# Chapter 7: Entity-Relationship Model

Database System Concepts, 6th Ed.

# 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

# The Entity-Relationship Model

- The **entity-relationship (E-R)** data model was developed to facilitate database design by allowing specification of an *enterprise schema* that represents the overall logical structure of a database.
- The E-R model is very useful in mapping the meanings and interactions of real-world enterprises onto a conceptual schema.
- Because of this usefulness, many database-design tools draw on concepts from the E-R model.
- The E-R data model employs three basic concepts:
  - entity sets,
  - relationship sets, and
  - attributes
- The E-R model also has an associated diagrammatic representation, called the E-R diagram.

# Entity

- A *database* can be modeled as:
  - a collection of entities,
  - relationship among entities.
- **Entity**
- An **entity** is a “thing” or “object” in the real world that is distinguishable from all other objects.
  - Example:  specific person, company, event, plant
- An entity has a set of properties and the values for some set of properties (**attributes)** may uniquely identify an entity.
  - Example: person (person_id, person_name, address)
  - Example: course (course_id, course_name, department, credits)
- An entity may be concrete, such as a person or a book, or it may be abstract, such as loan, or a holiday, or a concept.

# Entity Set

- **Entity Set:** An **entity set** is a set of entities of the same type that share the same properties or attributes.
  - Example: set of all persons, companies, trees, holidays
- Entity sets do not need to be disjoint. For example, it is possible to define the entity set of all people in a university (*person).* A *person* entity may be an *instructor* entity, a *student* entity, both, or neither.

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*

# Attributes

- An entity is represented by a set of attributes, that is descriptive properties possessed by all members of an entity set.
- The designation of an attribute for an entity set expresses that the database stores similar information concerning each entity in the entity set; however, each entity may have its own value for each attribute.
  - Example:

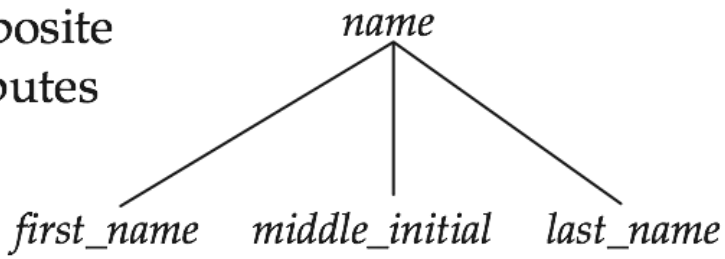
    *instructor* = (*ID, name, street, city, salary* )
    *course*= (*course_id, title, credits*)
- **Domain** – the set of permitted values for each attribute
  - customer_name – Set of all text strings of a certain length
- A database thus includes a collection of entity sets, each of which contains any number of entities of same type.

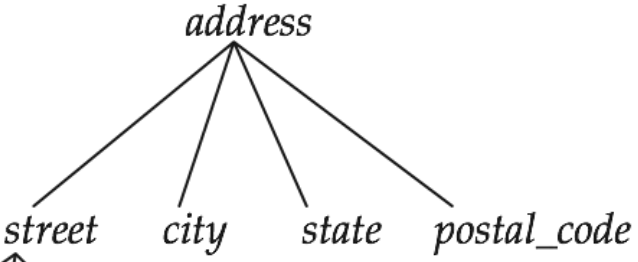
# Attribute Types

- **Simple** and **composite** attributes.
  - **Simple** – When an attribute is not possible to divide into subparts, customer city, country
  - **Composite** – Composite attributes can be divided into subparts i.e. other attributes. Composite attributes help us to group together the related attributes, making the modeling cleaner.
  - *name* – first_name, middle_name, last_name
    *address* – street, city, state, zip_code (postal code)
    *street* – street_number, street_name, apartment_number
- **Single-valued** and **multivalued** attributes
  - **Single-valued** – when there exist a single value for a particular entity
    - Example: *course_name, salary*
  - **Multivalued** - When there exist a set of values for a specific entity
    - Example: *phone_numbers, dependent_names*

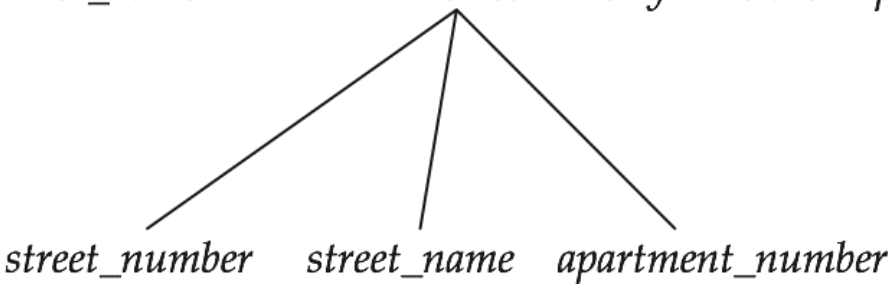
# Composite Attributes

composite attributes




component attributes



# Attribute Types

- An *instructor* may have zero, one, or several phone numbers, and different instructors may have different numbers of phones.
- Where appropriate, upper and lower bounds may be placed on the number of values in a multivalued attribute. For example, a university may limit the number of phone numbers recorded for a single instructor to two.

- **Derived** attributes
  - **Derived** attributes - The value of this type of attribute can be derived from the values of other related attributes or entities.
  - The value of a derived attribute is not stored, but is computed when required.
    - Example: age, given date_of_birth
    - Example: loan_held, given a customer's all loans
    - Example: service_length, given date_of_joining

# 7.2.2 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 set of relationships of the same type. Formally, it is a mathematical relation among $n \geq 2$ (possibly non-distinct) entities.

- If $E_1, E_2, \ldots, E_n$ are entity sets, then a relationship set $R$ is a subset of

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

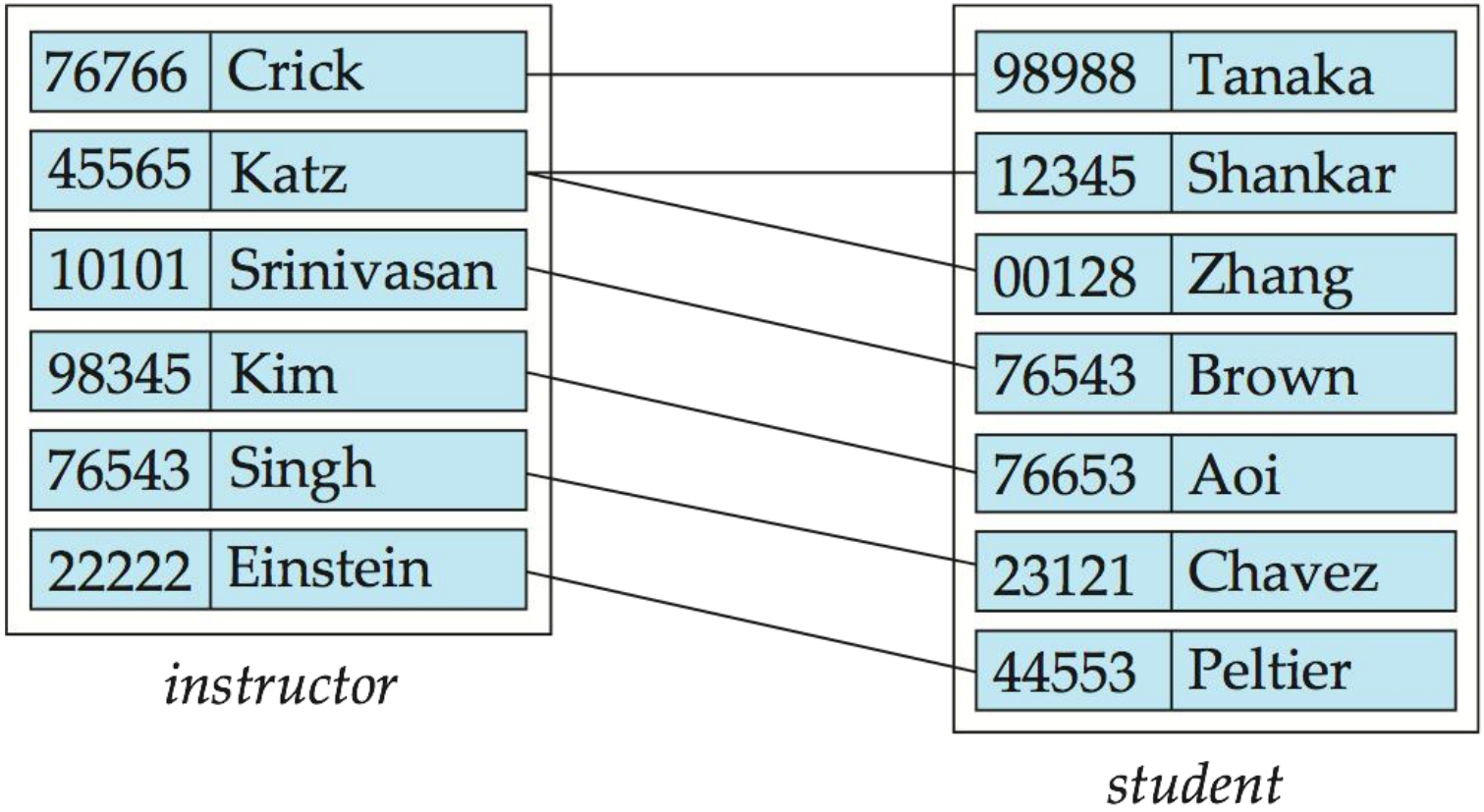
  where $(e_1, e_2, \ldots, e_n)$ is a relationship

  - Example:

$$(44553,22222) \in advisor$$

# Relationship Sets (Cont.)

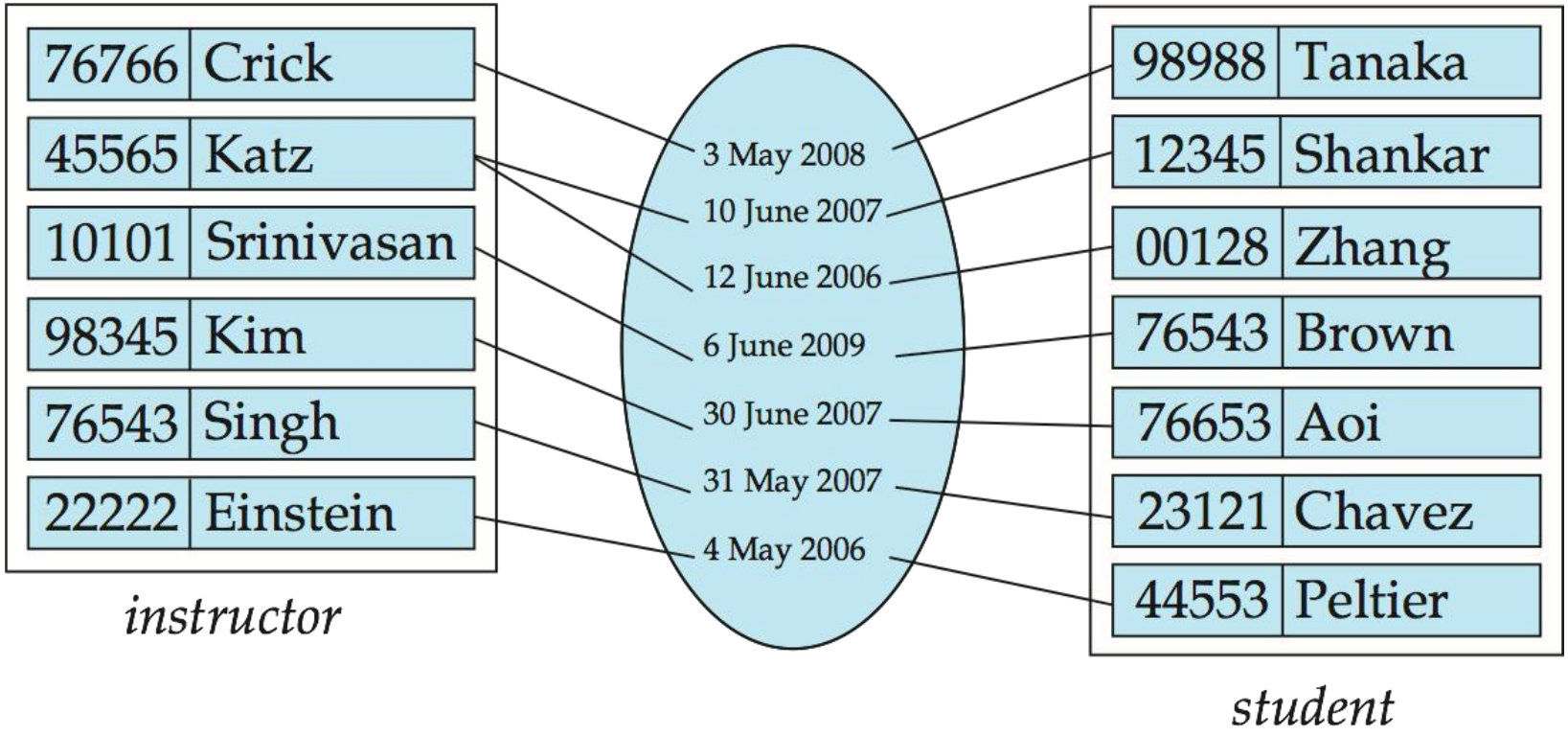
- The relationship set *advisor* to denote the association between instructors and students.




- The relationship set *takes* to denote the association between a student and the course sections in which that student is enrolled.

# 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.



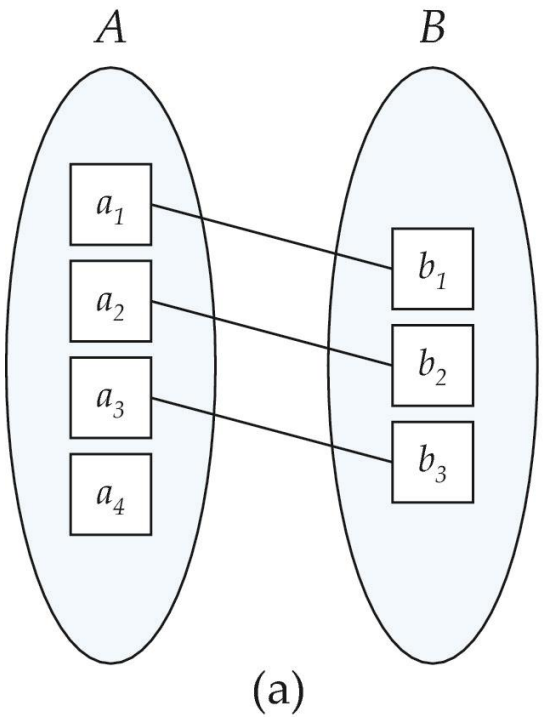
# Degree of a Relationship Set

- **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*

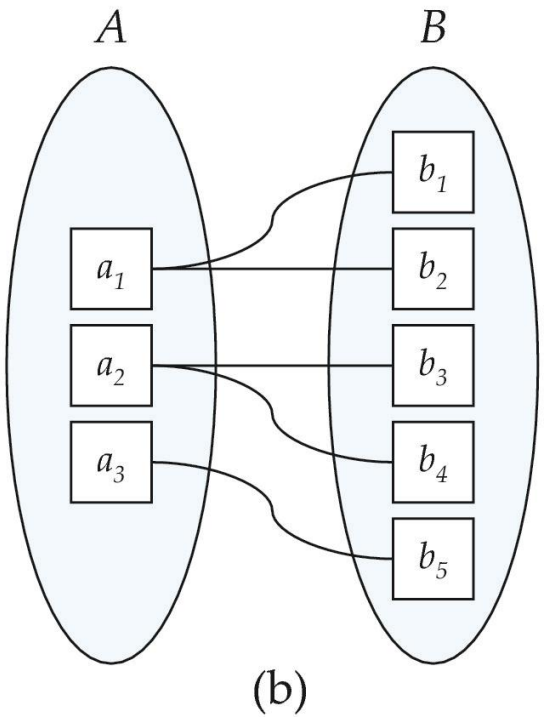
# Mapping Cardinality Constraints

- 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



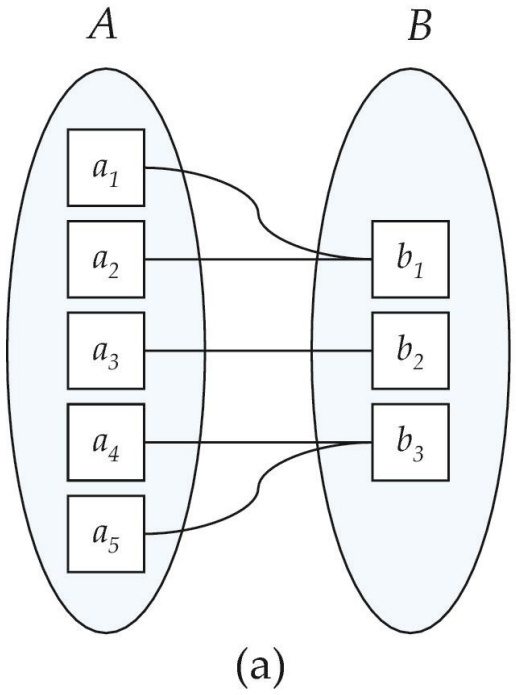

One to one



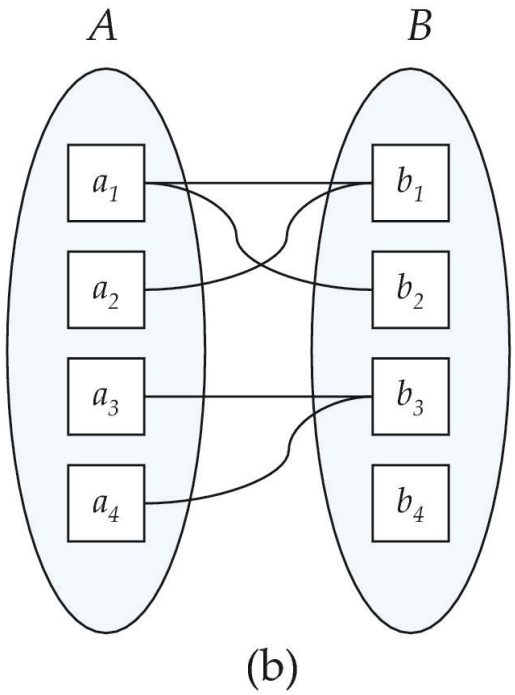

One to many

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

# Mapping Cardinalities




Many to one




Many to many

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

# Keys

- The values of the attributes of an entity must be such that they can *uniquely identify* the entity from the entity set.
- In other words, no two entities in an entity set are allowed to have exactly the same value for all attributes.
- A key allows us to identify a set of attributes that suffice to distinguish entities from each other.
- Keys also help uniquely identify relationships, and thus distinguish relationships from each other.

# Keys for Entity Sets

- **Superkey:** A superkey is a set of one or more attributes that, taken collectively, allow identifying uniquely an entity in the entity set. A superkey may contain extraneous attributes.
  - Exmp: instructor – {ID}, {ID, name}, {all attributes}
    name is not a superkey
- **Candidate key:** The superkey, for which no proper subset is a superkey, is a candidate key. So the minimul superkeys are called candidate keys.
  It is possible that several distinct sets of attributes could serve as a candidate key.
  - *ID* is candidate key of *instructor*
  - *course_id* is candidate key of *course*
- **Primary Key:** The primary key is to denote a candidate key that is chosen by the database designer as the principal means of identifying entities within the entity set.
- **So, Primary key $\subseteq$ Candidate Key $\subseteq$ Super 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

# Primary Key for Relationship Sets

- To distinguish among the various relationships of a relationship set we use the individual primary keys of the entities in the relationship set.
  - Let *R* be a relationship set involving entity sets E1, E2, .. En
  - The super key for R consists of the union of the primary keys of entity sets E1, E2, ..En
  - If the relationship set *R* has attributes a1, a2, .., am associated with it, then *R* also includes the attributes a1, a2, .., am
- Example: relationship set “advisor”.
  - The super key consists of *instructor.ID* and s*tudent.ID*
- The choice of the primary key for a relationship set depends on the mapping cardinality of the relationship set.

# Choice of Primary key for Binary Relationship

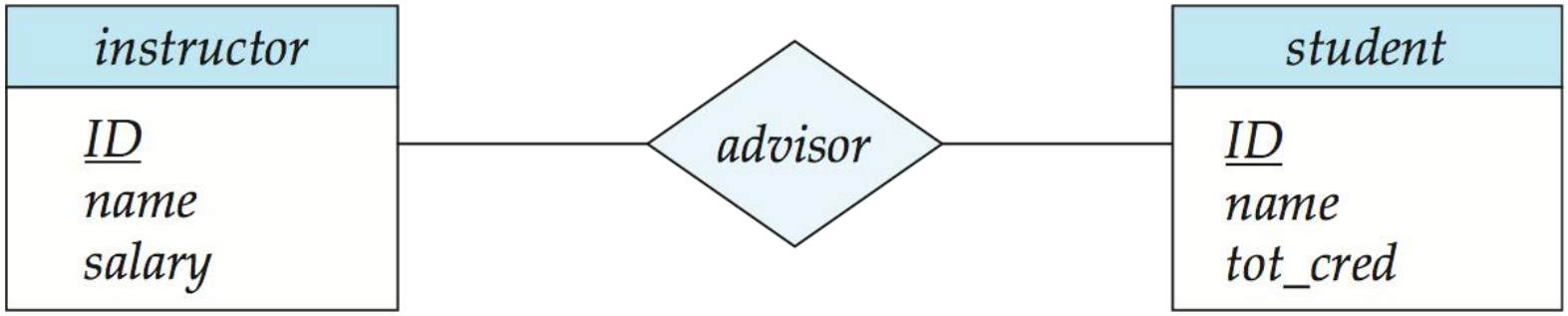
- Many-to-Many relationships. The preceding union of the primary keys is a minimal superkey and is chosen as the primary key.
- One-to-Many relationships . The primary key of the “Many” side is a minimal superkey and is used as the primary key.
- Many-to-one relationships. The primary key of the “Many” side is a minimal superkey and is used as the primary key.
- One-to-one relationships. The primary key of either one of the participating entity sets forms a minimal superkey, and either one can be chosen as the primary key.

# Redundant Attributes

- Suppose we have entity sets
  - *instructor*, with attributes including *dept_name*
  - *department*

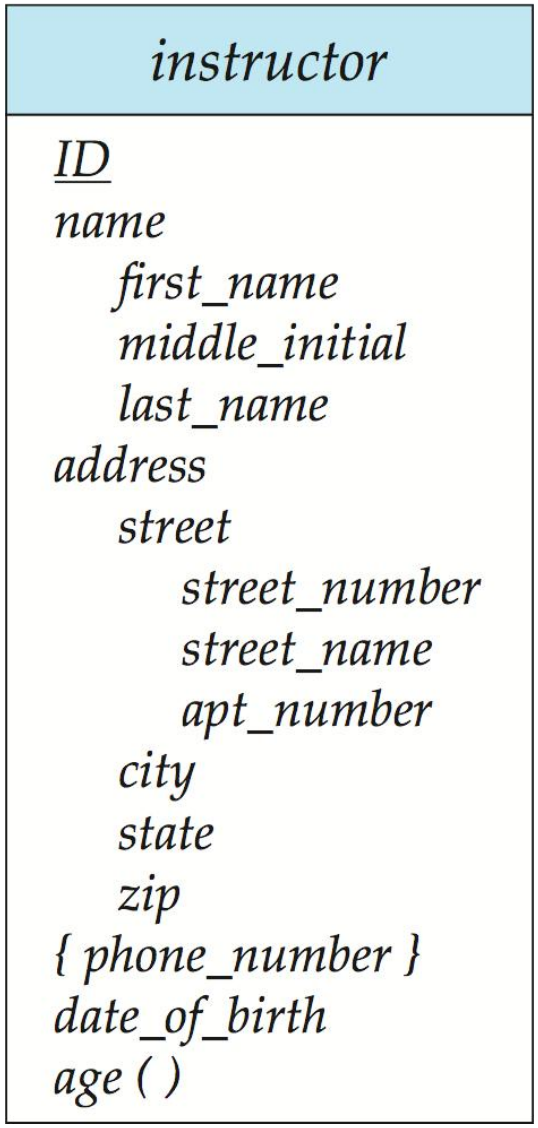
  and a relationship
  - *inst_dept* relating *instructor* and *department*
- Attribute *dept_name* in entity *instructor* is redundant since there is an explicit relationship *inst_dept* which relates instructors to departments
  - The attribute replicates information present in the relationship, and 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



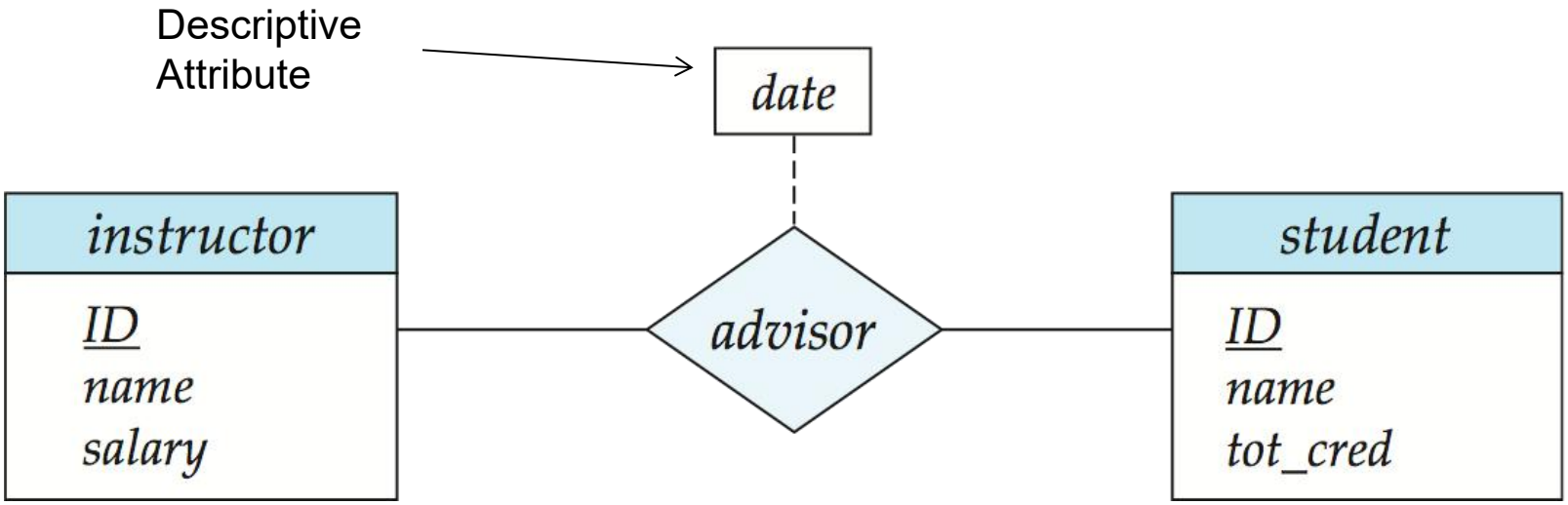

- Rectangles represent entity sets.
- Diamonds represent relationship sets.
- Attributes listed inside entity rectangle
- Underline indicates primary key attributes

# Entity With Composite, Multivalued, and Derived Attributes



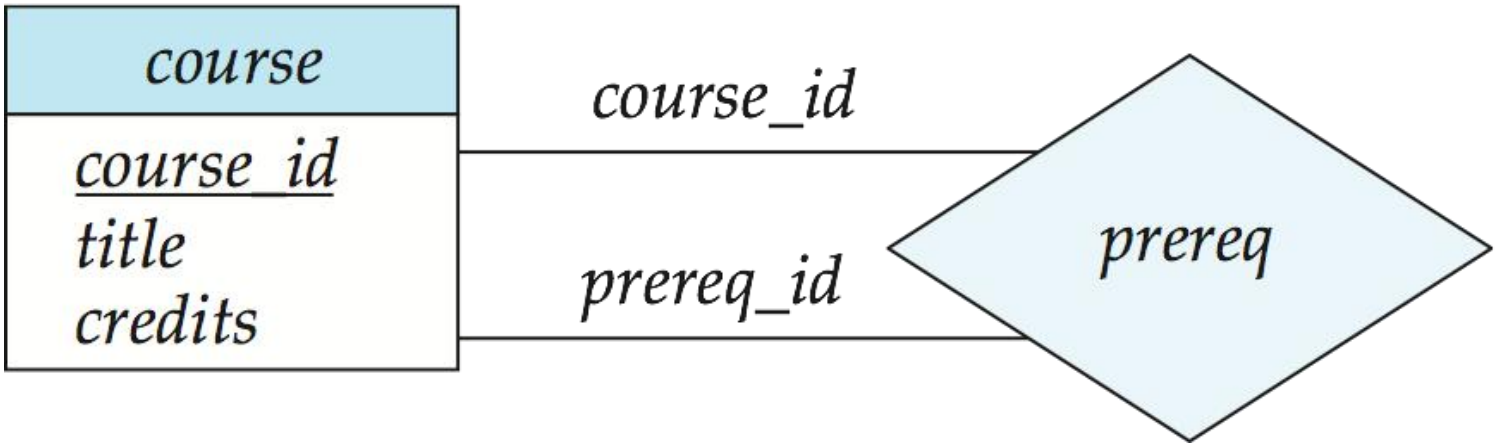

 

# Relationship Sets with Attributes



# 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**.



# Constraints in E-R Model

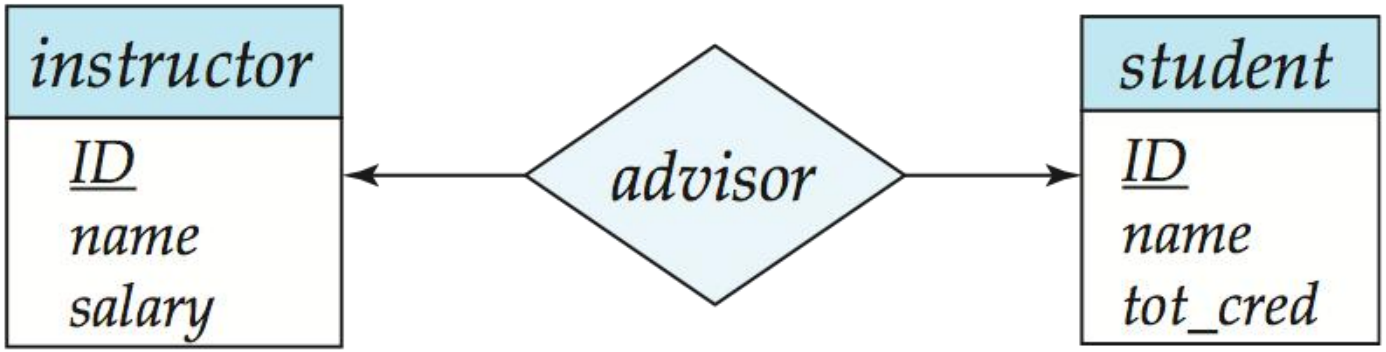
- An E-R enterprise schema may define certain constraints to which the contents of a database must conform. These are:
- 1. Mapping cardinalities or Cardinality ratios and
- 2. Participation constraints.

# Cardinality Constraints

- Express the number of entities to which another entity can be associated via a relationship set.
- 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*

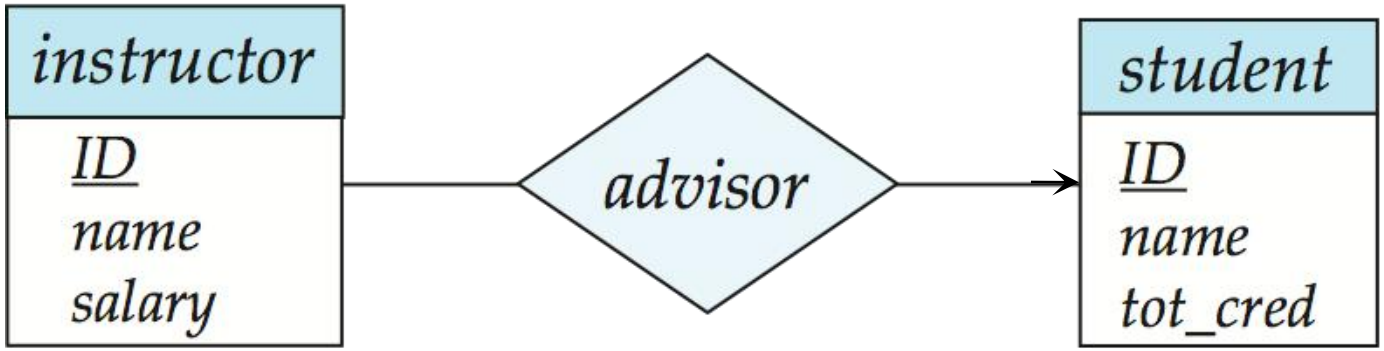


# One-to-Many Relationship

- 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,

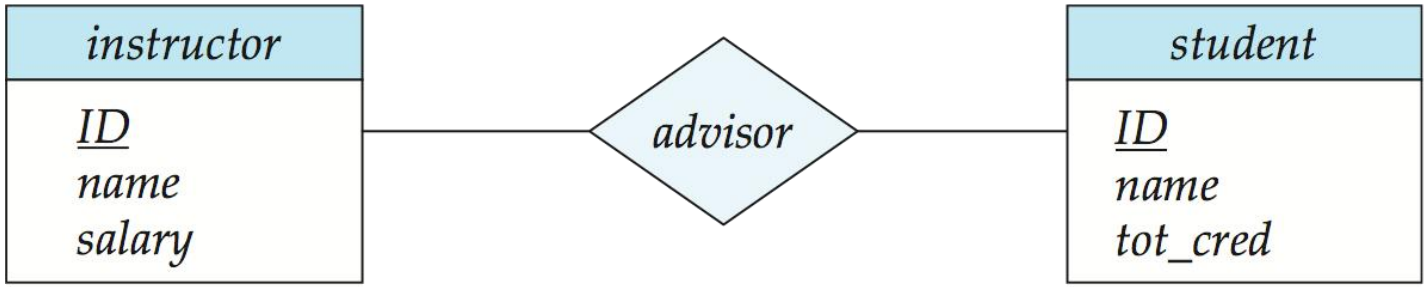
# Many-to-One Relationships

- 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*



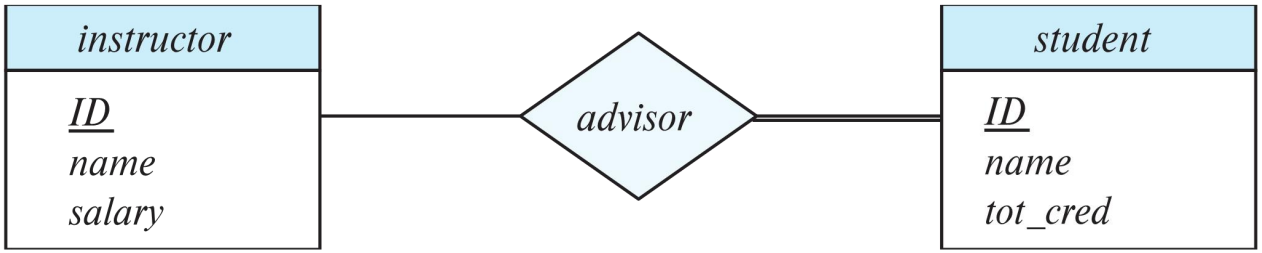
# Many-to-Many Relationship

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



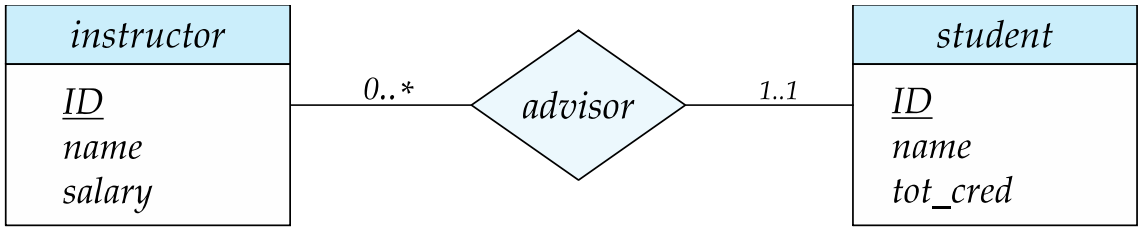
# Participation of an Entity Set in a Relationship Set

- 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
  - E.g., participation of *student* in *advisor* relation is total
    - every *student* must have an associated instructor
- Partial participation: some entities may not participate in any relationship in the relationship set
  - Example: participation of *instructor* in *advisor* is partial



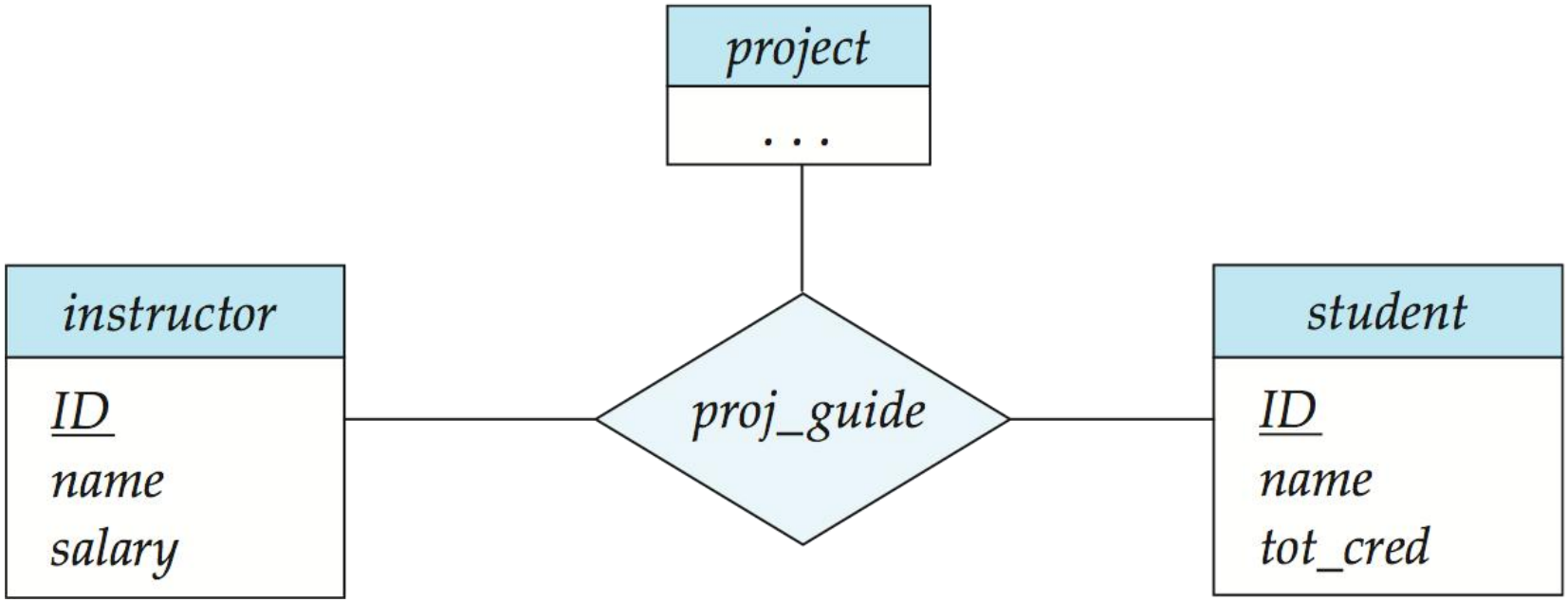
# Notation for Expressing More Complex Constraints

- A line may have an associated minimum and maximum cardinality, shown in the form *l..h*, where *l* is the minimum and *h* the maximum cardinality
  - A minimum value of 1 indicates total participation.
  - A maximum value of 1 indicates that the entity participates in at most one relationship
  - A maximum value of * indicates no limit.
- Example




  - Instructor can advise 0 or more students. A student must have 1 advisor; cannot have multiple advisors

# 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

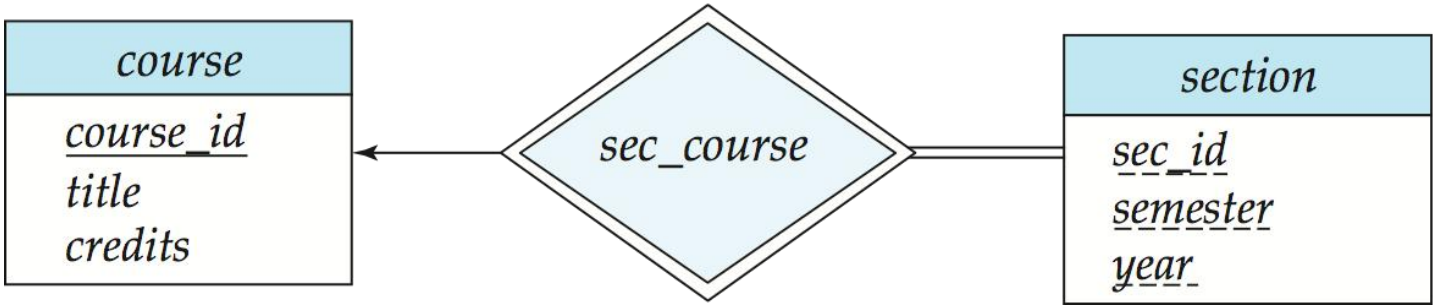
# How about doing an ER design interactively on the board? Suggest an application to be modeled.

# 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*)



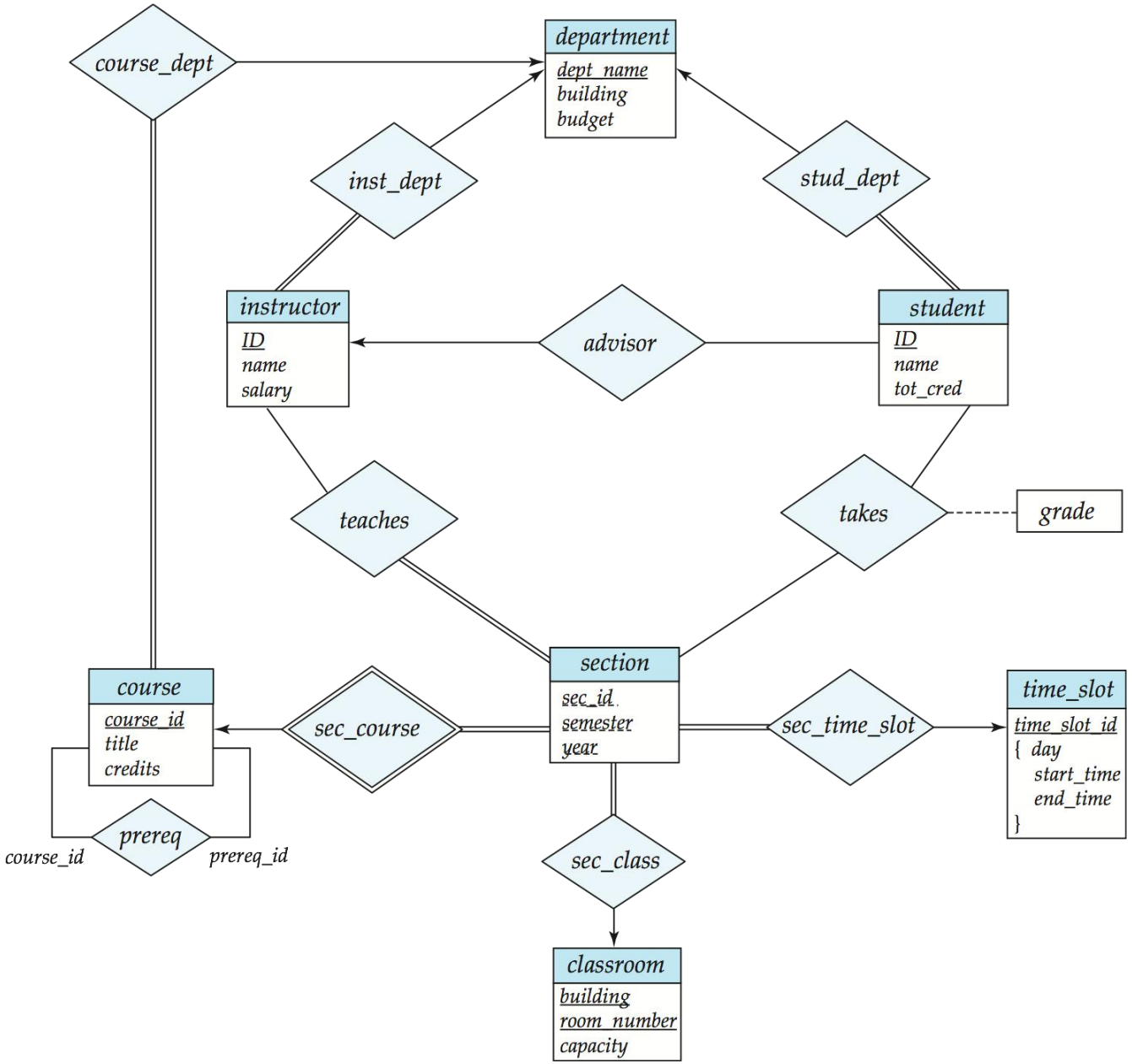
# Weak Entity Sets (Cont.)

- 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*

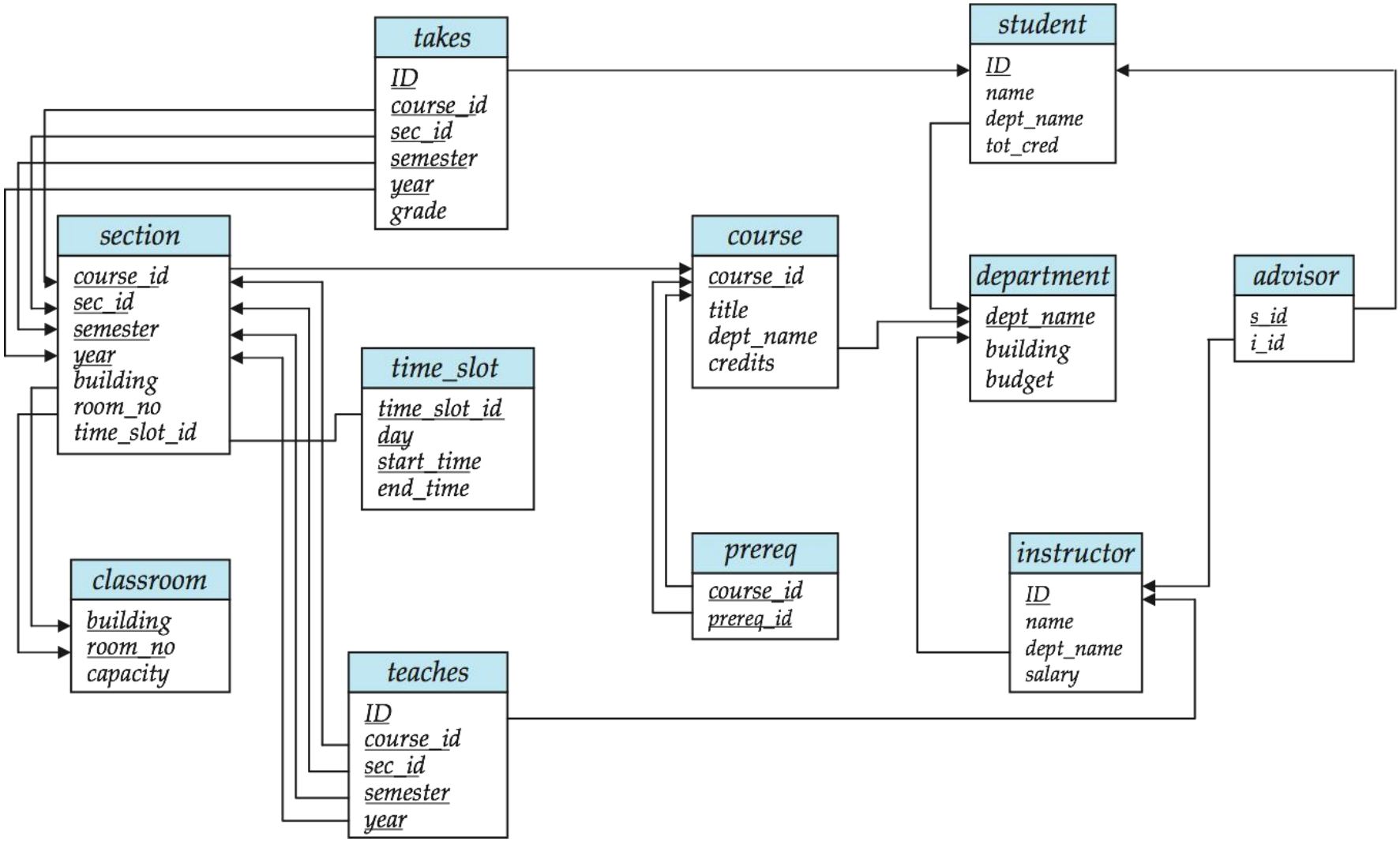
# Weak Entity Sets (Cont.)

- The identifying relationship set should have no descriptive attributes, since any required attributes can be associated with the weak entity set.
- 1. A weak entity set can participate in relationships other than the identifying relationship.
- 2. A weak entity set may participate as owner in an identifying relationship with another weak entity set.
- 3. It is possible to have a weak entity set with more than one identifying entity set. The primary key of the weak entity set would consist of the union of the primary keys of the identifying entity sets plus the discriminator of the weak entity set.

# E-R Diagram for a University Enterprise



# Schema Diagram for University Database



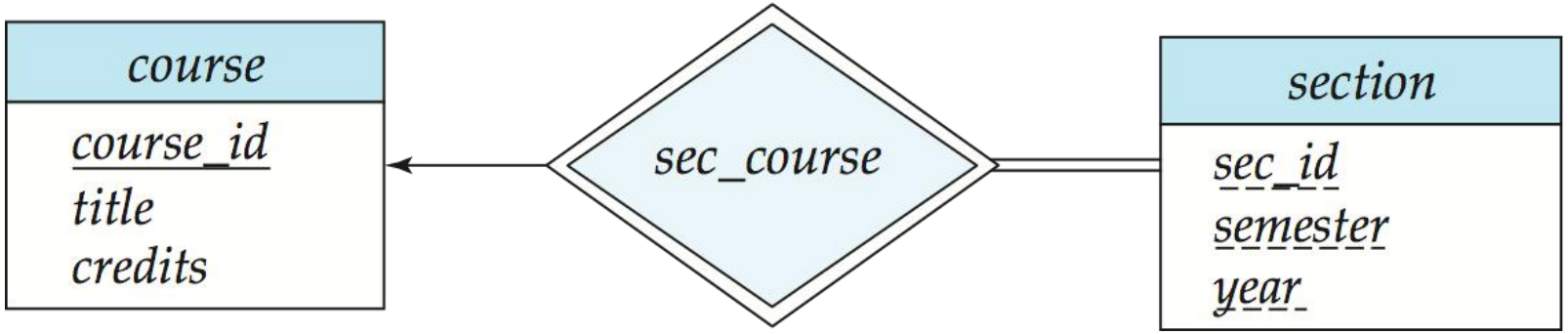
# Reduction to Relational Schemas

# 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.
- Both the E-R model and the relational database model are abstract, logical representations of real-world enterprises. Because the two models employ similar design principles, we can convert E-R design into a relational design.

# Representing Entity Sets With Simple Attributes

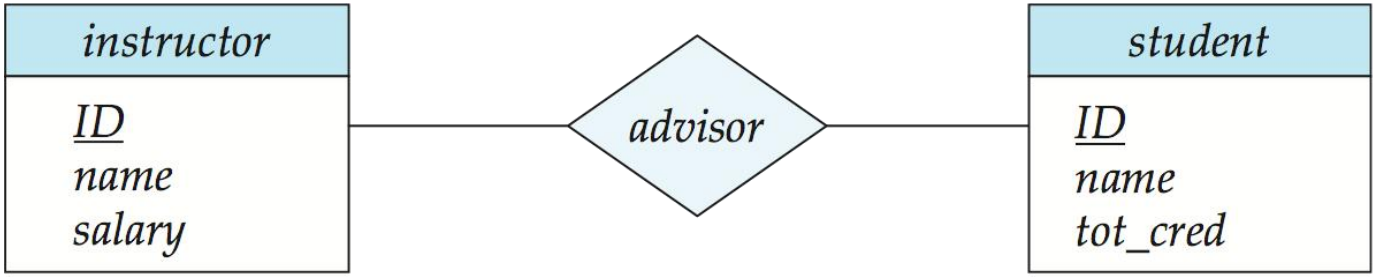
- A strong entity set reduces to a schema with the same attributes
  *course(course_id, title, credits)*
- A weak entity set becomes a table that includes a column for the primary key of the identifying strong entity set
  *section ( course_id, sec_id, sem, year )*



# Representing Relationship Sets

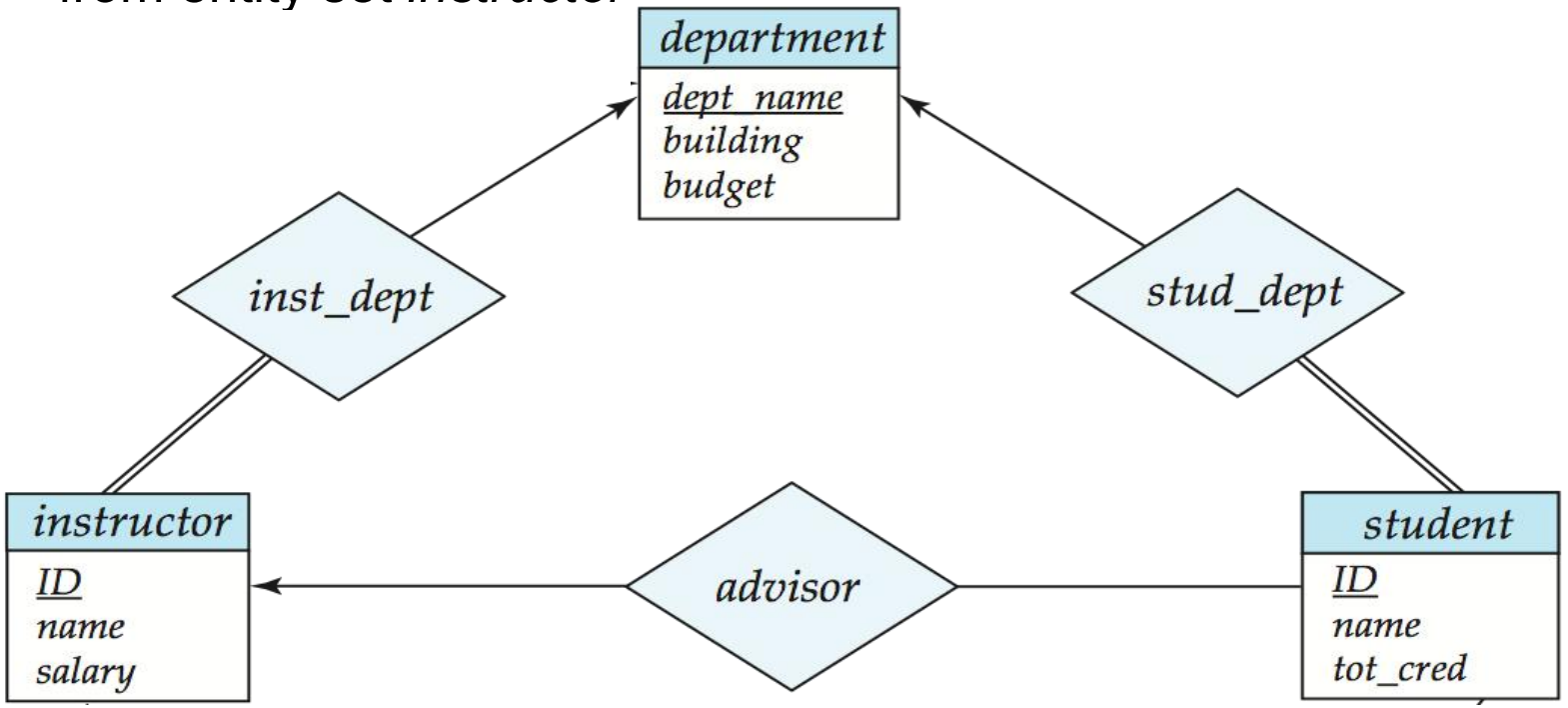
- A many-to-many relationship set is represented as a schema with attributes for the primary keys of the two participating entity sets, and any descriptive attributes of the relationship set.
- Example: schema for relationship set *advisor*

  *advisor* = (*s_id, i_id*)



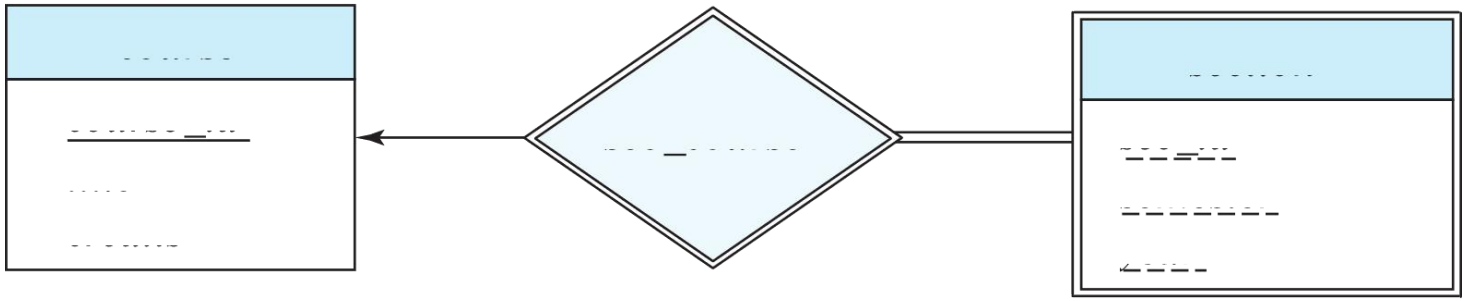
# Redundancy of Schemas

- 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



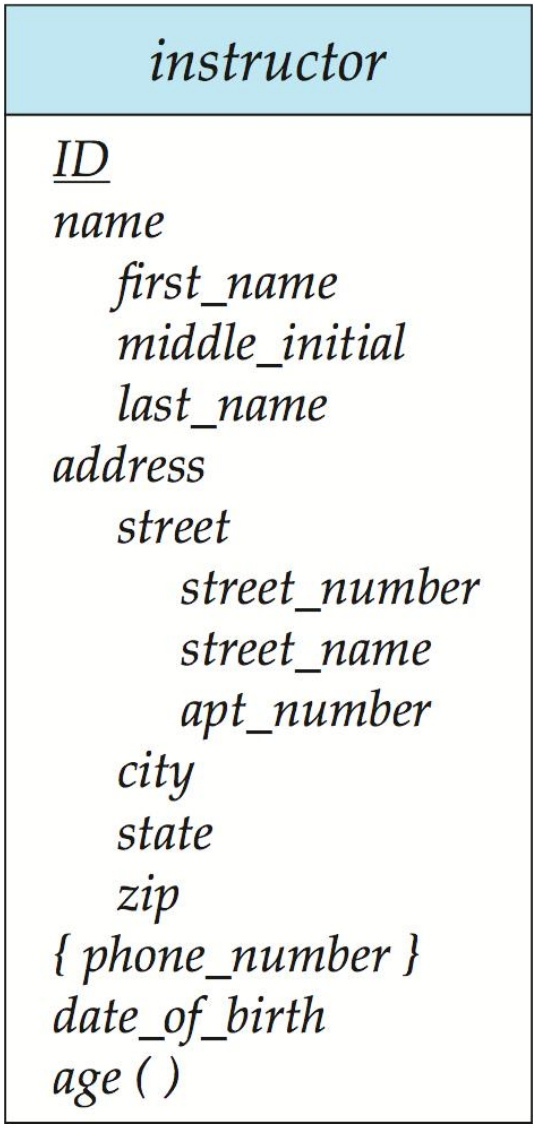
# Combination of Schemas

- If there are two entity sets A and B and there exist relationship set R between those, 3 schemas A, B, R can be generated using relational-schema construction scheme.
- 1. one to one relationship from A to B - the primary key of R is either of the primary keys of participating entity sets (A or B)
  - These can be combined into 2 schemas: AR, B or A, BR
- 2. one to many from A to B - the primary key of R is simply the primary key of B
  - These can be combined into 2 schemas: A, BR
- 3. many to one from A to B - the primary key of R is simply the primary key of A
  - These can be combined into 2 schemas: AR, B
- 4. many to many – the primary key of R consists of the union of the primary keys of A and B
  - There is no other alternative but to keep all the 3 schemas.

# Combination of Schemas (Cont.)

- For all the above cases there should be total participation of the entity sets in the relationship set.
- Combination of schemas works nicely in case of total participation.
- If there exists partial participation and we combine the schemas there will be null values for some entities those do not participate in relationship.

# Composite and Multivalued Attributes



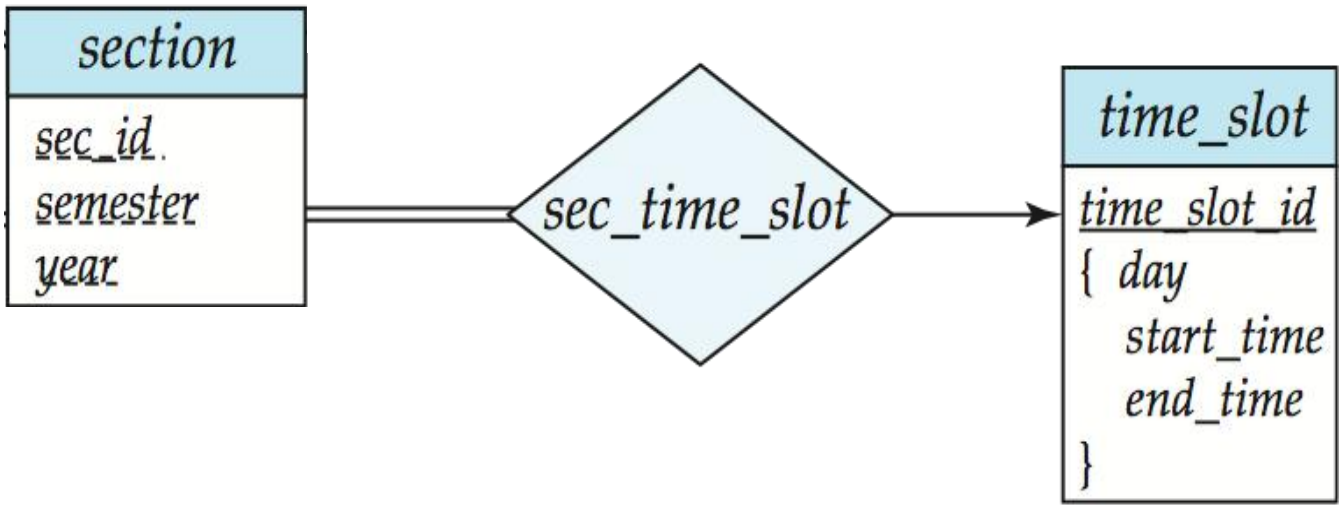

- Composite attributes are flattened out by creating a separate attribute for each component attribute
  - Example: given entity set *instructor* with composite attribute *name* with component attributes *first_name* and *last_name* the schema corresponding to the entity set has two attributes *name_first_name* 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*= ( *ID*, *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*)
  - Caution: *time_slot* attribute of *section (*from *sec_time_slot*) cannot be a foreign key due to this optimization

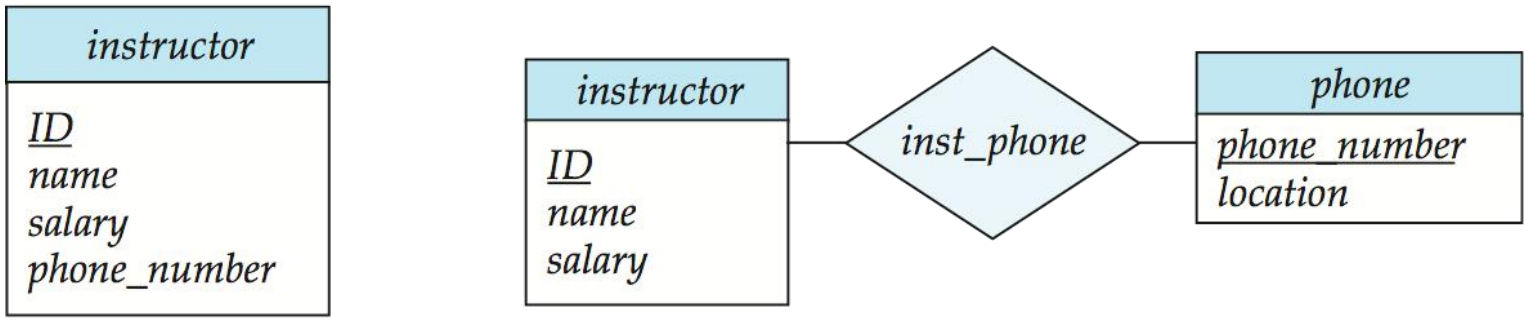


# Design Issues

- The notions of an entity set and a relationship set are not precise and it is possible to define a set of entities and the relationships among them in a number of different ways.

  1. Use of Entity Sets Vs Attributes
     - What constitutes an attribute and what constitutes an entity set?
  2. Use of Entity Sets Vs Relationship Sets
  3. Binary Vs n-ary Relationship Sets
  4. Placement of Relationship Attributes (Descriptive attributes)

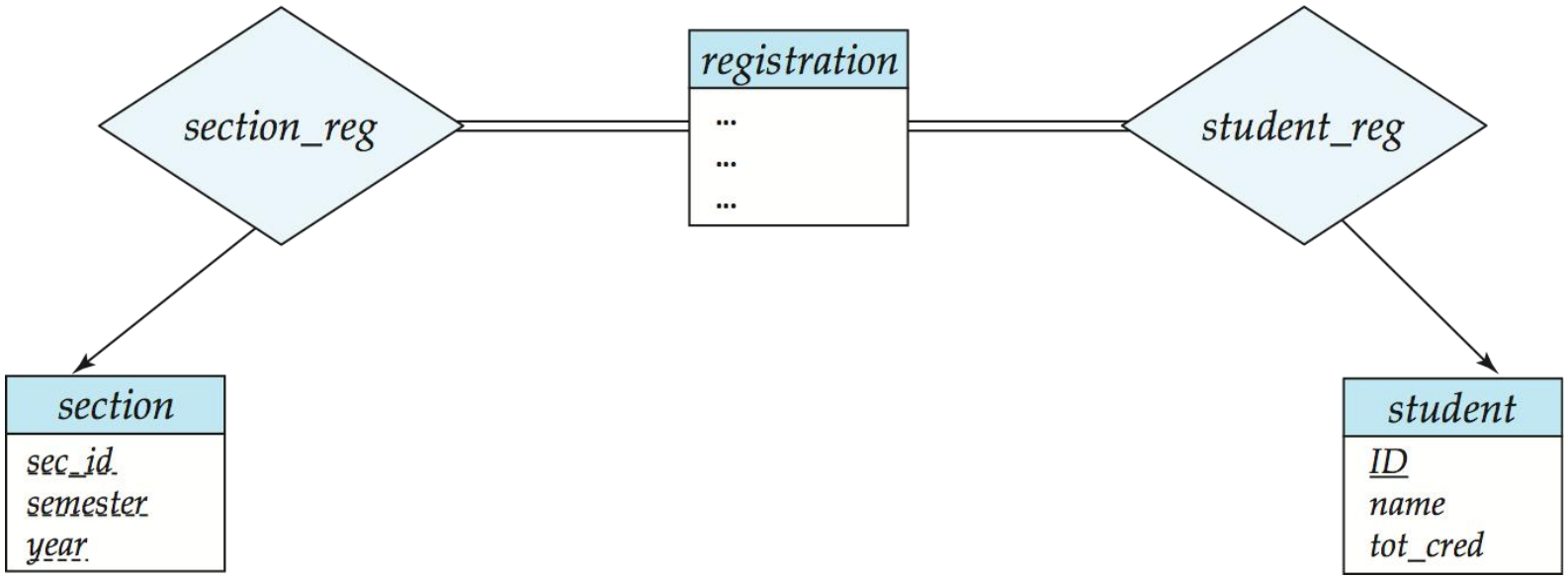
- **Use of entity sets vs. attributes**




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

# Design Issues

an action



# Design Issues

or $n > 2$)
p sets, a *n*-
ies

e of *student*

positioned to

ute can be

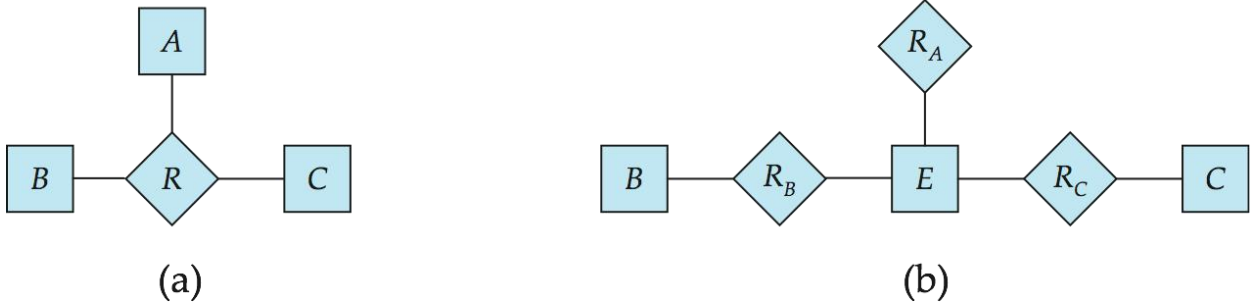
n
rately, that
tionship set.

# 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 know)
  - 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:
    1. $R_A$, relating $E$ and $A$
    2. $R_B$, relating $E$ and $B$
    3. $R_C$, 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
    1. a new entity $e_i$ in the entity set $E$
    2. add $(e_i , a_i)$ to $R_A$
    3. add $(e_i , b_i)$ to $R_B$
    4. 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

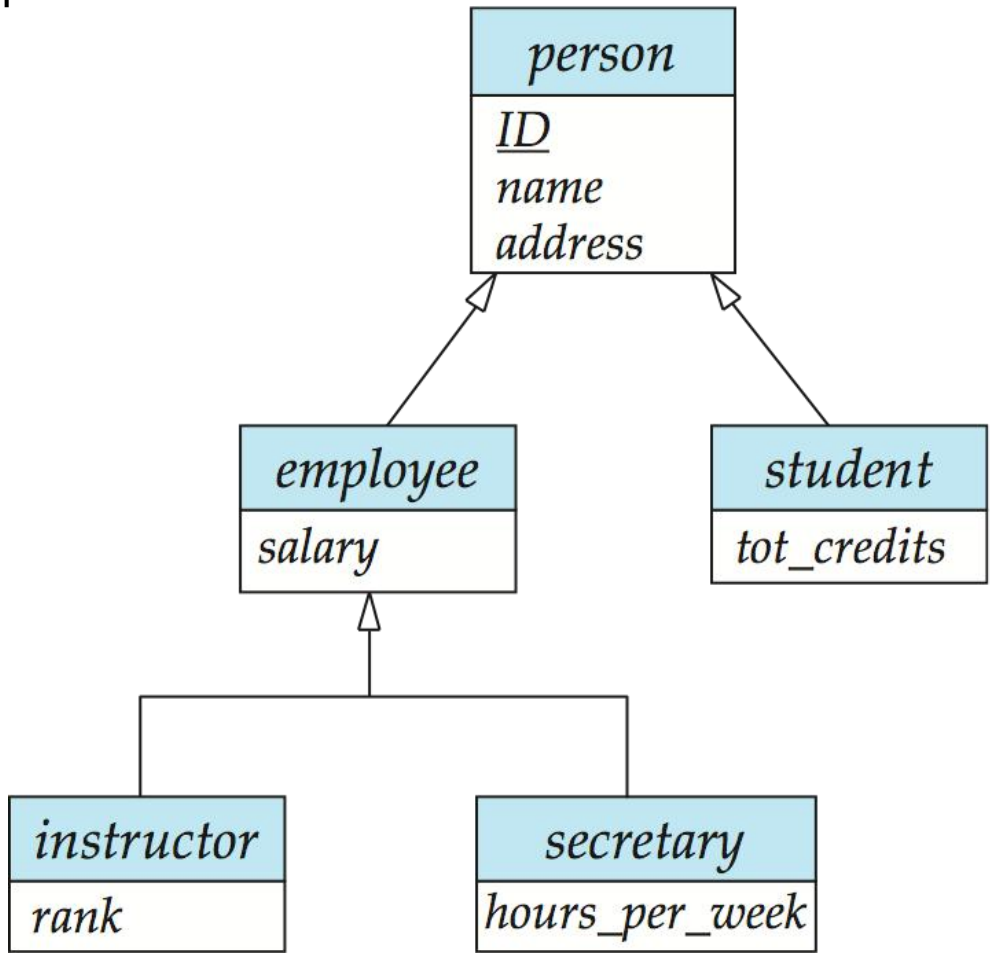
# Extended ER Features

# Extended E-R Features: Specialization

- Top-down design process: we designate sub-groupings within an entity set that are distinctive from other entities in the set.
- These sub-groupings 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

- **Overlapping** – *employee* and *student*
- **Disjoint** – *instructor* and *secretary*
- Total and partial



# 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* or *secretary*
- The ISA relationship also referred to as **superclass - subclass** relationship

# Design Constraints on a Specialization/Generalization

- **A. Type1:** Constraint on which entities can be members of a given lower-level entity set.
- **1. Condition-defined:** In condition-defined lower-level entity sets, membership is defined on the basis of whether or not an entity satisfies an explicit condition or predicate. It is also called attribute-defined generalization.
  - Example 1: Saving account and Checking account
  - Example 2: All customers over 65 years are members of *senior-citizen* entity set; *senior-citizen* ISA *person*.
- **2. User-defined:** Used-defined lower-level entity sets are not constrained by a membership condition; rather, the database user assigns to a given entity set.
  - Example: Employee and Team

# Design Constraints on a Specialization/Generalization

- **B. Type2:** Constraint on whether or not entities may belong to more than one lower-level entity set within a single generalization.
- **1. Disjoint:** A *disjointness constraint* requires that an entity 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
  - Example: Saving account and Checking account
- **2. Overlapping:** In *overlapping generalizations*, the same entity may belong to more than one lower-level entity sets within a single generalization.
  - Example: Employee and Team

# Design Constraints on a Specialization/Generalization

- **C. 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 the generalization/ specialization.
- **1. Total generalization or specialization:** Each higher-level entity must belong to one of the lower-level entity set.
  - Example: Saving account and Checking account
- **2. Partial generalization or specialization:** Some higher-level entities may not belong to any lower-level entity set.
  - Example: Employee and Team

# Completeness constraint (Cont.)

- Partial generalization is the default.
- We can specify total generalization in an ER diagram by adding the keyword **total** in the diagram and drawing a dashed line from the keyword to the corresponding hollow arrow-head to which it applies (for a total generalization), or to the set of hollow arrow-heads to which it applies (for an overlapping generalization).
- The *student* generalization is total: All student entities must be either graduate or undergraduate. Because the higher-level entity set arrived at through generalization is generally composed of only those entities in the lower-level entity sets, the completeness constraint for a generalized higher-level entity set is usually total

# 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 |
|---|---|
| *person* | *ID, name, street, city* |
| *student* | *ID, tot_cred* |
| *employee* | *ID, salary* |

  - 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.)

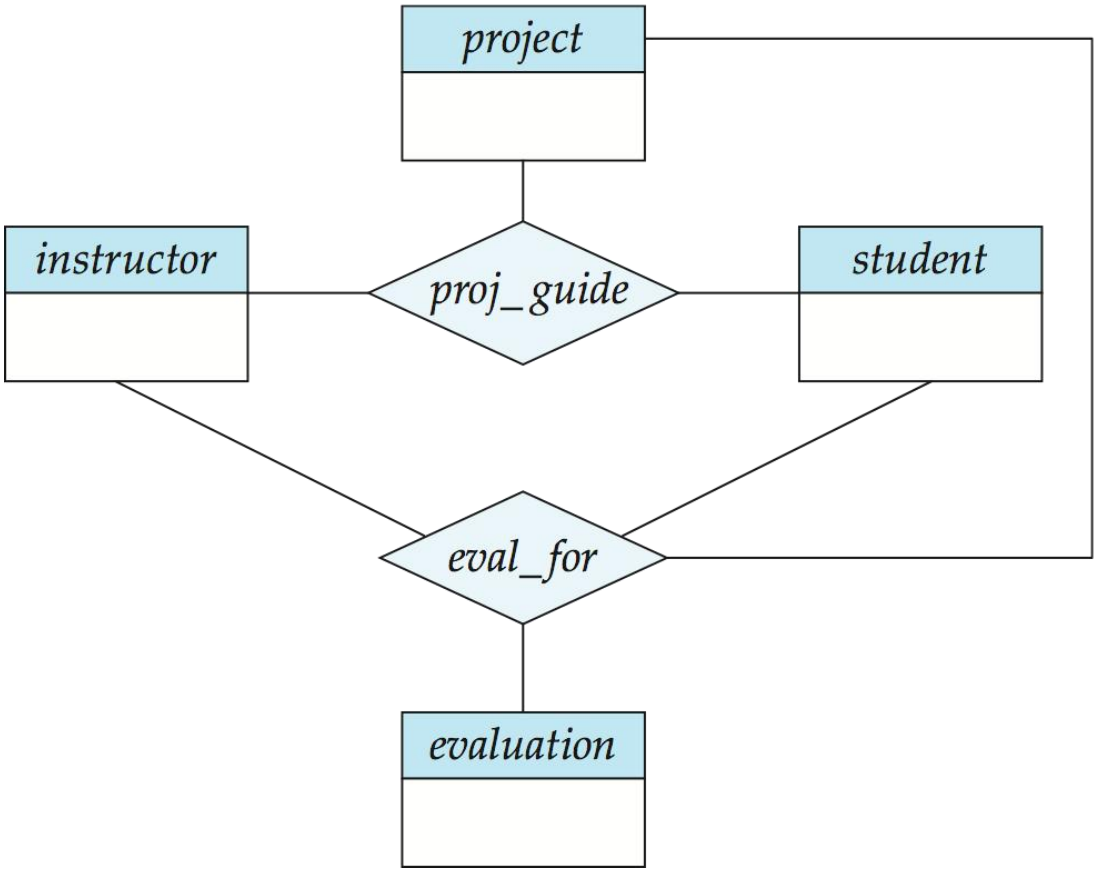
- Method 2:
  - Form a schema for each entity set with all local and inherited attributes

| schema | attributes |
|---|---|
| *person* | *ID, name, street, city* |
| *student* | *ID, name, street, city, tot_cred* |
| *employee* | *ID, name, street, city, salary* |

  - 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

# 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

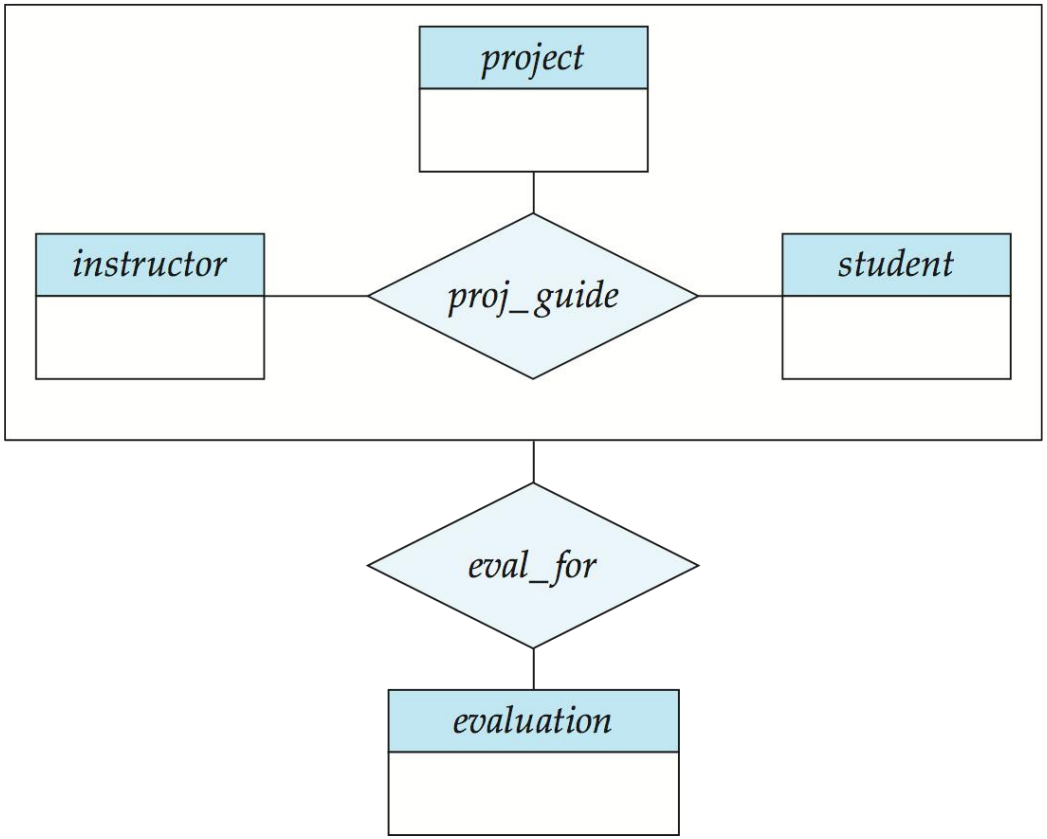


# Aggregation (Cont.)

- 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



# Schemas Corresponding to Aggregation

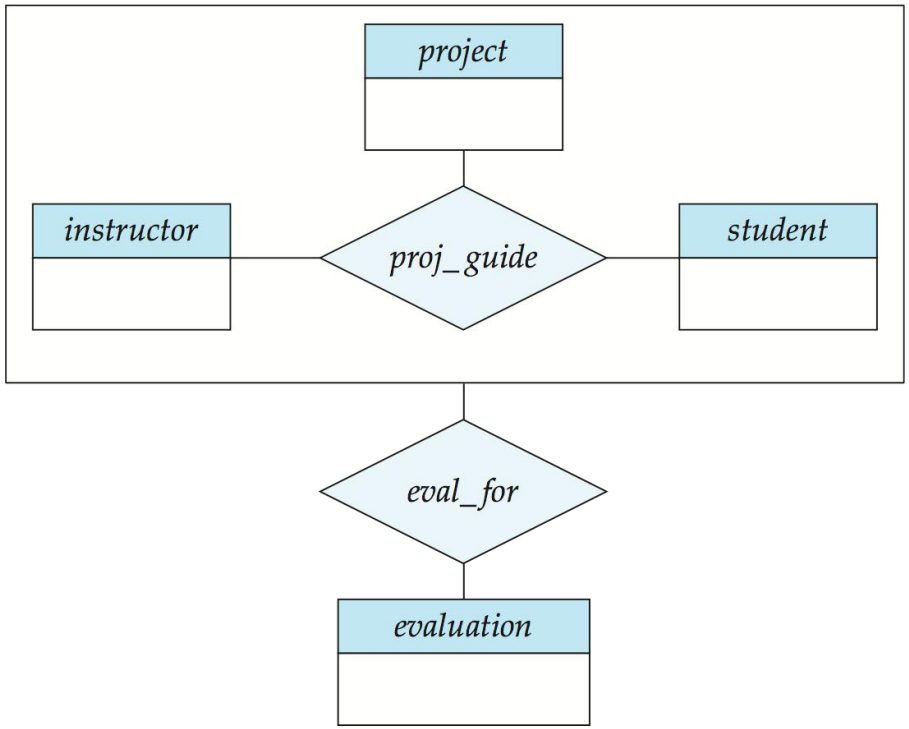
- To represent aggregation, create a schema containing
  - primary key of the aggregated relationship,
  - the primary key of the associated entity set
  - any descriptive attributes

# Schemas Corresponding to Aggregation (Cont.)

- For example, to represent aggregation manages between relationship works_on and entity set manager, create a schema

  *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*

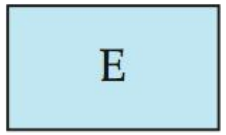


# E-R Design Decisions

- 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?

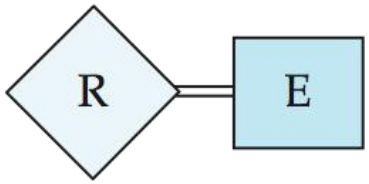
# Summary of Symbols Used in E-R Notation



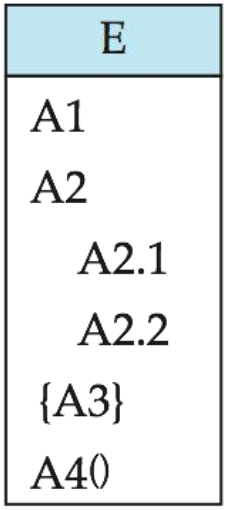

entity set


relationship set

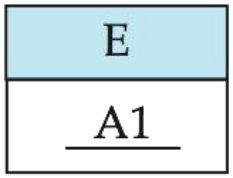

identifying relationship set for weak entity set



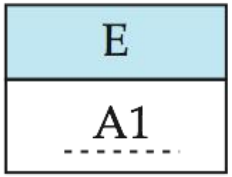

total participation of entity set in relationship




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

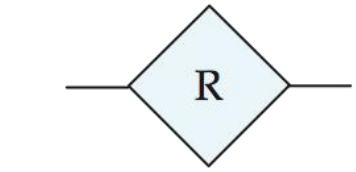



primary key

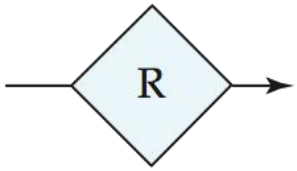



discriminating attribute of weak entity set

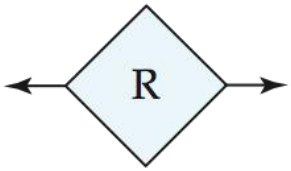
# Symbols Used in E-R Notation (Cont.)



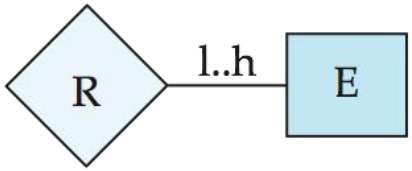

many-to-many relationship



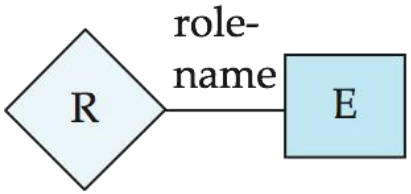

many-to-one relationship



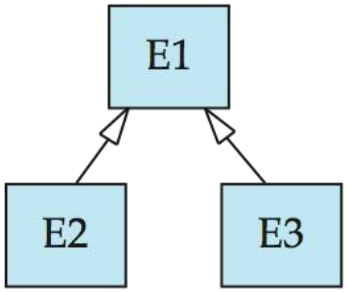

one-to-one relationship



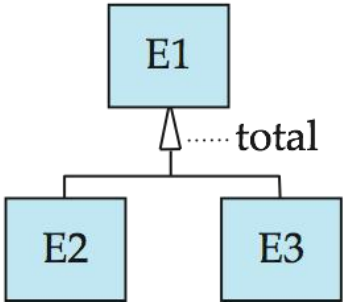

cardinality limits



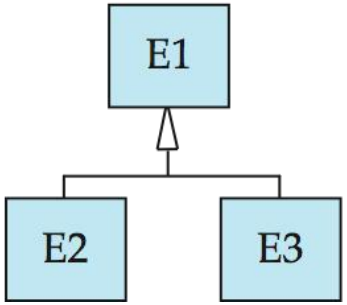

role indicator




ISA: generalization or specialization




total (disjoint) generalization

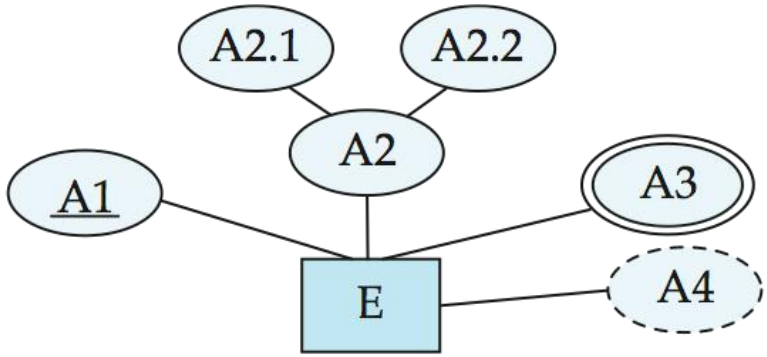



disjoint generalization

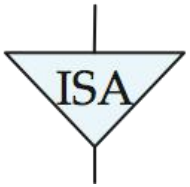
# Alternative ER Notations

- Chen, IDE1FX, …

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

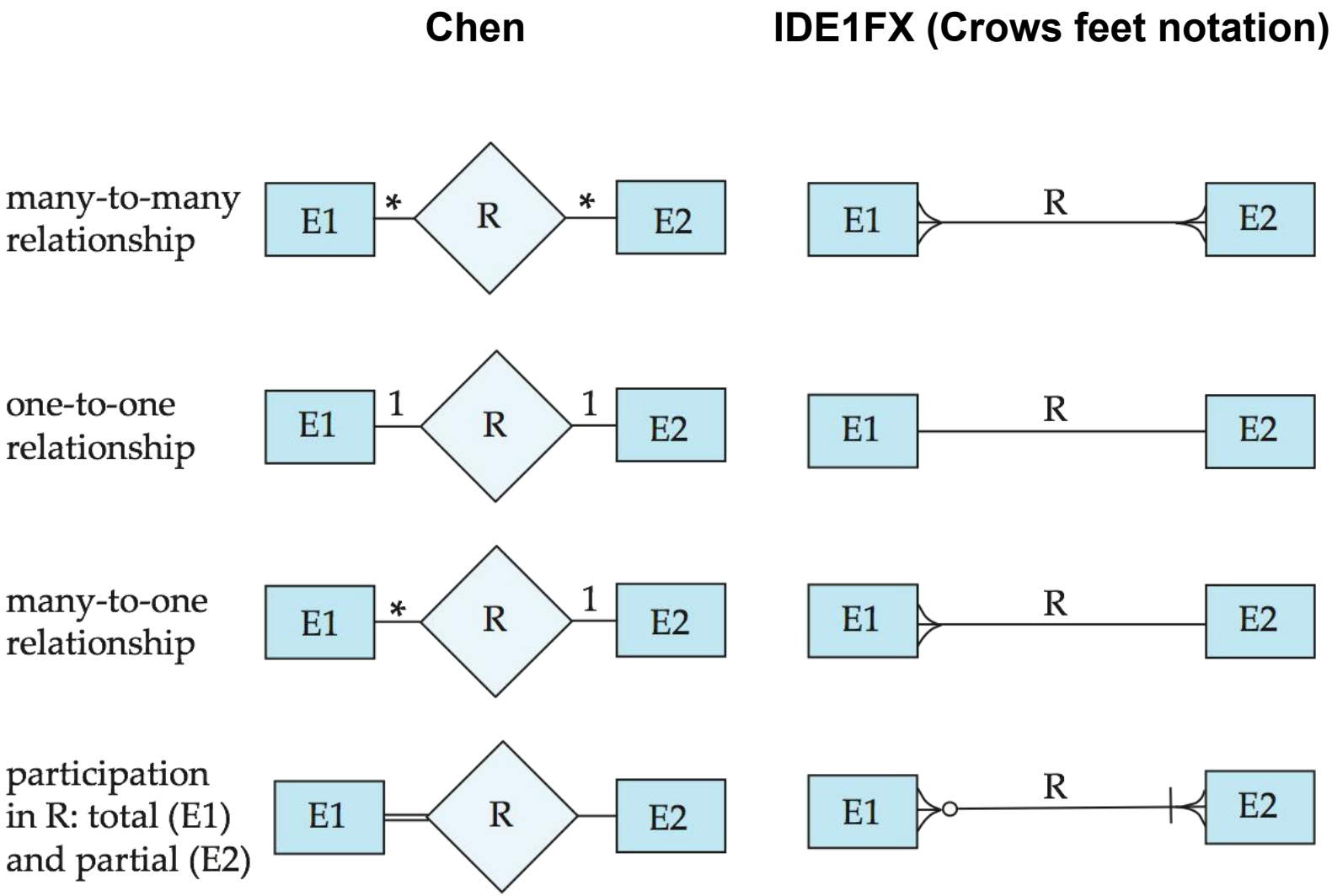



weak entity set 

generalization 


total generalization 

# Alternative ER Notations

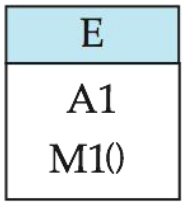


# UML

- **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.

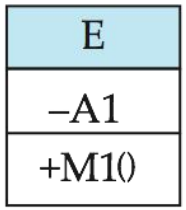
# ER vs. UML Class Diagrams

**ER Diagram Notation**

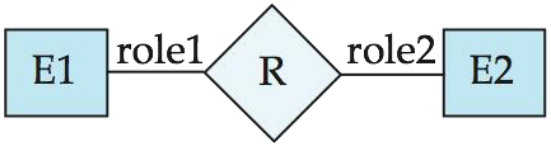



entity with attributes (simple, composite, multivalued, derived)

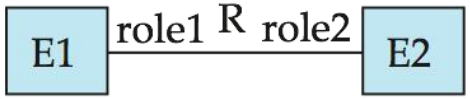
**Equivalent in UML**

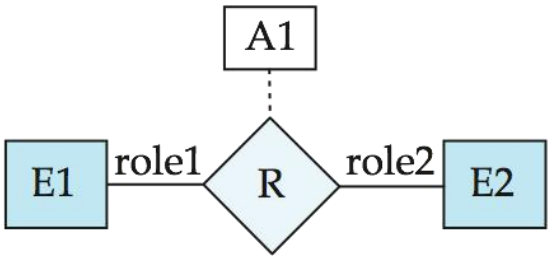



class with simple attributes and methods (attribute prefixes: + = public, – = private, # = protected)

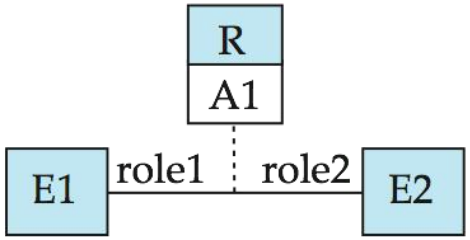


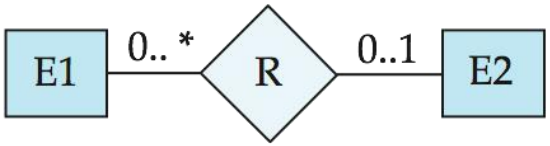

binary relationship



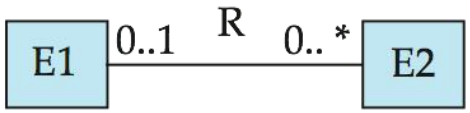




relationship attributes



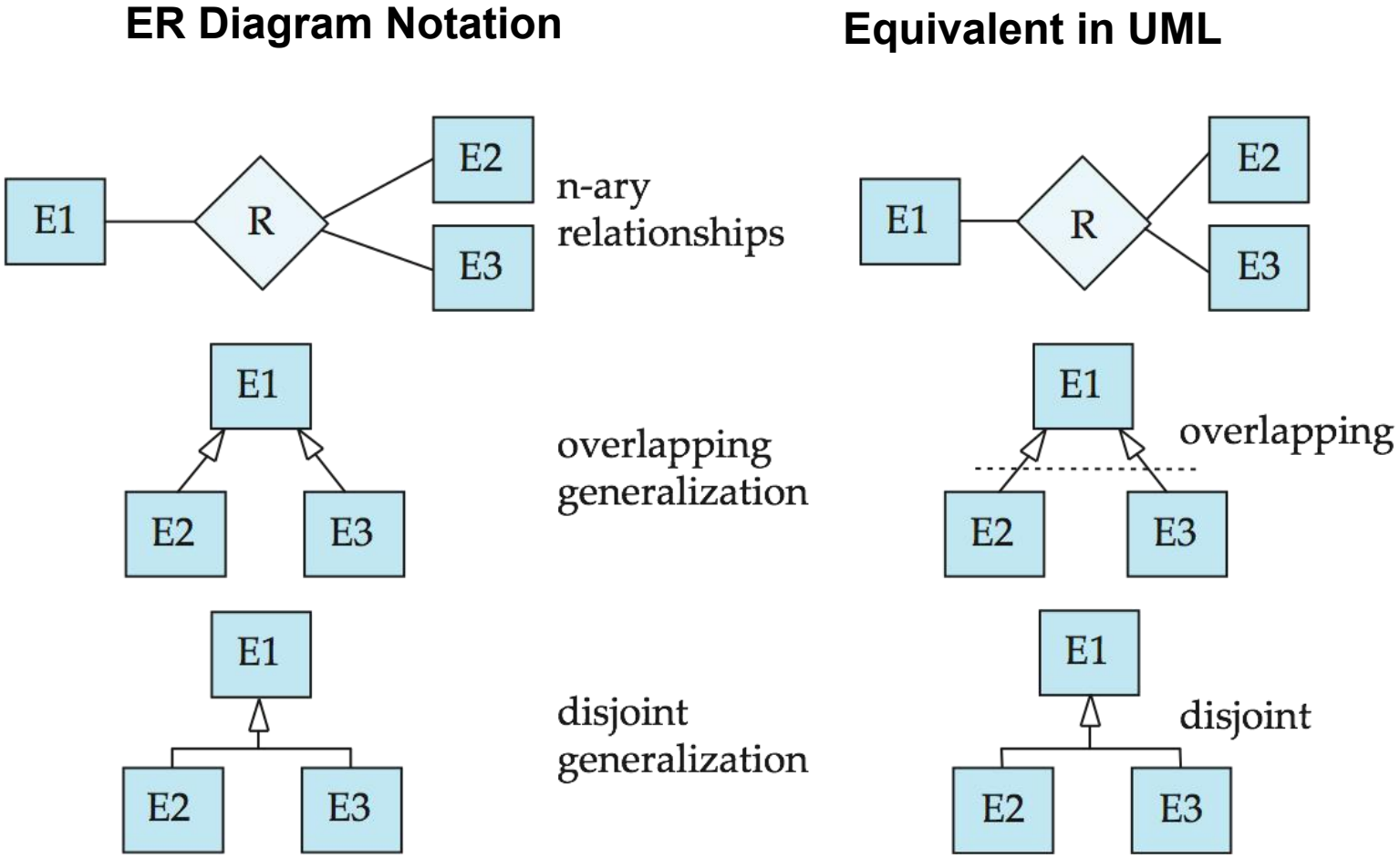




cardinality constraints




*Note reversal of position in cardinality constraint depiction

# ER vs. UML Class Diagrams




*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.

# End of Chapter 7

# Figure 7.01

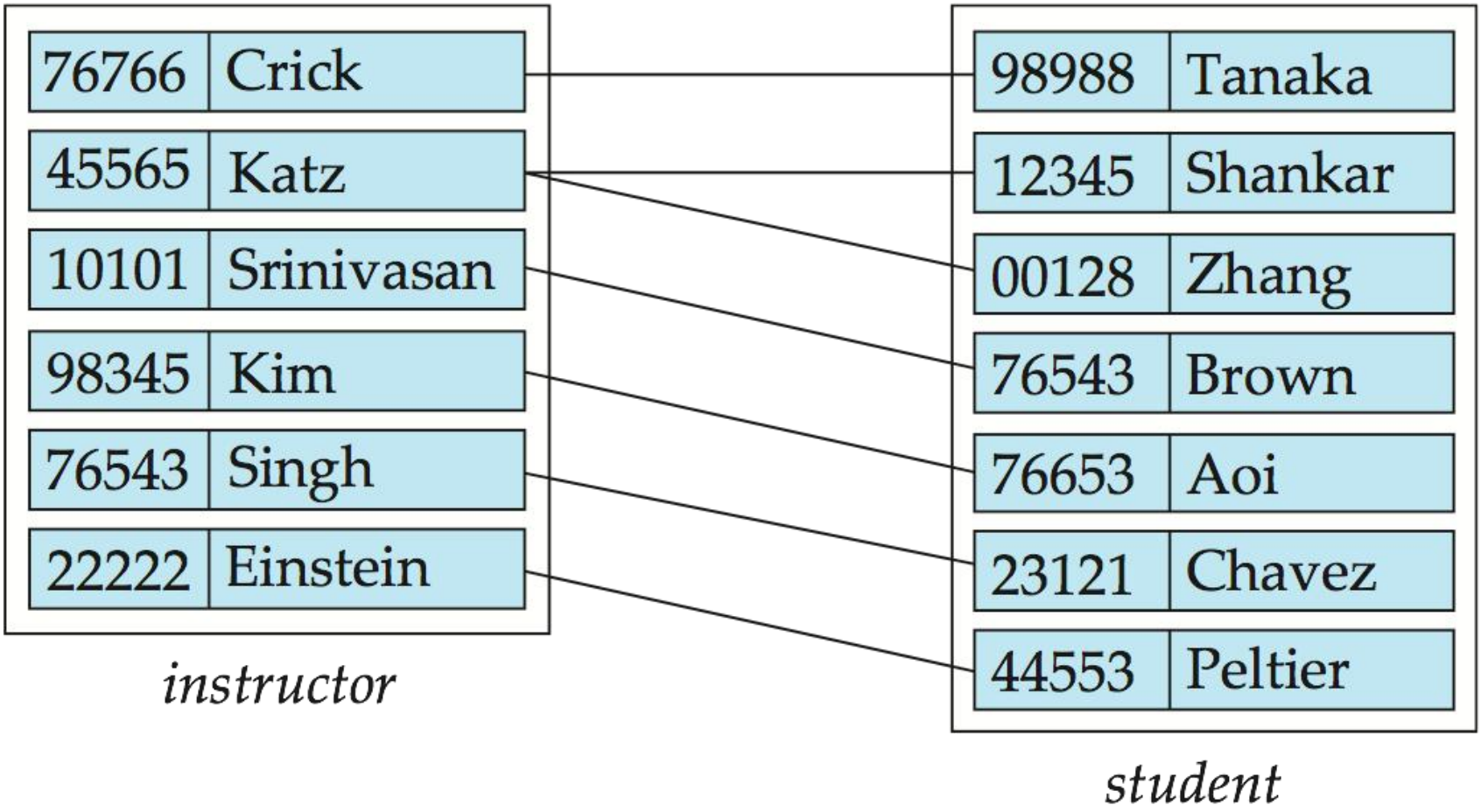
| | |
|---|---|
| 76766 | Crick |
| 45565 | Katz |
| 10101 | Srinivasan |
| 98345 | Kim |
| 76543 | Singh |
| 22222 | Einstein |

*instructor*

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

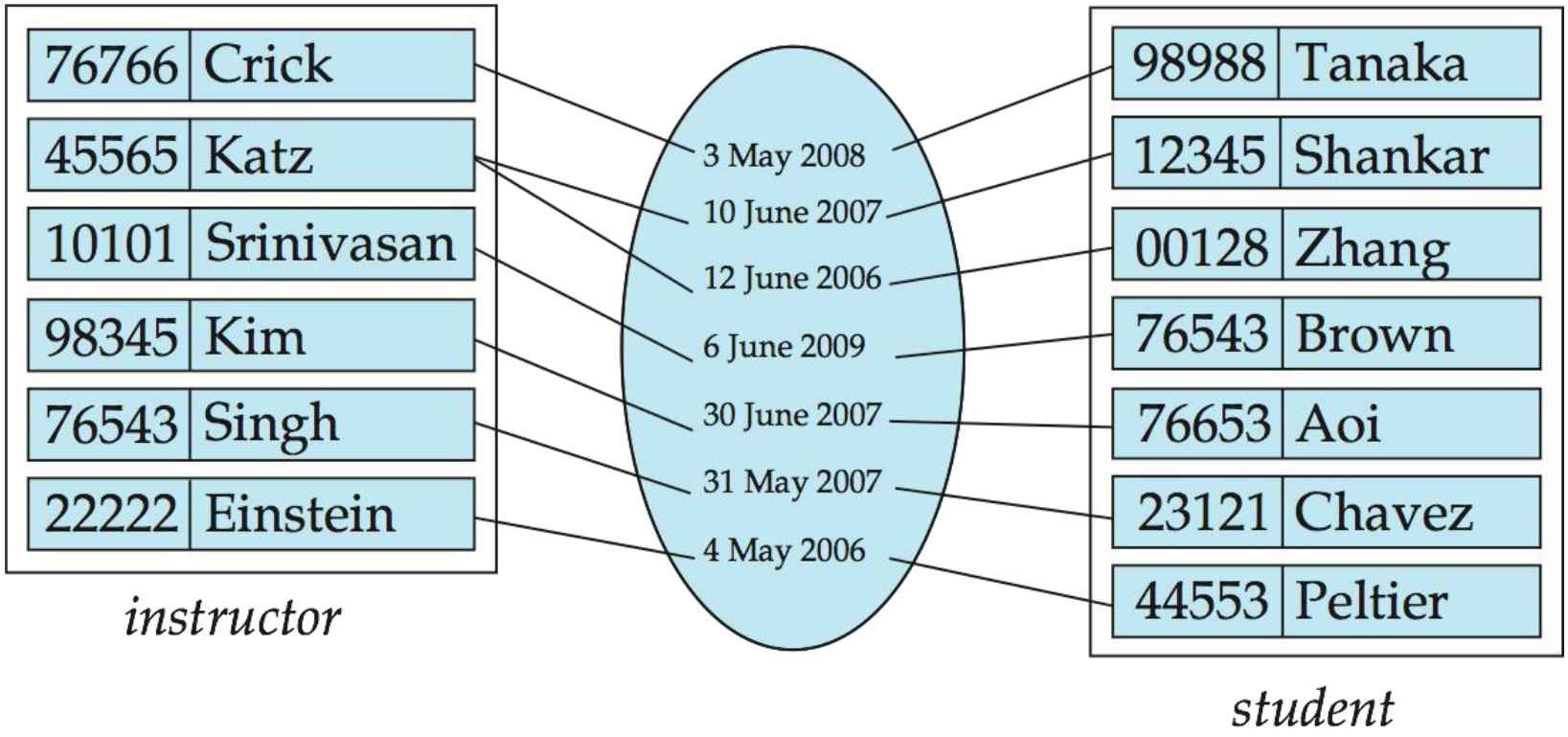
*student*

# Figure 7.02



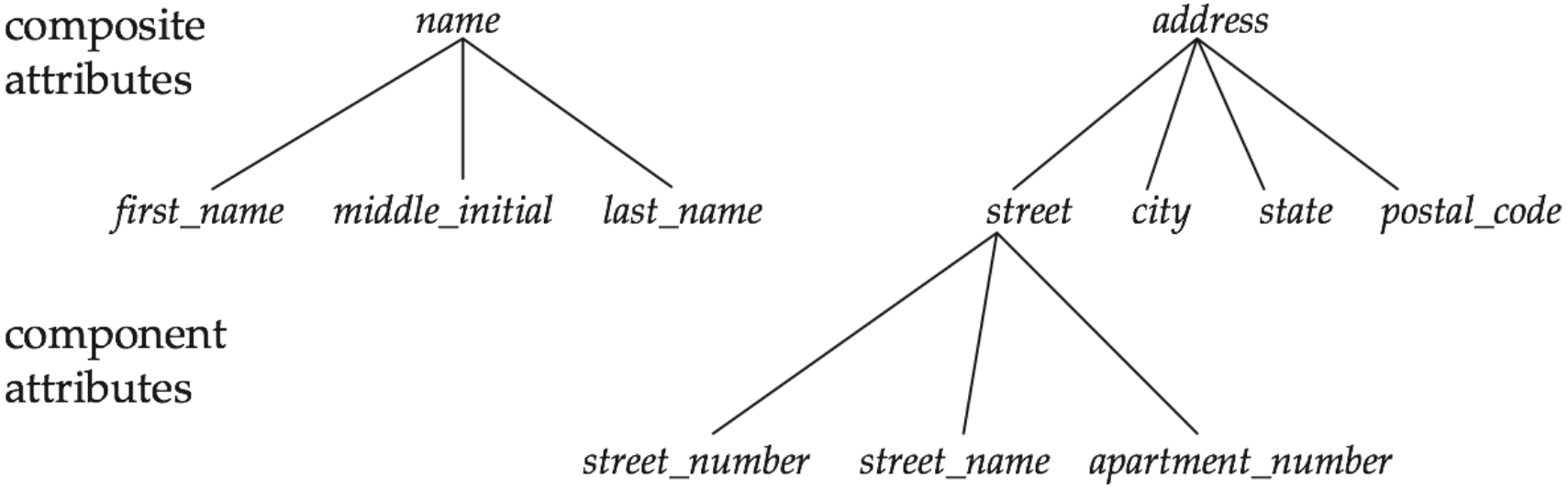

 

# Figure 7.03



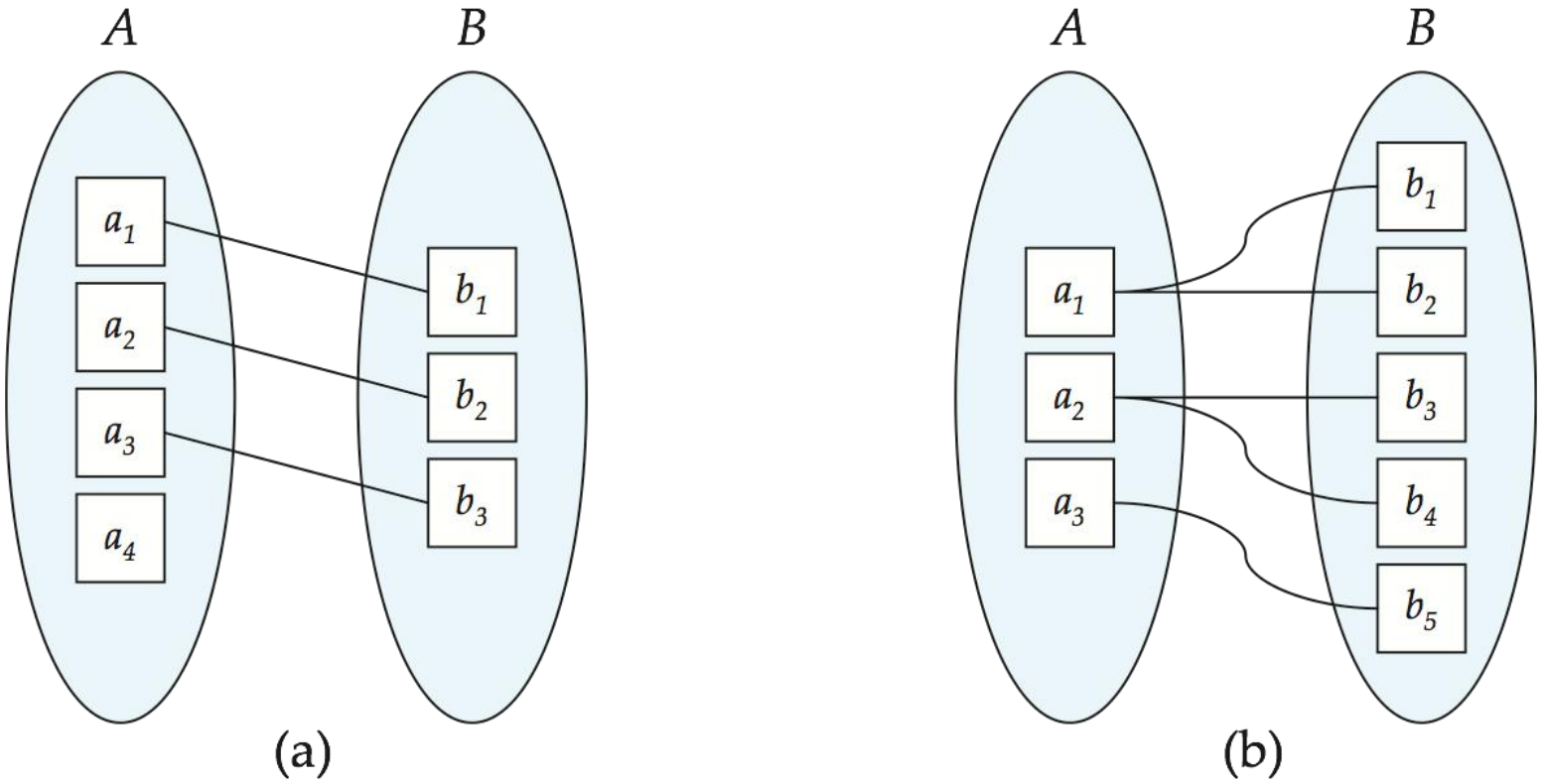

 

# Figure 7.04



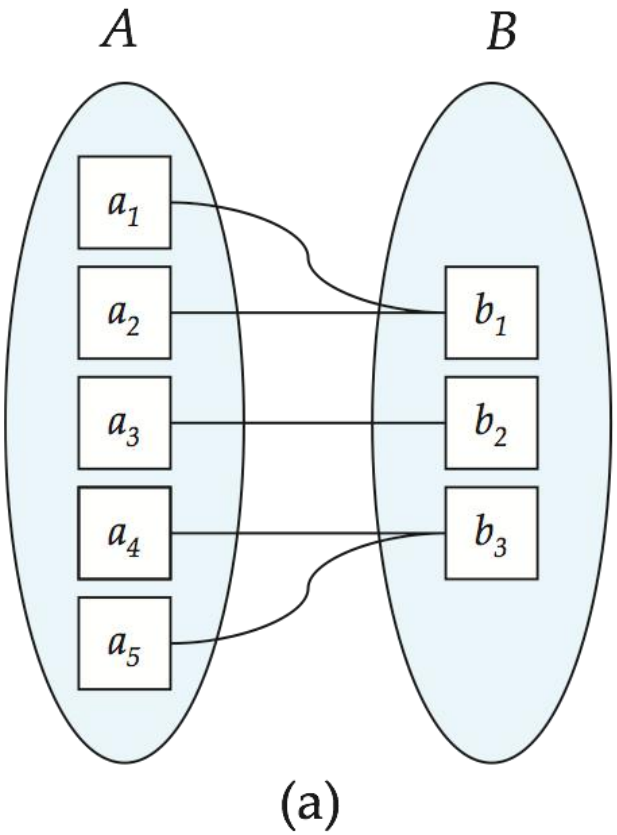

 

# Figure 7.05

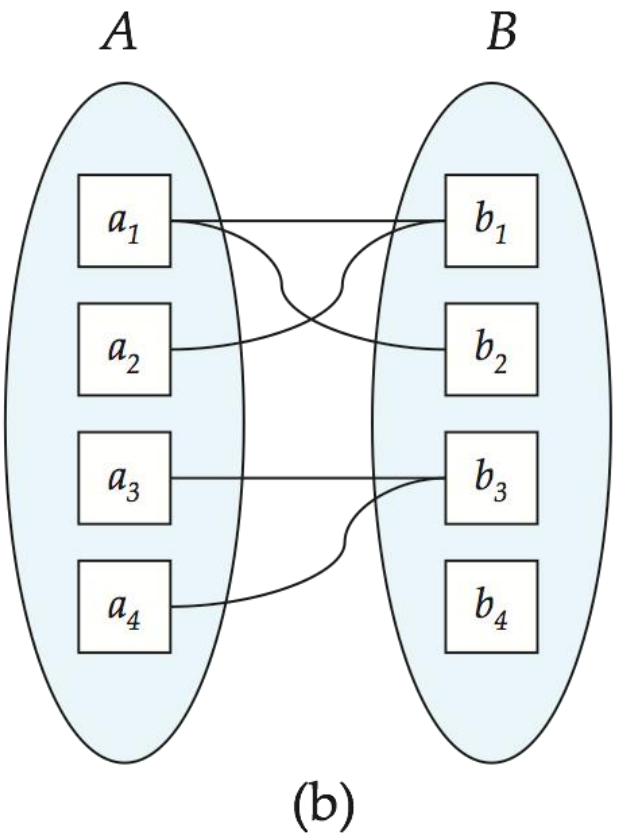


# Figure 7.06

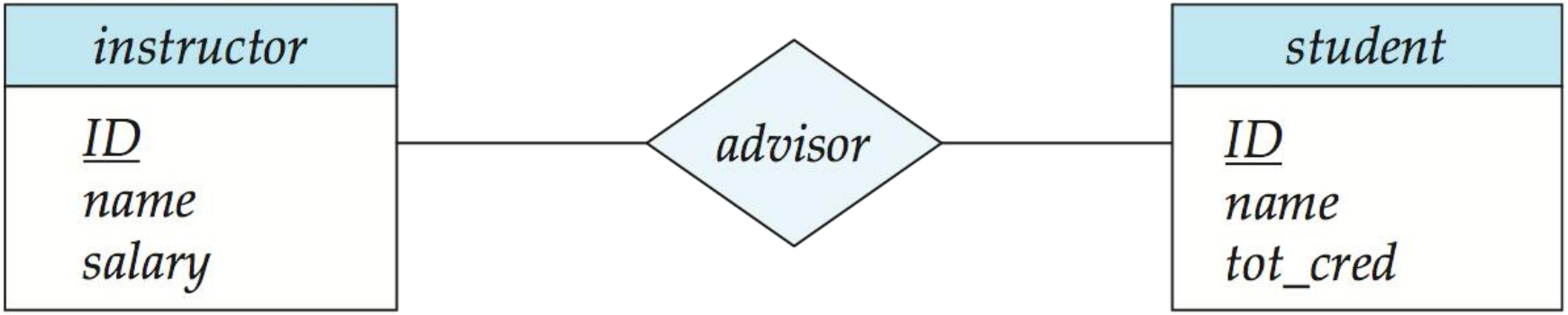



(a)

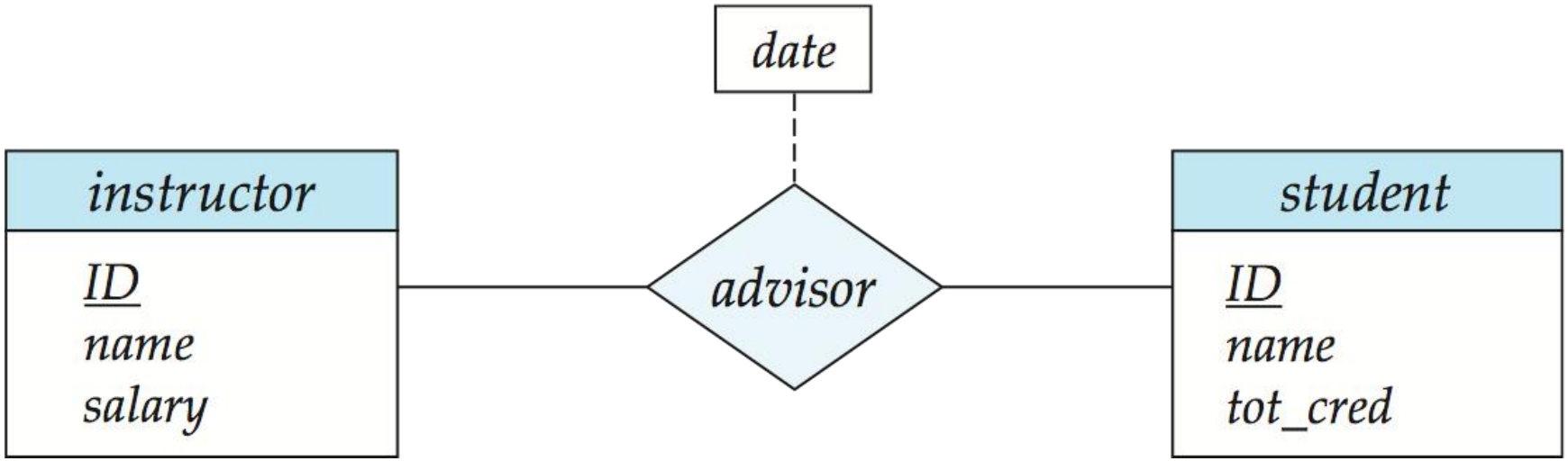



(b)

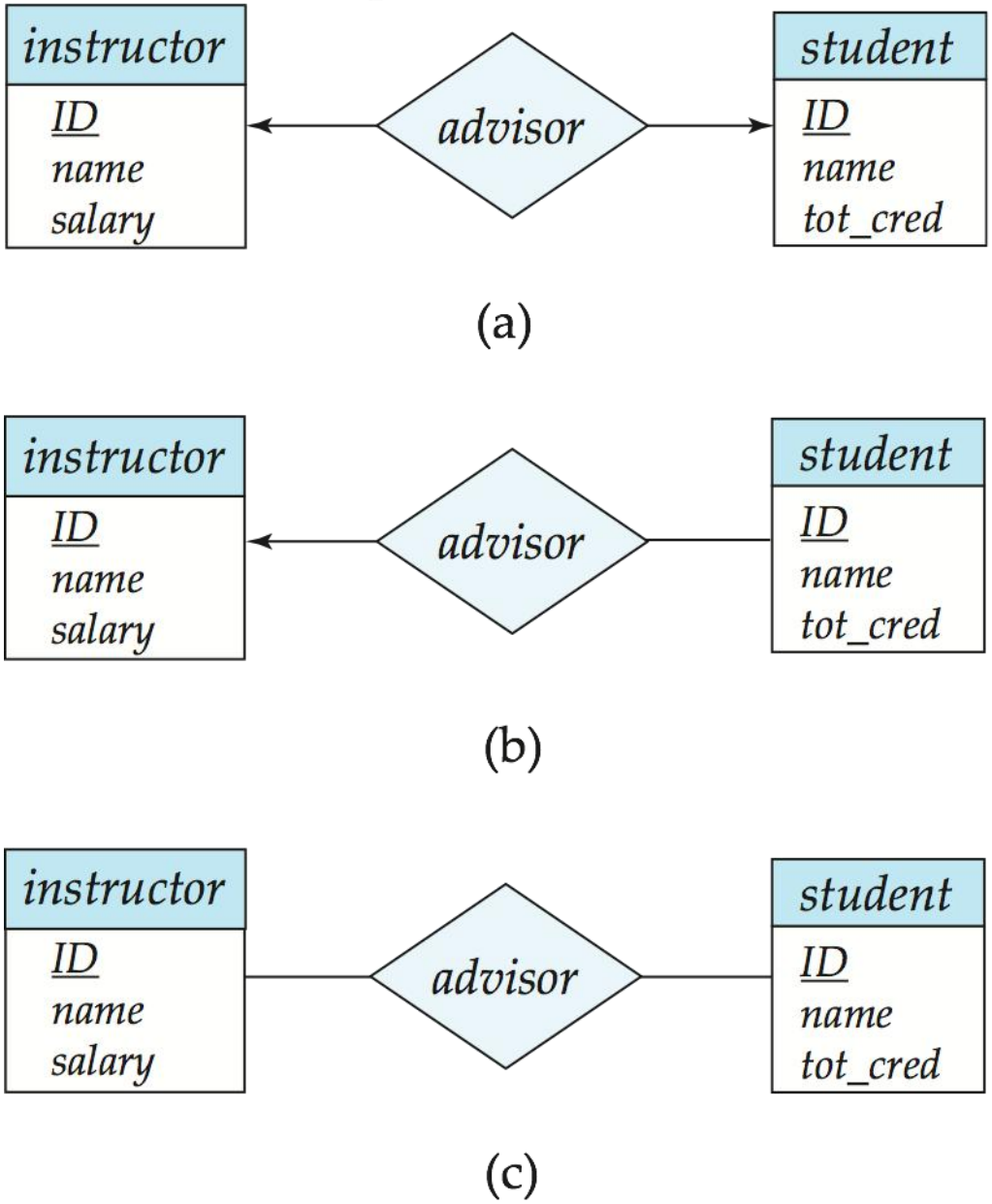
# Figure 7.07



# Figure 7.08

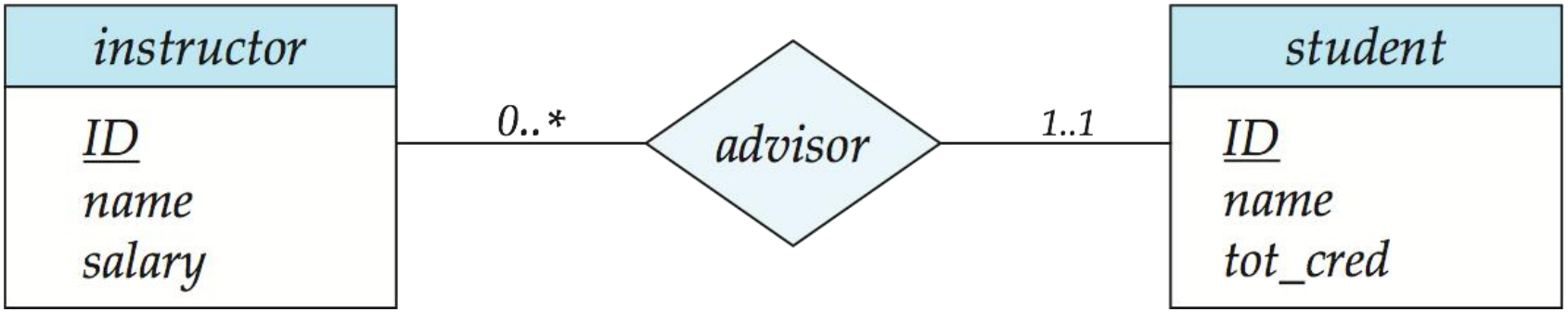

date
instructor
ID
name
salary
advisor
student
ID
name
tot_cred

# Figure 7.09

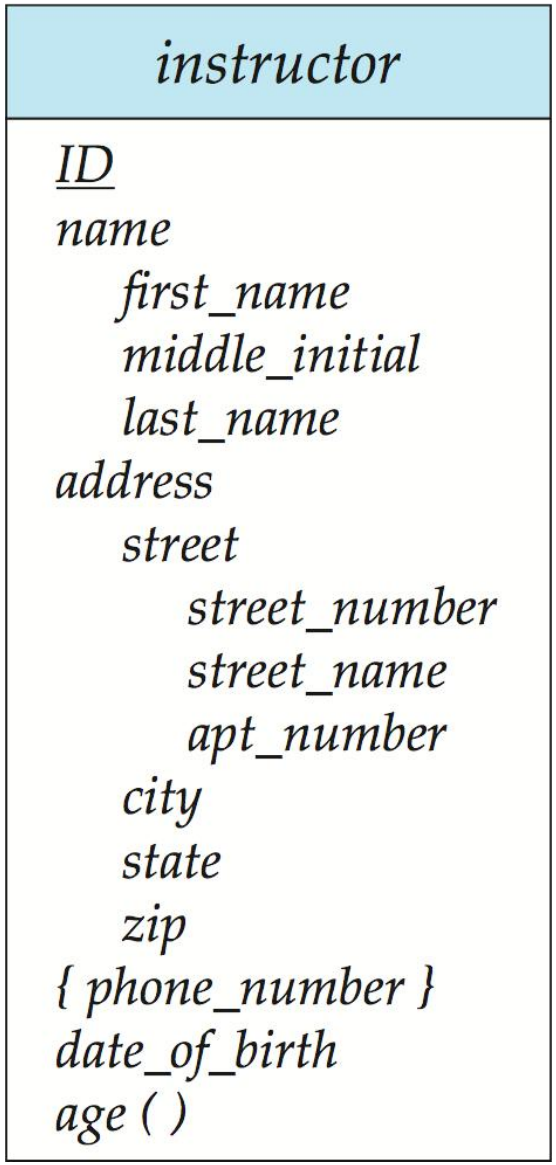


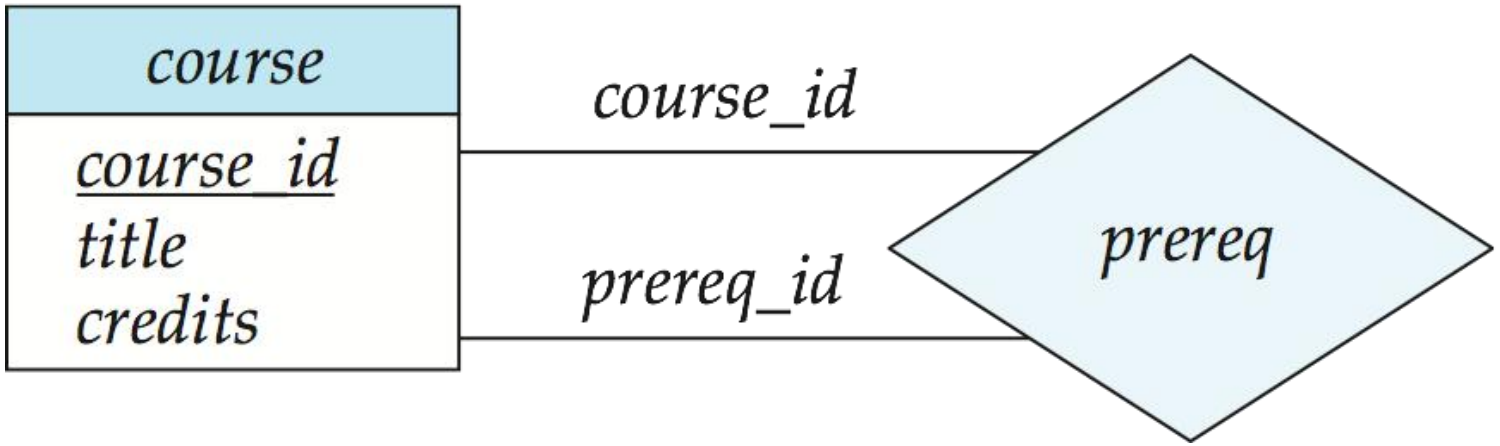
# Figure 7.10


instructor
ID
name
salary
0..*
advisor
1..1
student
ID
name
tot_cred

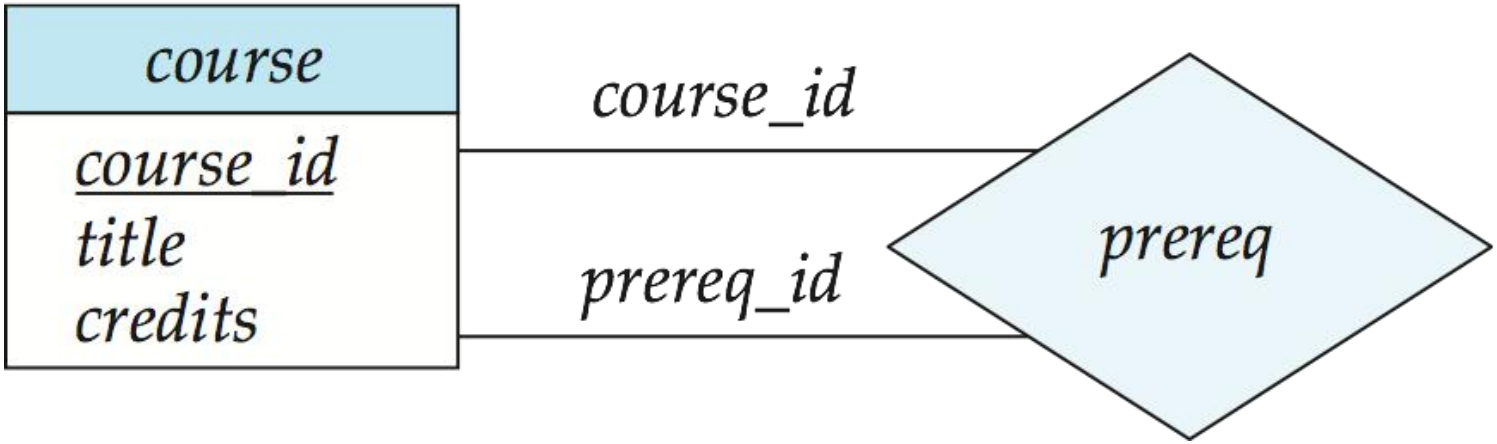
# Figure 7.11



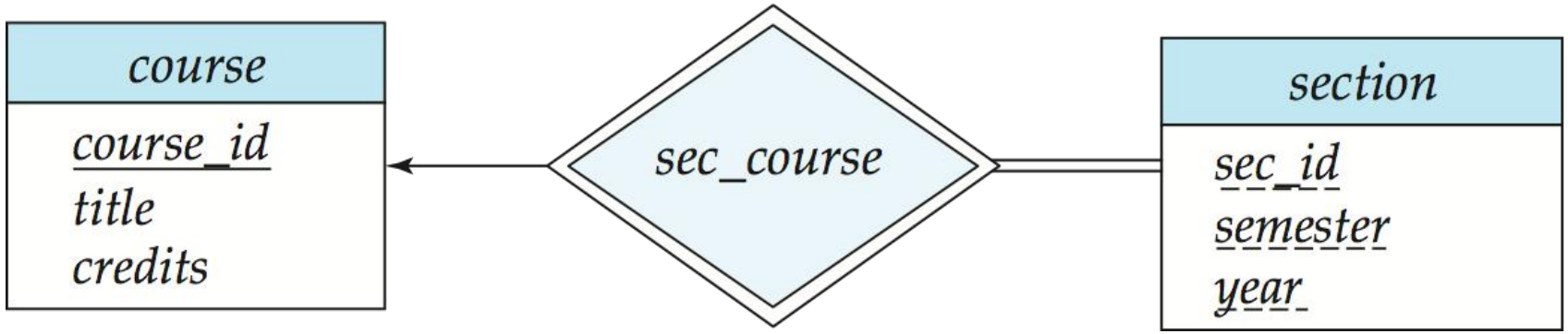
# Figure 7.12



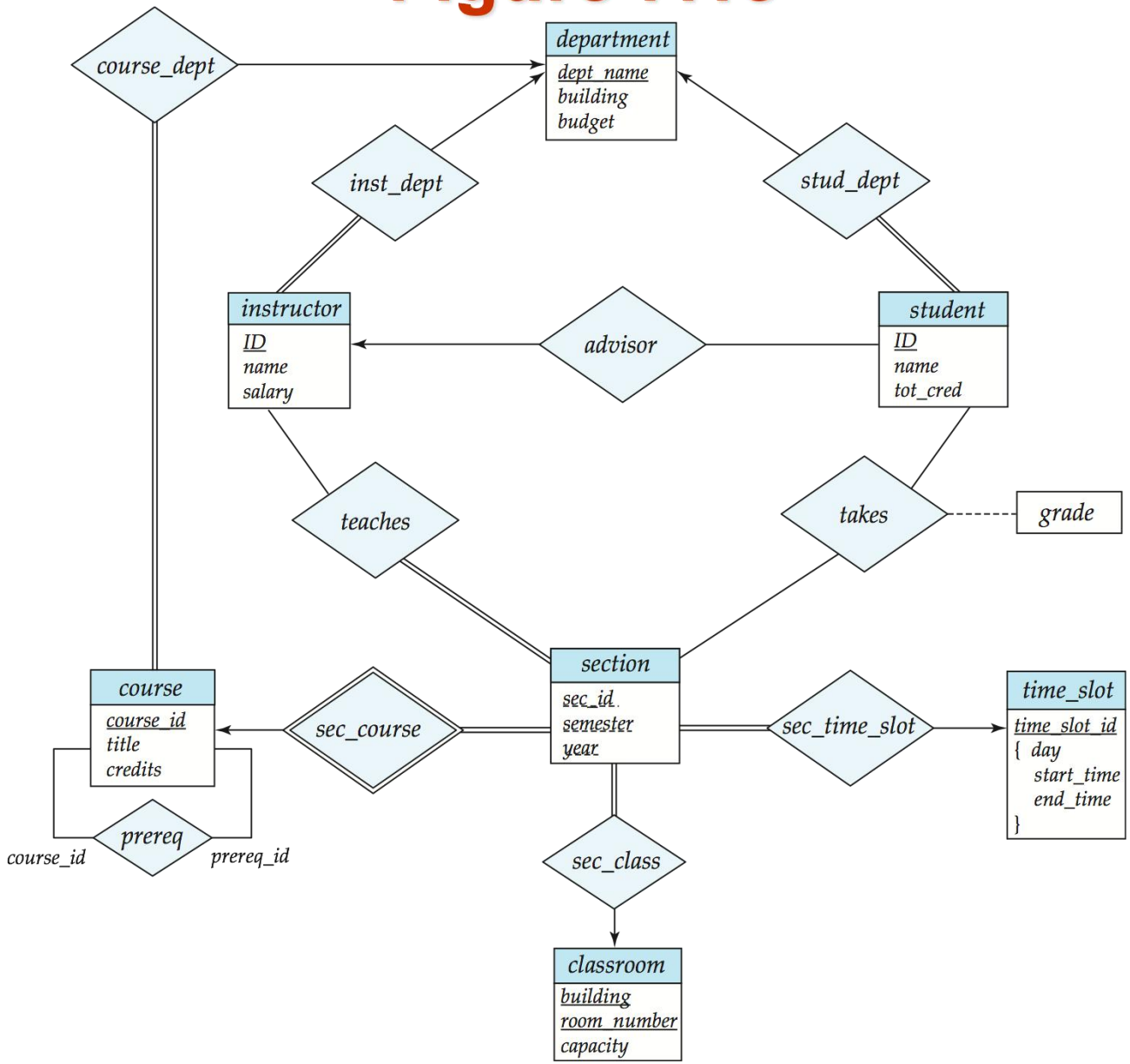
# Figure 7.13



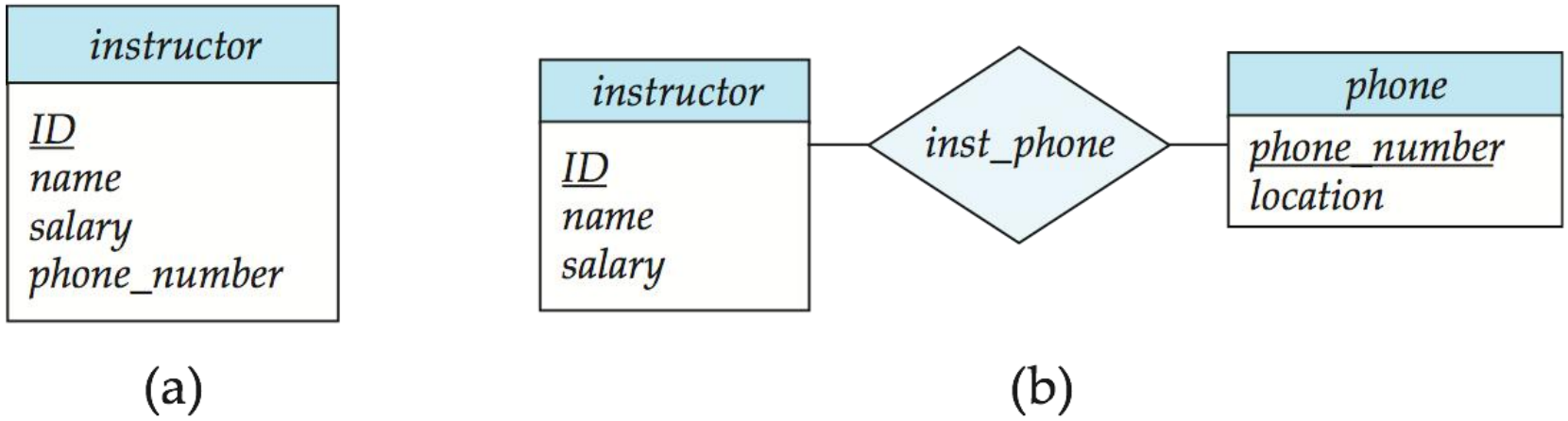
# Figure 7.14


course
course_id
title
credits
sec_course
section
sec_id
semester
year

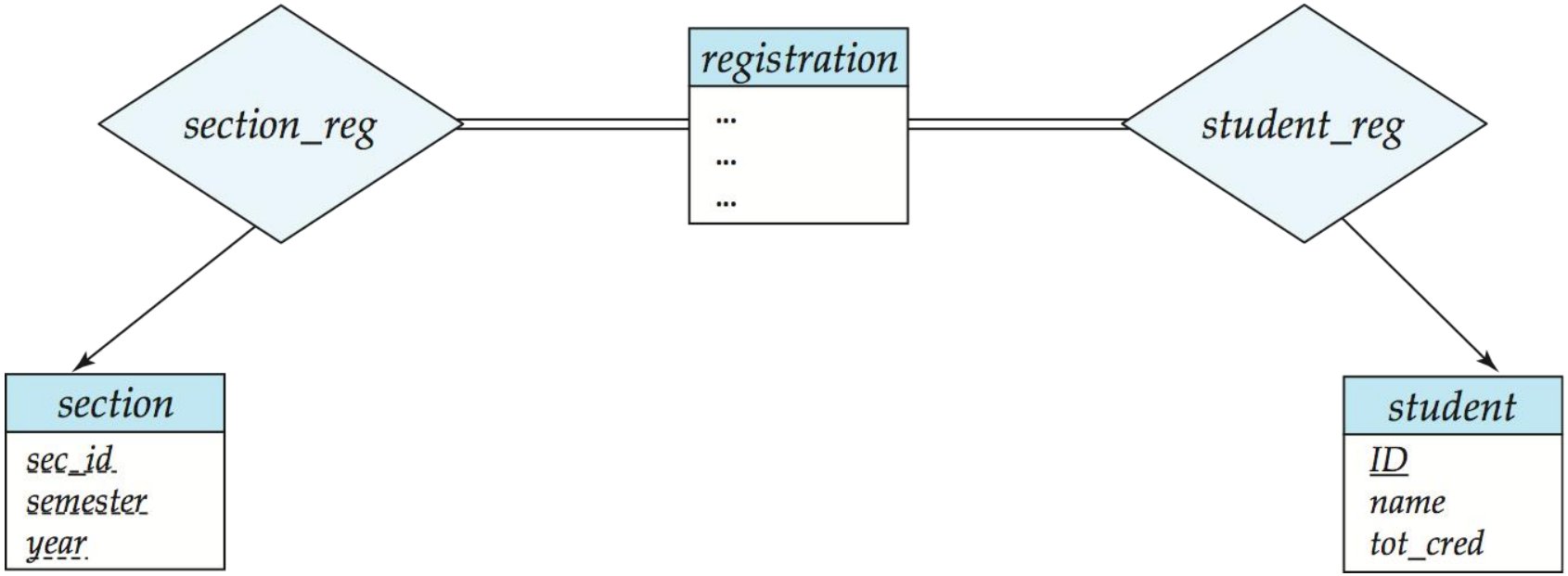
# Figure 7.15


department
dept_name
building
budget
course_dept
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

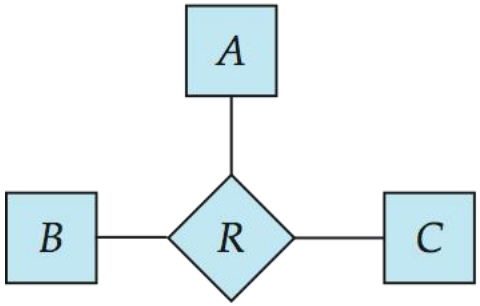
# Figure 7.17


instructor
ID
name
salary
phone_number
(a)
instructor
ID
name
salary
inst_phone
phone
phone_number
location
(b)

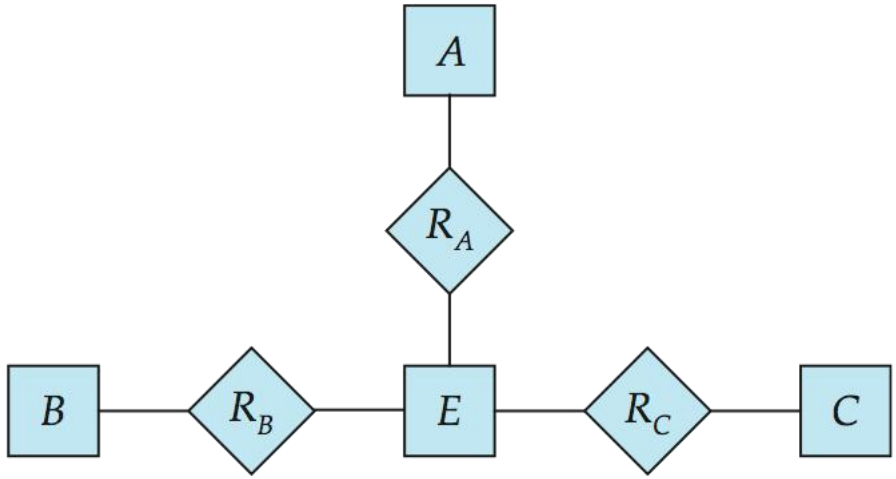
# Figure 7.18


section_reg
registration
...
...
...
student_reg
section
sec_id
semester
year
student
ID
name
tot_cred

# Figure 7.19

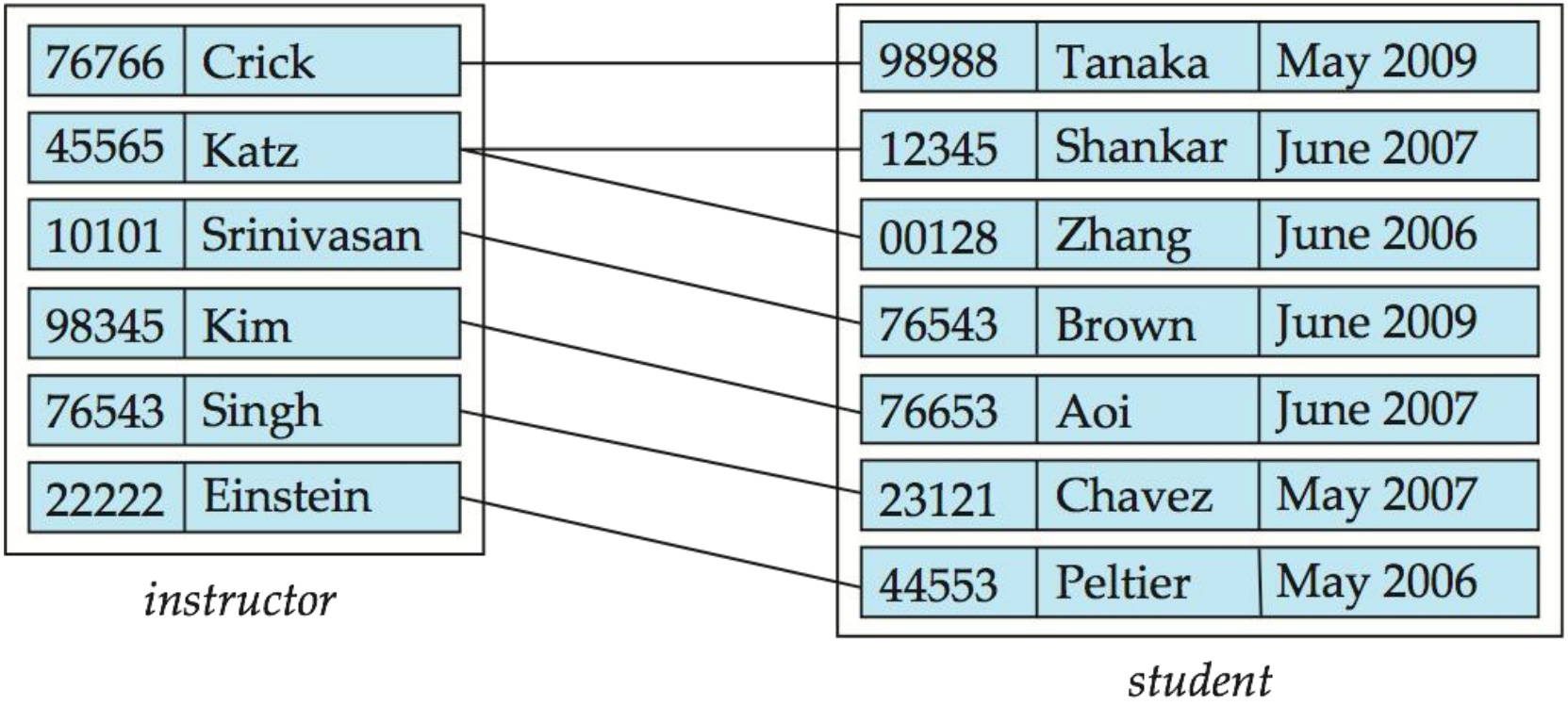



(a)



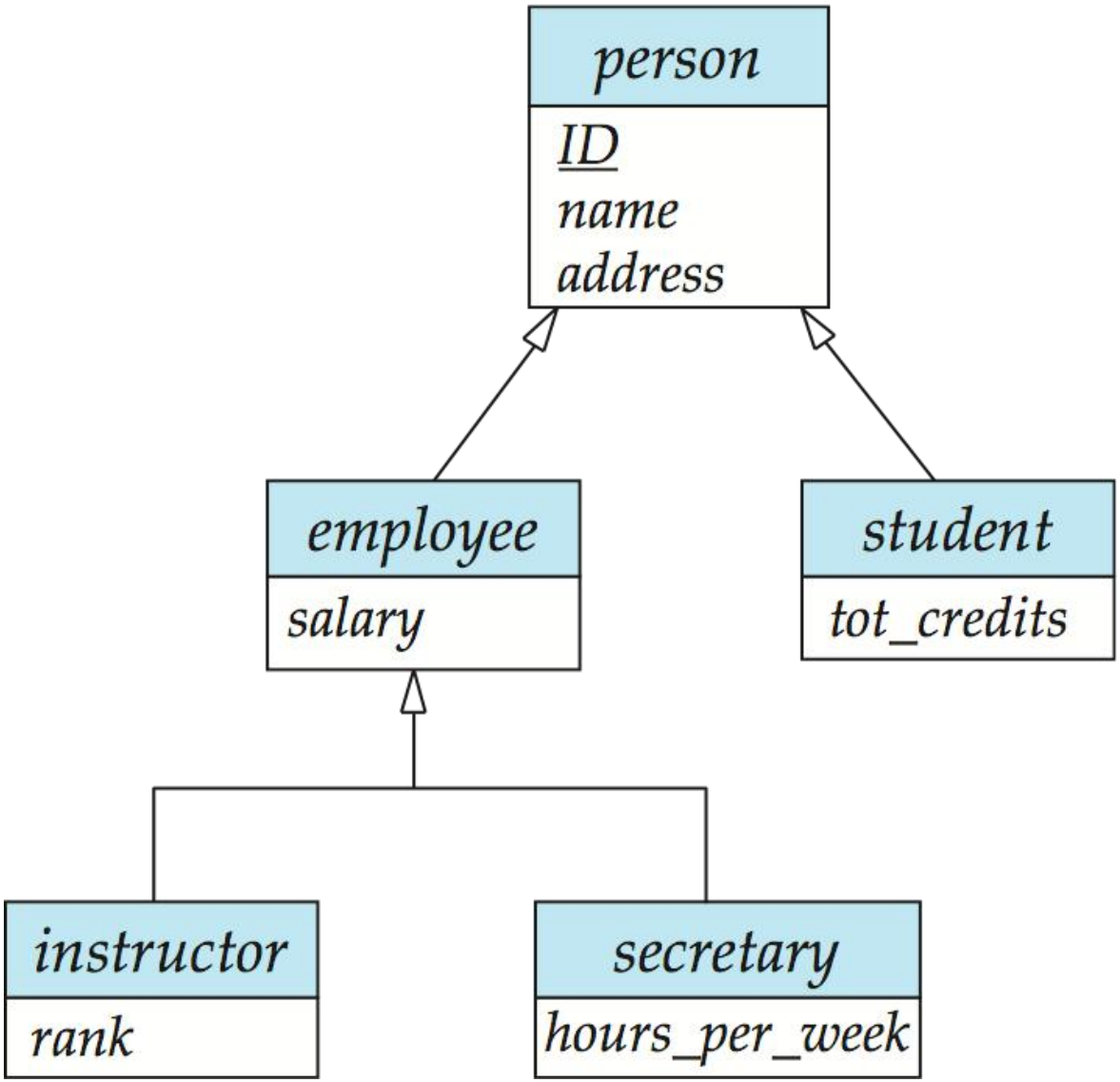

(b)

# Figure 7.20



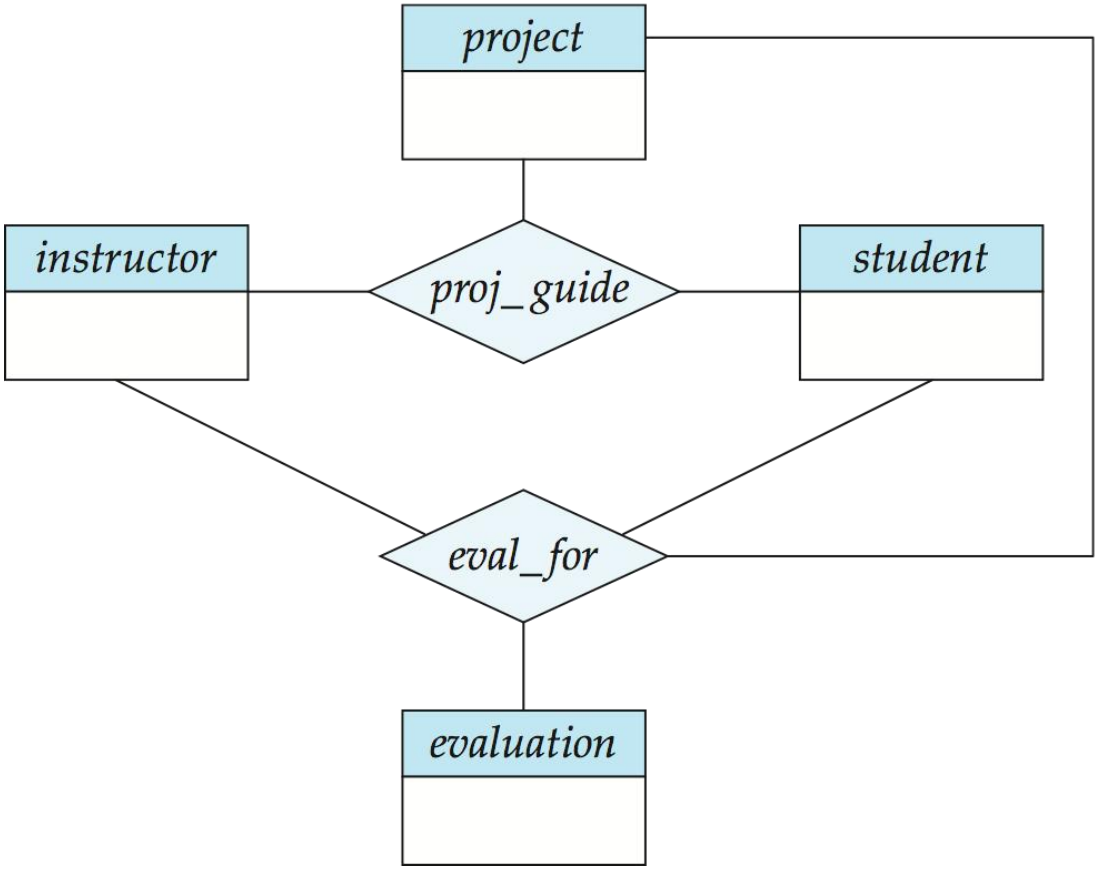

 

# Figure 7.21

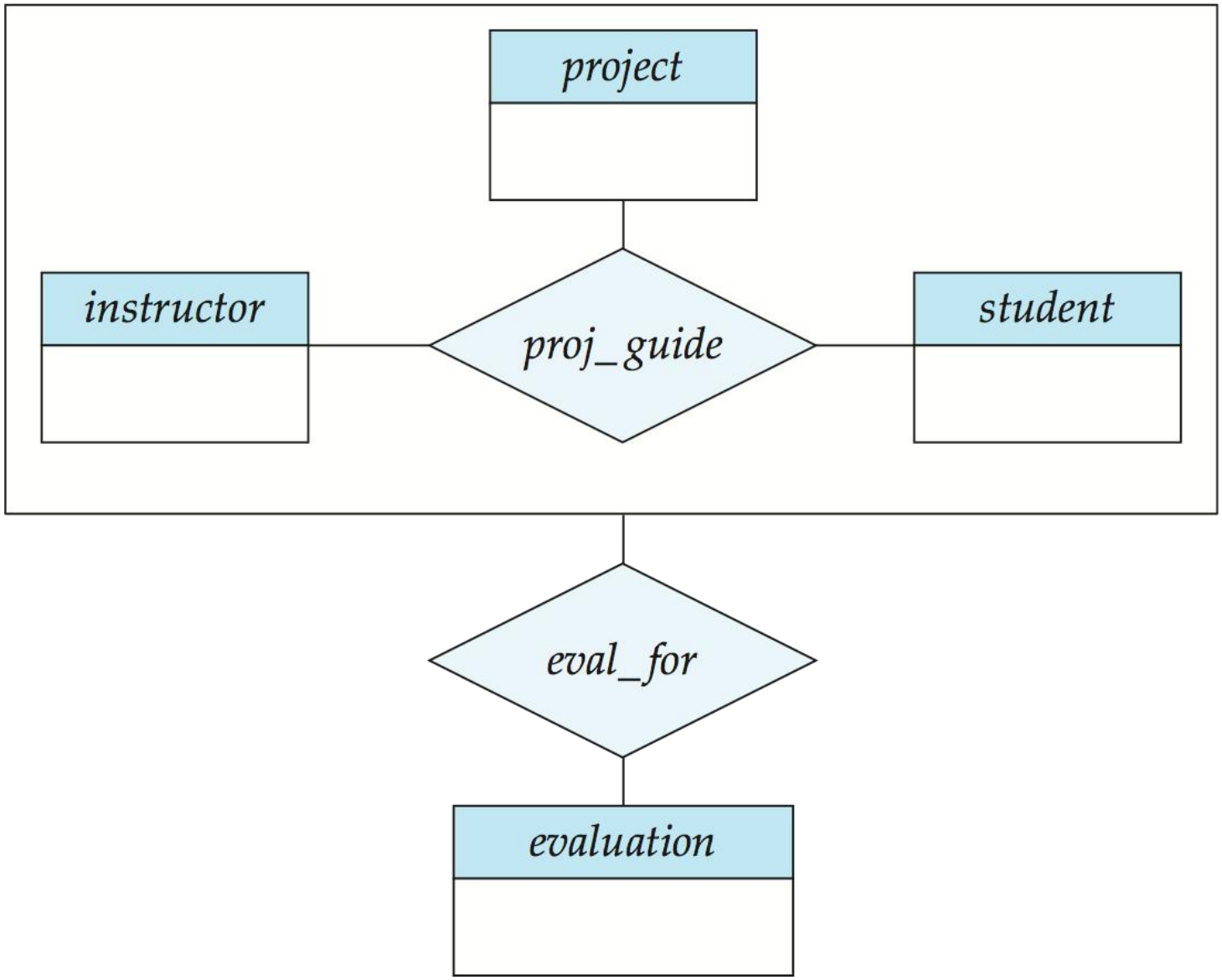


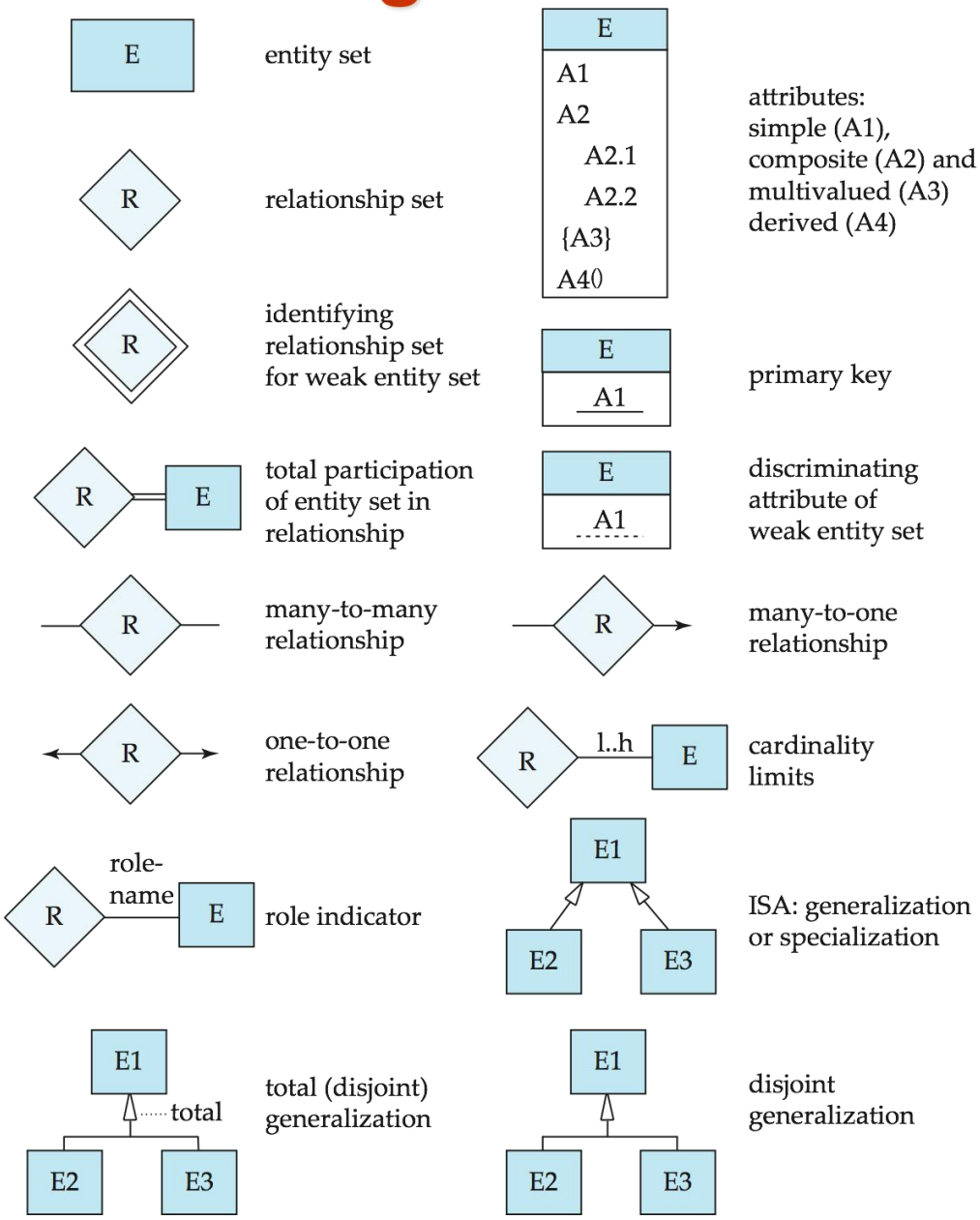
# Figure 7.22


project
instructor
proj_guide
student
eval_for
evaluation

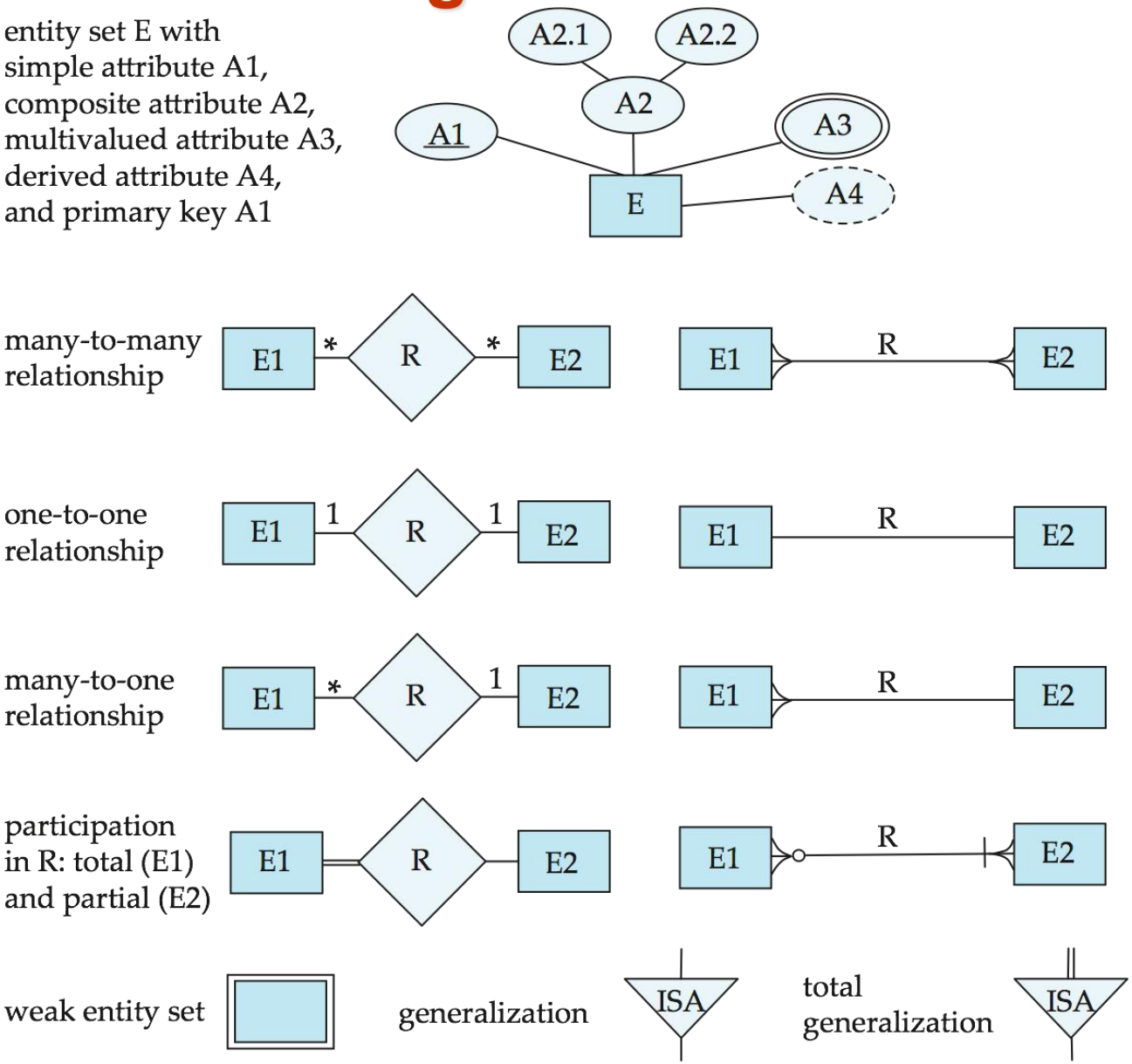
# Figure 7.23


project
instructor
proj_guide
student
eval_for
evaluation

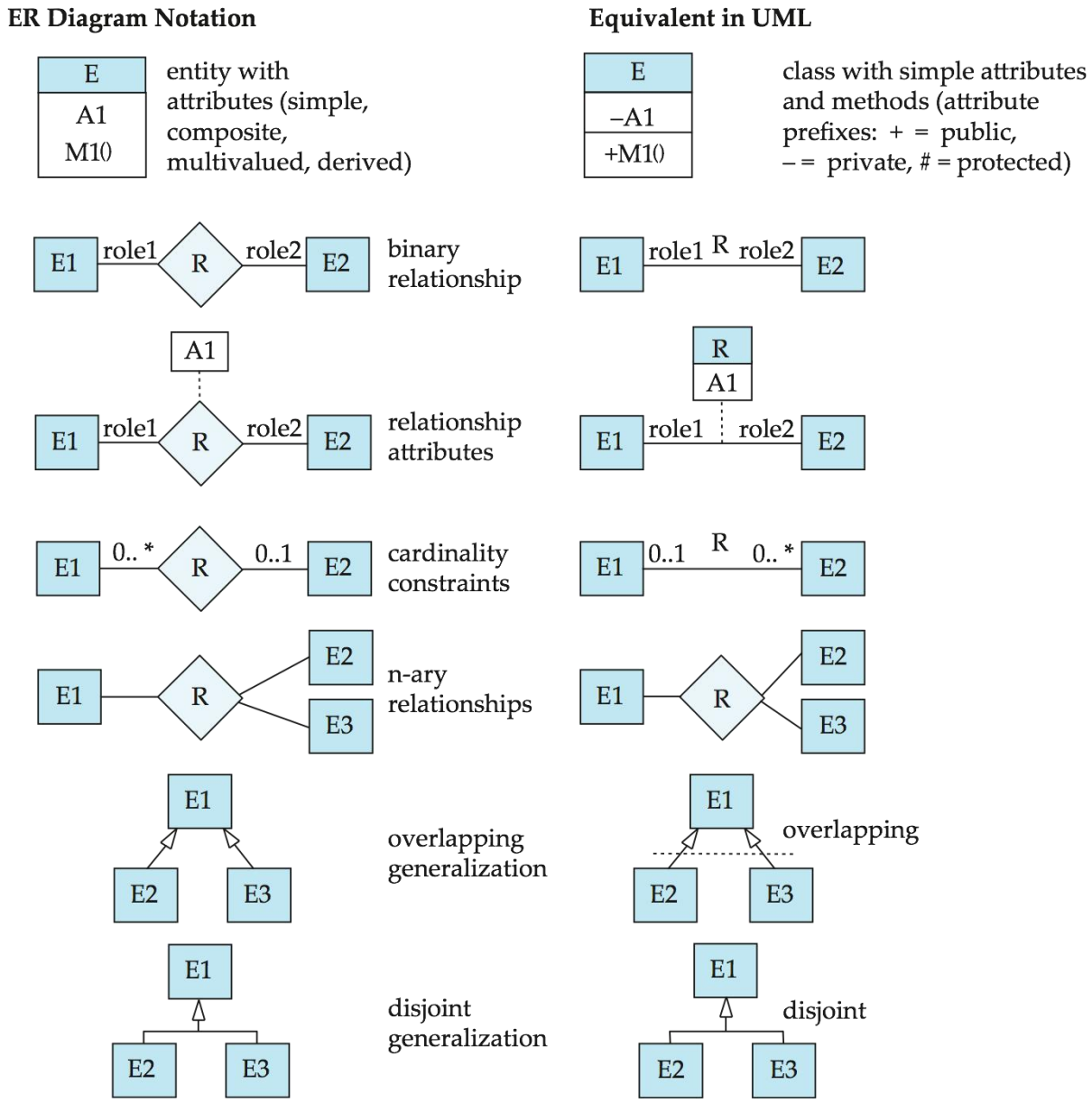
# Figure 7.24


E
entity set
R
relationship set
R
identifying relationship set for weak entity set
R
E
total participation of entity set in relationship
R
many-to-many relationship
R
one-to-one relationship
R
role-name
E
role indicator
E1
total
E2
E3
total (disjoint) generalization
E
A1
A2
A2.1
A2.2
{A3}
A40
attributes: simple (A1), composite (A2) and multivalued (A3) derived (A4)
E
A1
primary key
E
A1
discriminating attribute of weak entity set
R
many-to-one relationship
R
1..h
E
cardinality limits
E1
E2
E3
ISA: generalization or specialization
E1
E2
E3
disjoint generalization

# Figure 7.25

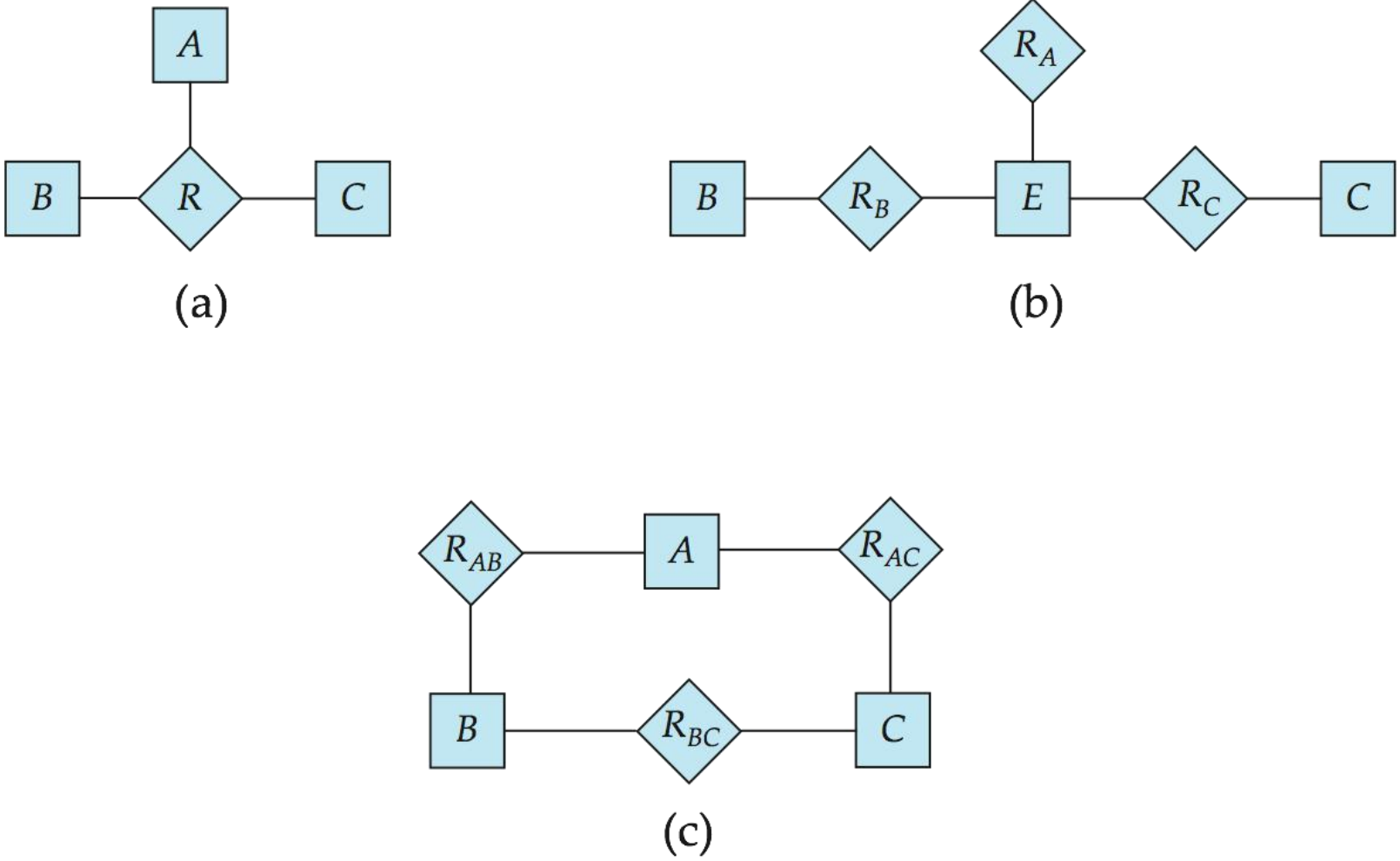

entity set E with
simple attribute A1,
composite attribute A2,
multivalued attribute A3,
derived attribute A4,
and primary key A1
A2.1
A2.2
A2
A1
A3
E
A4
many-to-many
relationship
E1
*
R
*
E2
E1
R
E2
one-to-one
relationship
E1
1
R
1
E2
E1
R
E2
many-to-one
relationship
E1
*
R
1
E2
E1
R
E2
participation
in R: total (E1)
and partial (E2)
E1
R
E2
E1
R
E2
weak entity set
generalization
ISA
total
generalization
ISA

# Figure 7.26



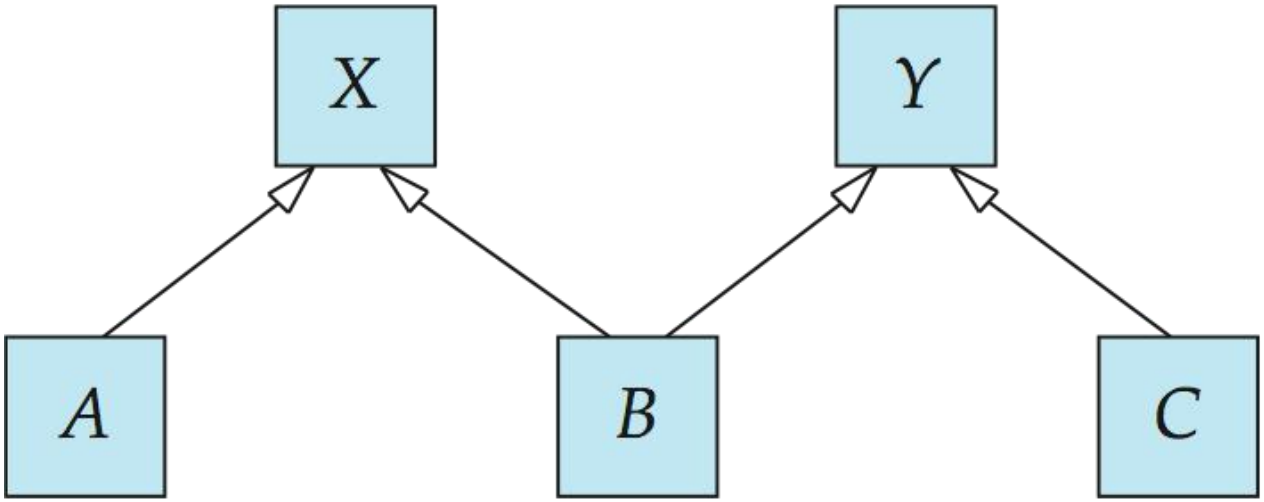

 

# Figure 7.27

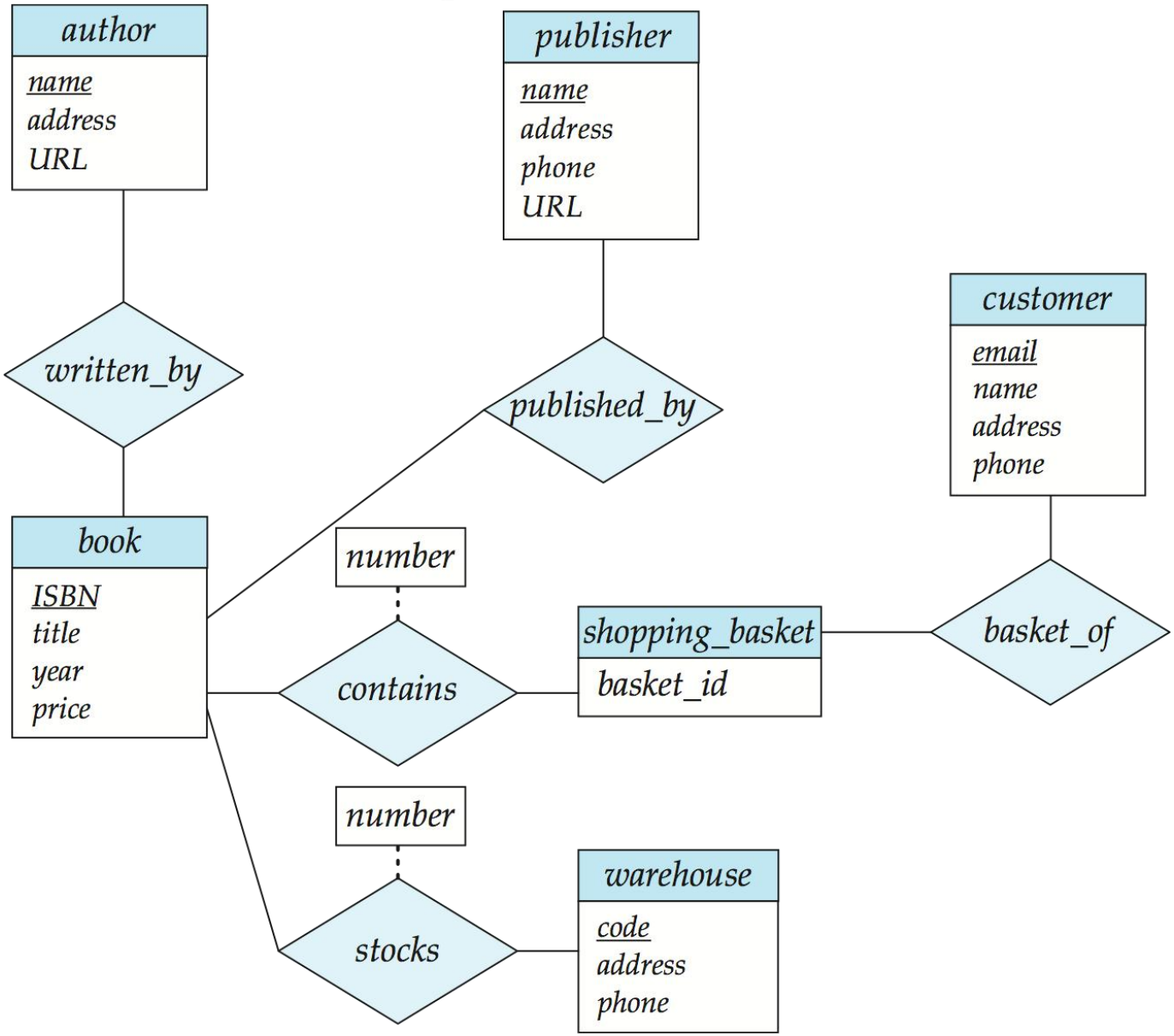


# Figure 7.28



# Figure 7.29


author
name
address
URL
publisher
name
address
phone
URL
customer
email
name
address
phone
written_by
published_by
book
ISBN
title
year
price
number
contains
shopping_basket
basket_id
basket_of
number
stocks
warehouse
code
address
phone