


# 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

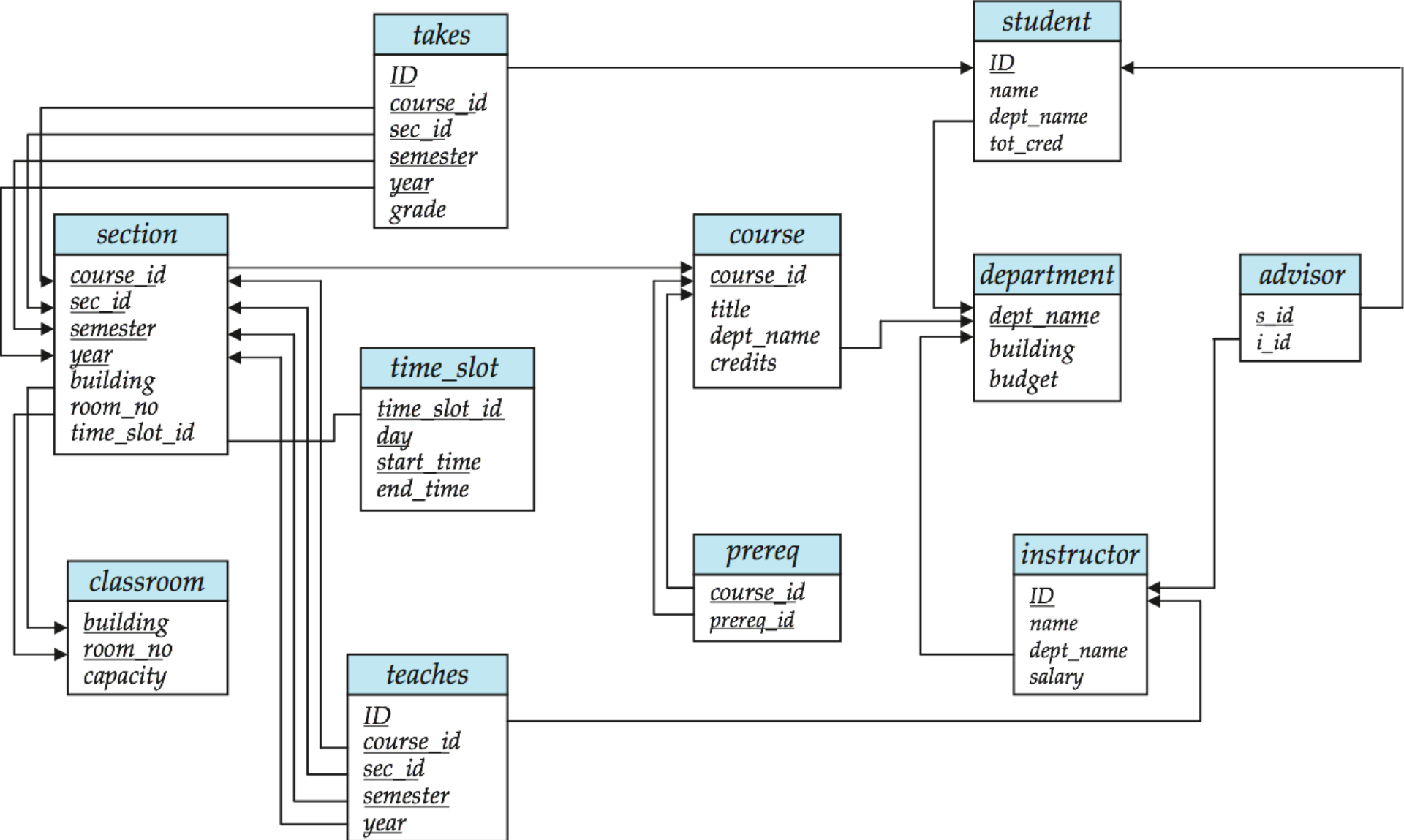
## ▶ Normalization (covered later) also deals with this issue

# 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)

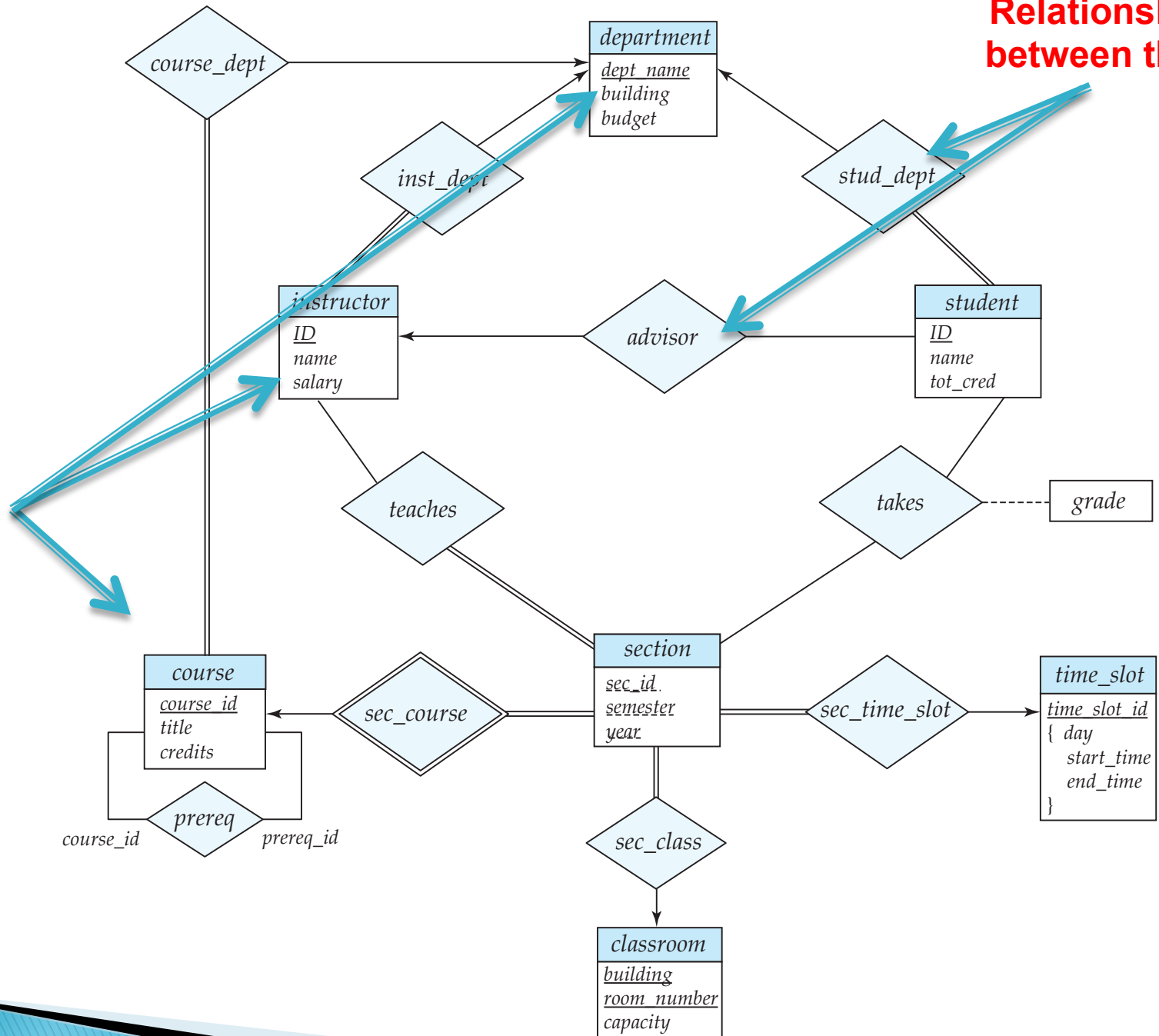


- Key entities and “relationships” between them, all mixed up.
  - Attributes appearing multiple times
  - Complicated foreign keys

VS.

Relationships  
between them

Entities



# Entity-Relationship Model

## ▶ Two key concepts

### ◦ Entities:

- An object that *exists* and is *distinguishable* from other objects
  - Examples: Bob Smith, BofA, CMSC424
- Have attributes (people have names and addresses)
- Form entity sets with other entities of the same type that share the same properties
  - Set of all people, set of all classes
- Entity sets may overlap
  - Customers and Employees

# Entity-Relationship Model

## ▶ Two key concepts

### ◦ Relationships:

- Relate 2 or more entities
  - E.g. Bob Smith has account at College Park Branch
- Form relationship sets with other relationships of the same type that share the same properties
  - Customers have accounts at Branches
- Can have attributes:
  - has account at may have an attribute *start-date*
- Can involve more than 2 entities
  - Employee *works at* Branch *at* Job

# Entities and relationships

## Two Entity Sets

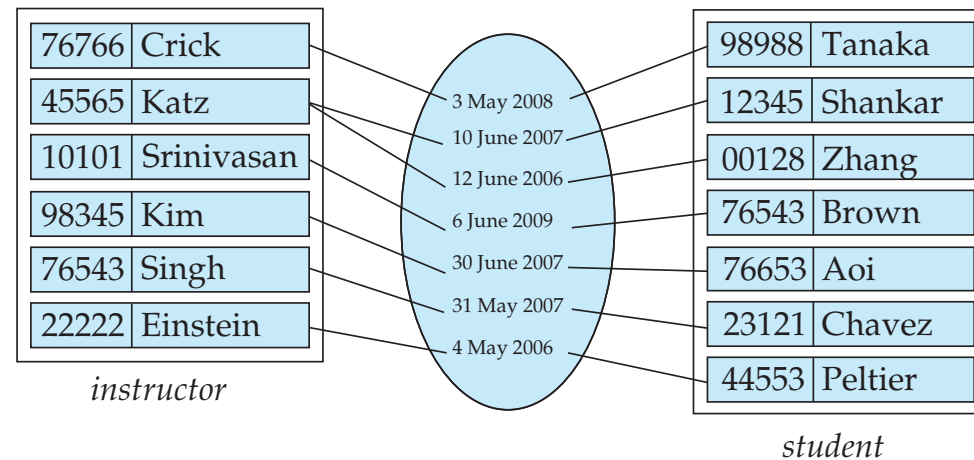
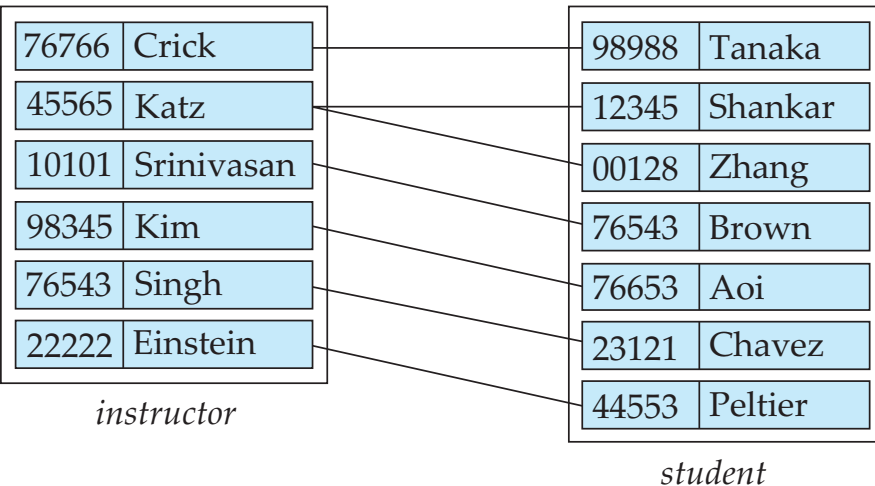
76766	Crick
45565	Katz
10101	Srinivasan
98345	Kim
76543	Singh
22222	Einstein

*instructor*

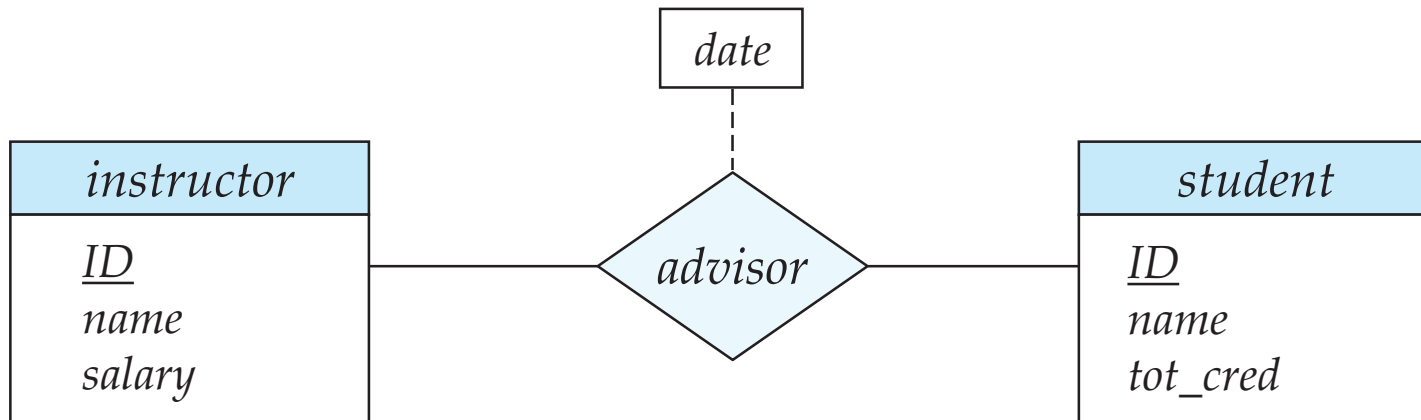
98988	Tanaka
12345	Shankar
00128	Zhang
76543	Brown
76653	Aoi
23121	Chavez
44553	Peltier

*student*

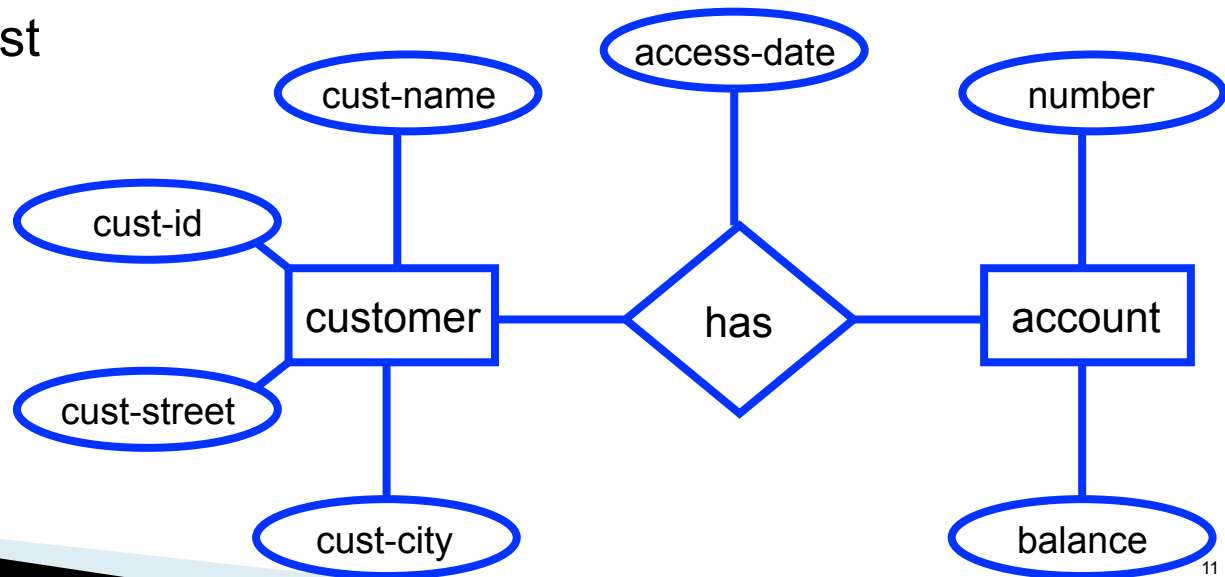
## Advisor Relationship, with and without attributes



# ER Diagram



Alternative representation,  
used in the book in the past



# Rest of the class

- ▶ Details of the ER Model
  - How to represent various types of constraints/semantic information etc.
- ▶ Design issues
- ▶ A detailed example

# Next: Relationship Cardinalities

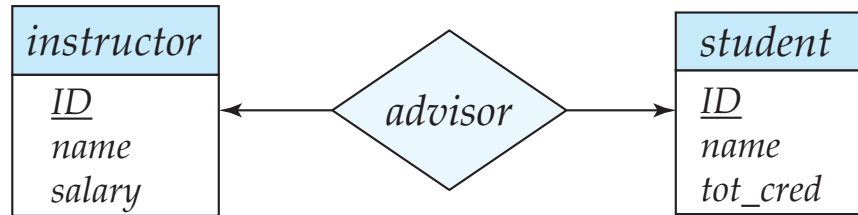
- ▶ We may know:
  - One student can only have one advisor
  - OR
  - One student can have multiple advisors
- ▶ Representing this is important
- ▶ Why ?
  - Better manipulation of data
    - If former, can store the advisor info in the student table
  - Can enforce such a constraint
    - Application logic will have to do it; NOT GOOD
  - Remember: If not represented in conceptual model, the domain knowledge may be lost

# Mapping Cardinalities

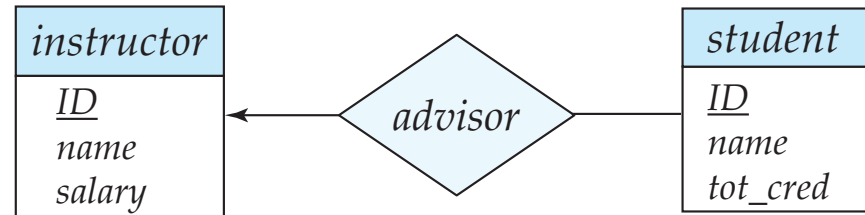
- ▶ Express the number of entities to which another entity can be associated via a relationship set
- ▶ Most useful in describing binary relationship sets

# Mapping Cardinalities

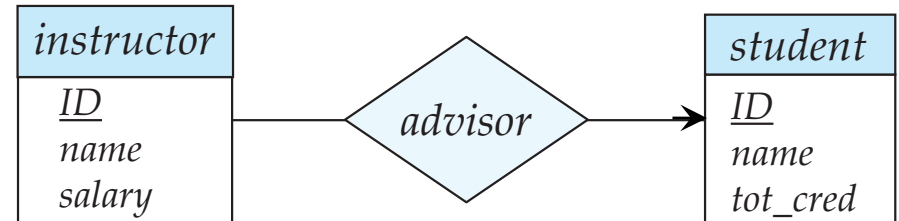
- ▶ One-to-One



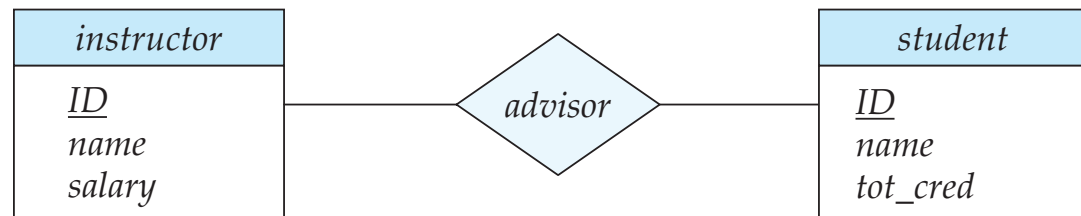
- ▶ One-to-Many



- ▶ Many-to-One



- ▶ Many-to-Many



# Mapping Cardinalities

- ▶ Express the number of entities to which another entity can be associated via a relationship set
- ▶ Most useful in describing binary relationship sets
- ▶ N-ary relationships ?
  - More complicated
  - Details in the book

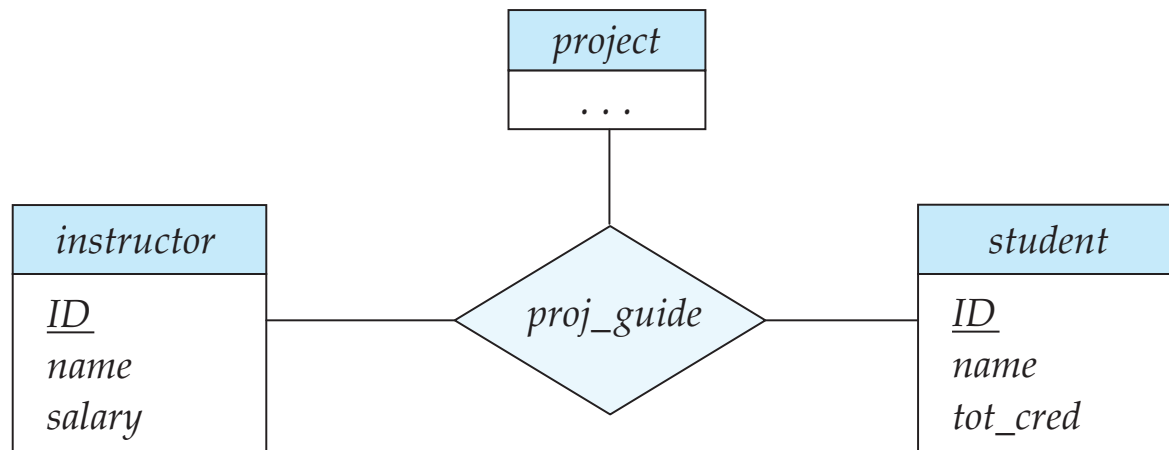
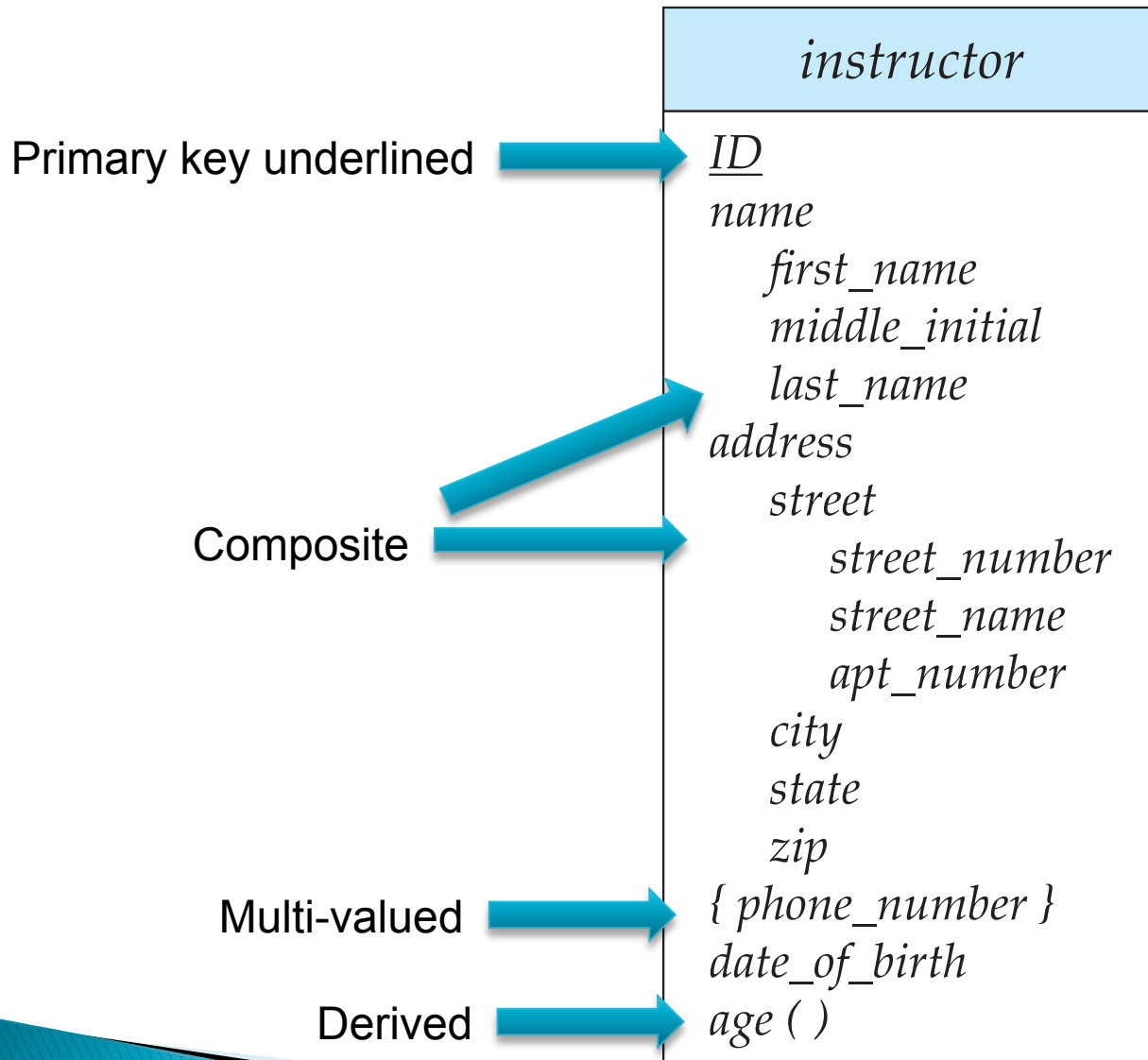


Figure 7.13 E-R diagram with a ternary relationship.

# Next: Types of Attributes

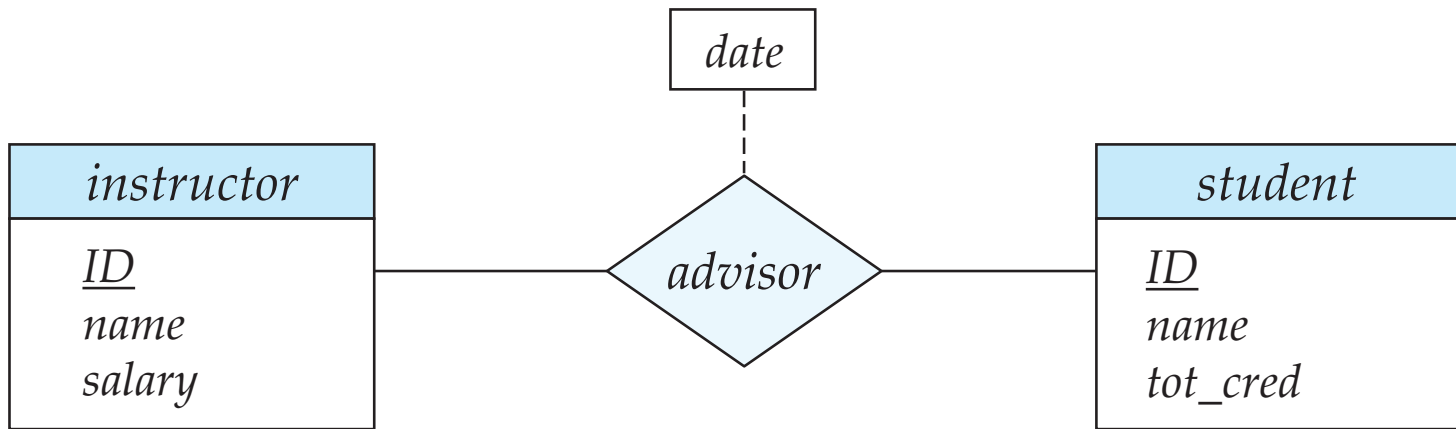
- ▶ Simple vs Composite
  - Single value per attribute ?
- ▶ Single-valued vs Multi-valued
  - E.g. Phone numbers are multi-valued
- ▶ Derived
  - If date-of-birth is present, age can be derived
  - Can help in avoiding redundancy, enforcing constraints etc...

# Types of Attributes



# Relationship Set Keys

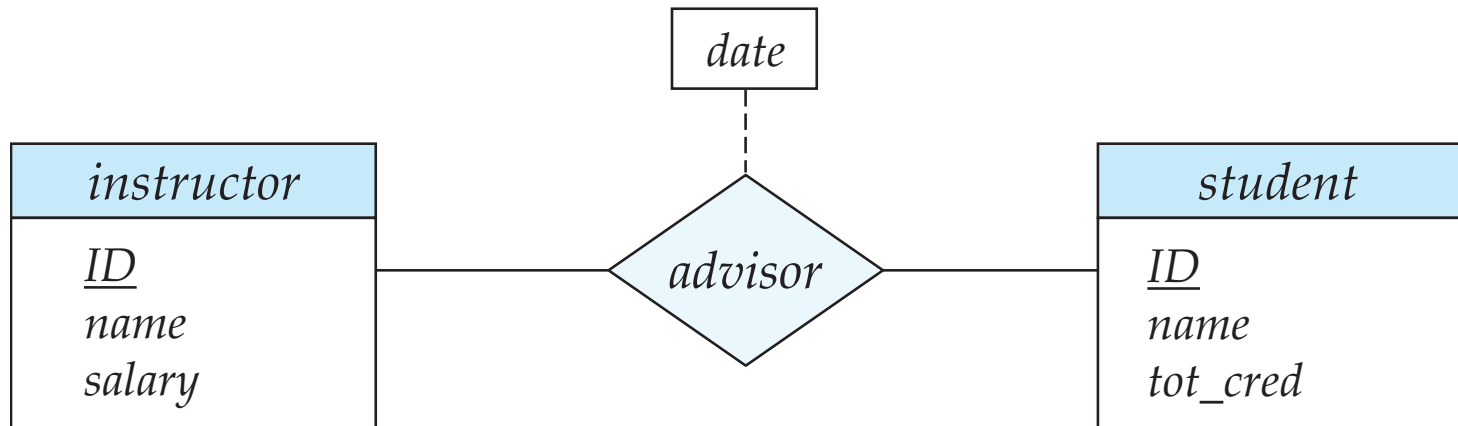
- ▶ What attributes are needed to represent a relationship completely and uniquely ?
  - Union of primary keys of the entities involved, and relationship attributes



- {instructor\_id, date, student\_id} describes a relationship completely

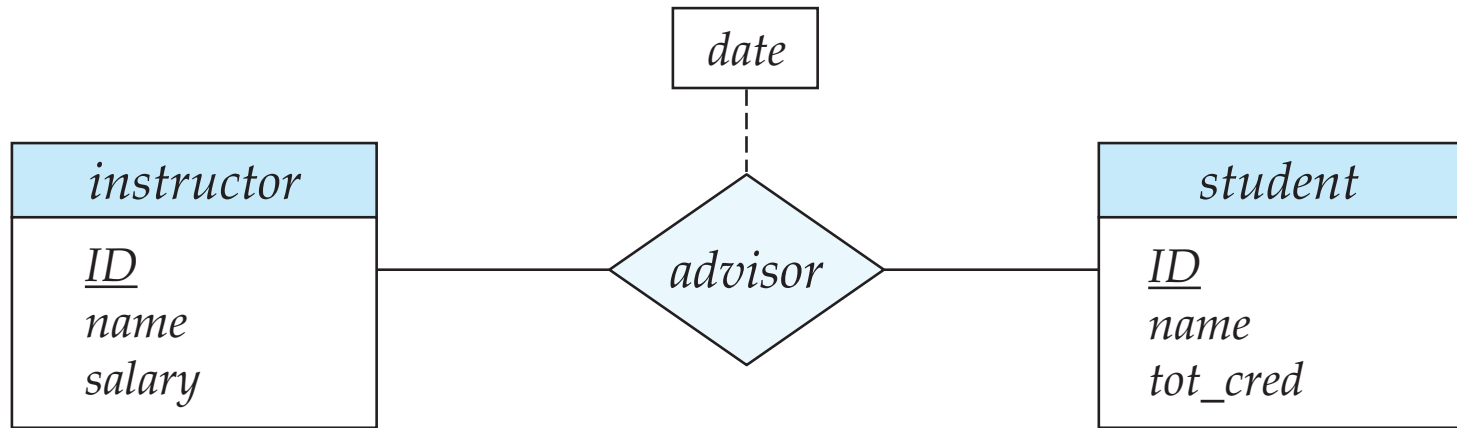
# Relationship Set Keys

- ▶ Is  $\{instructor\_id, date, student\_id\}$  a candidate key ?
  - No. Attribute *date* can be removed from this set without losing key-ness
    - By definition: there can only be one relationship of a given type between two different entities
    - There may be two different relationships between two entities
      - e.g., (instructor is member of a department) and (instructor is “chair” of a department)
  - In fact, union of primary keys of associated entities is always a superkey



# Relationship Set Keys

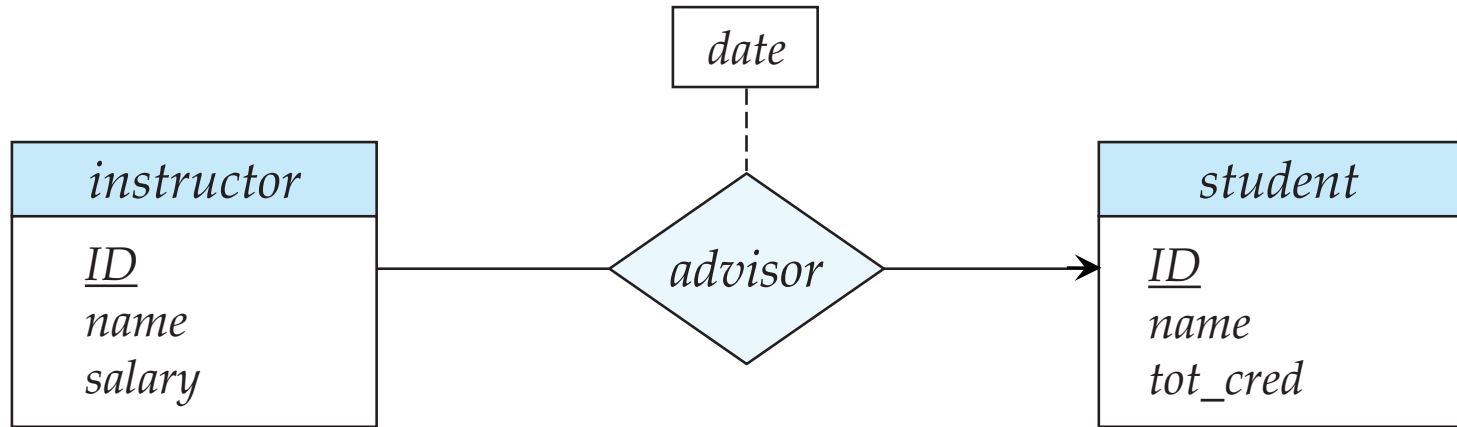
- ▶ Is {instructor-id, student-id} a candidate key ?
  - Depends



- If one-to-one relationship, either {*student\_id*} or {*instructor\_id*} sufficient
  - Since a given *student* can only have one *advisor*, she can only participate in one relationship
  - Ditto *instructor*

# Relationship Set Keys

- ▶ Is {instructor-id, student-id} a candidate key ?
  - Depends



- If one-to-many relationship (as shown), {*instructor\_id*} is a primary key
  - A given instructor can only have one advisee, but a student may have many advisors
  - So the *instructor\_id* is sufficient to uniquely identify a relationship

# Relationship Set Keys

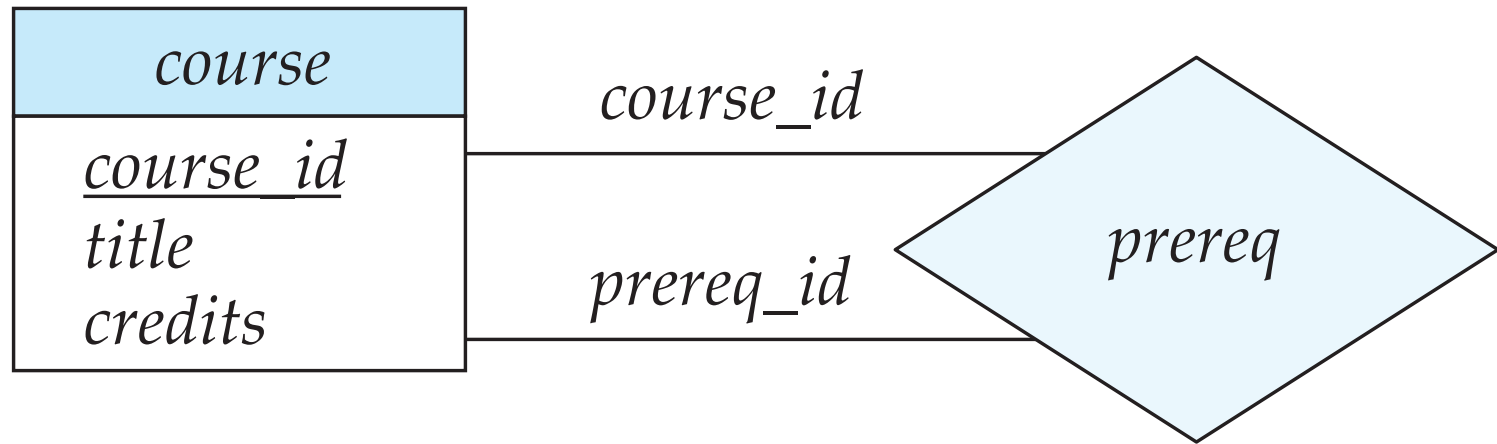
- ▶ General rule for binary relationships
  - one-to-one: primary key of either entity set
  - one-to-many: primary key of the entity set on the many side
  - many-to-many: union of primary keys of the associate entity sets
- ▶ n-ary relationships
  - More complicated rules



- ▶ What have we been doing
  - Try to record all the requirements and constraints
  
- ▶ Why ?
  - So we can design a faithful and easy-to-use schema
  
- ▶ Understanding this is important
  - Rest are details !!
  - That's what books/manuals are for.

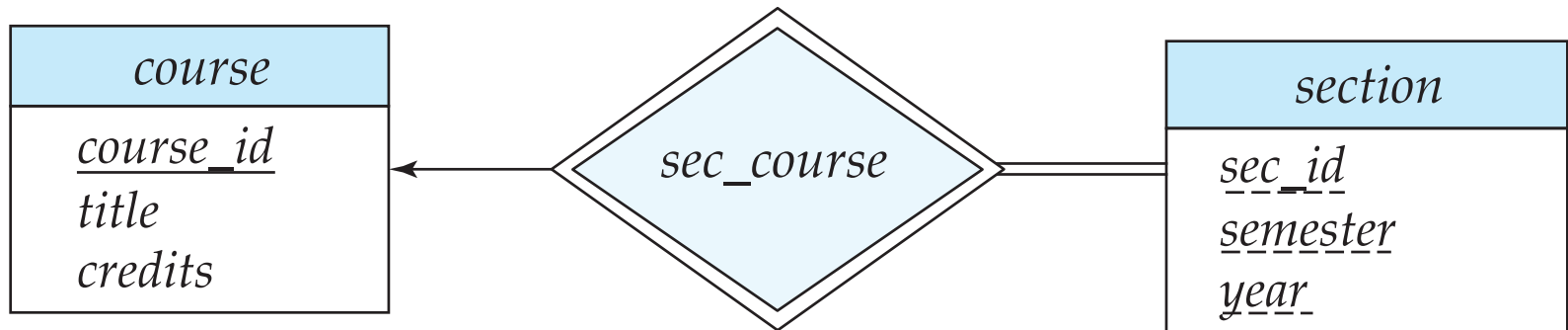
# Recursive Relationships

- ▶ Sometimes a relationship associates an entity set to itself
- ▶ Need “roles” to distinguish



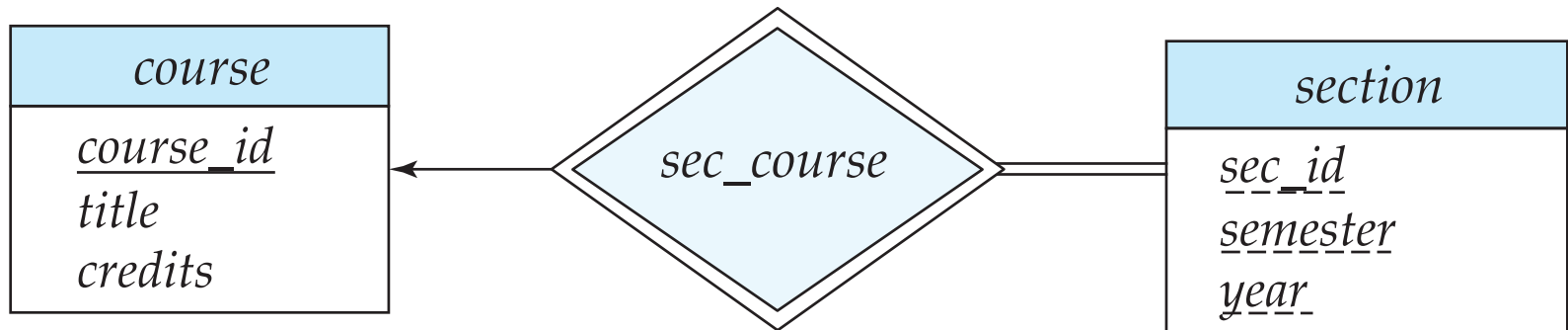
# Weak Entity Sets

- ▶ 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



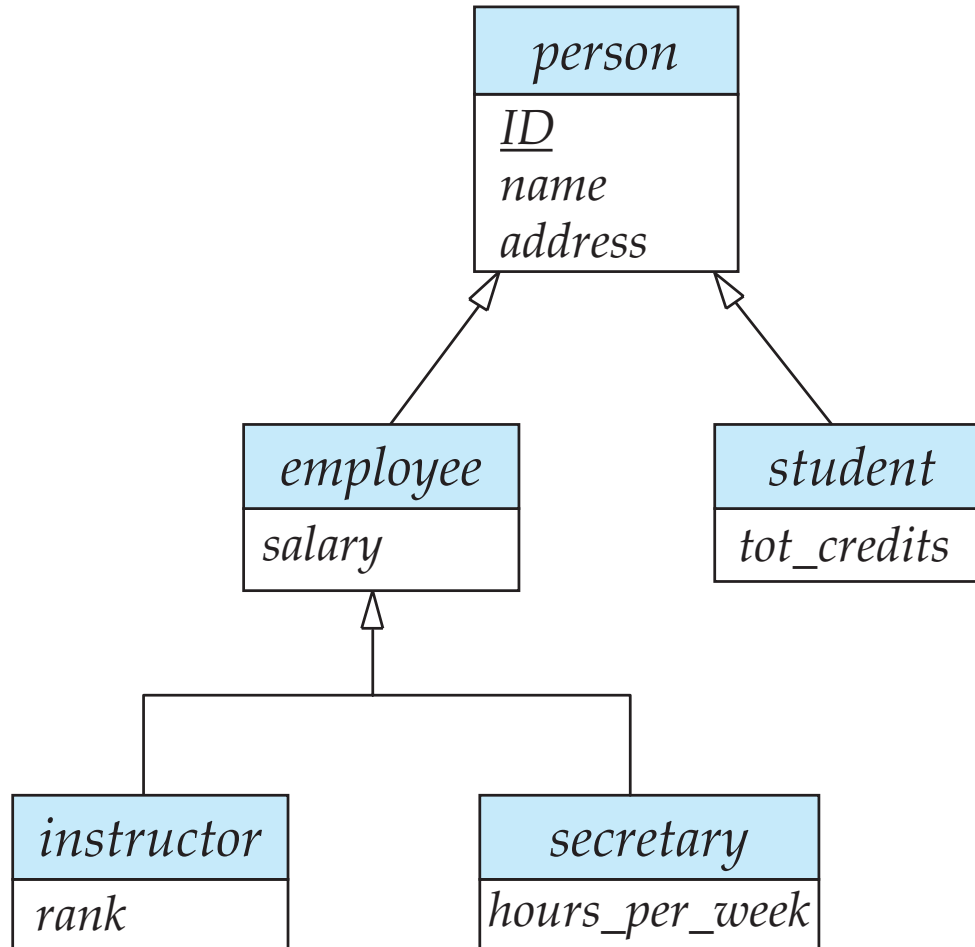
# Participation Constraints

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



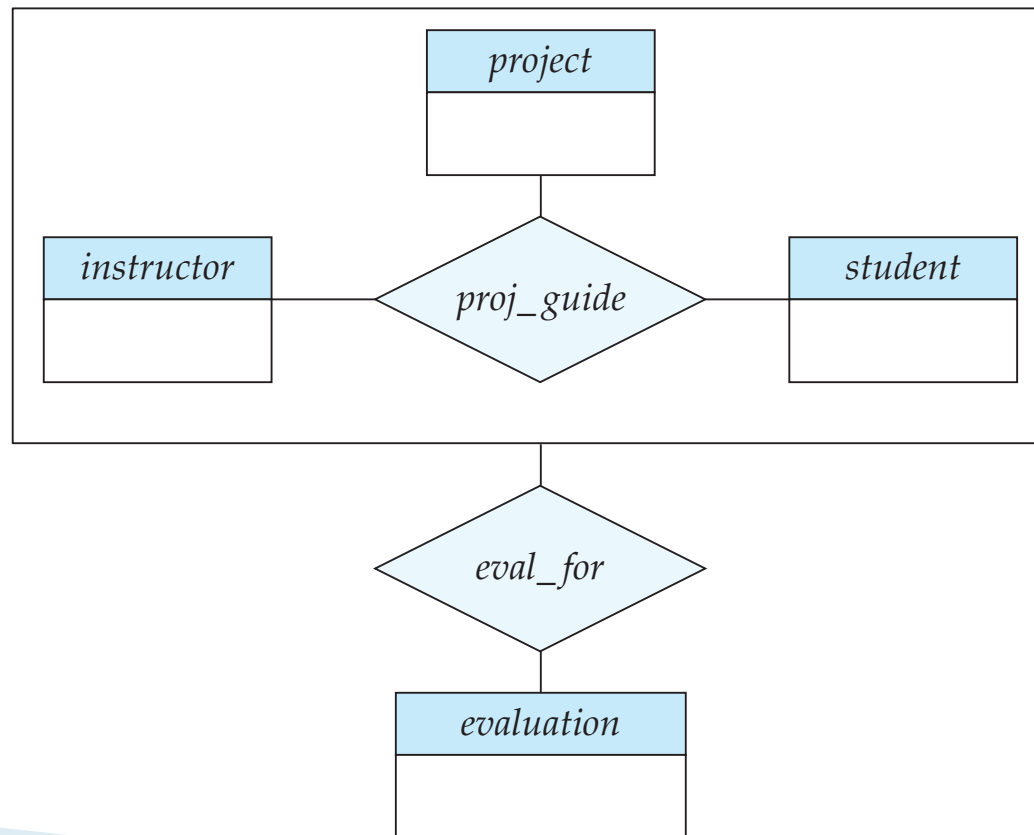
# Specialization/Generalization

Similar to object-oriented programming: allows inheritance etc.




# 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*
  - ▶ Entity sets vs Relationship 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

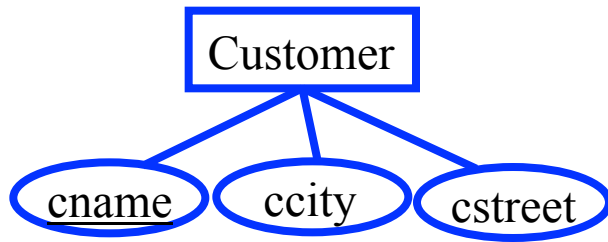
- No standardized model (as far as I know)
  - You will see different types of symbols/constructs
- Easy to reason about/understand/construct
- Not as easy to implement
  - Came after the relational model, so no real implementation was ever done
  - Mainly used in the design phase

# Outline

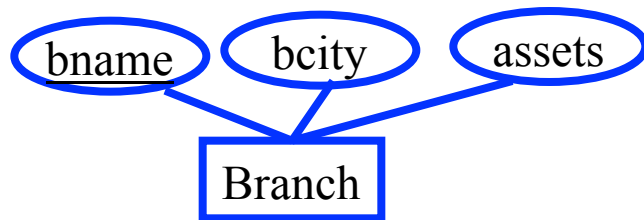
- ▶ Entity-relationship Model (E/R model)
- ▶ Converting from E/R to Relational
- ▶ Extra slides

# E/R Diagrams → Relations

- ▶ Convert entity sets into a relational schema with the same set of attributes



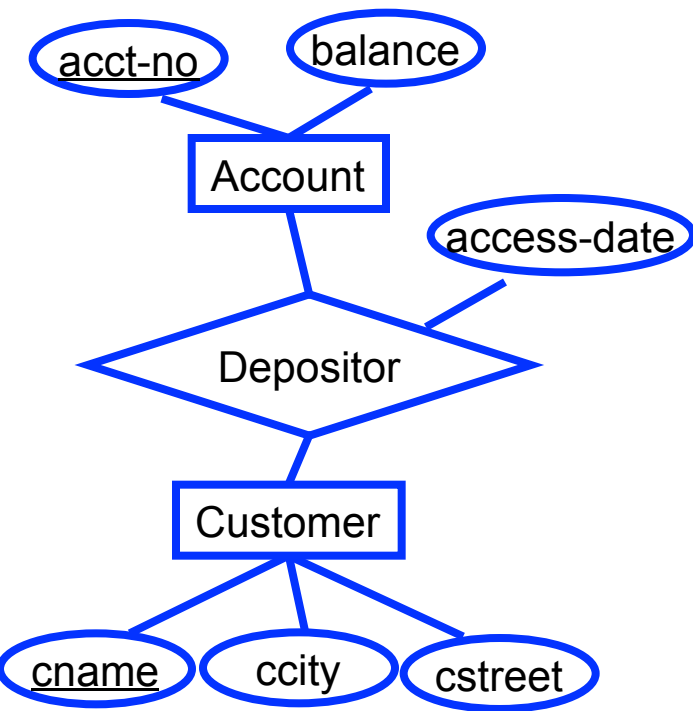
Customer\_Schema(cname, ccity, cstreet)



Branch\_Schema(bname, bcity, assets)

# E/R Diagrams → Relations

- ▶ Convert relationship sets *also* into a relational schema
- ▶ Remember: A relationship is completely described by primary keys of associate entities and its own attributes



Account\_Schema(acct-no, balance)

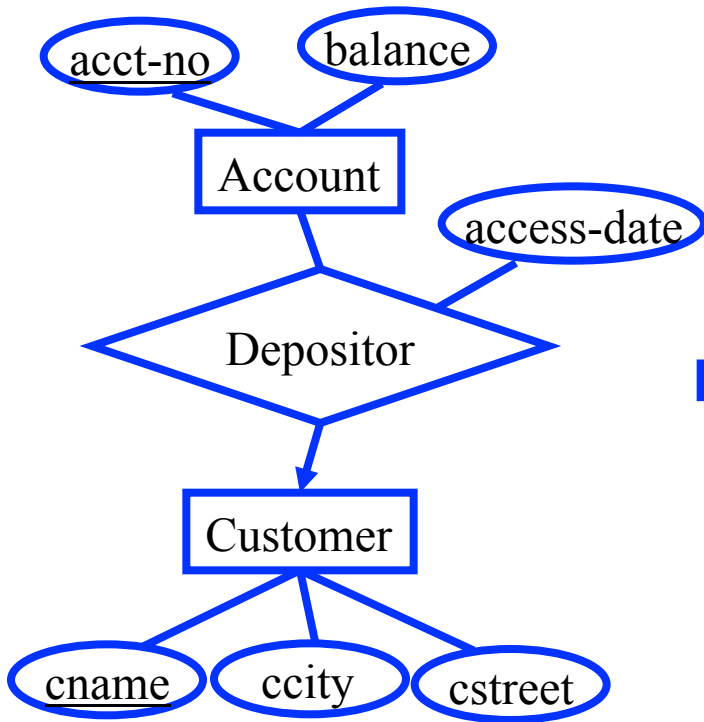
Depositor\_Schema(cname, acct-no, access-date)

Customer\_Schema(cname, ccity, cstreet)

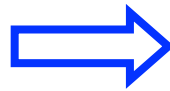
Well... Not quite. We can do better.  
It depends on the relationship cardinality

# E/R Diagrams → Relations

- ▶ Say One-to-Many Relationship from Customer to Account  
→ Many accounts per customer

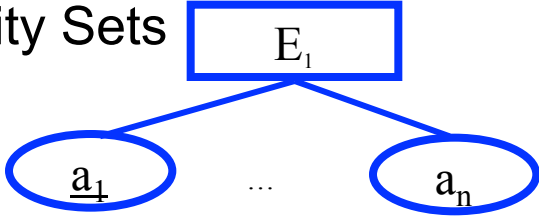
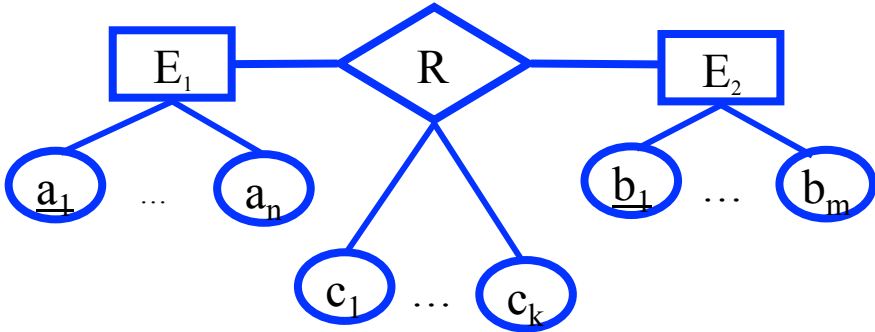


Account\_Schema(acct-no, balance, cname, access-date)



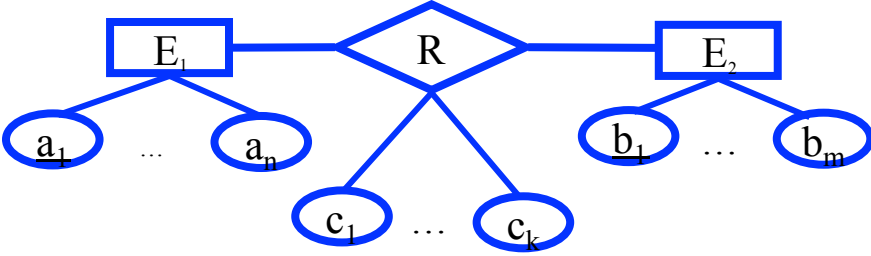




Customer\_Schema(cname, ccity, cstreet)

# E/R Diagrams → Relations

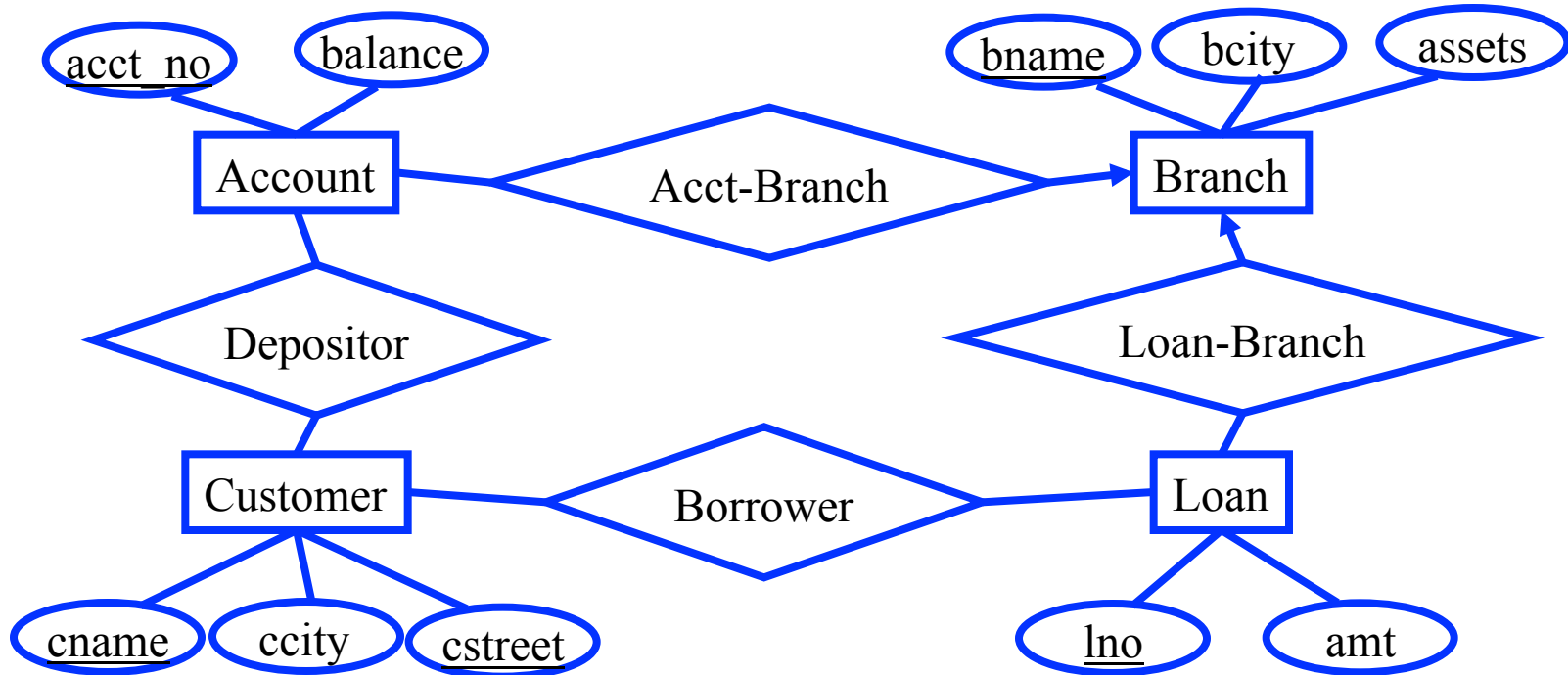
E/R	Relational Schema
<p>Entity Sets</p> 	$E = (\underline{a_1}, \dots, a_n)$
<p>Relationship Sets</p> 	$R = (\underline{a_1}, \underline{b_1}, c_1, \dots, c_n)$ <p><math>a_1</math>: <math>E_1</math>'s key <math>b_1</math>: <math>E_2</math>'s key <math>c_1, \dots, c_k</math>: attributes of R</p>

*Not the whole story for Relationship Sets ...*

# E/R Diagrams → Relations

Relationship Cardinality	Relational Schema
	
$n:m$ 	$E_1 = (\underline{a_1}, \dots, a_n)$ $E_2 = (\underline{b_1}, \dots, b_m)$ $R = (\underline{a_1}, \underline{b_1}, c_1, \dots, c_n)$
$n:1$ 	$E_1 = (\underline{a_1}, \dots, a_n, b_1, c_1, \dots, c_n)$ $E_2 = (\underline{b_1}, \dots, b_m)$
$1:n$ 	$E_1 = (\underline{a_1}, \dots, a_n)$ $E_2 = (\underline{b_1}, \dots, b_m, a_1, c_1, \dots, c_n)$
$1:1$ 	Treat as $n:1$ or $1:n$

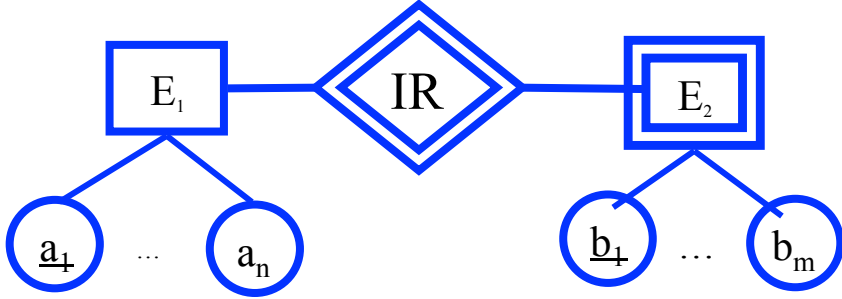
# Translating E/R Diagrams to Relations



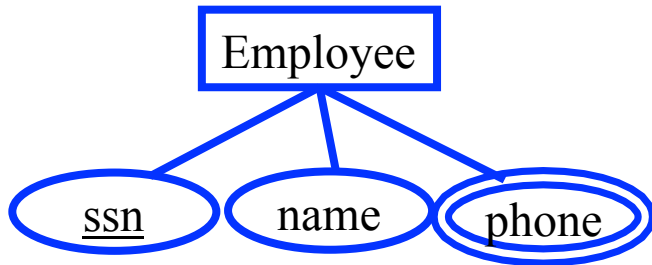
Q. How many tables does this get translated into?

A. 6 (*account, branch, customer, loan, depositor, borrower*)

# E/R Diagrams & Relations

E/R	Relational Schema
<i>Weak Entity Sets</i>	
 <p>The diagram illustrates two weak entity sets, <math>E_1</math> and <math>E_2</math>, connected by a double-lined diamond labeled <math>IR</math>. <math>E_1</math> is represented by a rectangle with a double border, and <math>E_2</math> is represented by a rectangle with a triple border. Both are connected to the <math>IR</math> diamond. <math>E_1</math> has attributes <math>a_1, \dots, a_n</math>, with <math>a_1</math> underlined. <math>E_2</math> has attributes <math>b_1, \dots, b_m</math>, with <math>b_1</math> underlined.</p>	$E_1 = (\underline{a_1}, \dots, a_n)$ $E_2 = (\underline{a_1}, \underline{b_1}, \dots, b_m)$

# E/R Diagrams & Relations

E/R	Relational Schema																
Multivalued Attributes																	
	<div>Emp = (<u>ssn</u>, name)</div> <div>Emp-Phones = (ssn, phone)</div> <div><table><tr><th>ssn</th><th>name</th><th>ssn</th><th>phone</th></tr><tr><td>001</td><td>Smith</td><td>001</td><td>4-1234</td></tr><tr><td>...</td><td>...</td><td>001</td><td>4-5678</td></tr><tr><td></td><td></td><td>...</td><td>...</td></tr></table></div> <div>Emp</div> <div>Emp-Phones</div>	ssn	name	ssn	phone	001	Smith	001	4-1234	...	...	001	4-5678			...	...
ssn	name	ssn	phone														
001	Smith	001	4-1234														
...	...	001	4-5678														
		...	...														

# E/R Diagrams & Relations

E/R	Relational Schema
<p data-bbox="102 348 374 396"><i>Subclasses</i></p> <pre data-bbox="193 456 927 1142">graph TD; E[E] --- ISA[ISA]; ISA --- E1[E1]; ISA --- E2[E2]; E --- a1((a1)); E --- dots1(...); E --- an((an)); E1 --- b1((b1)); E1 --- dots2(...); E1 --- bm((bm)); E2 --- c1((c1)); E2 --- dots3(...); E2 --- ck((ck));</pre>	<p data-bbox="993 448 1232 496">Method 1:</p> $E = (\underline{a_1}, \dots, a_n)$ $E_1 = (\underline{a_1}, b_1, \dots, b_m)$ $E_2 = (\underline{a_1}, c_1, \dots, c_k)$ <p data-bbox="993 828 1232 876">Method 2:</p> $E_1 = (\underline{a_1}, \dots, a_n, b_1, \dots, b_m)$ $E_2 = (\underline{a_1}, \dots, a_n, c_1, \dots, c_k)$

# E/R Diagrams & Relations

- ▶ Subclasses example:

- ▶ Method 1:

- ▶ Account = (acct\_no, balance)
- ▶ SAccount = (acct\_no, interest)
- ▶ CAccount = (acct\_no, overdraft)

- ▶ Method 2:

- ▶ SAccount = (acct\_no, balance, interest)
- ▶ CAccount = (acct\_no, balance, overdraft)