

# Chapter 7: Entity-Relationship Model

Database System Concepts, 6<sup>th</sup> Ed.

©Silberschatz, Korth and Sudarshan

See [www.db-book.com](http://www.db-book.com) for conditions on re-use



# Chapter 7: Entity-Relationship Model

- Design Process
- Modeling
- Constraints
- E-R Diagram
- Design Issues
- Weak Entity Sets
- Extended E-R Features
- Design of the Bank Database
- Reduction to Relation Schemas
- Database Design
- UML



# Modeling

- A *database* can be modeled as:
  - a collection of entities,
  - relationship among entities.
- An **entity** is an object that exists and is distinguishable from other objects.
  - Example: specific person, company, event, plant
- Entities have **attributes**
  - Example: people have *names* and *addresses*
- An **entity set** is a set of entities of the same type that share the same properties.
  - Example: set of all persons, companies, trees, holidays



# Entity Sets *instructor* and *student*

instructor\_ID instructor\_name

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

*instructor*

student-ID student\_name

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

*student*



# Relationship Sets

- A **relationship** is an association among several entities

Example:

44553 (Peltier)                  advisor                  22222 (Einstein)  
*student entity*                  *relationship set*                  *instructor entity*

- A **relationship set** is a mathematical relation among  $n \geq 2$  entities, each taken from entity sets

$$\{(e_1, e_2, \dots, e_n) \mid e_1 \in E_1, e_2 \in E_2, \dots, e_n \in E_n\}$$

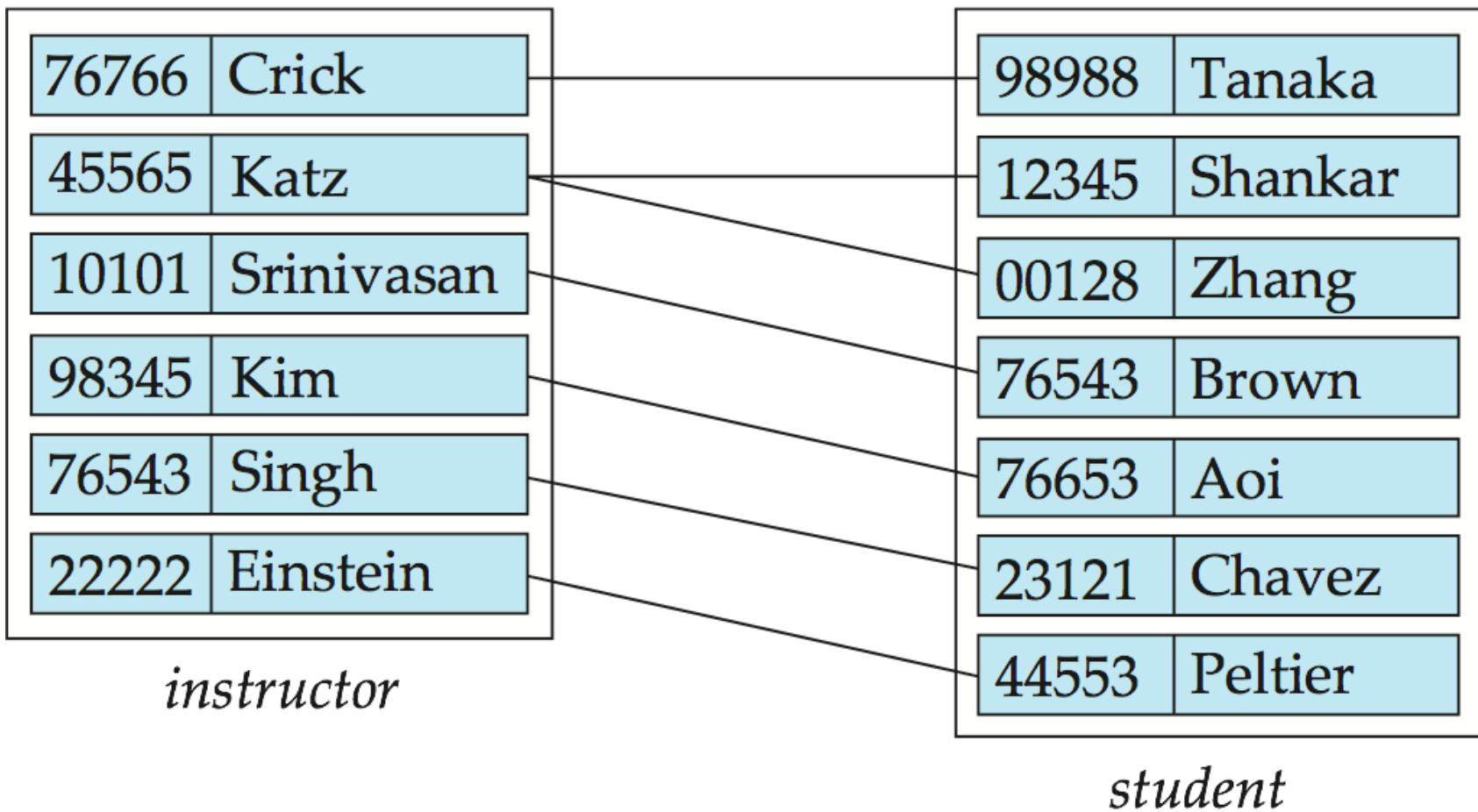
where  $(e_1, e_2, \dots, e_n)$  is a relationship

- Example:

$$(44553, 22222) \in \text{advisor}$$



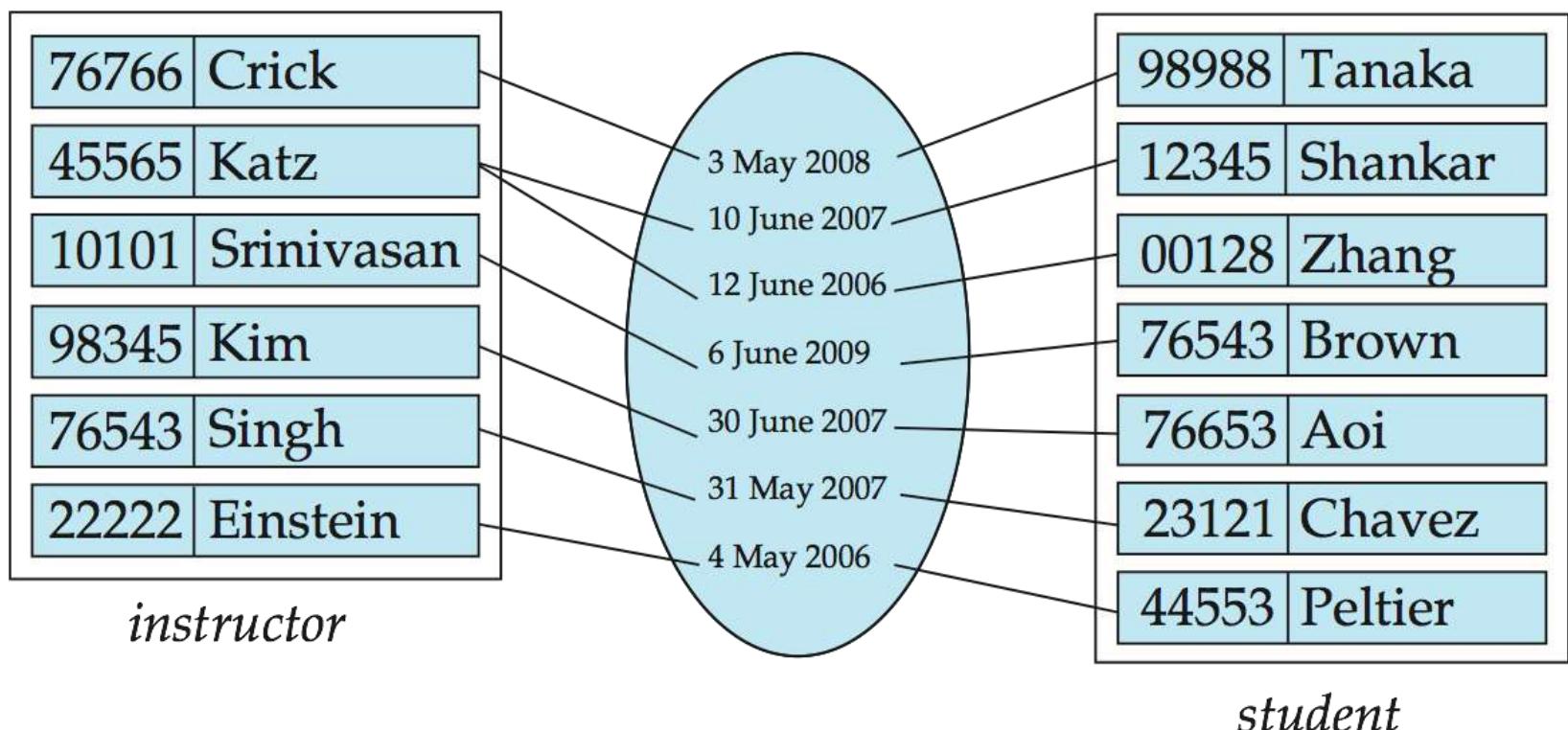
# Relationship Set *advisor*





# Relationship Sets (Cont.)

- An **attribute** can also be property of a relationship set.
- For instance, the *advisor* relationship set between entity sets *instructor* and *student* may have the attribute *date* which tracks when the student started being associated with the advisor





# Degree of a Relationship Set

## ■ **binary relationship**

- involve two entity sets (or degree two).
- most relationship sets in a database system are binary.

## ■ Relationships between more than two entity sets are rare. Most relationships are binary. (More on this later.)

- ▶ Example: *students* work on research *projects* under the guidance of an *instructor*.
- ▶ relationship *proj\_guide* is a ternary relationship between *instructor*, *student*, and *project*



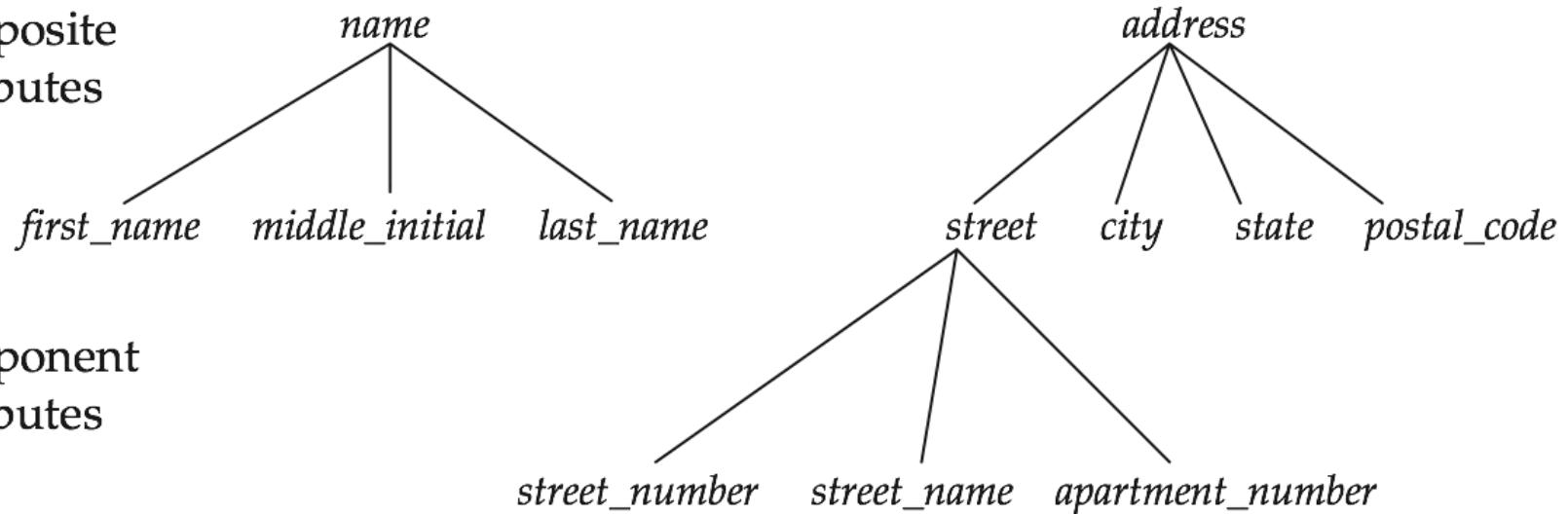
# Attributes

- An entity is represented by a set of attributes, that is descriptive properties possessed by all members of an entity set.
  - Example:  
 $\text{instructor} = (\text{ID}, \text{name}, \text{street}, \text{city}, \text{salary})$   
 $\text{course} = (\text{course\_id}, \text{title}, \text{credits})$
- **Domain** – the set of permitted values for each attribute
- Attribute types:
  - **Simple** and **composite** attributes.
  - **Single-valued** and **multivalued** attributes
    - ▶ Example: multivalued attribute: *phone\_numbers*
  - **Derived** attributes
    - ▶ Can be computed from other attributes
    - ▶ Example: *age*, *given\_date\_of\_birth*



# Composite Attributes

composite  
attributes



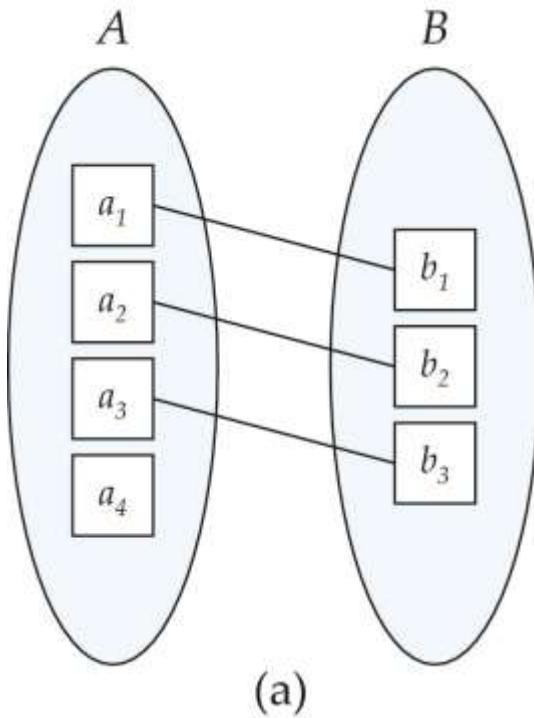


# Mapping Cardinality Constraints

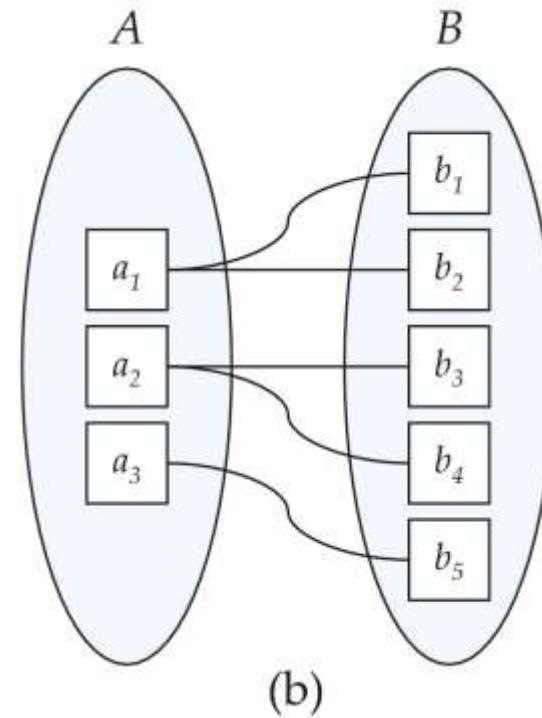
- Express the number of entities to which another entity can be associated via a relationship set.
- Most useful in describing binary relationship sets.
- For a binary relationship set the mapping cardinality must be one of the following types:
  - One to one
  - One to many
  - Many to one
  - Many to many



# Mapping Cardinalities



One to one

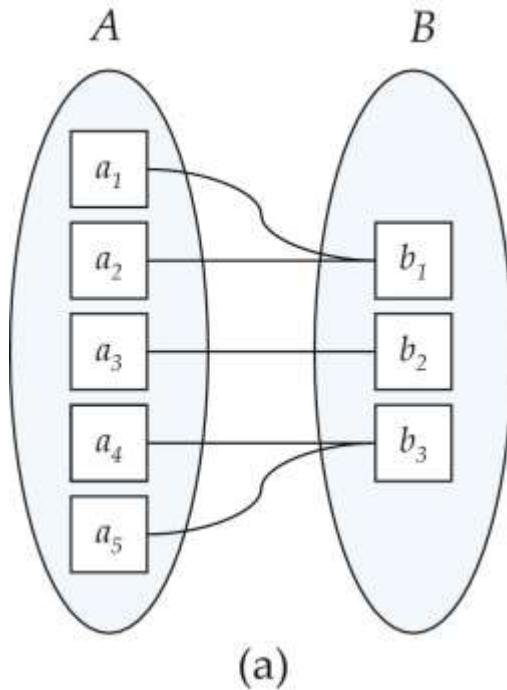


One to many

Note: Some elements in  $A$  and  $B$  may not be mapped to any elements in the other set

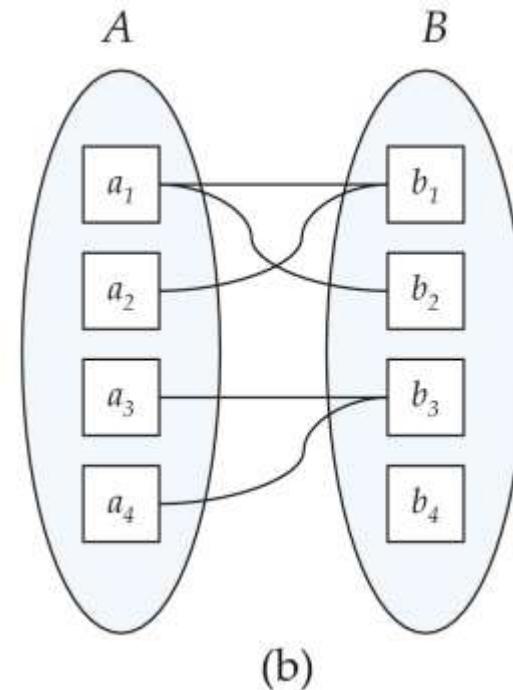


# Mapping Cardinalities



(a)

Many to  
one



(b)

Many to many

Note: Some elements in A and B may not be mapped to any elements in the other set



# Keys

- A **super key** of an entity set is a set of one or more attributes whose values uniquely determine each entity.
- A **candidate key** of an entity set is a minimal super key
  - *ID* is candidate key of *instructor*
  - *course\_id* is candidate key of *course*
- Although several candidate keys may exist, one of the candidate keys is selected to be the **primary key**.



# Keys for Relationship Sets

- The combination of primary keys of the participating entity sets forms a super key of a relationship set.
  - $(s\_id, i\_id)$  is the super key of *advisor*
  - ***NOTE: this means a pair of entity sets can have at most one relationship in a particular relationship set.***
    - ▶ Example: if we wish to track multiple meeting dates between a student and her advisor, we cannot assume a relationship for each meeting. We can use a multivalued attribute though
- Must consider the mapping cardinality of the relationship set when deciding what are the candidate keys
- Need to consider semantics of relationship set in selecting the *primary key* in case of more than one candidate key

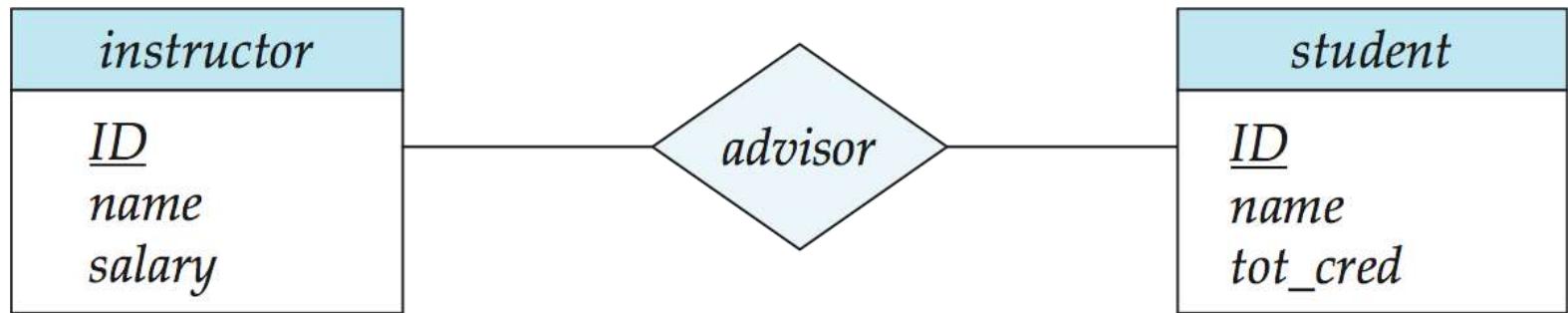


# Redundant Attributes

- Suppose we have entity sets
  - *instructor*, with attributes including *dept\_name*
  - *department*
- and a relationship
  - *inst\_dept* relating *instructor* and *department*
- Attribute *dept\_name* in entity *instructor* is redundant since there is an explicit relationship *inst\_dept* which relates instructors to departments
  - The attribute replicates information present in the relationship, and should be removed from *instructor*
  - BUT: when converting back to tables, in some cases the attribute gets reintroduced, as we will see.



# E-R Diagrams



- Rectangles represent entity sets.
- Diamonds represent relationship sets.
- Attributes listed inside entity rectangle
- Underline indicates primary key attributes

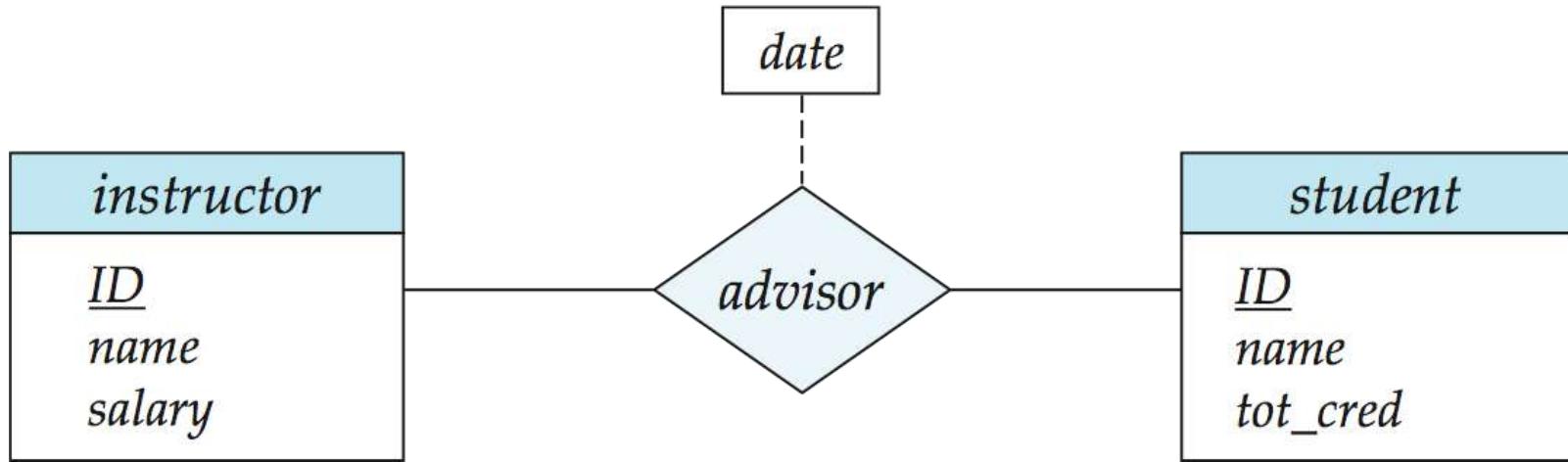


# Entity With Composite, Multivalued, and Derived Attributes

<i>instructor</i>	
<u><i>ID</i></u>	
<i>name</i>	
	<i>first_name</i>
	<i>middle_initial</i>
	<i>last_name</i>
<i>address</i>	
	<i>street</i>
	<i>street_number</i>
	<i>street_name</i>
	<i>apt_number</i>
<i>city</i>	
<i>state</i>	
<i>zip</i>	
{ <i>phone_number</i> }	
<i>date_of_birth</i>	
<i>age</i> ( )	



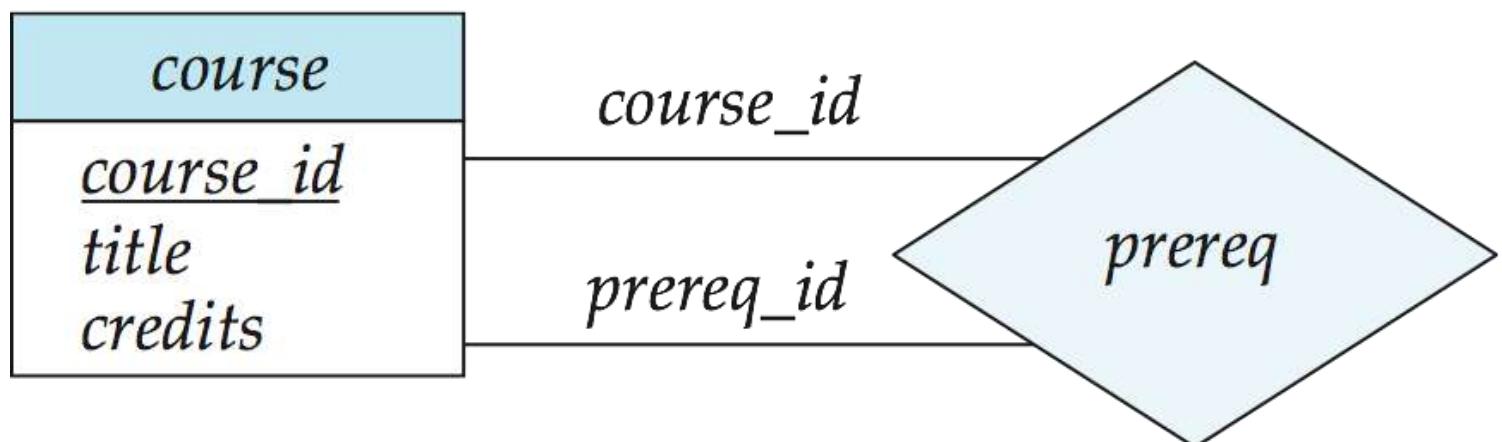
# Relationship Sets with Attributes





# Roles

- Entity sets of a relationship need not be distinct
  - Each occurrence of an entity set plays a “role” in the relationship
- The labels “*course\_id*” and “*prereq\_id*” are called **roles**.





# Cardinality Constraints

- We express cardinality constraints by drawing either a directed line ( $\rightarrow$ ), signifying “one,” or an undirected line ( $-$ ), signifying “many,” between the relationship set and the entity set.
- One-to-one relationship:
  - A student is associated with at most one *instructor* via the relationship *advisor*
  - A *student* is associated with at most one *department* via *stud\_dept*



# One-to-One Relationship

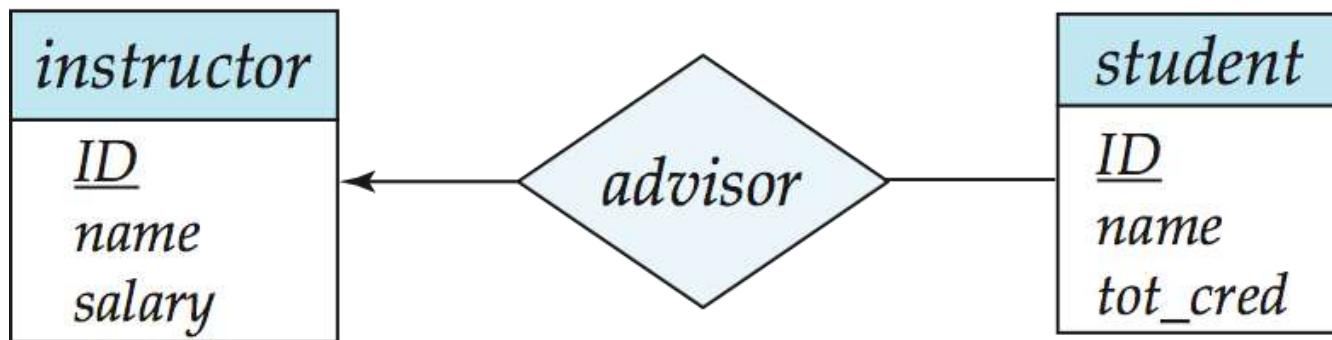
- one-to-one relationship between an *instructor* and a *student*
  - an instructor is associated with at most one student via *advisor*
  - and a student is associated with at most one instructor via *advisor*





# One-to-Many Relationship

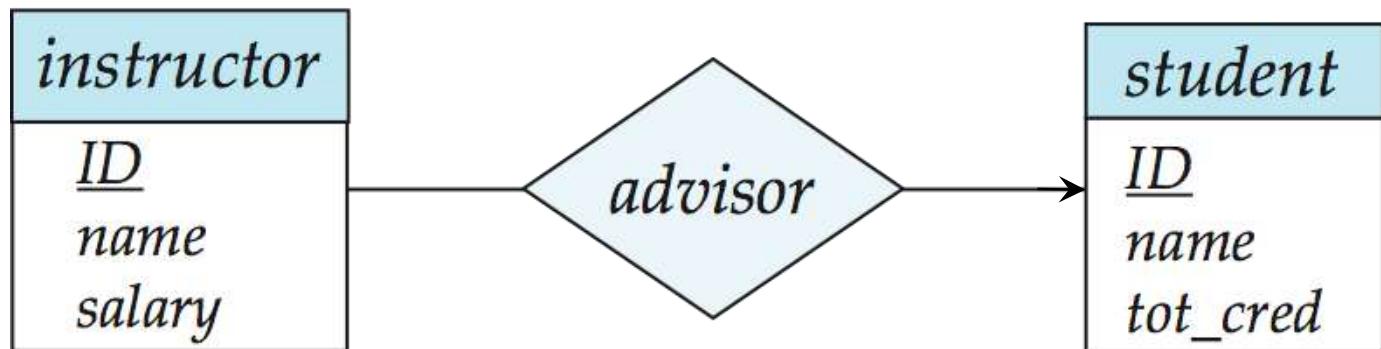
- one-to-many relationship between an *instructor* and a *student*
  - an instructor is associated with several (including 0) students via *advisor*
  - a student is associated with at most one instructor via advisor,





# Many-to-One Relationships

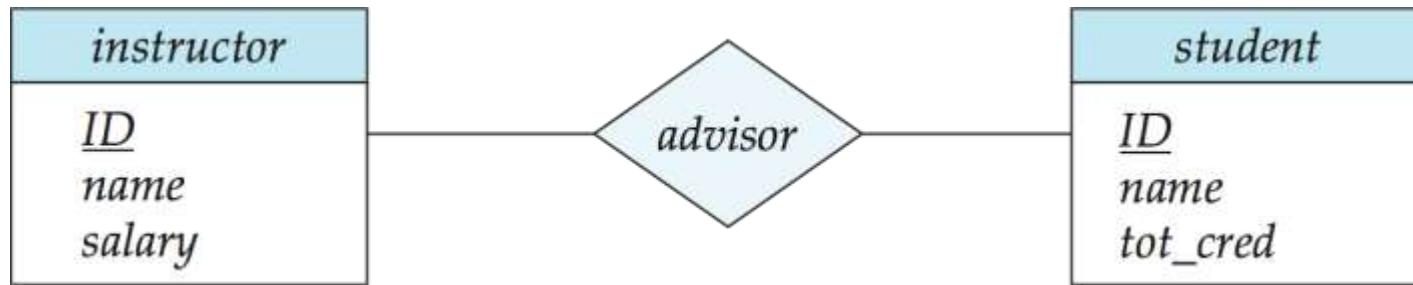
- In a many-to-one relationship between an *instructor* and a *student*,
  - an *instructor* is associated with at most one *student* via *advisor*,
  - and a *student* is associated with several (including 0) *instructors* via *advisor*





# Many-to-Many Relationship

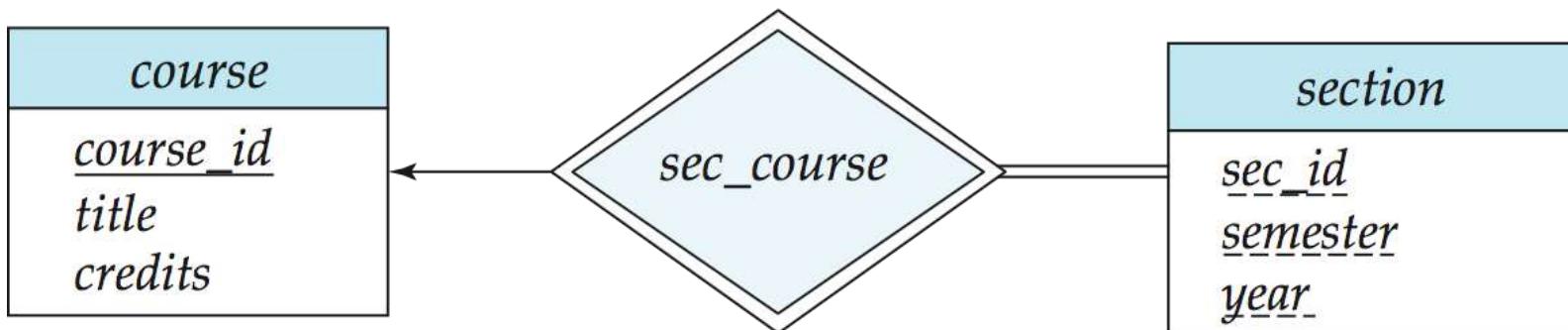
- An instructor is associated with several (possibly 0) students via *advisor*
- A student is associated with several (possibly 0) instructors via *advisor*





# Participation of an Entity Set in a Relationship Set

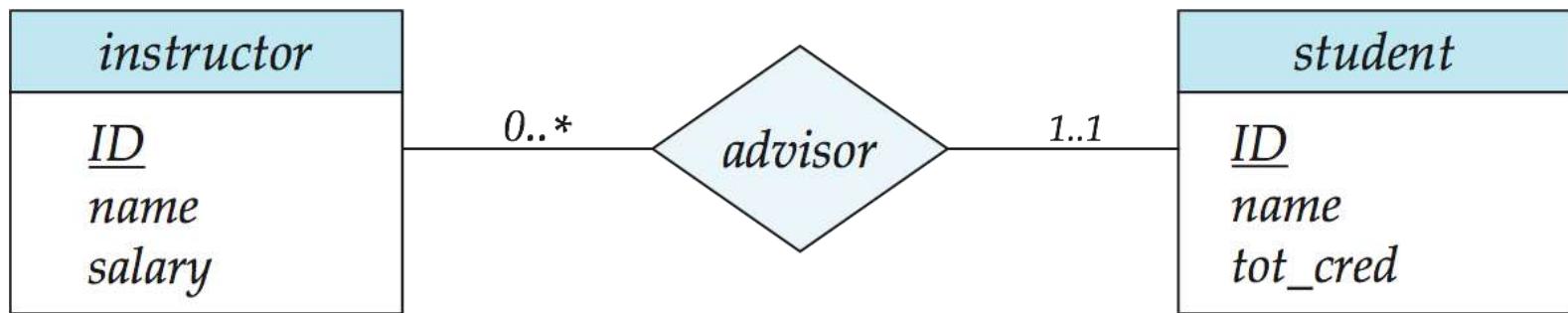
- Total participation (indicated by double line): every entity in the entity set participates in at least one relationship in the relationship set
  - E.g., participation of *section* in *sec\_course* is total
    - ▶ every *section* must have an associated course
- Partial participation: some entities may not participate in any relationship in the relationship set
  - Example: participation of *instructor* in *advisor* is partial





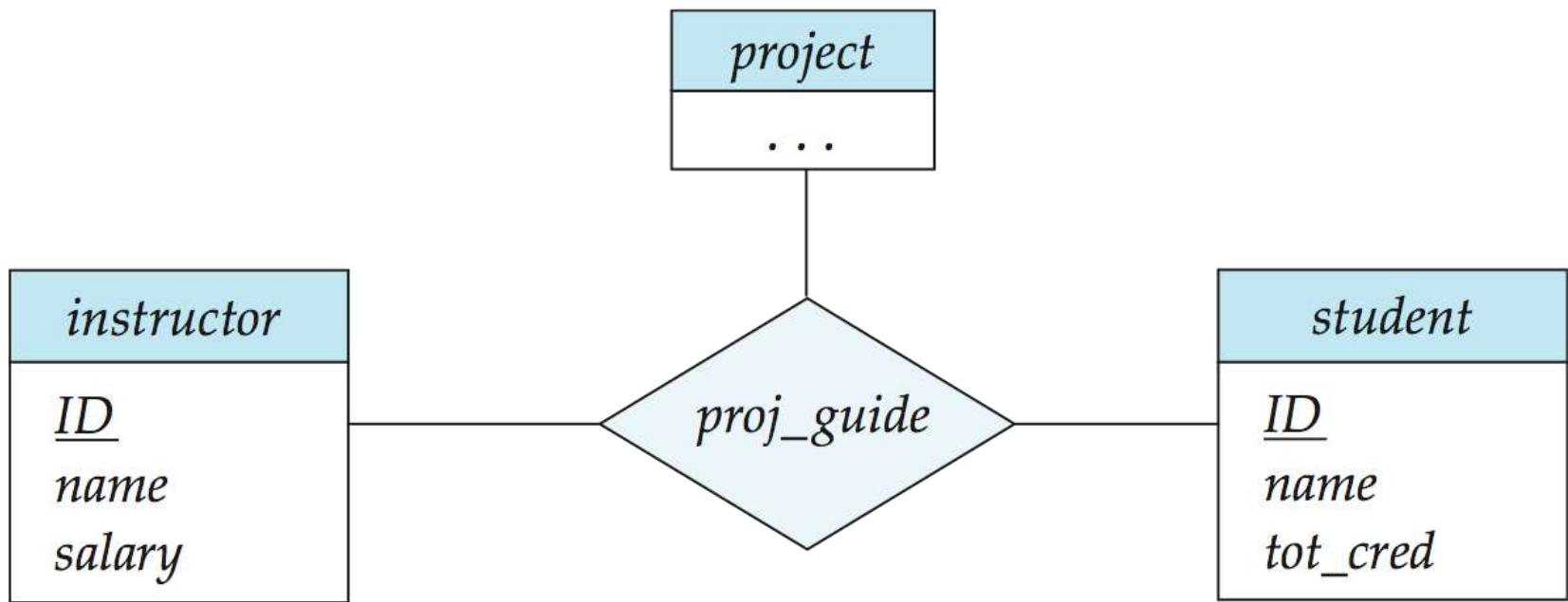
# Alternative Notation for Cardinality Limits

- Cardinality limits can also express participation constraints





# E-R Diagram with a Ternary Relationship





# Cardinality Constraints on Ternary Relationship

- We allow at most one arrow out of a ternary (or greater degree) relationship to indicate a cardinality constraint
- E.g., an arrow from *proj\_guide* to *instructor* indicates each student has at most one guide for a project
- If there is more than one arrow, there are two ways of defining the meaning.
  - E.g., a ternary relationship  $R$  between  $A$ ,  $B$  and  $C$  with arrows to  $B$  and  $C$  could mean
    1. each  $A$  entity is associated with a unique entity from  $B$  and  $C$  or
    2. each pair of entities from  $(A, B)$  is associated with a unique  $C$  entity, and each pair  $(A, C)$  is associated with a unique  $B$
  - Each alternative has been used in different formalisms
  - To avoid confusion we outlaw more than one arrow



The picture can't be displayed.

**How about doing an ER design  
interactively on the board?  
Suggest an application to be modeled.**

**Database System Concepts, 6<sup>th</sup> Ed.**

©Silberschatz, Korth and Sudarshan

See [www.db-book.com](http://www.db-book.com) for conditions on re-use



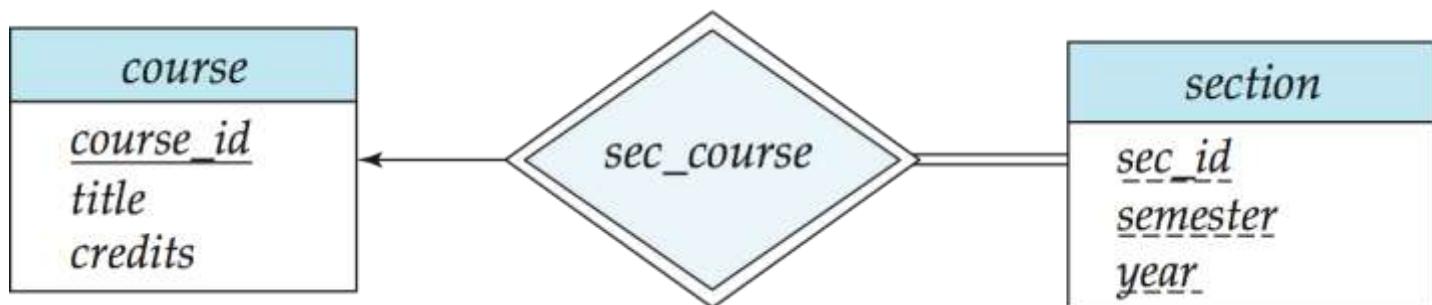
# Weak Entity Sets

- An entity set that does not have a primary key is referred to as a **weak entity set**.
- The existence of a weak entity set depends on the existence of a **identifying entity set**
  - It must relate to the identifying entity set via a total, one-to-many relationship set from the identifying to the weak entity set
  - **Identifying relationship** depicted using a double diamond
- The **discriminator** (*or partial key*) of a weak entity set is the set of attributes that distinguishes among all the entities of a weak entity set.
- The primary key of a weak entity set is formed by the primary key of the strong entity set on which the weak entity set is existence dependent, plus the weak entity set's discriminator.



# Weak Entity Sets (Cont.)

- We underline the discriminator of a weak entity set with a dashed line.
- We put the identifying relationship of a weak entity in a double diamond.
- Primary key for *section* – (*course\_id*, *sec\_id*, *semester*, *year*)



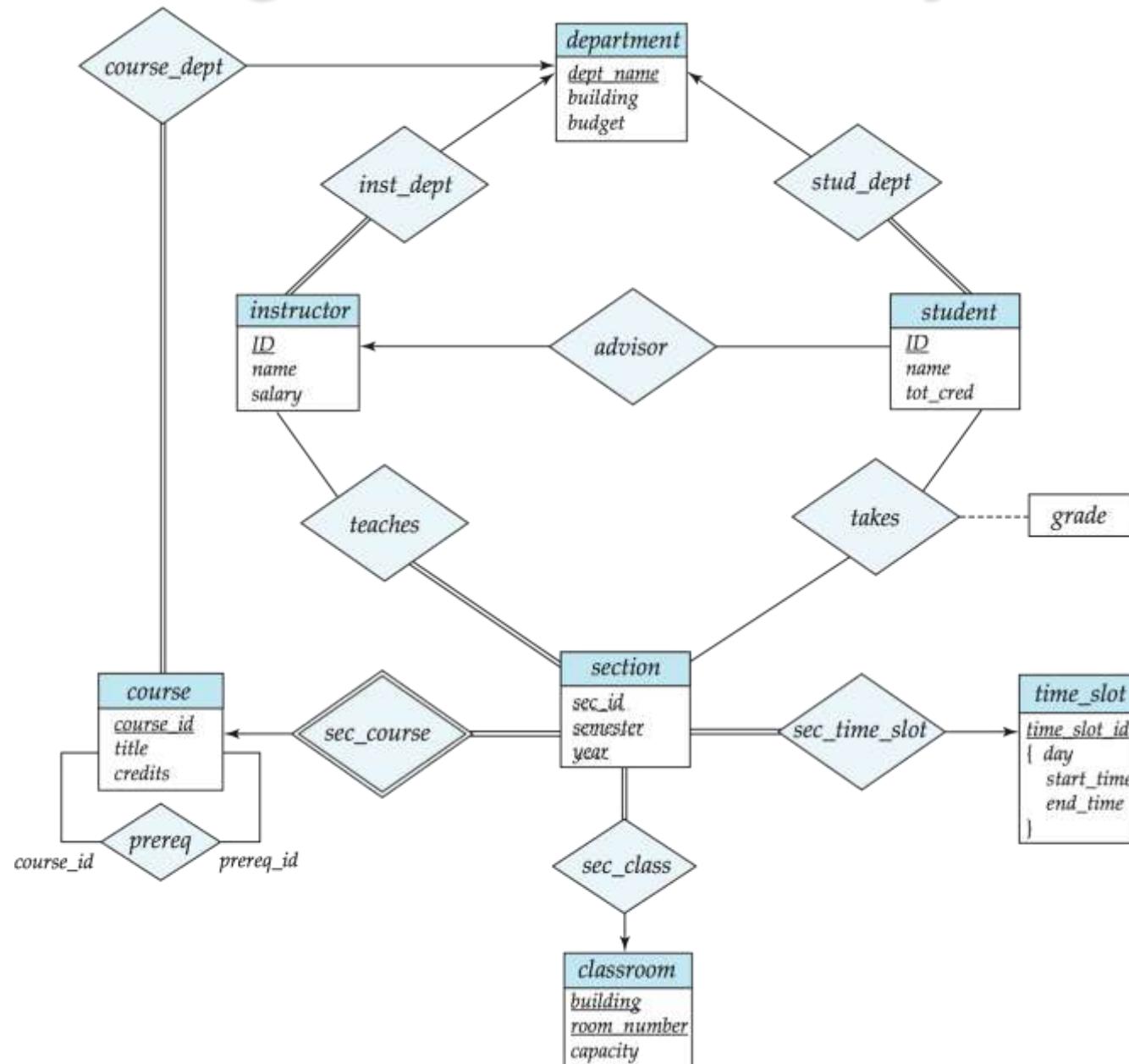


# Weak Entity Sets (Cont.)

- Note: the primary key of the strong entity set is not explicitly stored with the weak entity set, since it is implicit in the identifying relationship.
- If *course\_id* were explicitly stored, *section* could be made a strong entity, but then the relationship between *section* and *course* would be duplicated by an implicit relationship defined by the attribute *course\_id* common to *course* and *section*



# E-R Diagram for a University Enterprise





# Reduction to Relational Schemas



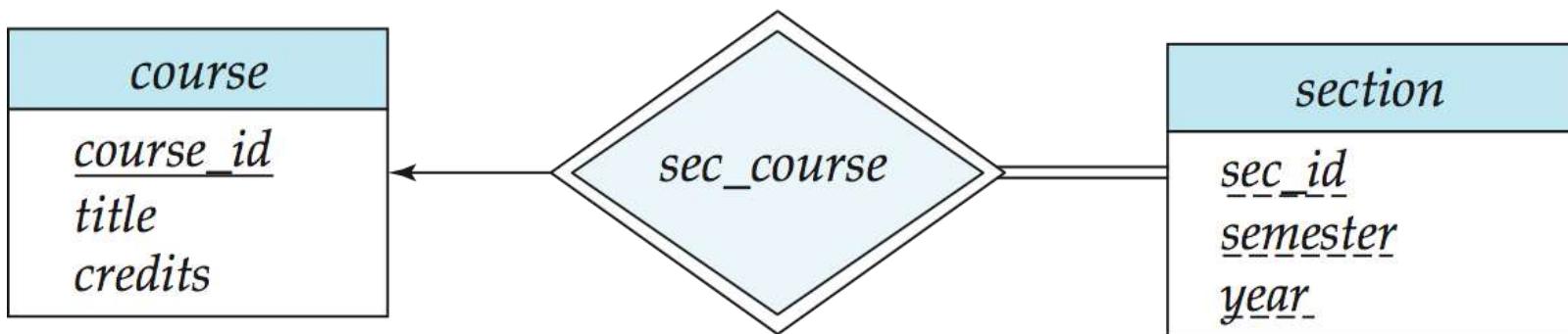
# Reduction to Relation Schemas

- Entity sets and relationship sets can be expressed uniformly as *relation schemas* that represent the contents of the database.
- A database which conforms to an E-R diagram can be represented by a collection of schemas.
- For each entity set and relationship set there is a unique schema that is assigned the name of the corresponding entity set or relationship set.
- Each schema has a number of columns (generally corresponding to attributes), which have unique names.



# Representing Entity Sets with Simple Attributes

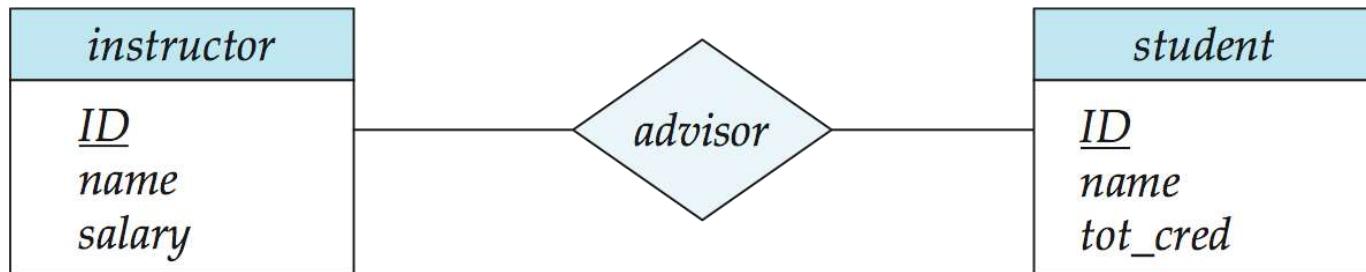
- A strong entity set reduces to a schema with the same attributes  
*student( ID, name, tot\_cred )*
- A weak entity set becomes a table that includes a column for the primary key of the identifying strong entity set  
*section ( course\_id, sec\_id, sem, year )*





# Representing Relationship Sets

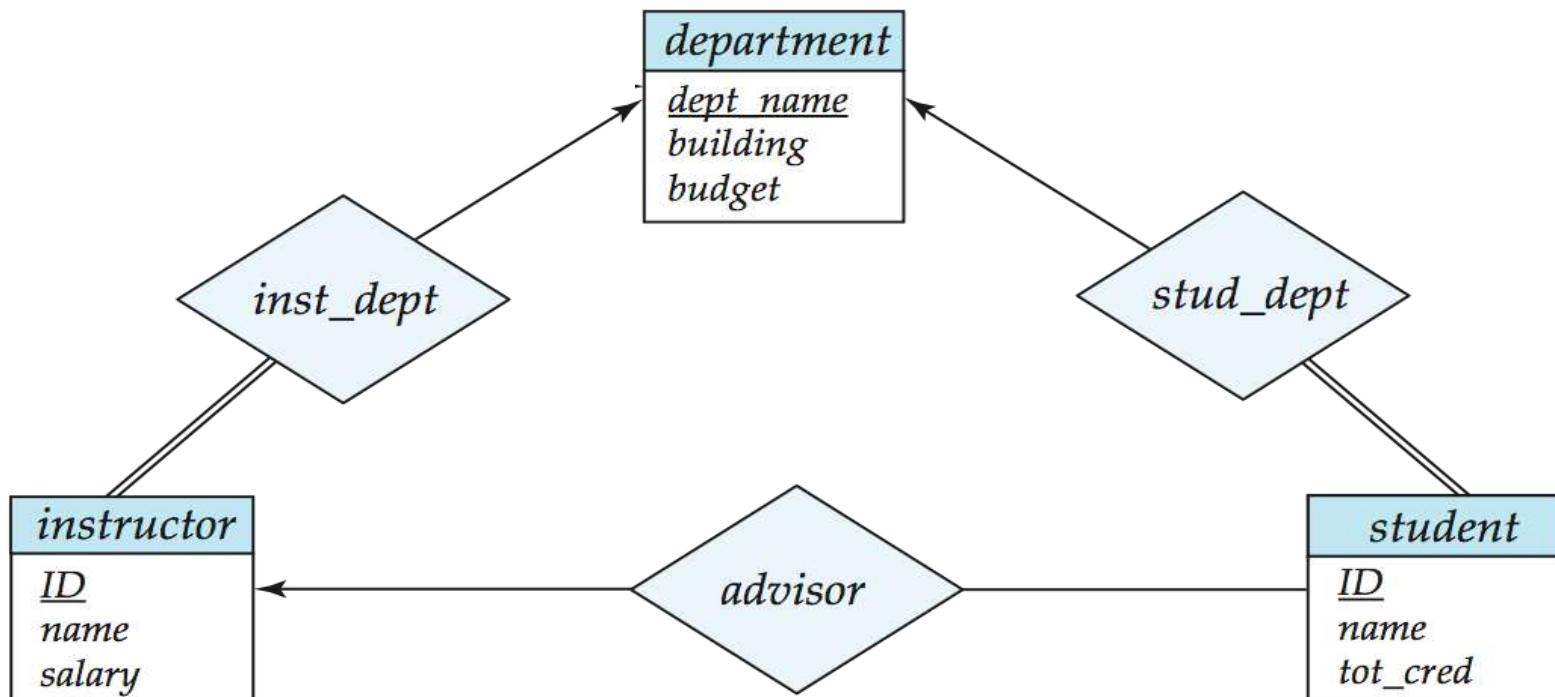
- A many-to-many relationship set is represented as a schema with attributes for the primary keys of the two participating entity sets, and any descriptive attributes of the relationship set.
- Example: schema for relationship set *advisor*  
 $\text{advisor} = (\underline{s\_id}, \underline{i\_id})$





# Redundancy of Schemas

- Many-to-one and one-to-many relationship sets that are total on the many-side can be represented by adding an extra attribute to the “many” side, containing the primary key of the “one” side
- Example: Instead of creating a schema for relationship set *inst\_dept*, add an attribute *dept\_name* to the schema arising from entity set *instructor*





# Redundancy of Schemas (Cont.)

- For one-to-one relationship sets, either side can be chosen to act as the “many” side
  - That is, extra attribute can be added to either of the tables corresponding to the two entity sets
- If participation is *partial* on the “many” side, replacing a schema by an extra attribute in the schema corresponding to the “many” side could result in null values
- The schema corresponding to a relationship set linking a weak entity set to its identifying strong entity set is redundant.
  - Example: The *section* schema already contains the attributes that would appear in the *sec\_course* schema



# Composite and Multivalued Attributes

<i>instructor</i>	
<u>ID</u>	
<i>name</i>	
<i>first_name</i>	
<i>middle_initial</i>	
<i>last_name</i>	
<i>address</i>	
<i>street</i>	
<i>street_number</i>	
<i>street_name</i>	
<i>apt_number</i>	
<i>city</i>	
<i>state</i>	
<i>zip</i>	
{ <i>phone_number</i> }	
<i>date_of_birth</i>	
<i>age ()</i>	

- Composite attributes are flattened out by creating a separate attribute for each component attribute
  - Example: given entity set *instructor* with composite attribute *name* with component attributes *first\_name* and *last\_name* the schema corresponding to the entity set has two attributes *name\_first\_name* and *name\_last\_name*
    - ▶ Prefix omitted if there is no ambiguity
- Ignoring multivalued attributes, extended instructor schema is
  - *instructor(ID, first\_name, middle\_initial, last\_name, street\_number, street\_name, apt\_number, city, state, zip\_code, date\_of\_birth)*



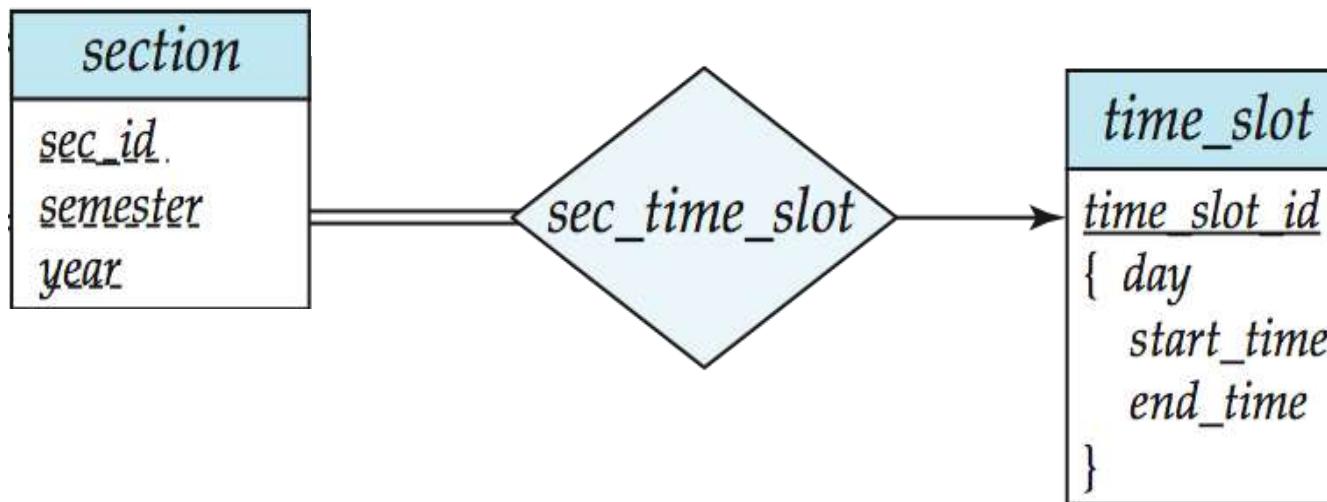
# Composite and Multivalued Attributes

- A multivalued attribute  $M$  of an entity  $E$  is represented by a separate schema  $EM$ 
  - Schema  $EM$  has attributes corresponding to the primary key of  $E$  and an attribute corresponding to multivalued attribute  $M$
  - Example: Multivalued attribute  $phone\_number$  of  $instructor$  is represented by a schema:  
 $inst\_phone = (\underline{ID}, \underline{phone\_number})$
  - Each value of the multivalued attribute maps to a separate tuple of the relation on schema  $EM$ 
    - ▶ For example, an  $instructor$  entity with primary key 22222 and phone numbers 456-7890 and 123-4567 maps to two tuples:  
(22222, 456-7890) and (22222, 123-4567)



# Multivalued Attributes (Cont.)

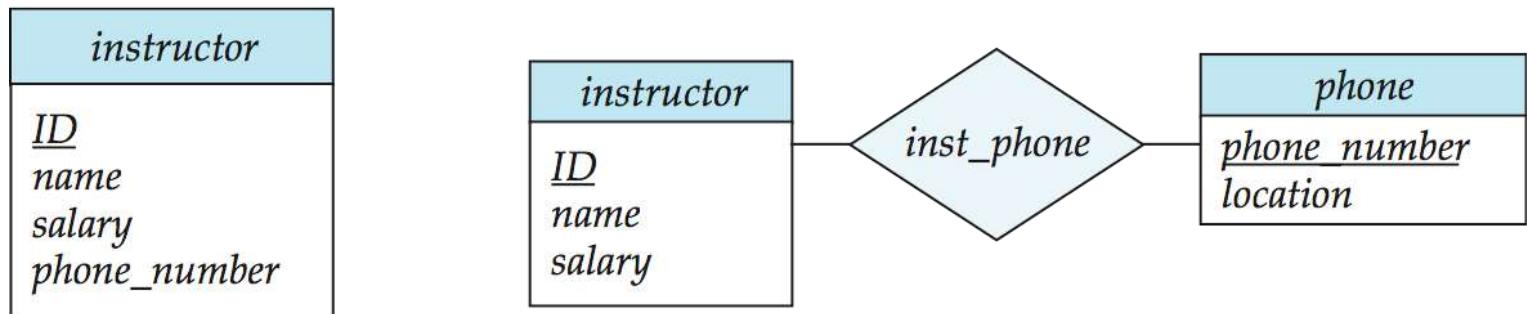
- Special case: entity *time\_slot* has only one attribute other than the primary-key attribute, and that attribute is multivalued
  - Optimization: Don't create the relation corresponding to the entity, just create the one corresponding to the multivalued attribute
  - *time\_slot*(*time\_slot\_id*, *day*, *start\_time*, *end\_time*)
  - Caveat: *time\_slot* attribute of *section* (from *sec\_time\_slot*) cannot be a foreign key due to this optimization





# Design Issues

## ■ Use of entity sets vs. attributes



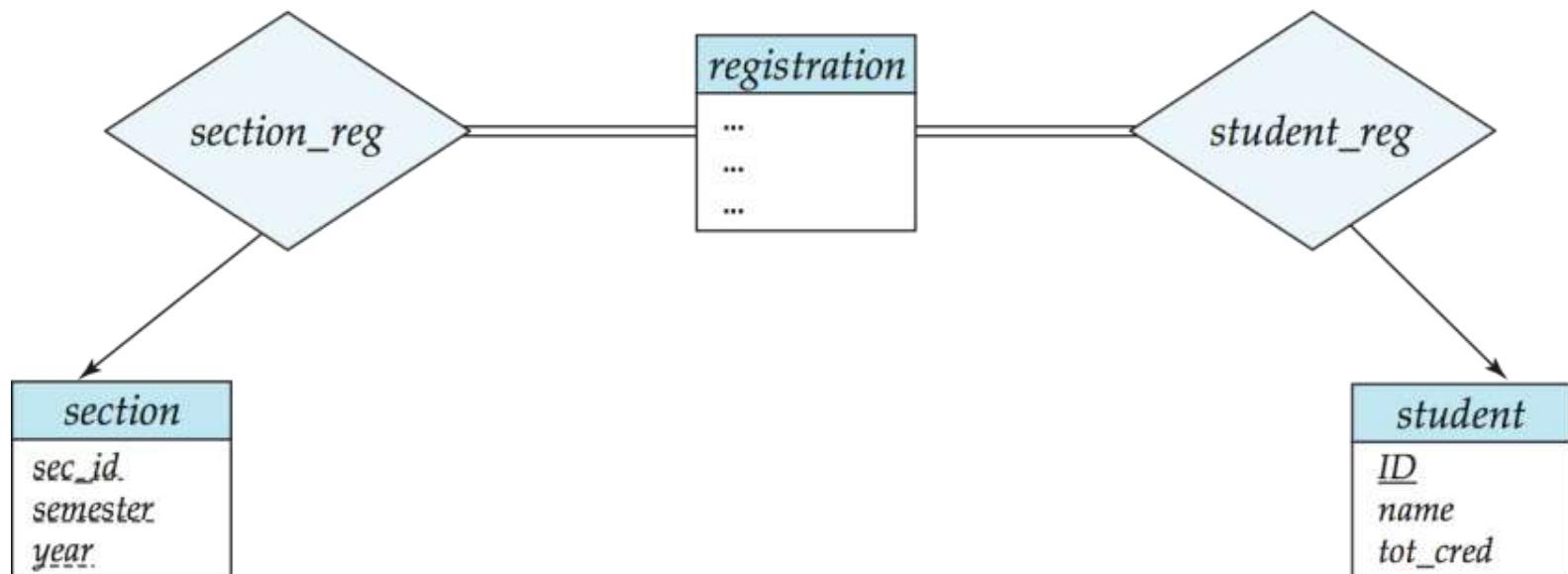
- Use of phone as an entity allows extra information about phone numbers (plus multiple phone numbers)



# Design Issues

## ■ Use of entity sets vs. relationship sets

Possible guideline is to designate a relationship set to describe an action that occurs between entities





# Design Issues

## ■ **Binary versus n-ary relationship sets**

Although it is possible to replace any nonbinary ( $n$ -ary, for  $n > 2$ ) relationship set by a number of distinct binary relationship sets, a  $n$ -ary relationship set shows more clearly that several entities participate in a single relationship.

## ■ **Placement of relationship attributes**

e.g., attribute *date* as attribute of *advisor* or as attribute of *student*



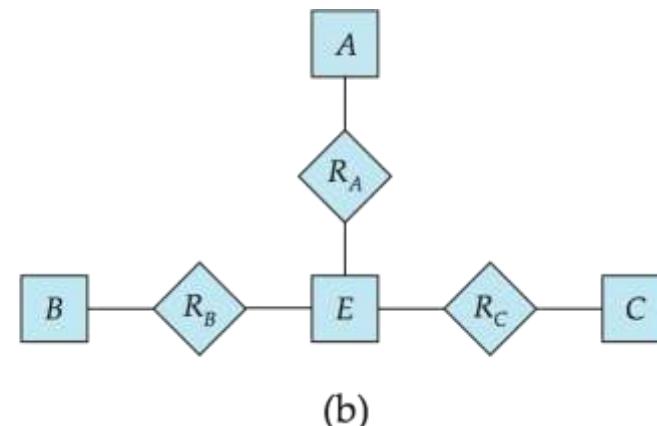
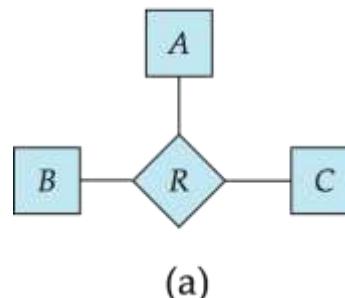
# Binary Vs. Non-Binary Relationships

- Some relationships that appear to be non-binary may be better represented using binary relationships
  - E.g., A ternary relationship *parents*, relating a child to his/her father and mother, is best replaced by two binary relationships, *father* and *mother*
    - ▶ Using two binary relationships allows partial information (e.g., only mother being known)
  - But there are some relationships that are naturally non-binary
    - ▶ Example: *proj\_guide*



# Converting Non-Binary Relationships to Binary Form

- In general, any non-binary relationship can be represented using binary relationships by creating an artificial entity set.
  - Replace  $R$  between entity sets A, B and C by an entity set  $E$ , and three relationship sets:
    - $R_A$ , relating  $E$  and A
    - $R_B$ , relating  $E$  and B
    - $R_C$ , relating  $E$  and C
  - Create a special identifying attribute for  $E$
  - Add any attributes of  $R$  to  $E$
  - For each relationship  $(a_i, b_i, c_i)$  in  $R$ , create
    - a new entity  $e_i$  in the entity set  $E$
    - add  $(e_i, a_i)$  to  $R_A$
    - add  $(e_i, b_i)$  to  $R_B$
    - add  $(e_i, c_i)$  to  $R_C$





# Converting Non-Binary Relationships (Cont.)

- Also need to translate constraints
  - Translating all constraints may not be possible
  - There may be instances in the translated schema that cannot correspond to any instance of  $R$ 
    - ▶ Exercise: *add constraints to the relationships  $R_A$ ,  $R_B$  and  $R_C$  to ensure that a newly created entity corresponds to exactly one entity in each of entity sets  $A$ ,  $B$  and  $C$*
  - We can avoid creating an identifying attribute by making  $E$  a weak entity set (described shortly) identified by the three relationship sets



# Extended ER Features

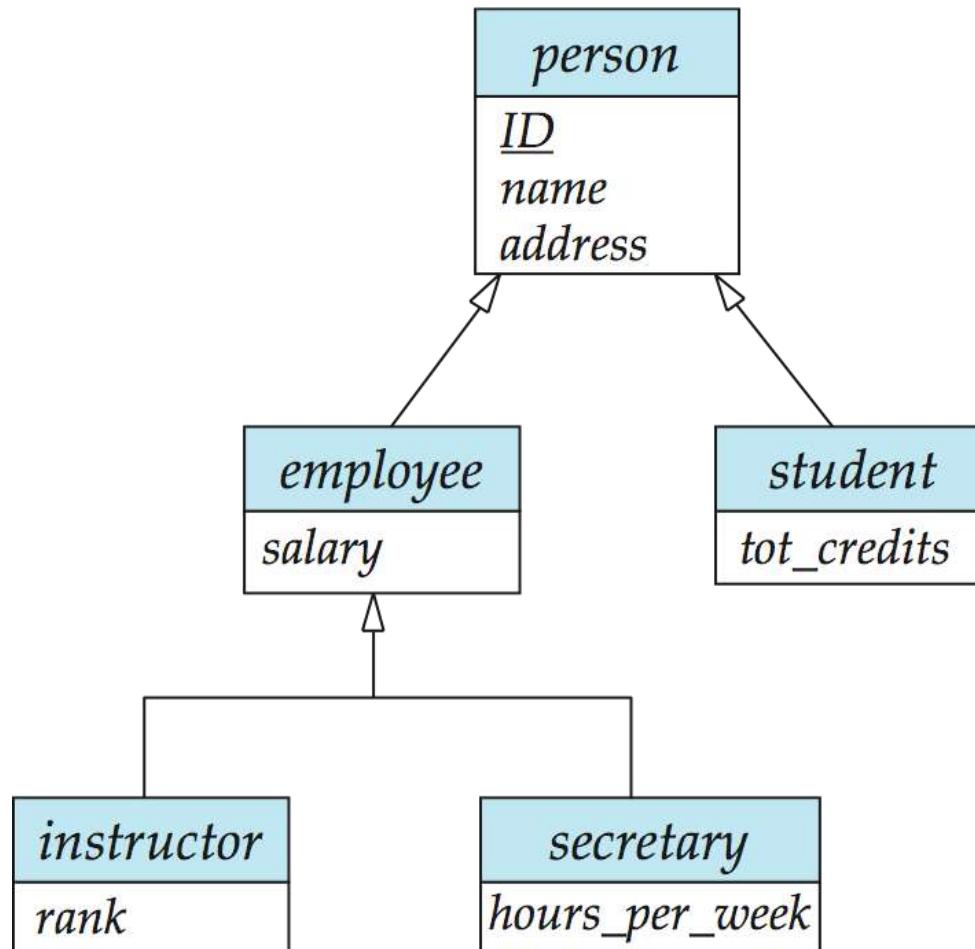


# Extended E-R Features: Specialization

- Top-down design process; we designate subgroupings within an entity set that are distinctive from other entities in the set.
- These subgroupings become lower-level entity sets that have attributes or participate in relationships that do not apply to the higher-level entity set.
- Depicted by a *triangle* component labeled ISA (E.g., *instructor* “is a” *person*).
- **Attribute inheritance** – a lower-level entity set inherits all the attributes and relationship participation of the higher-level entity set to which it is linked.



# Specialization Example





# Extended ER Features: Generalization

- **A bottom-up design process** – combine a number of entity sets that share the same features into a higher-level entity set.
- Specialization and generalization are simple inversions of each other; they are represented in an E-R diagram in the same way.
- The terms specialization and generalization are used interchangeably.



# Specialization and Generalization (Cont.)

- Can have multiple specializations of an entity set based on different features.
- E.g., *permanent\_employee* vs. *temporary\_employee*, in addition to *instructor* vs. *secretary*
- Each particular employee would be
  - a member of one of *permanent\_employee* or *temporary\_employee*,
  - and also a member of one of *instructor*, *secretary*
- The ISA relationship also referred to as **superclass - subclass** relationship



# Design Constraints on a Specialization/Generalization

- Constraint on which entities can be members of a given lower-level entity set.
  - condition-defined
    - ▶ Example: all customers over 65 years are members of *senior-citizen* entity set; *senior-citizen* ISA *person*.
  - user-defined
- Constraint on whether or not entities may belong to more than one lower-level entity set within a single generalization.
  - **Disjoint**
    - ▶ an entity can belong to only one lower-level entity set
    - ▶ Noted in E-R diagram by having multiple lower-level entity sets link to the same triangle
  - **Overlapping**
    - ▶ an entity can belong to more than one lower-level entity set



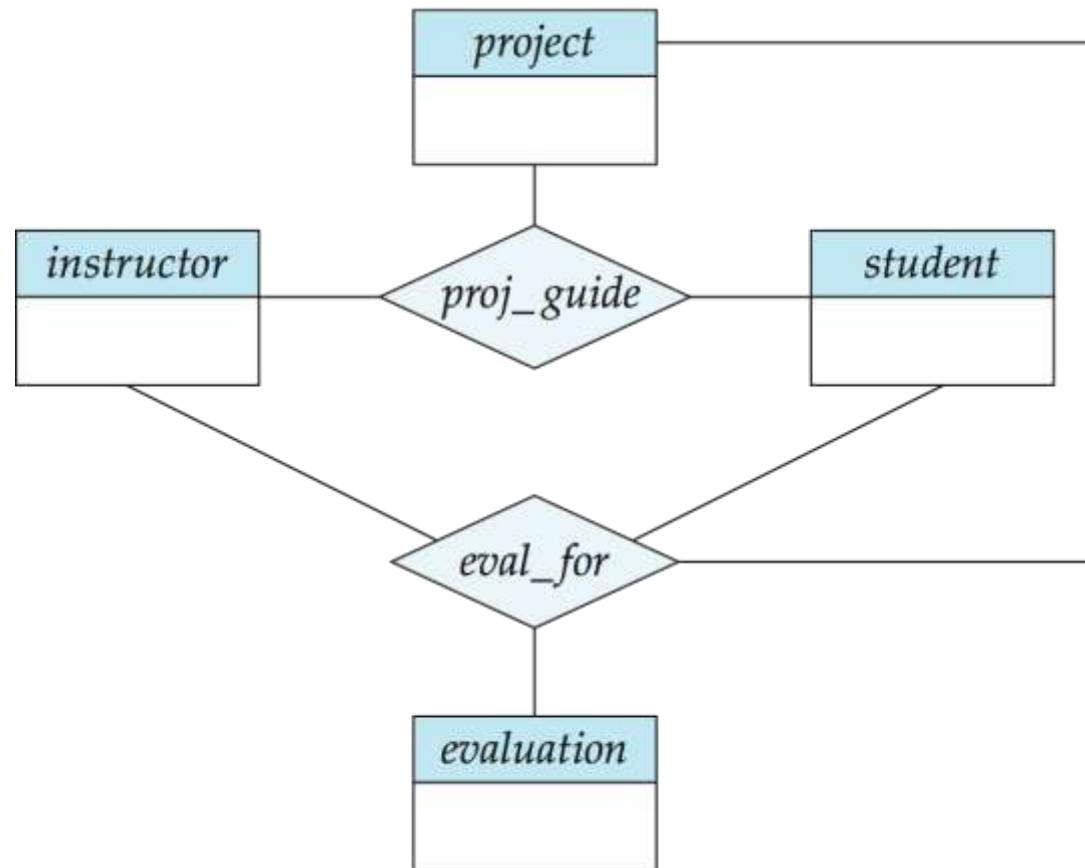
# Design Constraints on a Specialization/Generalization (Cont.)

- **Completeness constraint** -- specifies whether or not an entity in the higher-level entity set must belong to at least one of the lower-level entity sets within a generalization.
  - **total**: an entity must belong to one of the lower-level entity sets
  - **partial**: an entity need not belong to one of the lower-level entity sets



# Aggregation

- Consider the ternary relationship *proj\_guide*, which we saw earlier
- Suppose we want to record evaluations of a student by a guide on a project





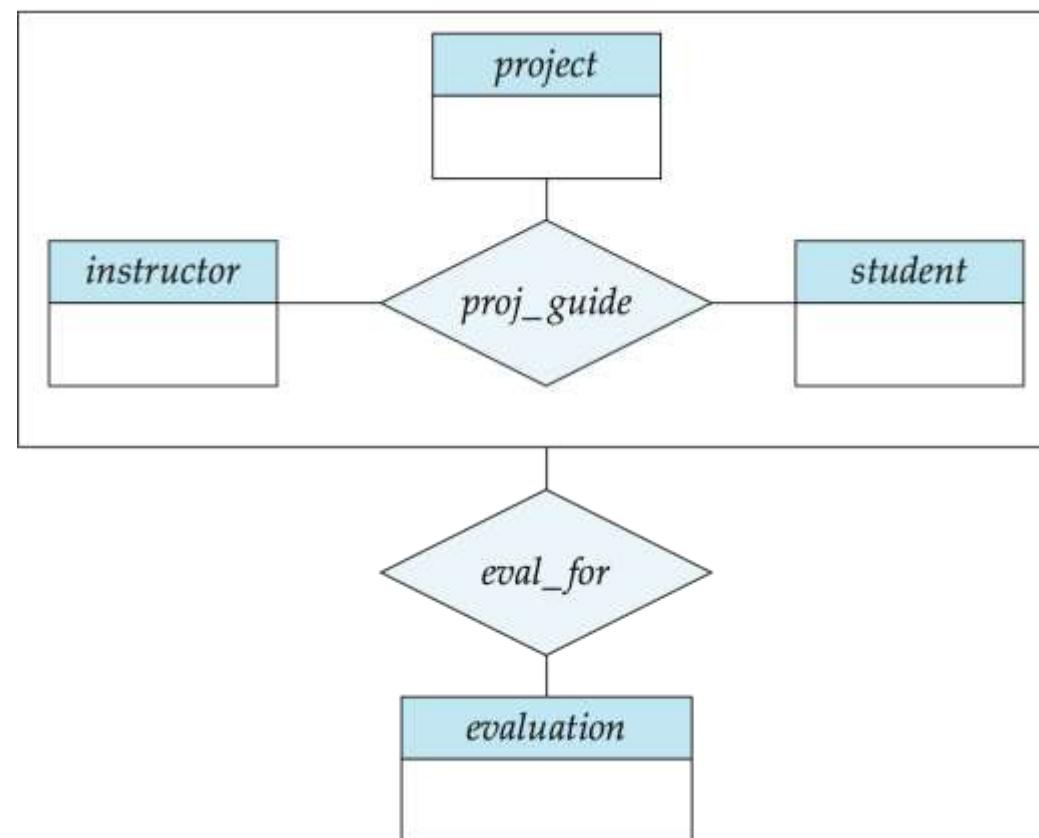
# Aggregation (Cont.)

- Relationship sets *eval\_for* and *proj\_guide* represent overlapping information
  - Every *eval\_for* relationship corresponds to a *proj\_guide* relationship
  - However, some *proj\_guide* relationships may not correspond to any *eval\_for* relationships
    - ▶ So we can't discard the *proj\_guide* relationship
- Eliminate this redundancy via *aggregation*
  - Treat relationship as an abstract entity
  - Allows relationships between relationships
  - Abstraction of relationship into new entity



# Aggregation (Cont.)

- Without introducing redundancy, the following diagram represents:
  - A student is guided by a particular instructor on a particular project
  - A student, instructor, project combination may have an associated evaluation





# Representing Specialization via Schemas

## ■ Method 1:

- Form a schema for the higher-level entity
- Form a schema for each lower-level entity set, include primary key of higher-level entity set and local attributes

schema	attributes
<i>person</i>	<i>ID, name, street, city</i>
<i>student</i>	<i>ID, tot_cred</i>
<i>employee</i>	<i>ID, salary</i>

- Drawback: getting information about, an *employee* requires accessing two relations, the one corresponding to the low-level schema and the one corresponding to the high-level schema



# Representing Specialization as Schemas (Cont.)

## Method 2:

- Form a schema for each entity set with all local and inherited attributes

schema	attributes
<i>person</i>	<i>ID, name, street, city</i>
<i>student</i>	<i>ID, name, street, city, tot_cred</i>
<i>employee</i>	<i>ID, name, street, city, salary</i>

- If specialization is total, the schema for the generalized entity set (*person*) not required to store information
  - Can be defined as a “view” relation containing union of specialization relations
  - But explicit schema may still be needed for foreign key constraints
- Drawback: *name, street* and *city* may be stored redundantly for people who are both students and employees



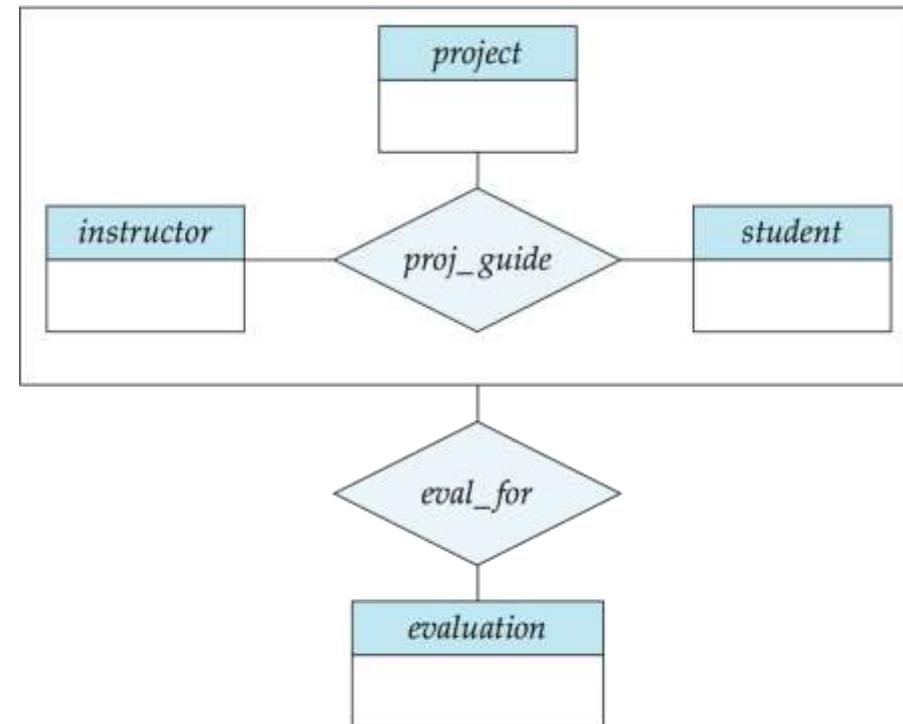
# Schemas Corresponding to Aggregation

- To represent aggregation, create a schema containing
  - primary key of the aggregated relationship,
  - the primary key of the associated entity set
  - any descriptive attributes



# Schemas Corresponding to Aggregation (Cont.)

- For example, to represent aggregation manages between relationship works\_on and entity set manager, create a schema  $\text{eval\_for}(s\_ID, project\_id, i\_ID, evaluation\_id)$
- Schema *proj\_guide* is redundant provided we are willing to store null values for attribute *manager\_name* in relation on schema *manages*





# E-R Design Decisions

- The use of an attribute or entity set to represent an object.
- Whether a real-world concept is best expressed by an entity set or a relationship set.
- The use of a ternary relationship versus a pair of binary relationships.
- The use of a strong or weak entity set.
- The use of specialization/generalization – contributes to modularity in the design.
- The use of aggregation – can treat the aggregate entity set as a single unit without concern for the details of its internal structure.



**How about doing another ER design  
interactively on the board?**

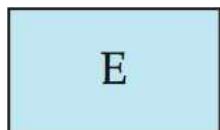
**Database System Concepts, 6<sup>th</sup> Ed.**

©Silberschatz, Korth and Sudarshan

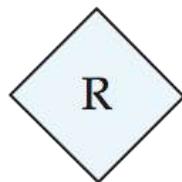
See [www.db-book.com](http://www.db-book.com) for conditions on re-use



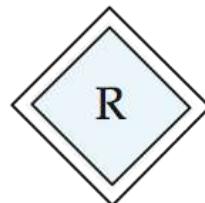
# Summary of Symbols Used in E-R Notation



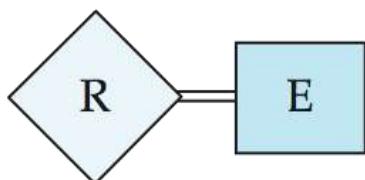
entity set



relationship set



identifying  
relationship set  
for weak entity set



total participation  
of entity set in  
relationship

E
A1
A2
A2.1
A2.2
{A3}
A4()

attributes:  
simple (A1),  
composite (A2) and  
multivalued (A3)  
derived (A4)

E
<u>A1</u>

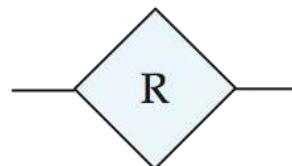
primary key

E
A1
.....

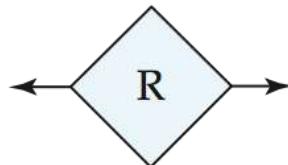
discriminating  
attribute of  
weak entity set



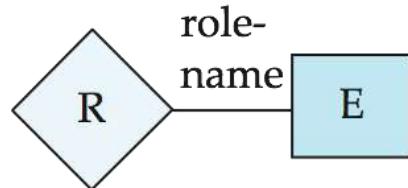
# Symbols Used in E-R Notation (Cont.)



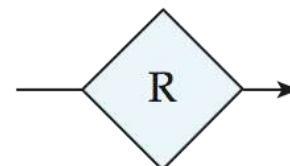
many-to-many  
relationship



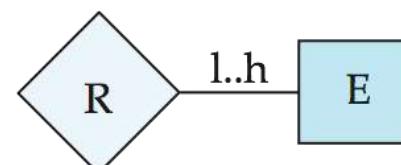
one-to-one  
relationship



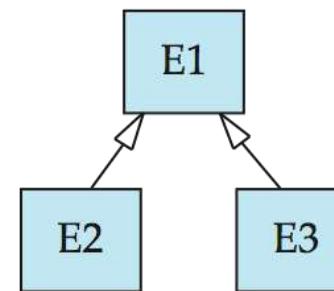
role indicator



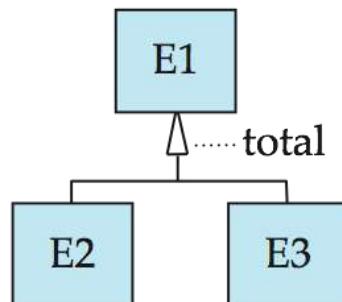
many-to-one  
relationship



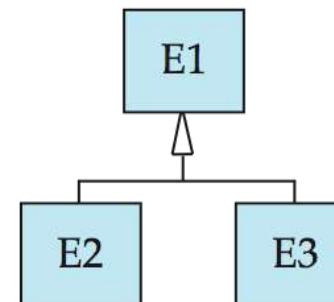
cardinality  
limits



ISA: generalization  
or specialization



total (disjoint)  
generalization



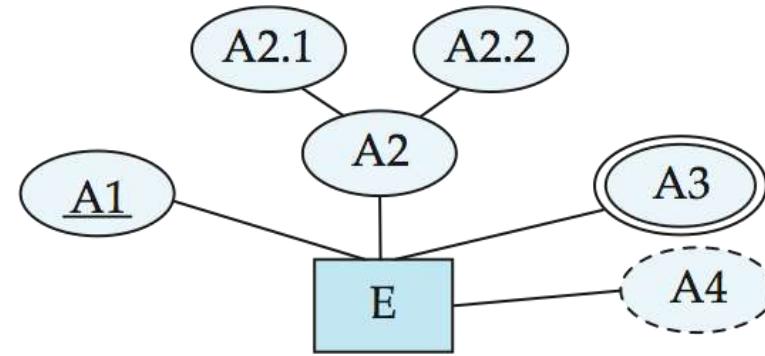
disjoint  
generalization



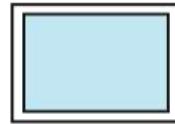
# Alternative ER Notations

- Chen, IDE1FX, ...

entity set E with  
simple attribute A1,  
composite attribute A2,  
multivalued attribute A3,  
derived attribute A4,  
and primary key A1



weak entity set



generalization



total  
generalization

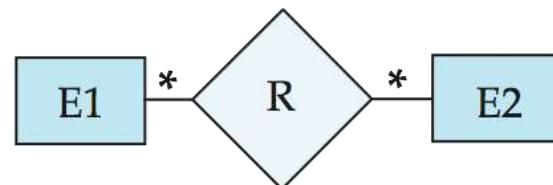




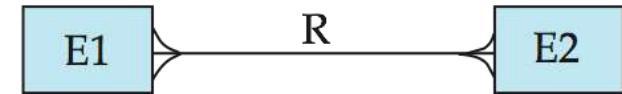
# Alternative ER Notations

Chen

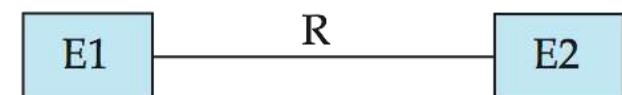
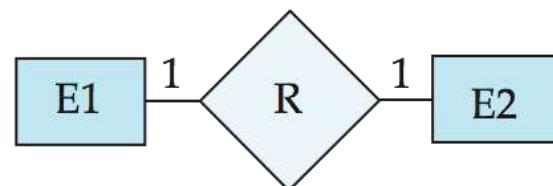
many-to-many  
relationship



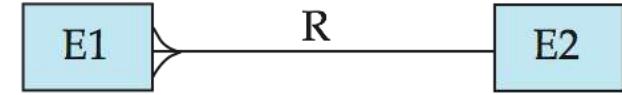
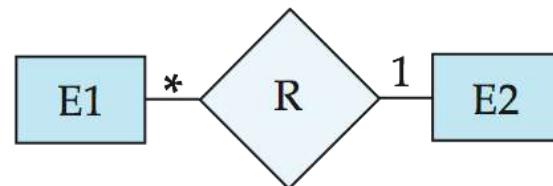
IDE1FX (Crows feet notation)



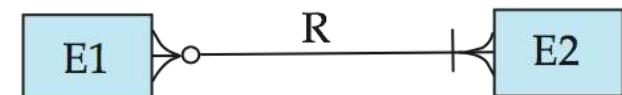
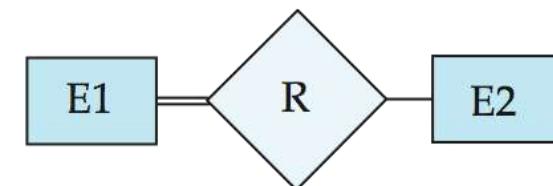
one-to-one  
relationship



many-to-one  
relationship



participation  
in R: total (E1)  
and partial (E2)





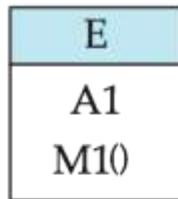
# UML

- **UML**: Unified Modeling Language
- UML has many components to graphically model different aspects of an entire software system
- UML Class Diagrams correspond to E-R Diagram, but several differences.

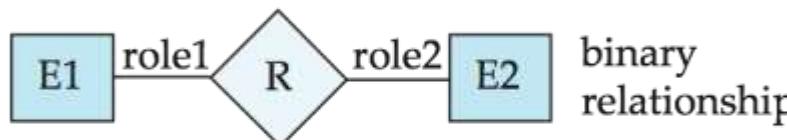


# ER vs. UML Class Diagrams

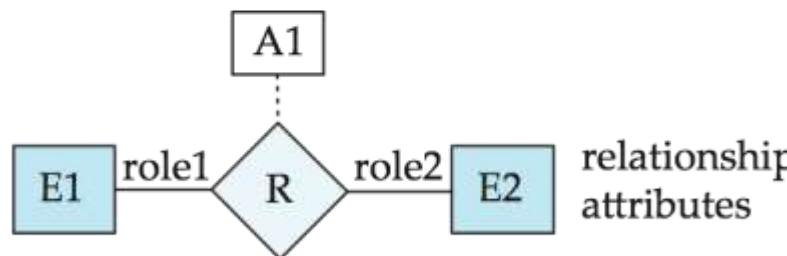
## ER Diagram Notation



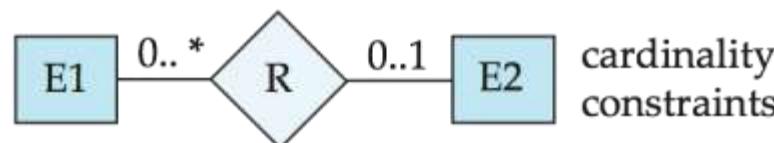
entity with attributes (simple, composite, multivalued, derived)



binary relationship

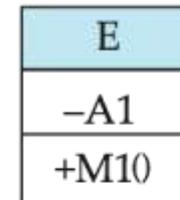


relationship attributes

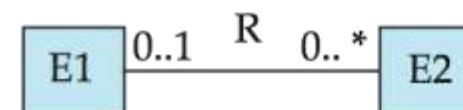
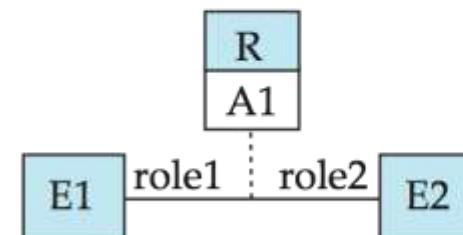
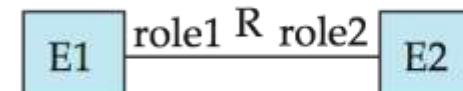


cardinality constraints

## Equivalent in UML



class with simple attributes and methods (attribute prefixes: + = public, - = private, # = protected)

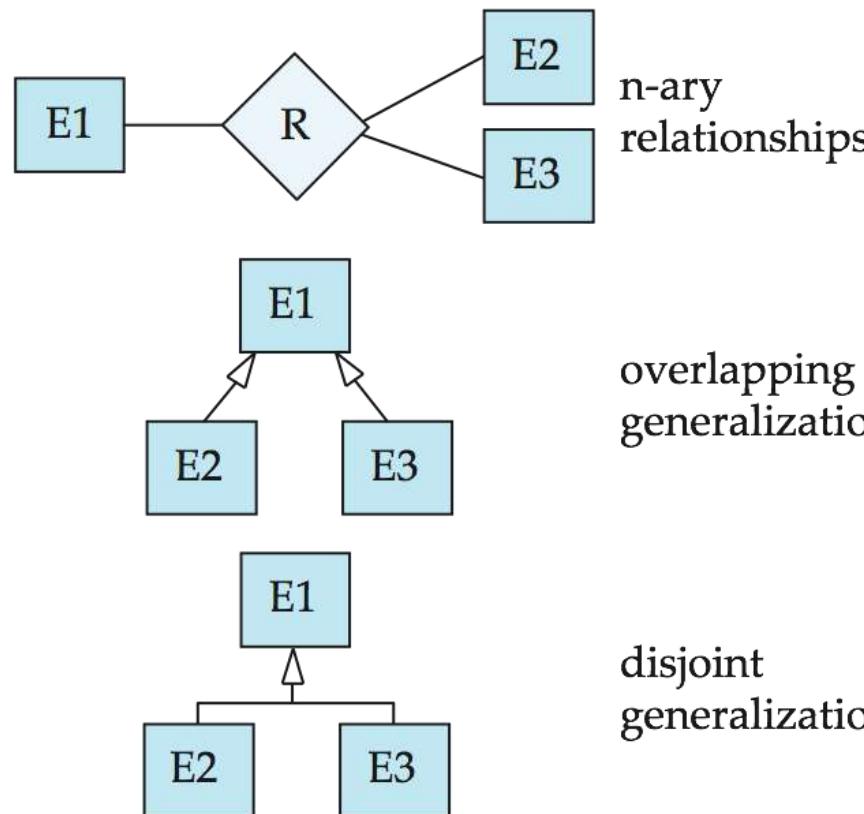


\*Note reversal of position in cardinality constraint depiction

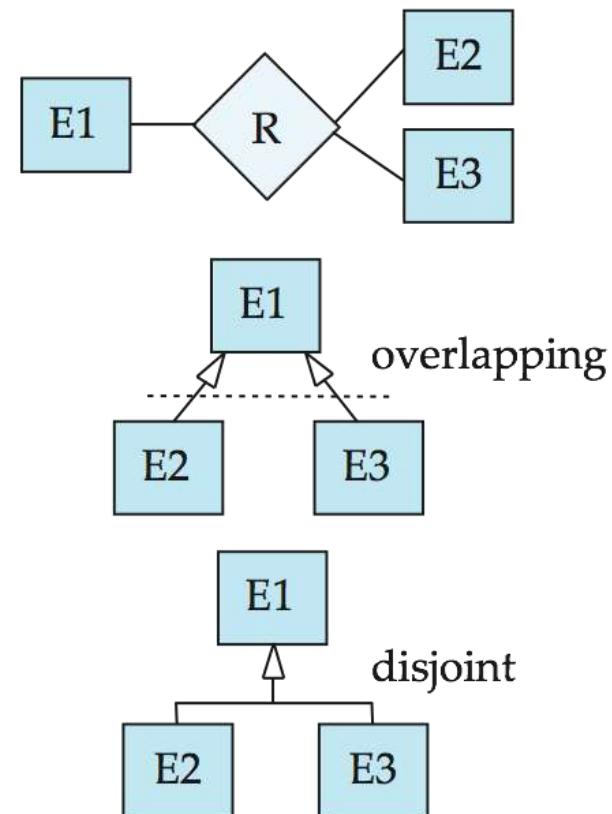


# ER vs. UML Class Diagrams

## ER Diagram Notation



## Equivalent in UML



\*Generalization can use merged or separate arrows independent of disjoint/overlapping



# UML Class Diagrams (Cont.)

- Binary relationship sets are represented in UML by just drawing a line connecting the entity sets. The relationship set name is written adjacent to the line.
- The role played by an entity set in a relationship set may also be specified by writing the role name on the line, adjacent to the entity set.
- The relationship set name may alternatively be written in a box, along with attributes of the relationship set, and the box is connected, using a dotted line, to the line depicting the relationship set.



The picture can't be displayed.

# End of Chapter 7

**Database System Concepts, 6<sup>th</sup> Ed.**

©Silberschatz, Korth and Sudarshan

See [www.db-book.com](http://www.db-book.com) for conditions on re-use



# Figure 7.01

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

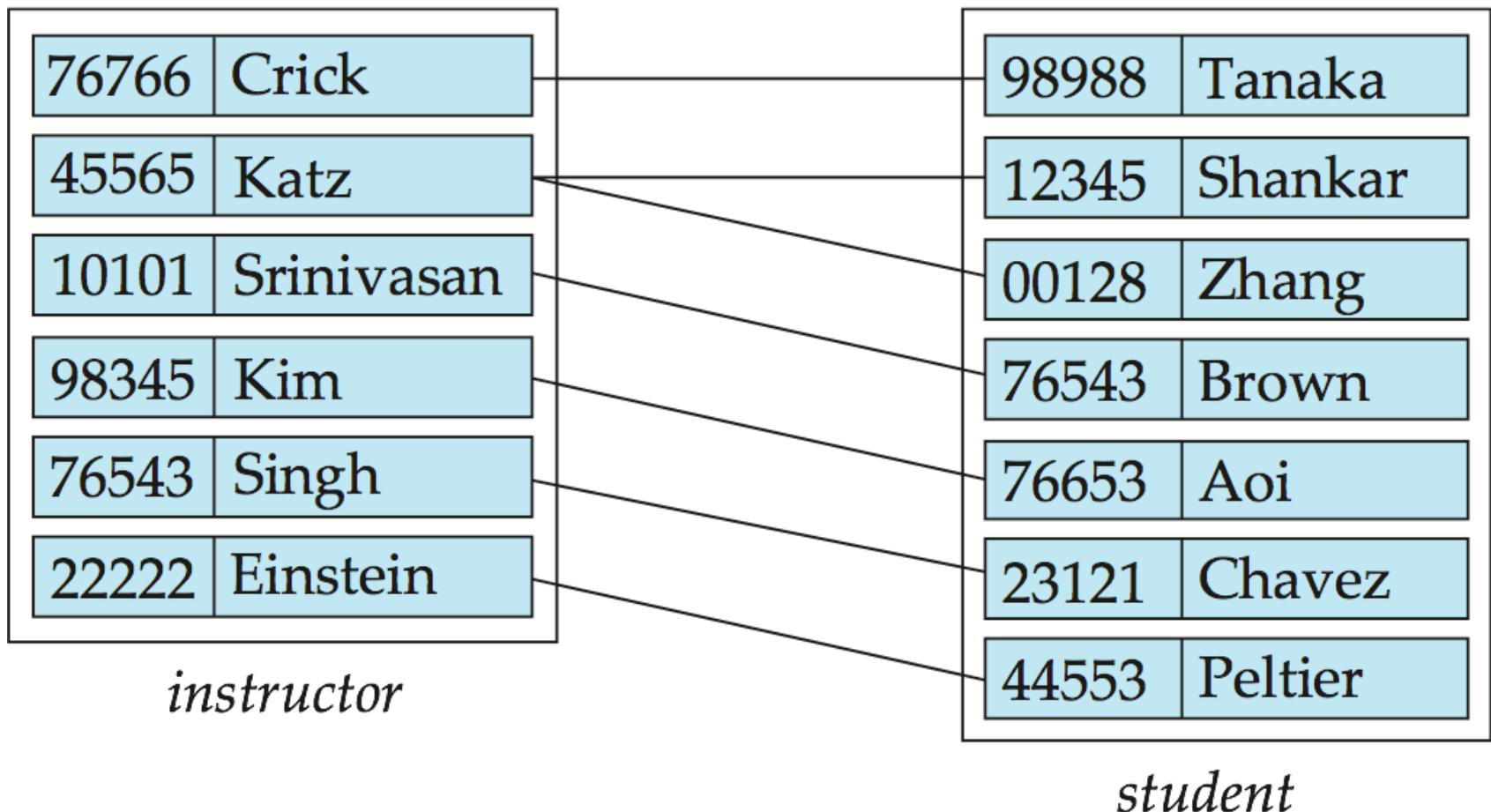
*instructor*

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

*student*

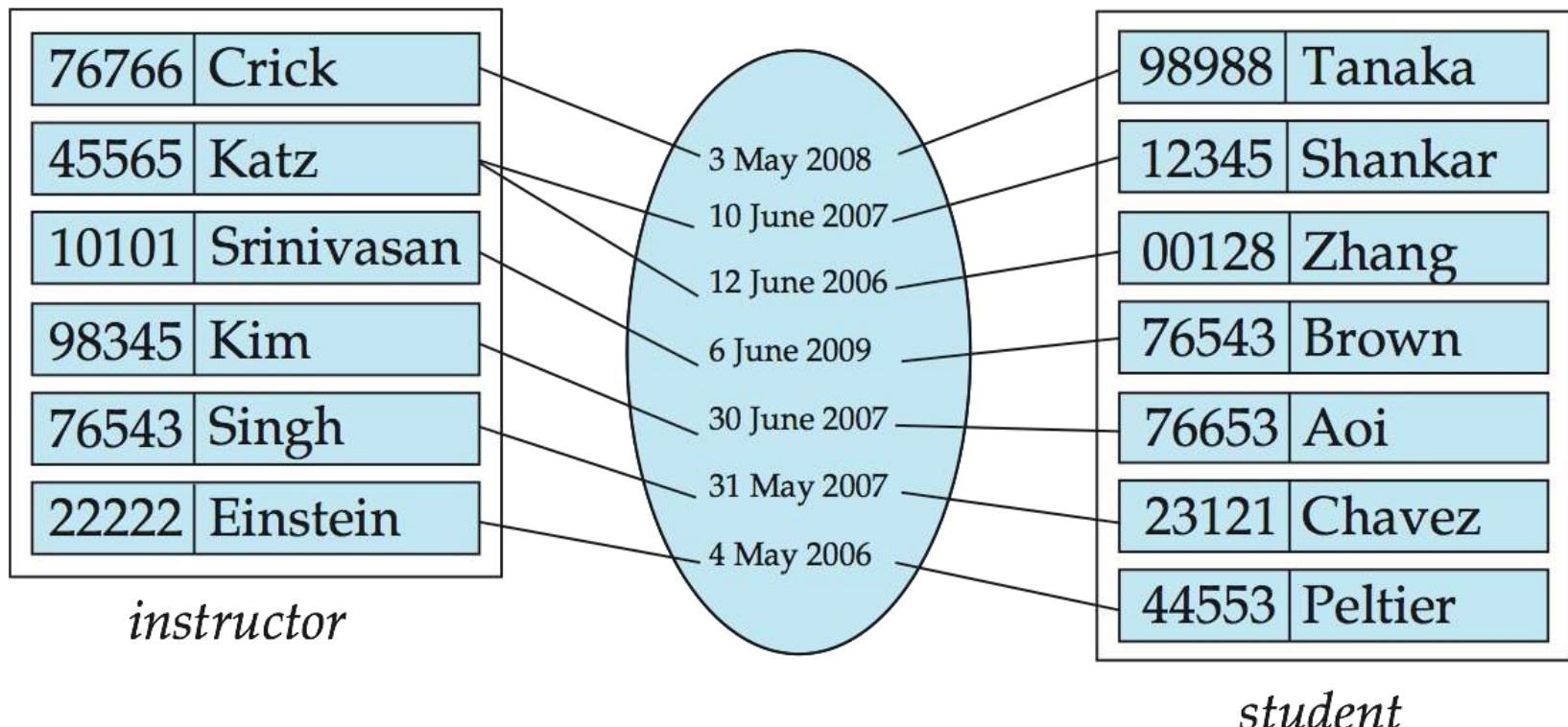


# Figure 7.02



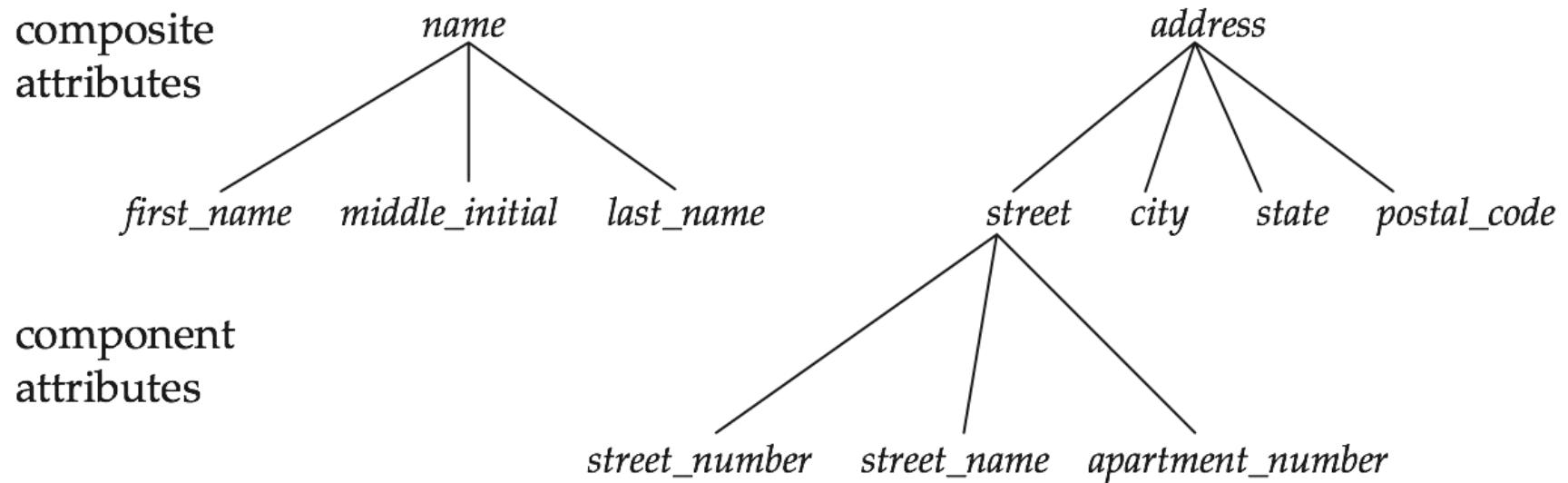


# Figure 7.03



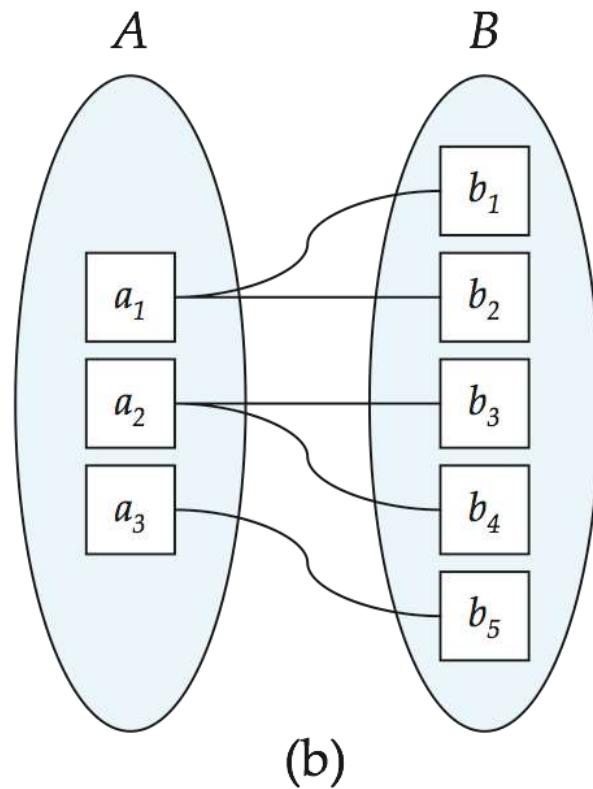
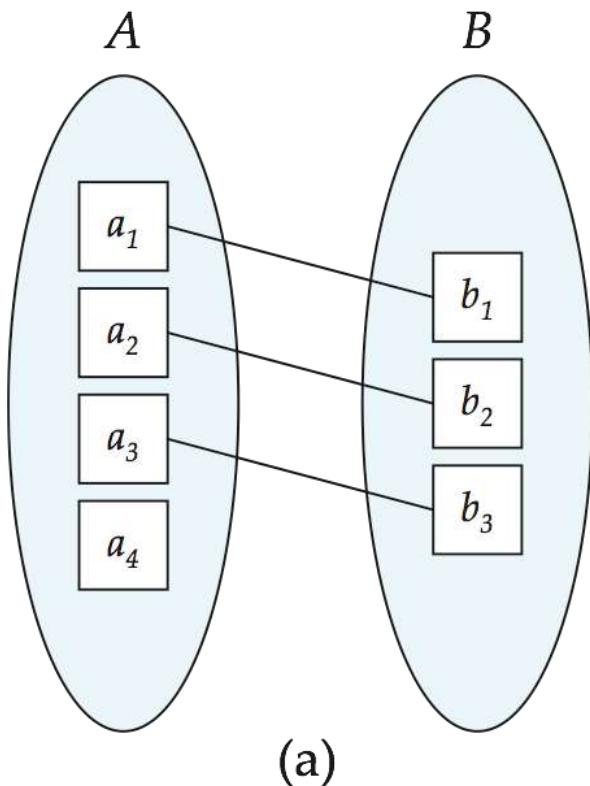


# Figure 7.04



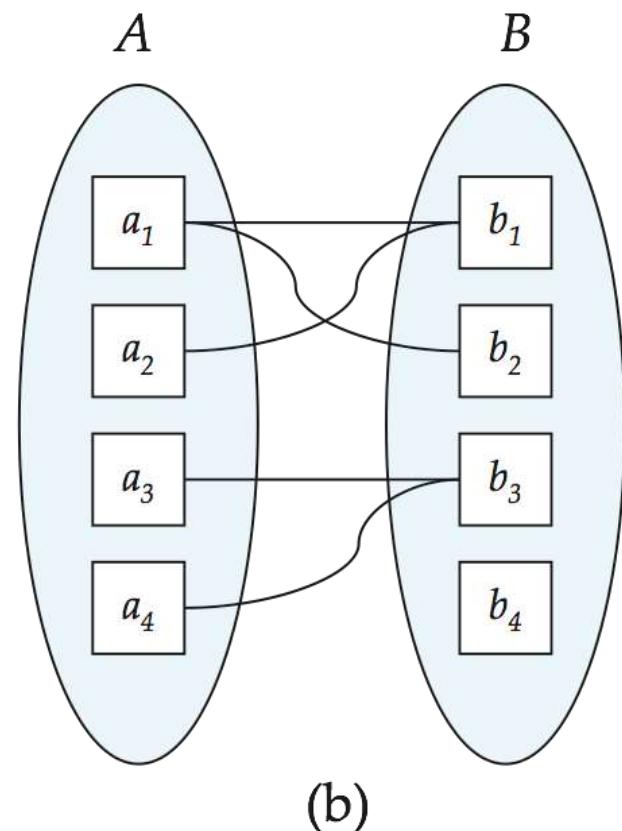
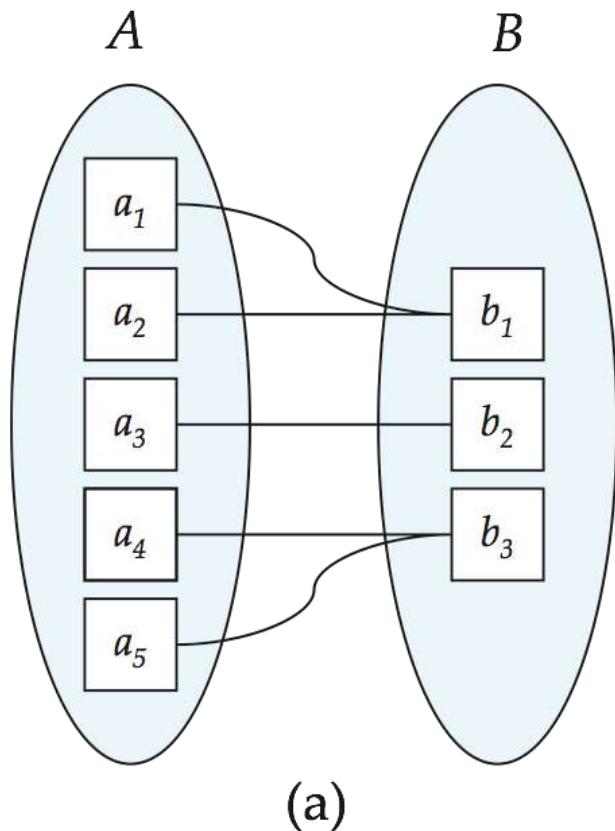


# Figure 7.05



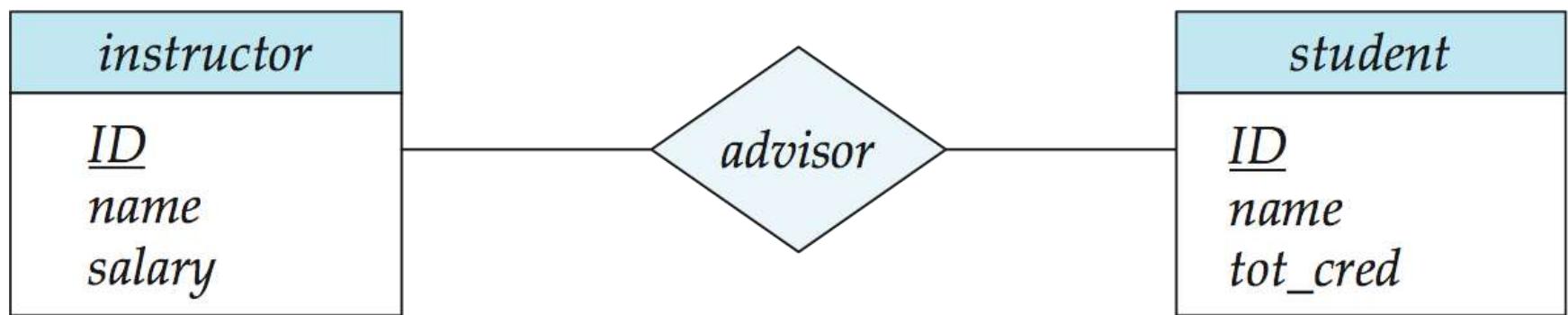


# Figure 7.06



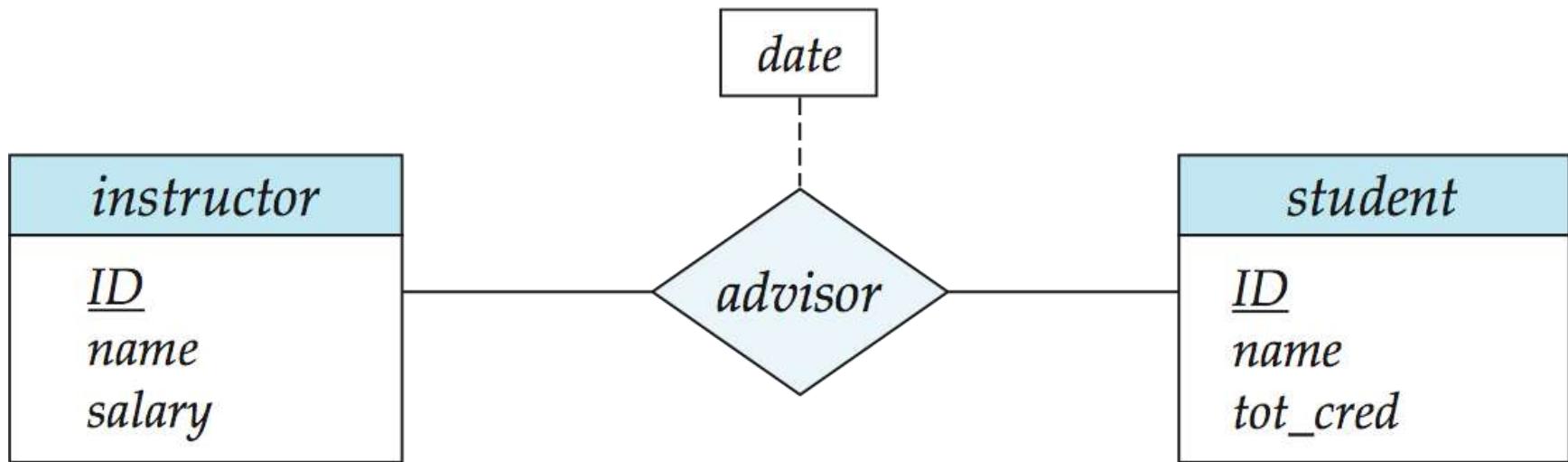


# Figure 7.07



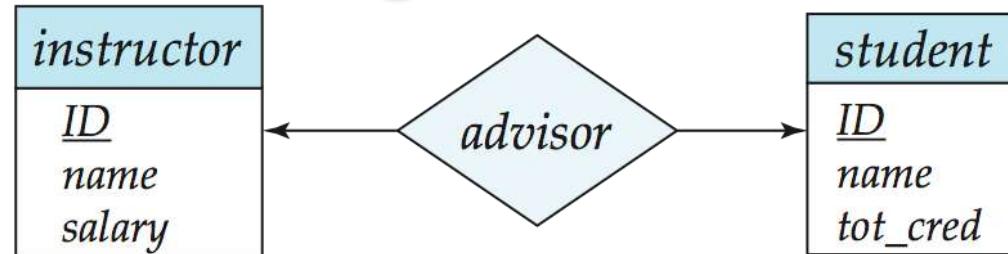


# Figure 7.08

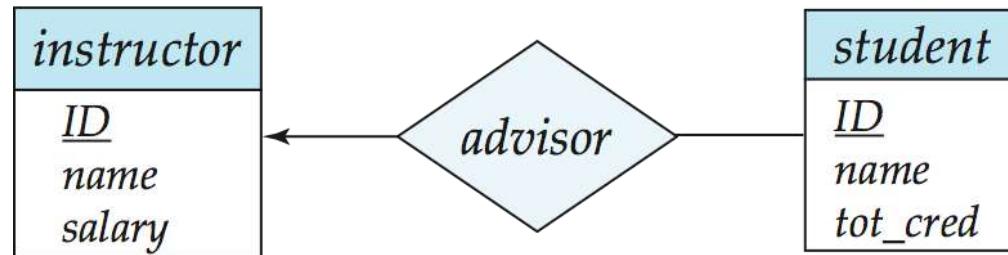




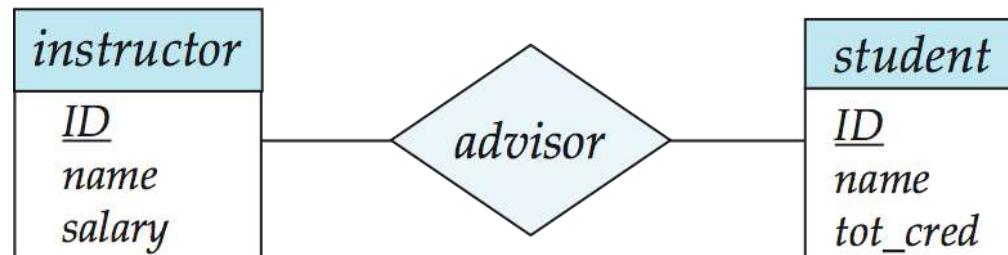
# Figure 7.09



(a)



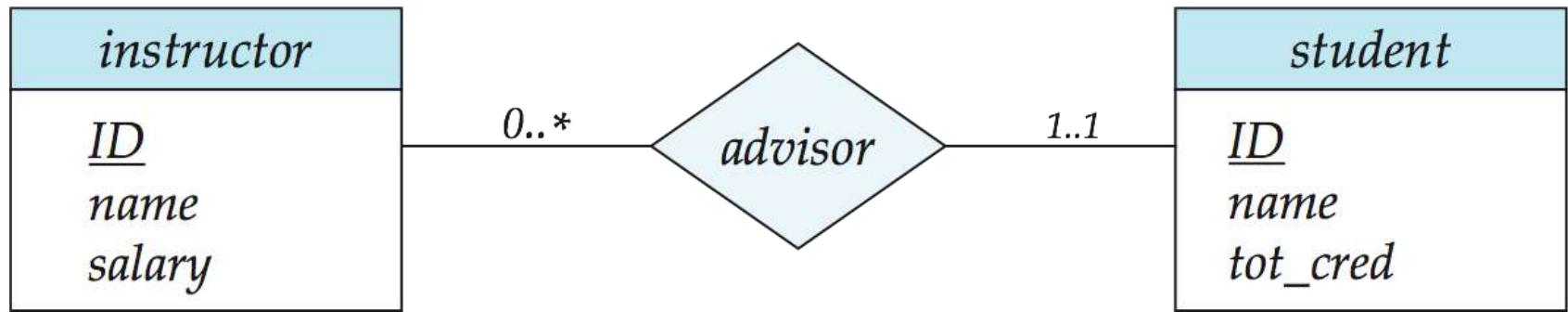
(b)



(c)



# Figure 7.10



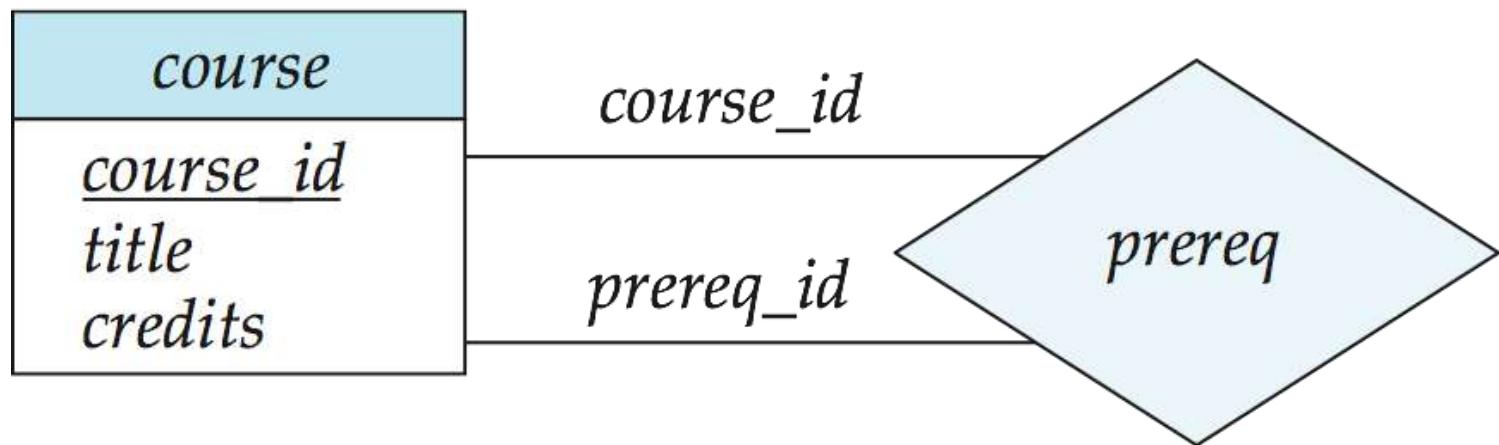


# Figure 7.11

<i>instructor</i>
<u>ID</u>
<i>name</i>
<i>first_name</i>
<i>middle_initial</i>
<i>last_name</i>
<i>address</i>
<i>street</i>
<i>street_number</i>
<i>street_name</i>
<i>apt_number</i>
<i>city</i>
<i>state</i>
<i>zip</i>
{ <i>phone_number</i> }
<i>date_of_birth</i>
<i>age</i> ( )

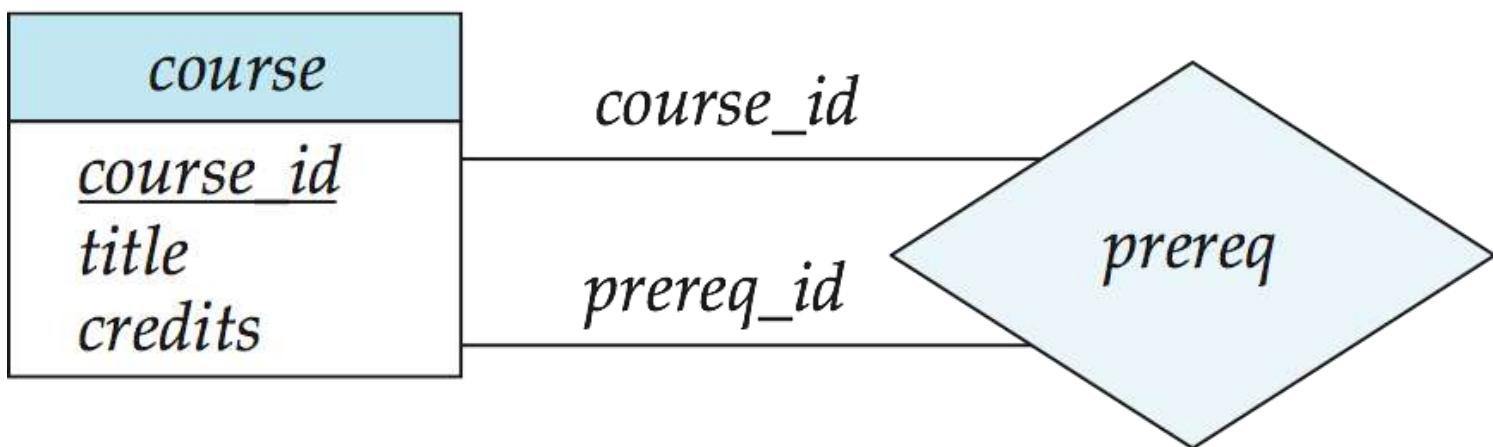


# Figure 7.12



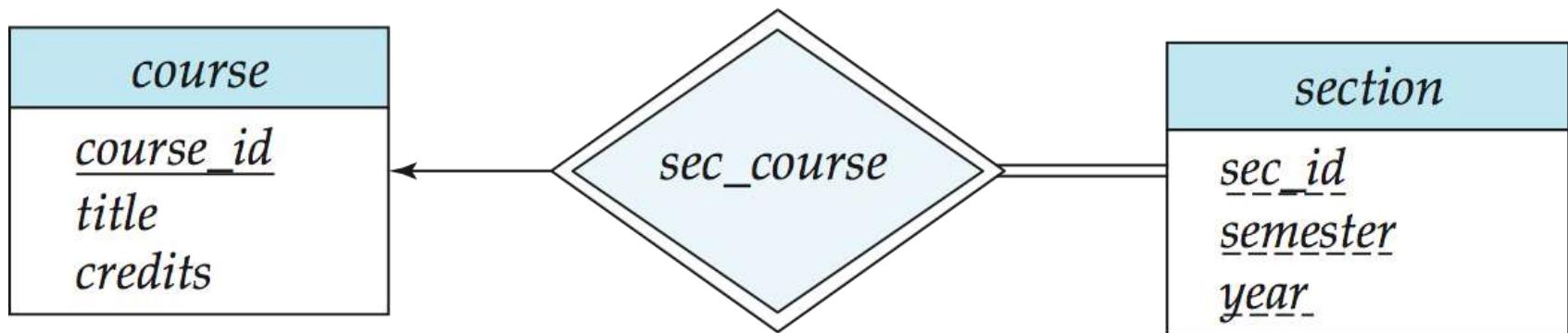


# Figure 7.13



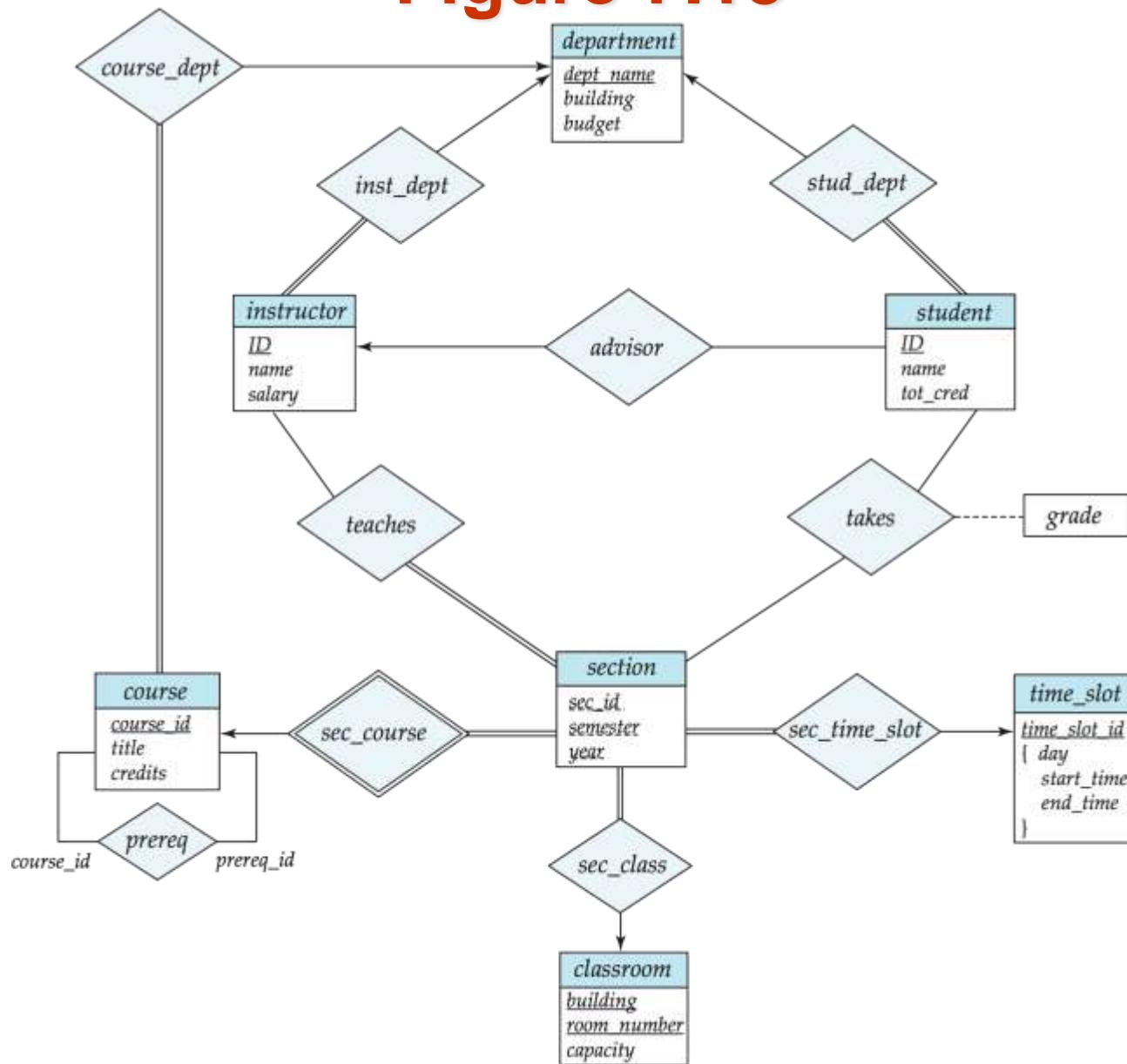


# Figure 7.14





# Figure 7.15

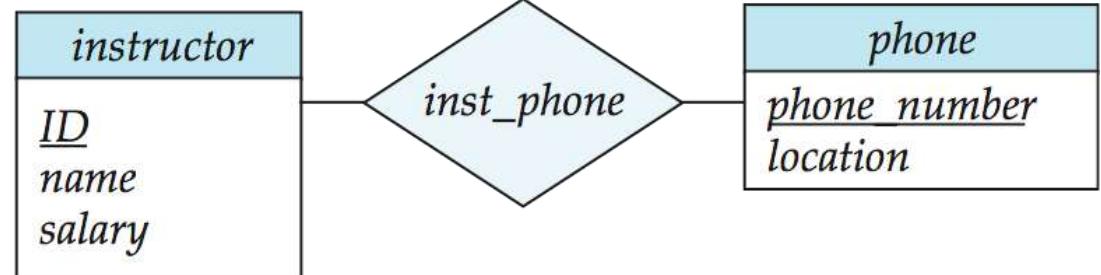




# Figure 7.17

<i>instructor</i>
<u>ID</u>
<i>name</i>
<i>salary</i>
<i>phone_number</i>

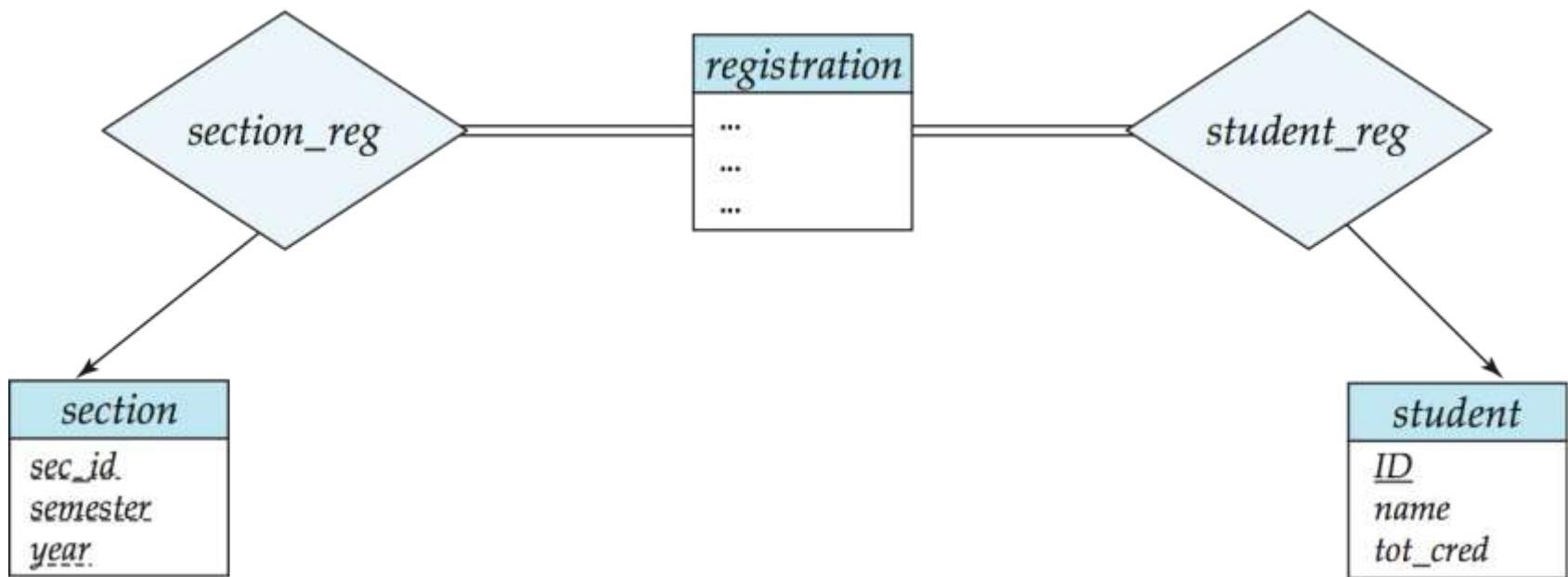
(a)



(b)

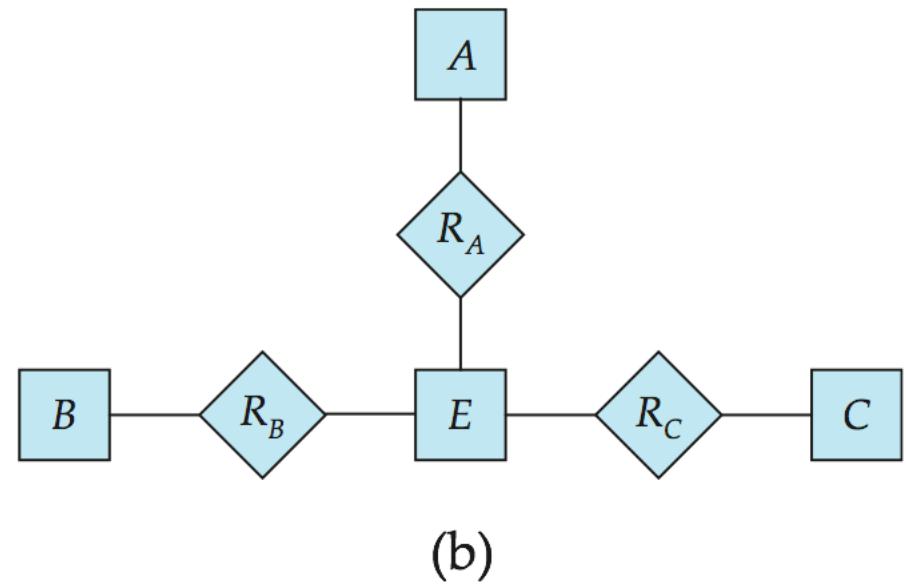
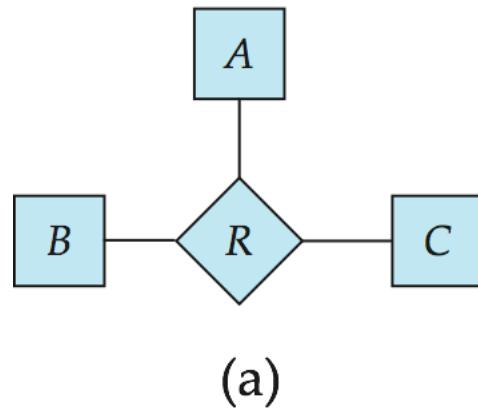


# Figure 7.18



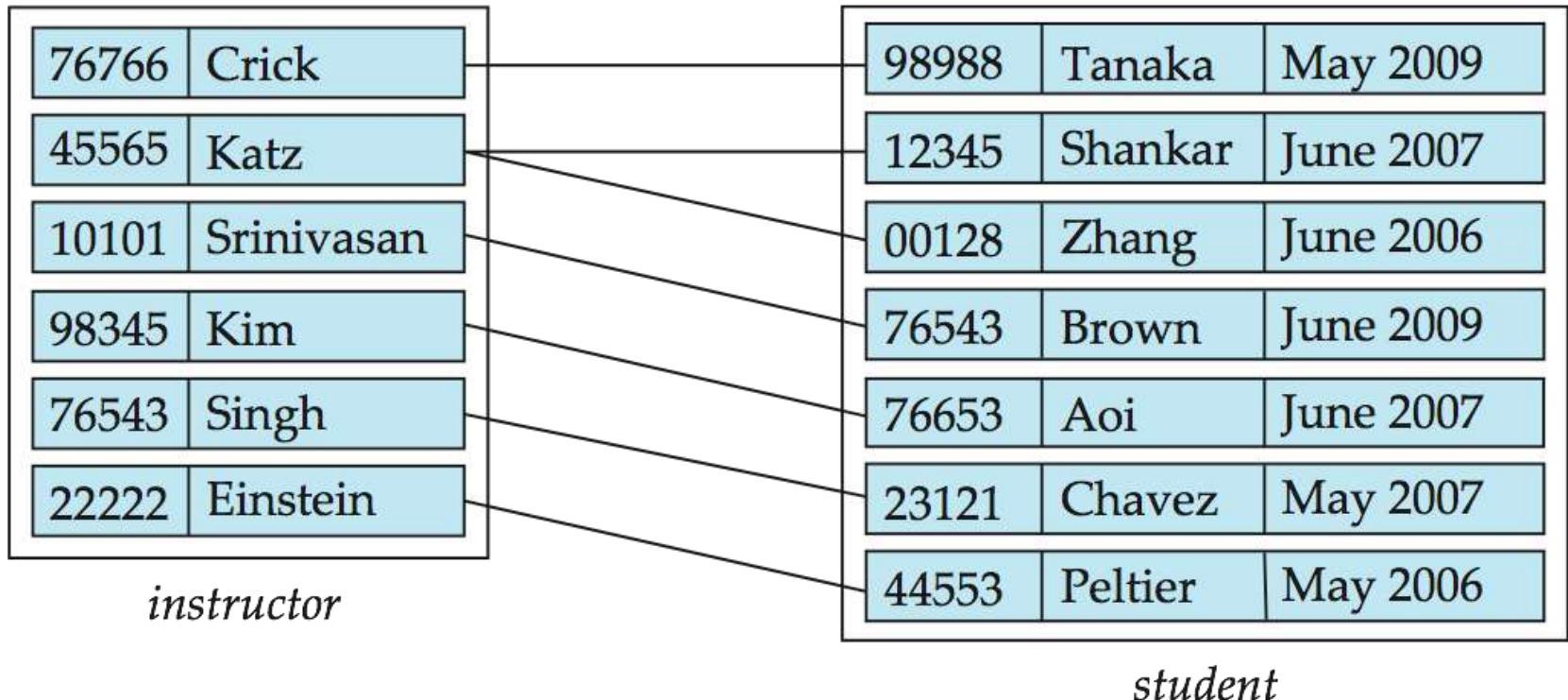


# Figure 7.19



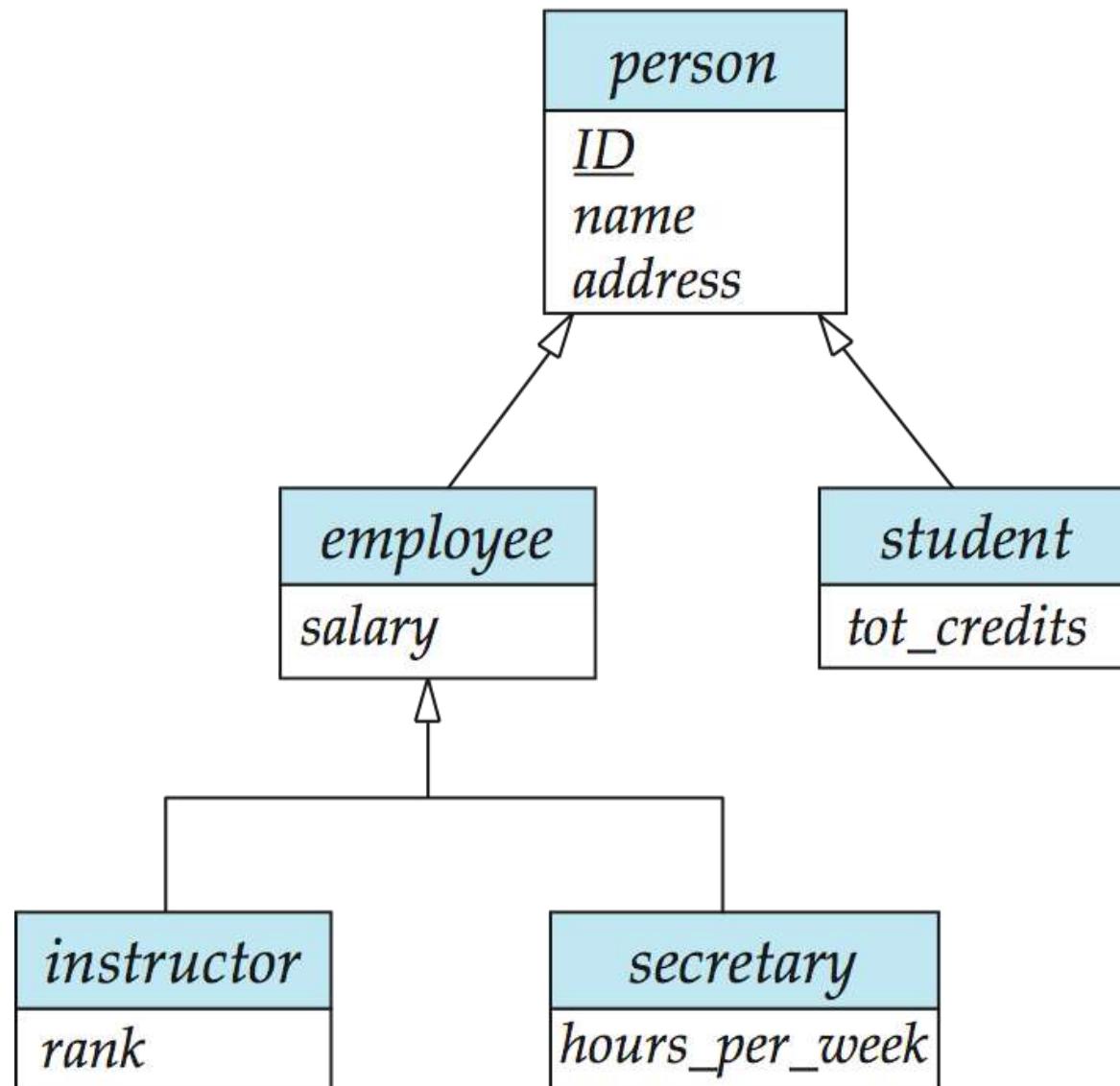


# Figure 7.20



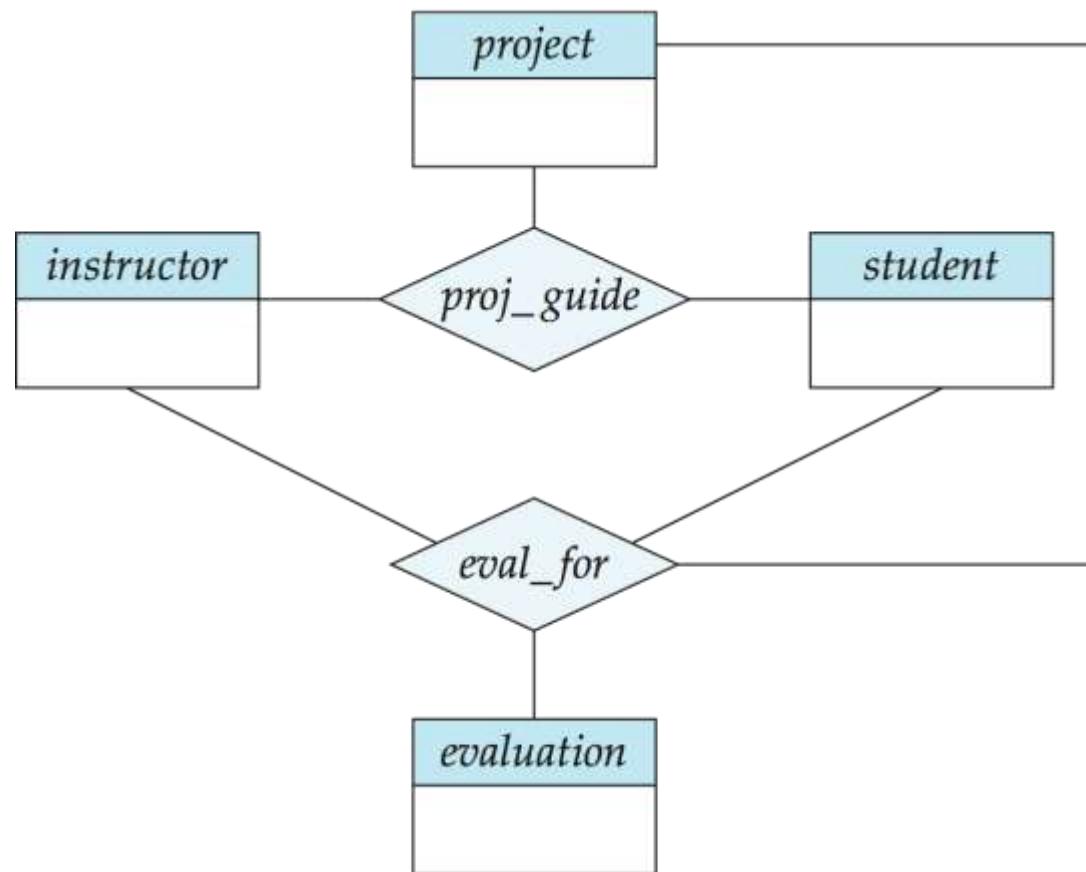


# Figure 7.21



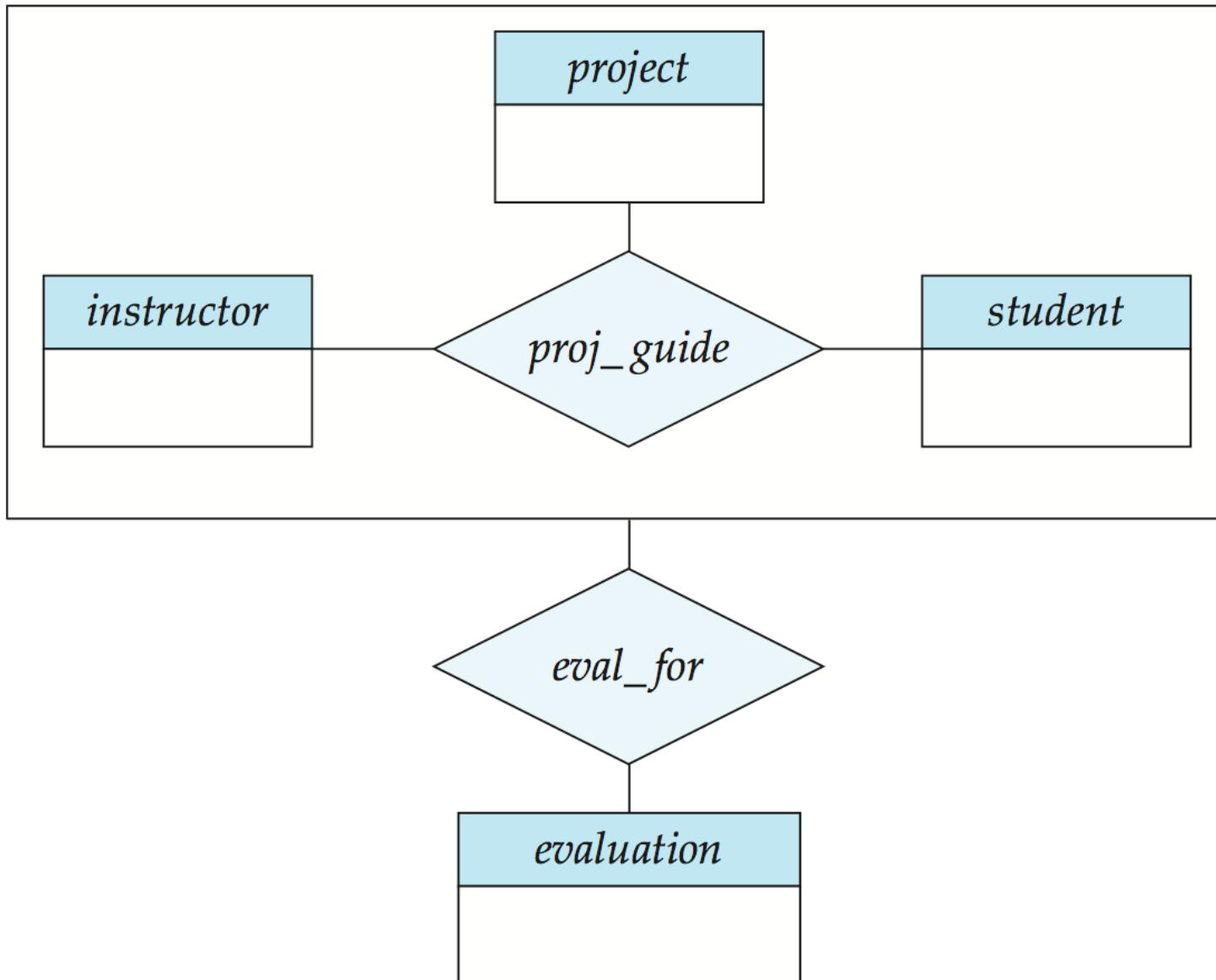


# Figure 7.22



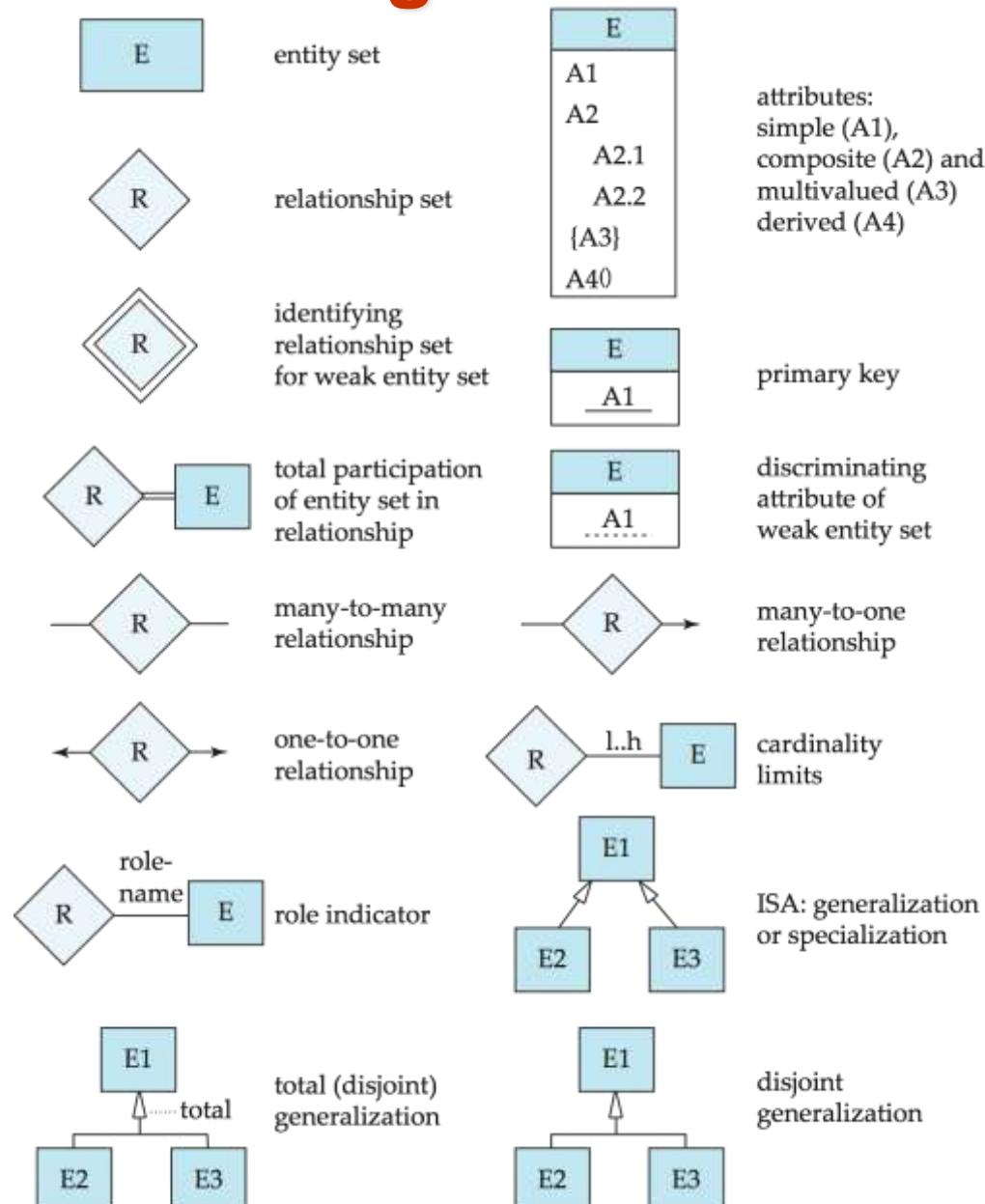


# Figure 7.23





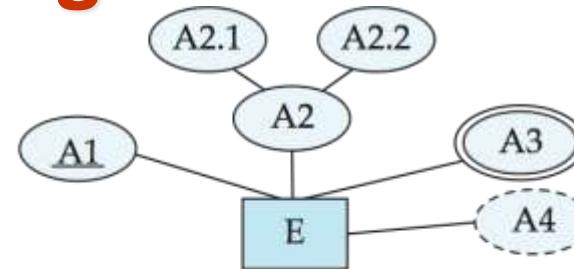
# Figure 7.24



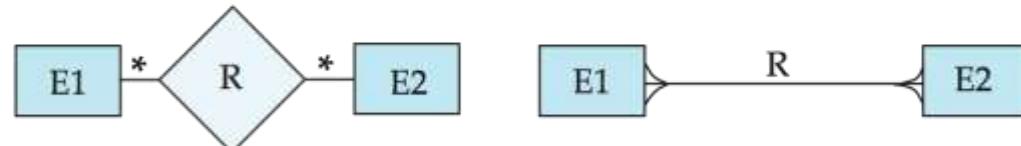


# Figure 7.25

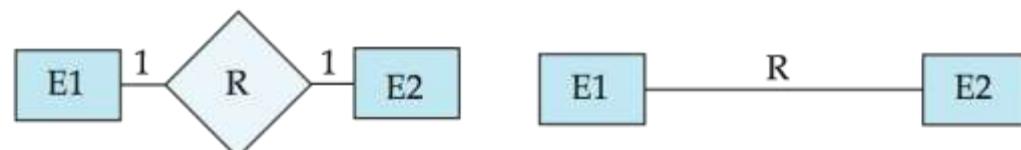
entity set E with simple attribute A1, composite attribute A2, multivalued attribute A3, derived attribute A4, and primary key A1



many-to-many relationship



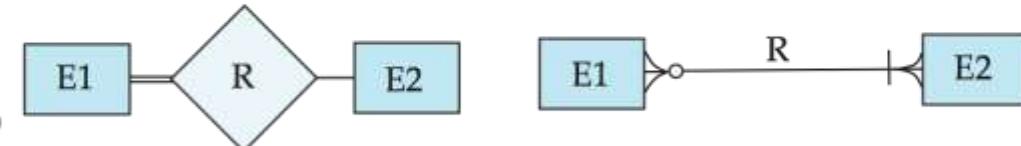
one-to-one relationship



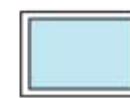
many-to-one relationship



participation in R: total (E1) and partial (E2)



weak entity set



generalization



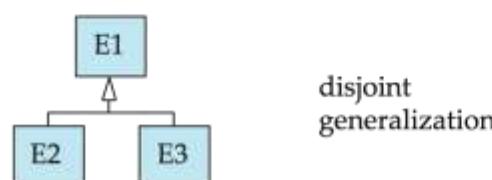
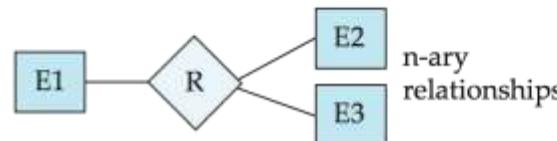
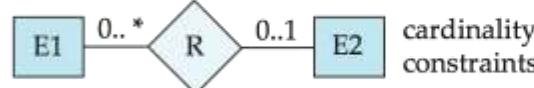
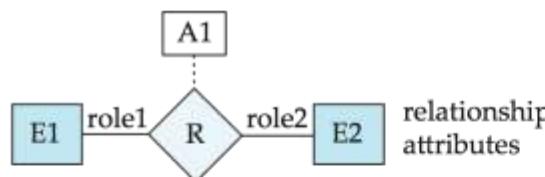
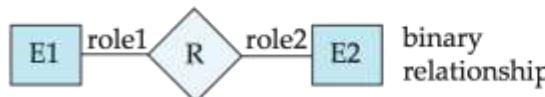
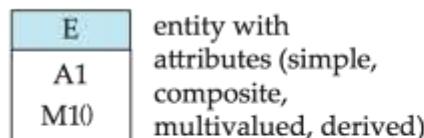
total generalization



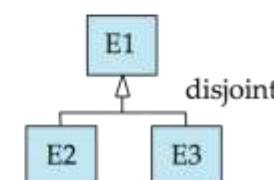
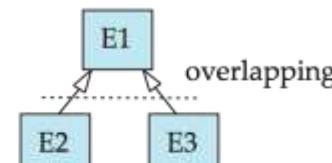
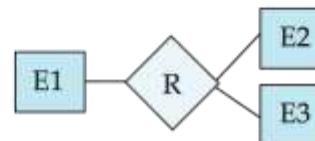
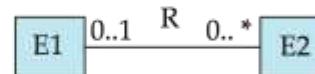
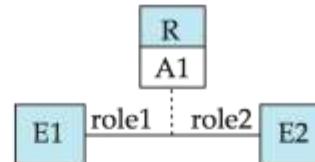
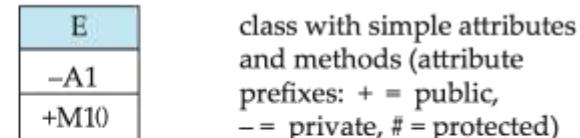


# Figure 7.26

## ER Diagram Notation

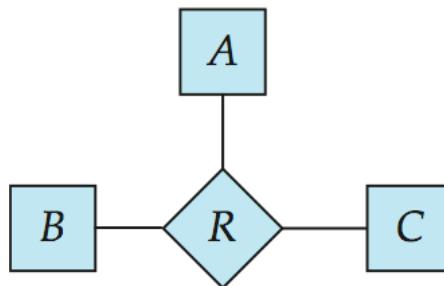


## Equivalent in UML

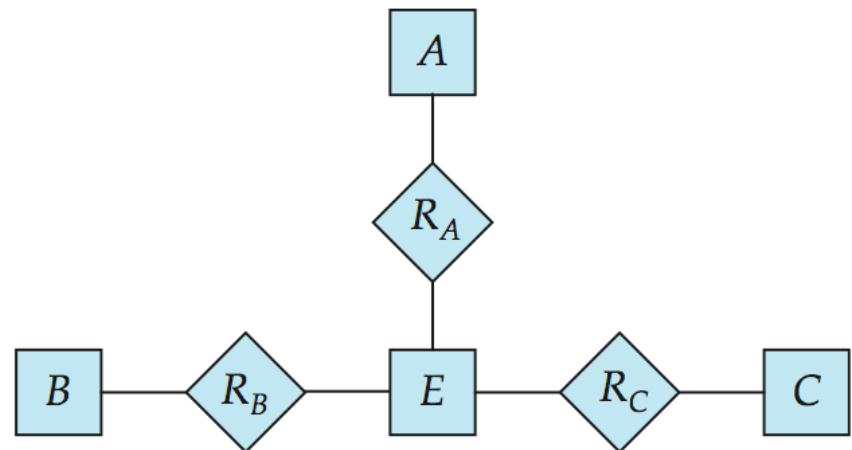




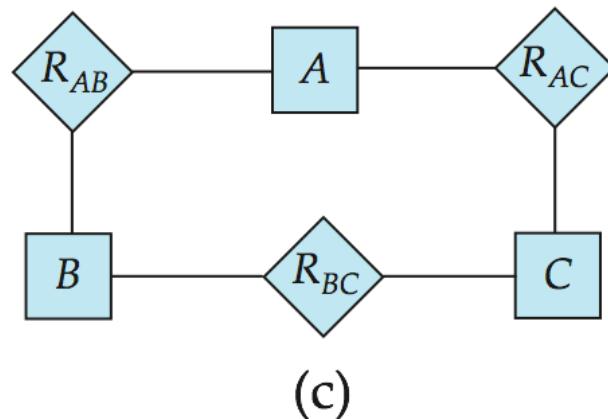
# Figure 7.27



(a)



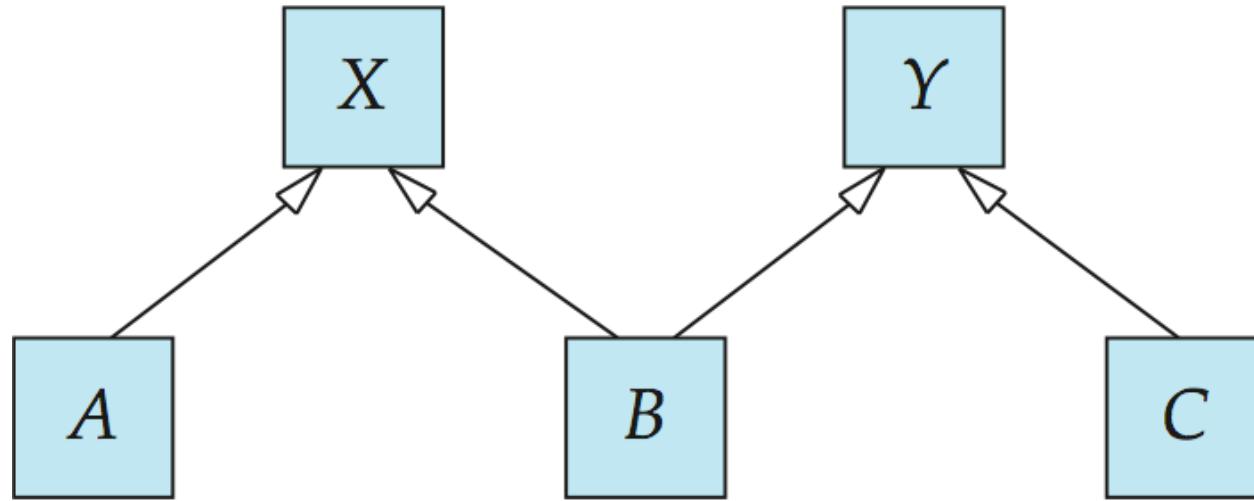
(b)



(c)



# Figure 7.28





# Figure 7.29

