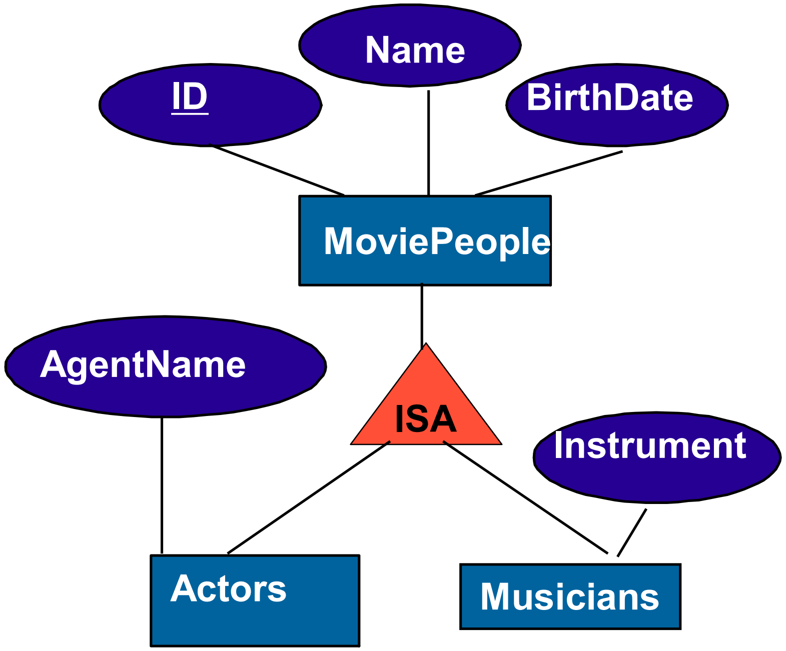
Totally unsatisfactory attempt: Safest but with lots of duplication (not in book)



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

- MoviePeople(ID, Name, BirthDate, AgentName, Instrument)
- Actors(ID, Name, BirthDate, AgentName, Instrument)
- Musicians(ID, Name, BirthDate, AgentName, Instrument)

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



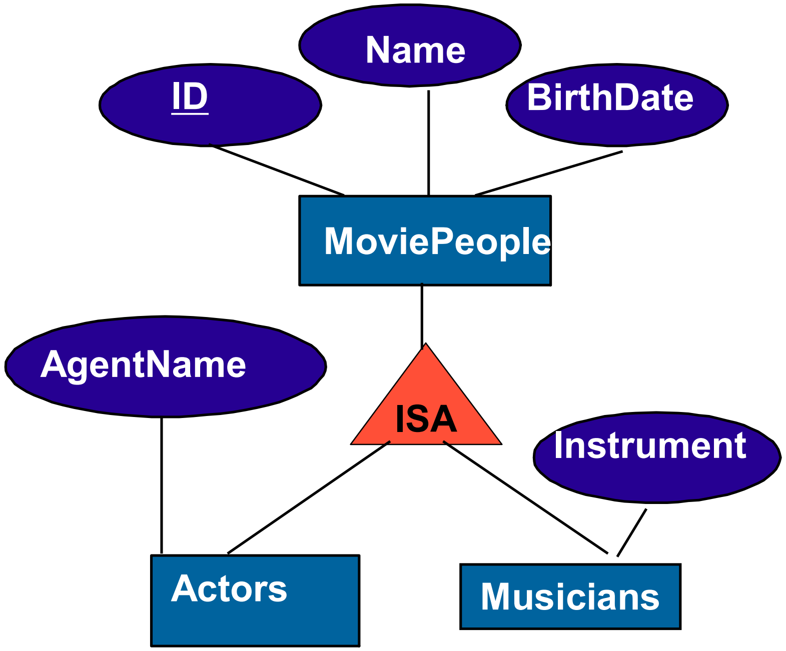
MoviePeople(ID, Name, BirthDate, AgentName, Instrument)

~~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~~)

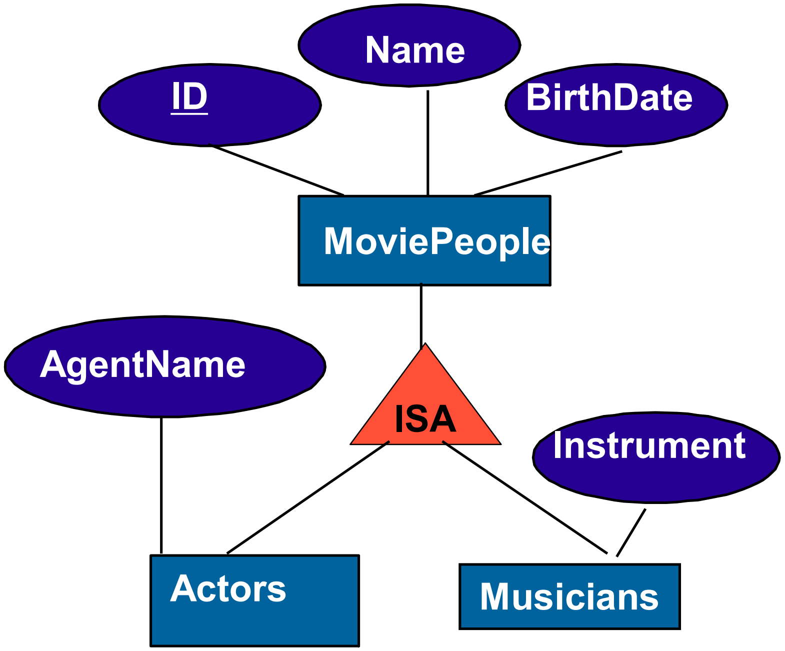
Actors(**ID**, ~~Name, BirthDate~~, AgentName, ~~Instrument~~)

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

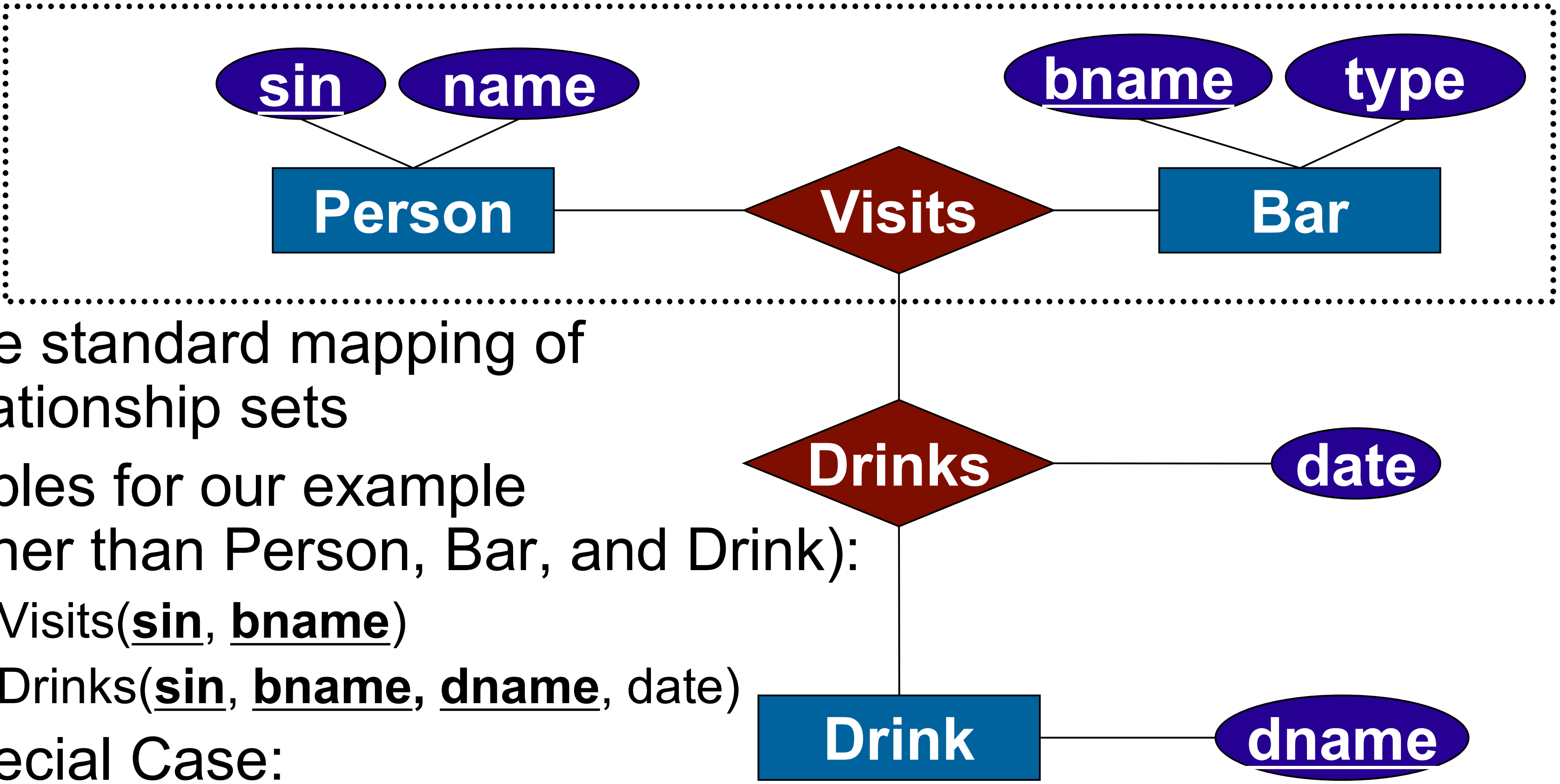
Musicians(ID, Name, BirthDate, ~~AgentName~~, Instrument)

☒Works poorly with relationships to superclass

☒ If ISA-relation is partial, it cannot be applied (loose entities)

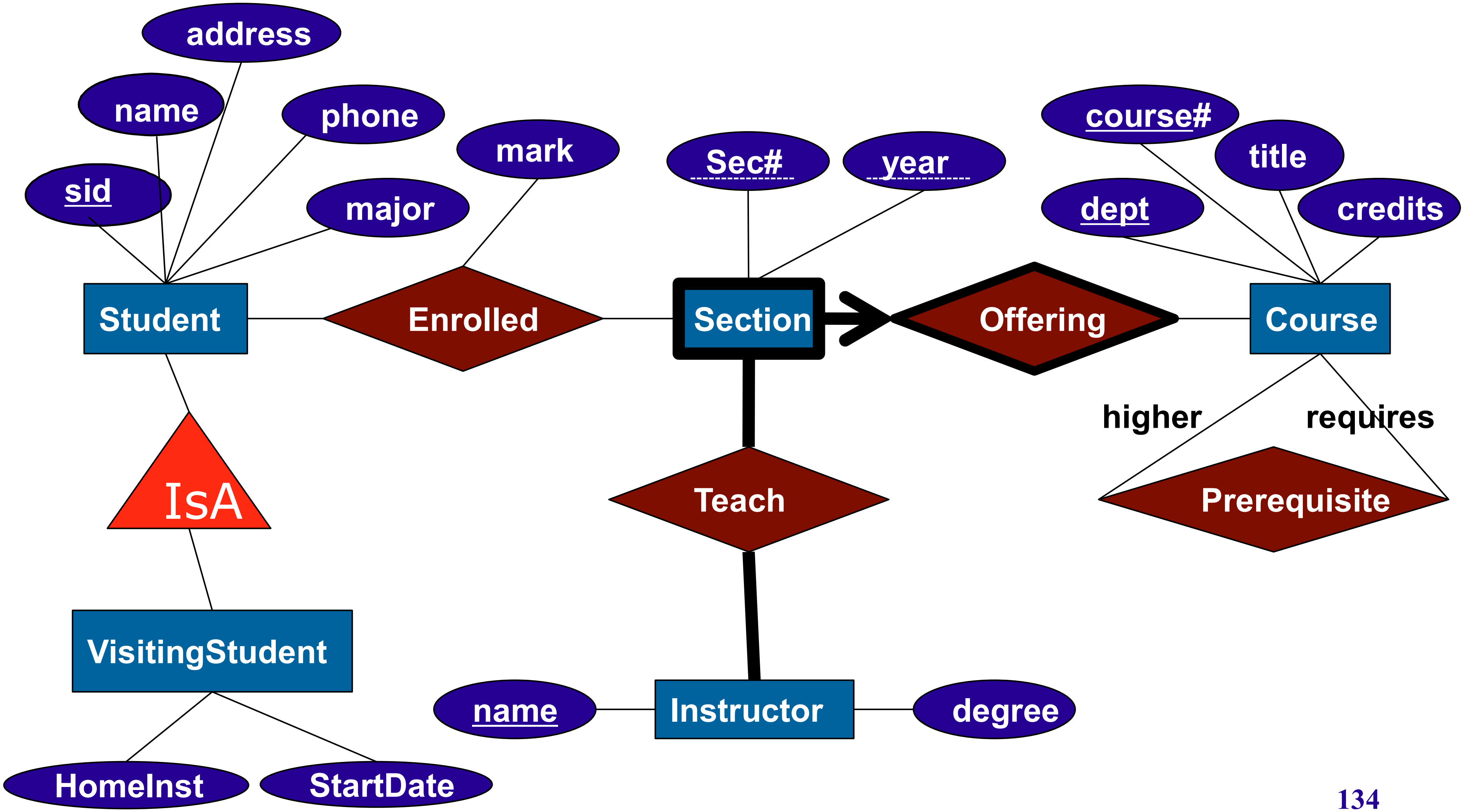
☒ If ISA-relation is not disjoint, it duplicates info

Translating Aggregation



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

Consider the following diagram for a university. List the tables, keys, and foreign keys when converted to relational. Do not write SQL DDL.



Sample ER to Relational Solution (foreign keys are bolded)

- *Student (sid, name, address, phone, major)*
- *VisitStudent (**sid**, homeInst, startDate)*
- *Course (dept, course#, title, credits)*
- *Instructor(insName, degree)*
- *SectionOffering*(**dept**, **course#**, sec#, year)
- *Teach(**dept, course#, sec#, year**, **insName**)*
  - Total participation constraint cannot be enforced for now
- *Enrolled (**sid, dept, course#, sec#, year**, mark)*
- *Prerequisite (**courseDept , course#, preDept, pre#**)*

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.