CMSC424: Database Design
Entity-Relationship Model

Instructor: Amol Deshpande
amol@cs.umd.edu

Outline

- Database Design Process
- Entity-relationship Model (E/R model)
- Converting from E/R to Relational
- Extra slides

Database Design Process

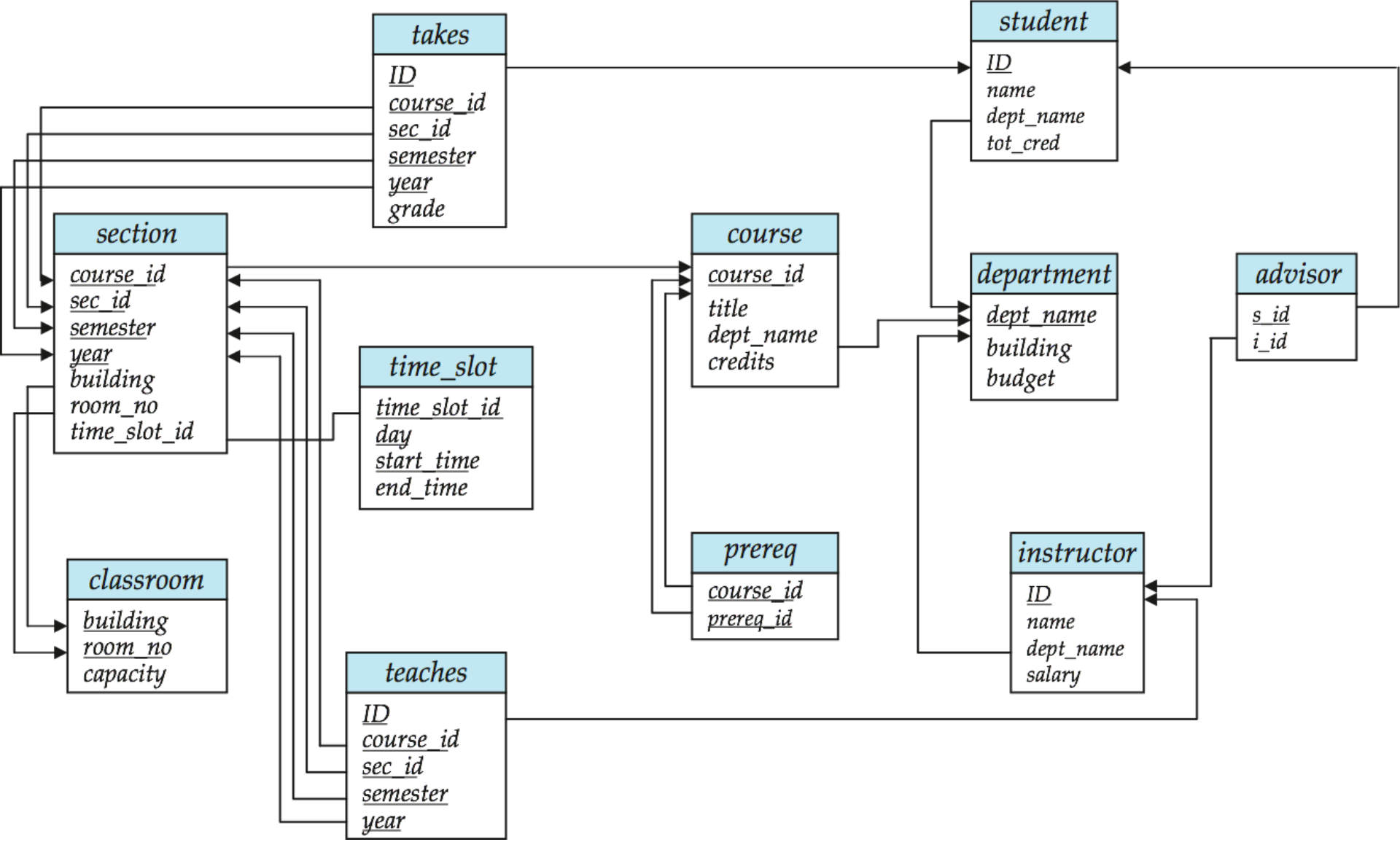
- Why?
  - Difficult to directly create schemas for complex domains
  - Need significant back-and-forth between the developer and the users
- Common Steps:
  - Initial design: Characterize the data needs of the users, including functional requirements (what types of queries/transactions)
  - Choose a data model appropriate for the data needs
  - Translate the requirements into a "conceptual schema"
  - Logical Design Step: Convert to the logical schema, typically relational
  - Physical Design Steps: Decide physical layout of the database
- Normalization (covered later) also deals with this issue

Outline

- Database Design Process
- Entity-relationship Model (E/R model)
- Converting from E/R to Relational
- Extra slides

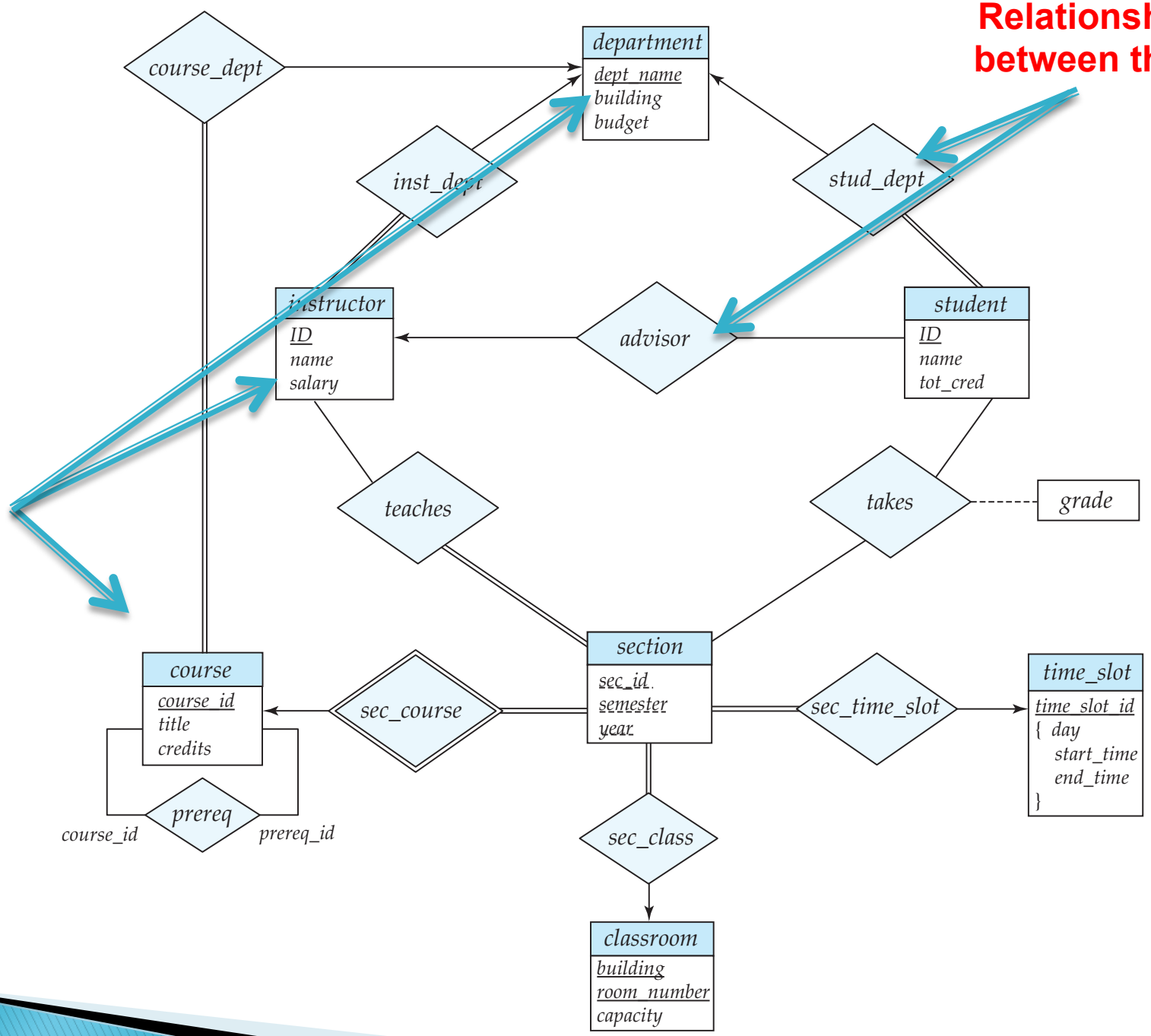
Entity-Relationship Model

- Conceptual schema often done in the E/R Model
- Why?
  - Why not just use the relational model directly?
  - Relational model too impoverished
    - Hard to understand what's going on
    - No distinction between different types of entities or relationships
      - Everything is a table
    - Too much detail
- E/R models have an associated diagrammatic representation
  - Easier to work with in the initial design phases
- At the end: easy to convert to a relational schema (almost mechanical)



- Key entities and “relationships” between them, all mixed up.
- Attributes appearing multiple times
- Complicated foreign keys

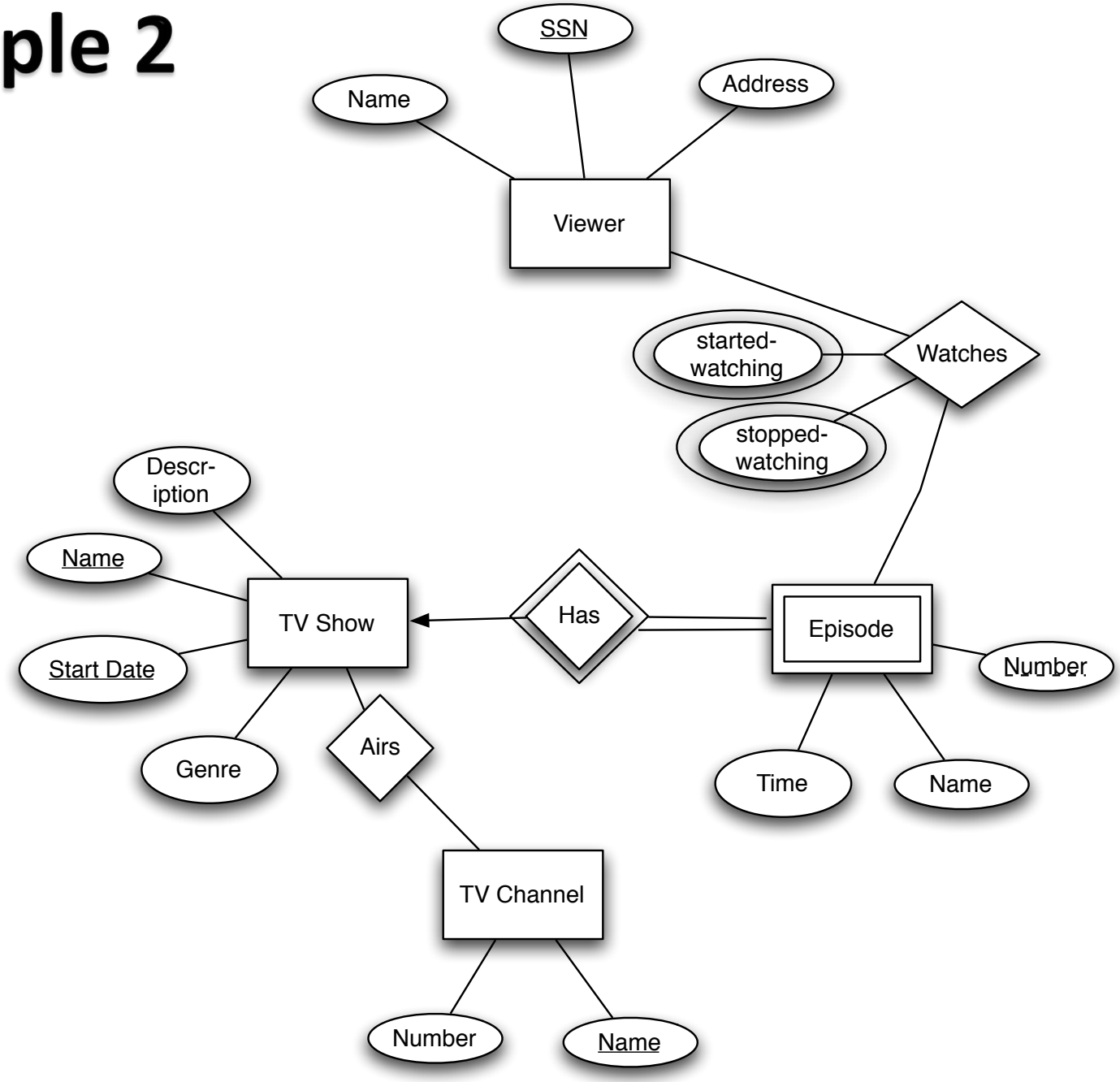
VS.


Relationships between them
Entities
course_dept
department
dept_name
building
budget
inst_dept
stud_dept
instructor
ID
name
salary
advisor
student
ID
name
tot_cred
teaches
takes
grade
section
sec_id
semester
year
course
course_id
title
credits
sec_course
sec_time_slot
time_slot
time_slot_id
{ day
start_time
end_time
}
prereq
course_id
prereq_id
sec_class
classroom
building
room_number
capacity

Example 1

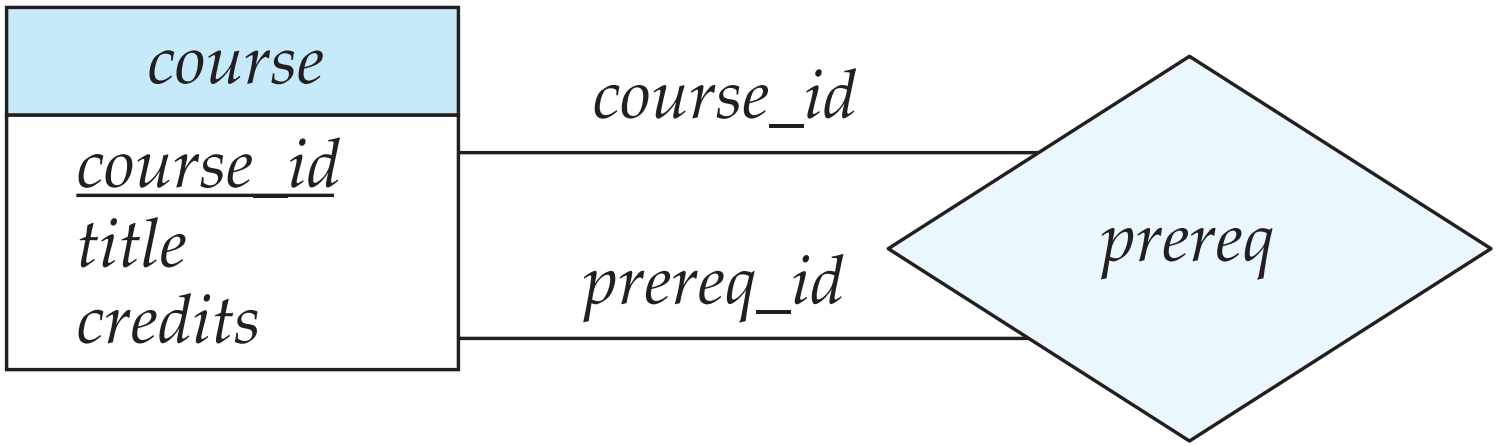
- Let's consider a application like AirBnB
- So we have:
  - Properties
  - Owners
  - Customers
  - Stays

Example 2


SSN
Name
Address
Viewer
started-watching
Watches
stopped-watching
Descr-iption
Name
TV Show
Has
Episode
Start Date
Number
Genre
Airs
Time
Name
TV Channel
Number
Name

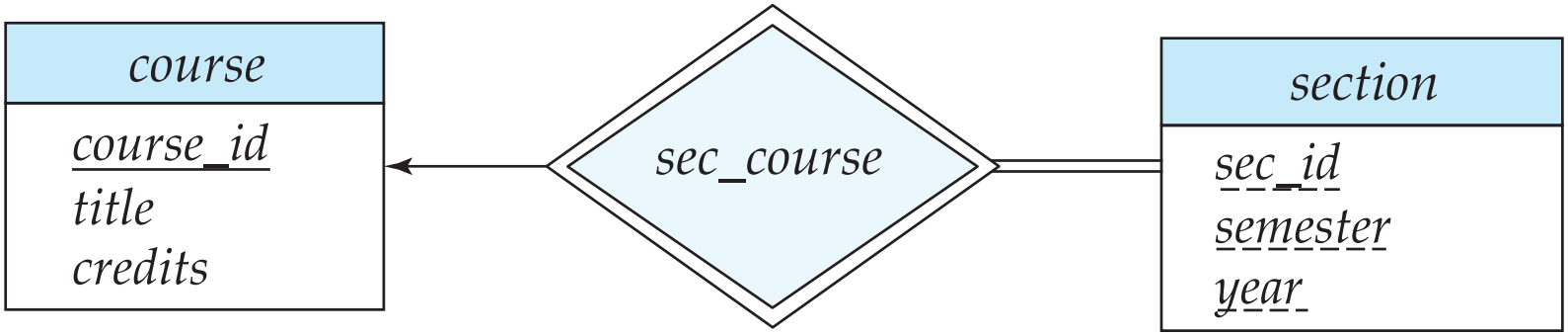
Recursive Relationships

- Sometimes a relationship associates an entity set to itself
- Need "roles" to distinguish



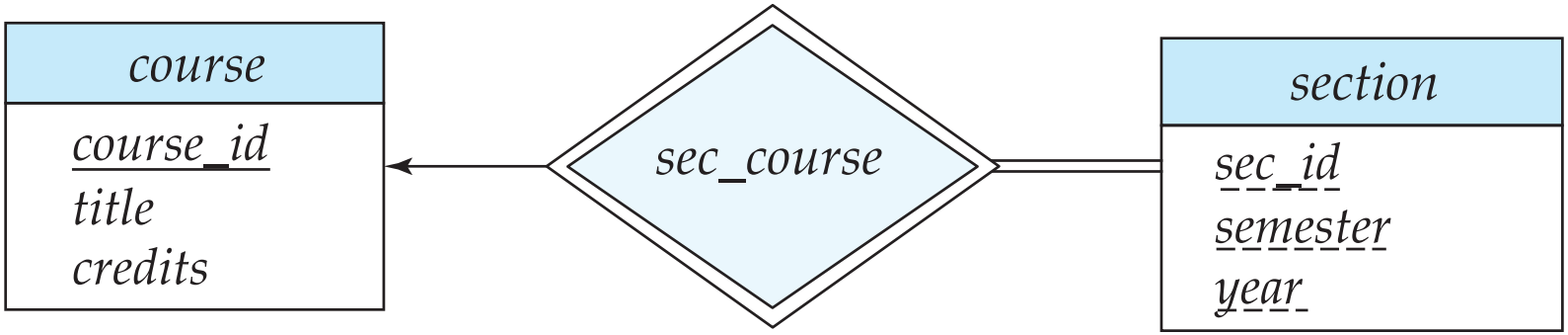
Weak Entity Sets

- An entity set without enough attributes to have a primary key
- E.g. Section Entity
- Still need to be able to distinguish between weak entities
  - Called "discriminator attributes": dashed underline



Participation Constraints

- Records the information that any entity in an entity set must participate in at least one relationship of that type



Specialization/Generalization

Similar to object-oriented programming: allows inheritance etc.

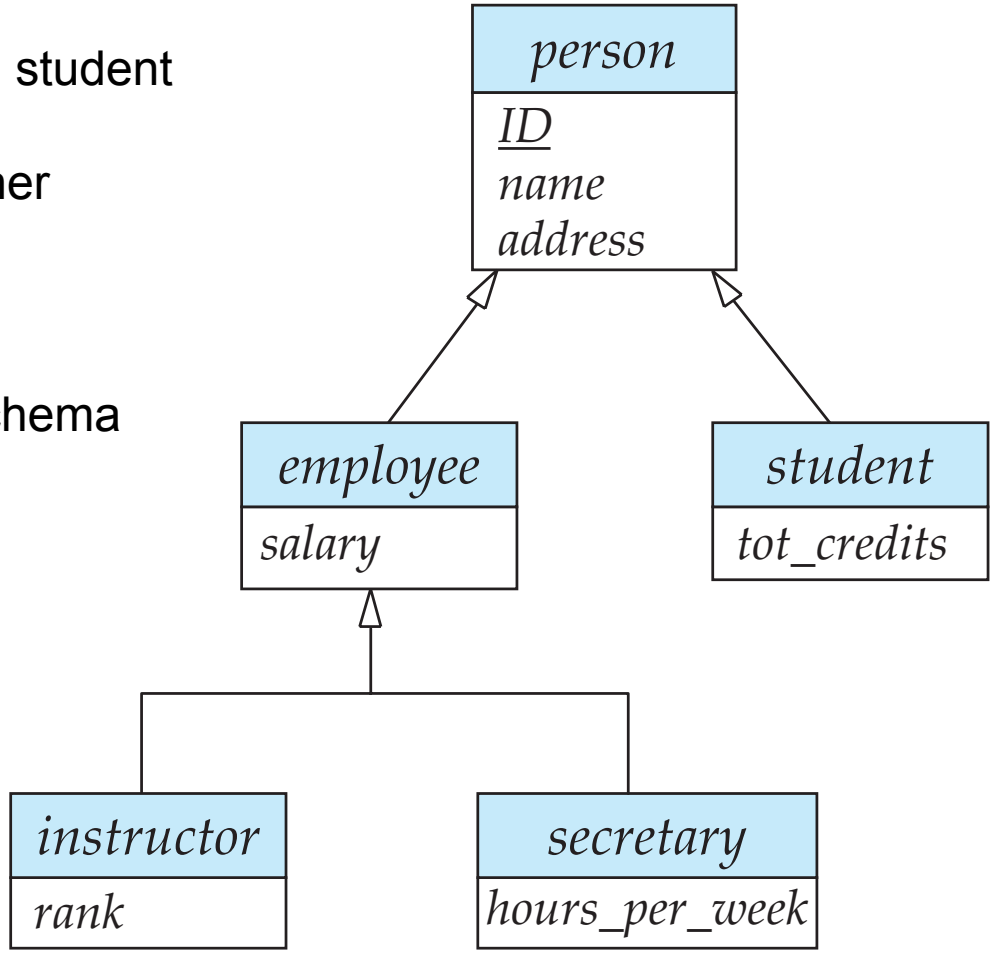
Disjoint vs Overlapping:

No person can be both employee and student

Partial vs Total

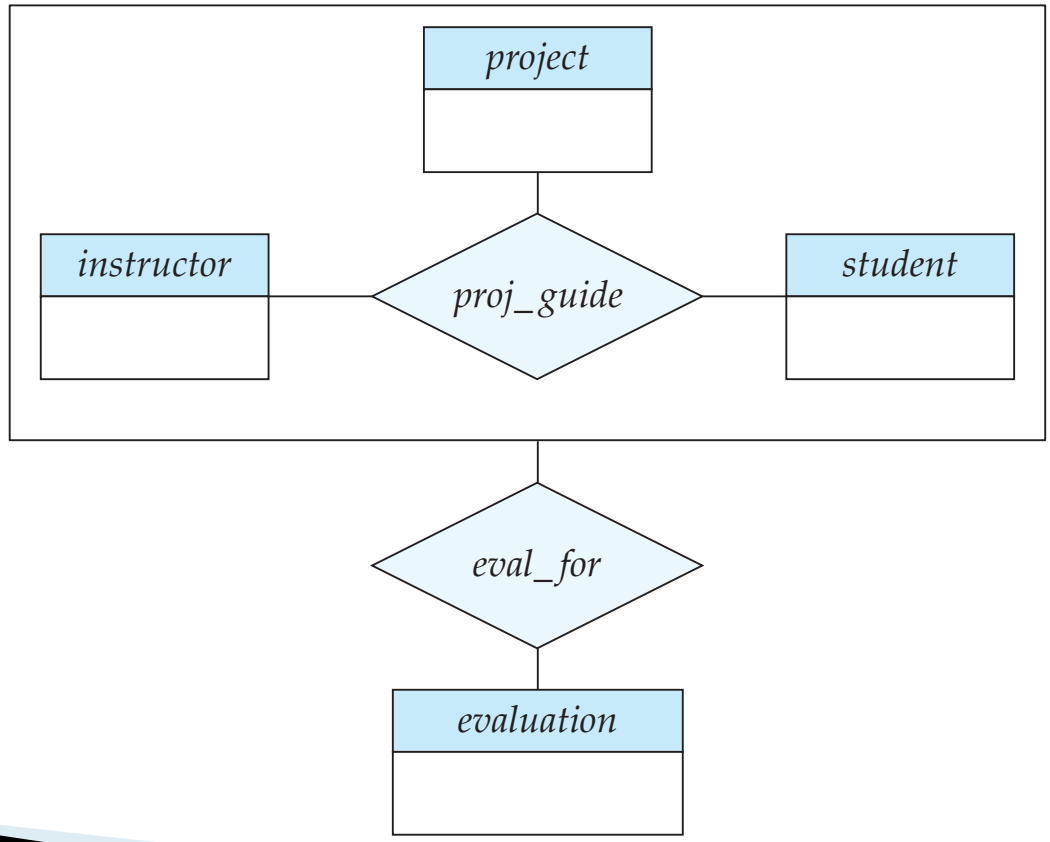
There may be “Persons” who are neither employee or student

Different ways to convert to a Relational schema based on the above issues



Aggregation

- No relationships allowed between relationships
- Suppose we want to record evaluations of a student by a guide on a project



Thoughts…

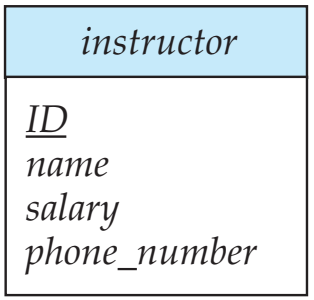
- Nothing about actual data
  - How is it stored ?
- No talk about the query languages
  - How do we access the data ?
- Semantic vs Syntactic Data Models
  - Remember: E/R Model is used for conceptual modeling
  - Many conceptual models have the same properties
- They are much more about representing the knowledge than about database storage/querying

Thoughts…

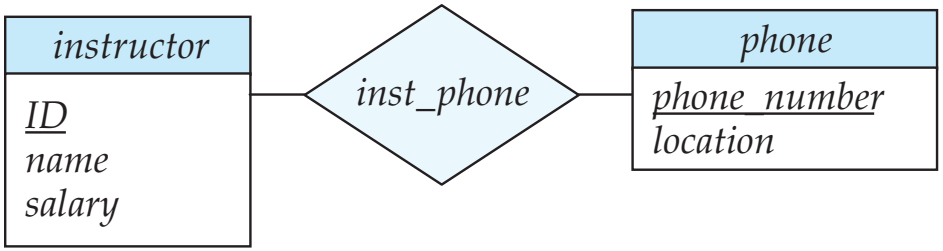
- Basic design principles
  - Faithful
    - Must make sense
  - Satisfies the application requirements
  - Models the requisite domain knowledge
    - If not modeled, lost afterwards
  - Avoid redundancy
    - Potential for inconsistencies
  - Go for simplicity
- Typically an iterative process that goes back and forth

Design Issues

- Entity sets vs attributes
  - Depends on the semantics of the application
  - Consider *telephone*



(a)



(b)

Design Issues

- Entity sets vs Relationsihp sets
  - Consider *takes*

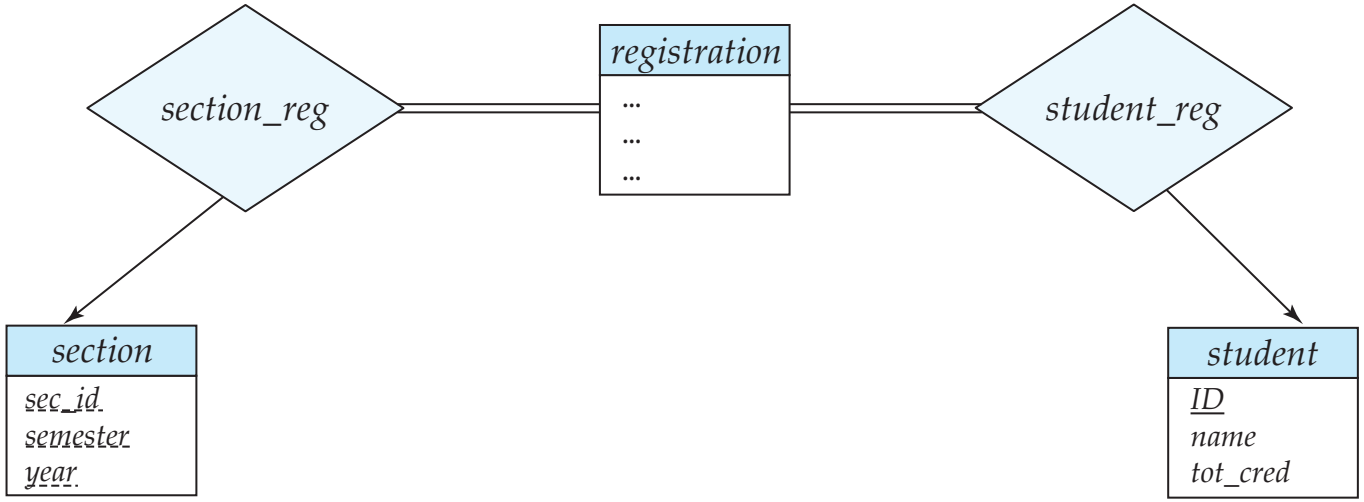


Figure 7.18 Replacement of *takes* by *registration* and two relationship sets

Design Issues

- Entity sets vs attributes
  - Depends on the semantics of the application
  - Consider *telephone*
- Entity sets vs Relationsihp sets
  - Consider *loan*
- N-ary vs binary relationships
  - Possible to avoid n-ary relationships, but there are some cases where it is advantageous to use them
- It is not an exact science !!

Recap

- Entity-relationship Model
  - Intuitive diagram-based representation of domain knowledge, data properties etc…
  - Two key concepts:
    - Entities
    - Relationships
  - We also looked at:
    - Relationship cardinalities
    - Keys
    - Weak entity sets
    - …

Recap

- Entity-relationship Model
  - No standardized model (as far as I know)
    - You will see different types of symbols/constructs
  - Easy to reason about/understand/construct
  - Not as easy to implement
    - Came after the relational model, so no real implementation was ever done
    - Mainly used in the design phase

Django: Overview

- Web application framework written in Python
- Uses a Model-Template-View pattern
- Very similar to the Model-View-Controller pattern that others (e.g., Ruby on Rails) use

- The slides we covered are from an old talk on Django by Simon Willison, a co-creator of Django
  - The talk is from 2006, but mostly still seems correct
  - http://www.slideshare.net/simon/the-django-web-application-framework

Project

- Basic skeleton already created for you
- You have to change some of the files
- Separately, generalize the E/R model that is provided