CS 4092 Database Design and Development (DDD)

06: ER Model

Seokki Lee

Slides are adapted from:

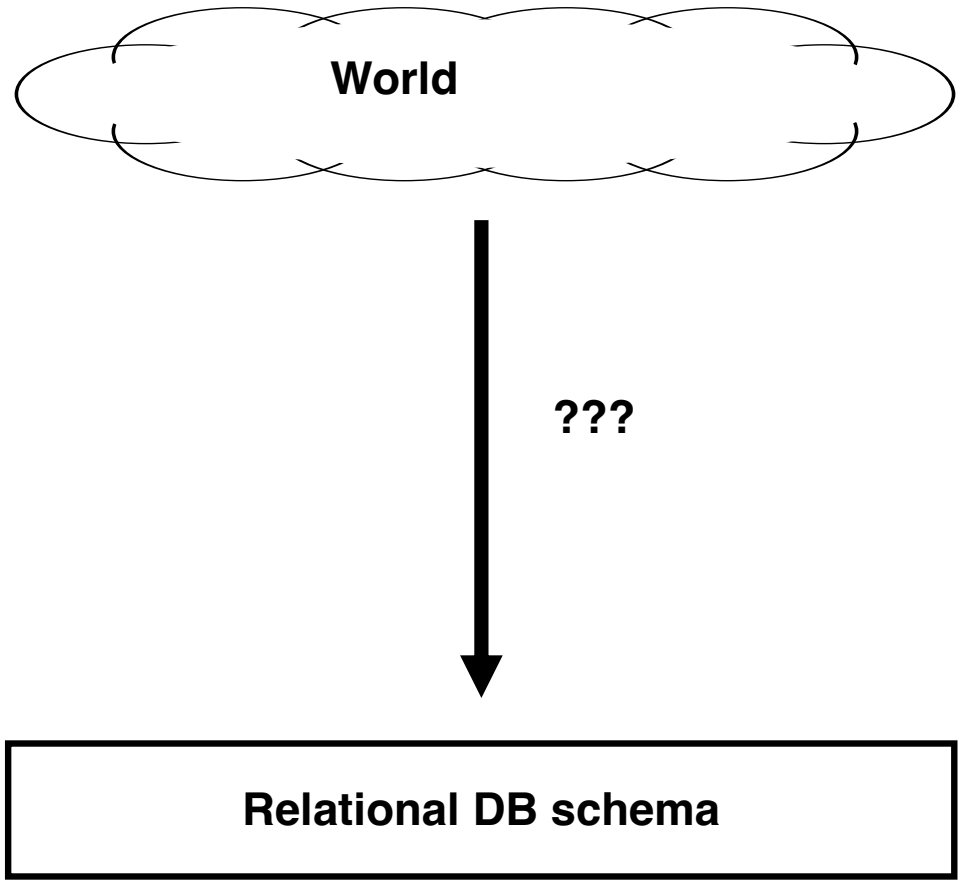
Database System Concepts, 6th & 7th Ed. ©Silberschatz, Korth and Sudarshan

Outline

- Overview of the Design Process
- The Entity-Relationship Model
- Complex Attributes
- Mapping Cardinalities
- Primary Key
- Removing Redundant Attributes in Entity Sets
- Reducing ER Diagrams to Relational Schemas
- Extended E-R Features
- Entity-Relationship Design Issues
- Alternative Notations for Modeling Data
- Other Aspects of Database Design

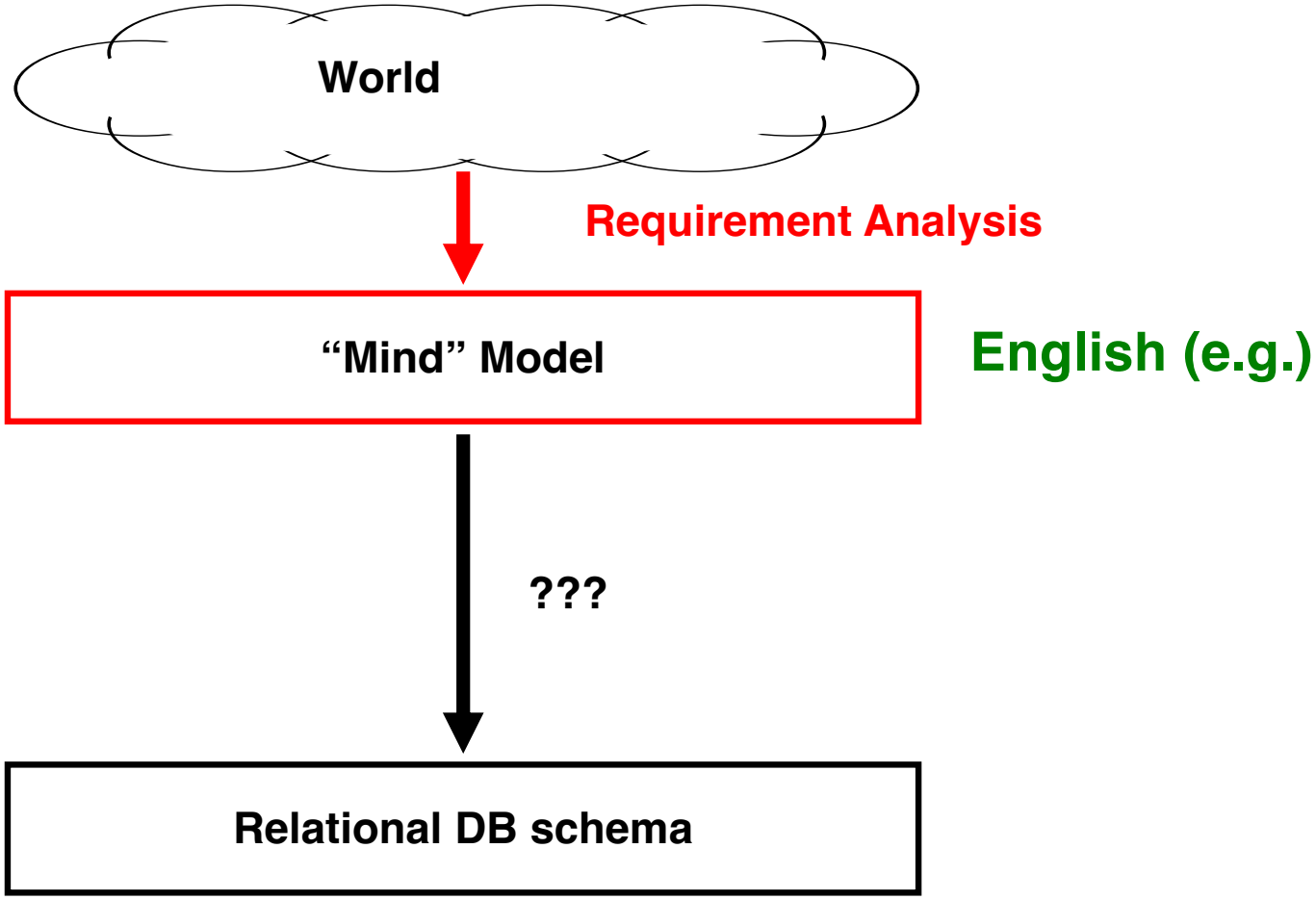
Database Design

- Understanding requirements is difficult for real-world applications.



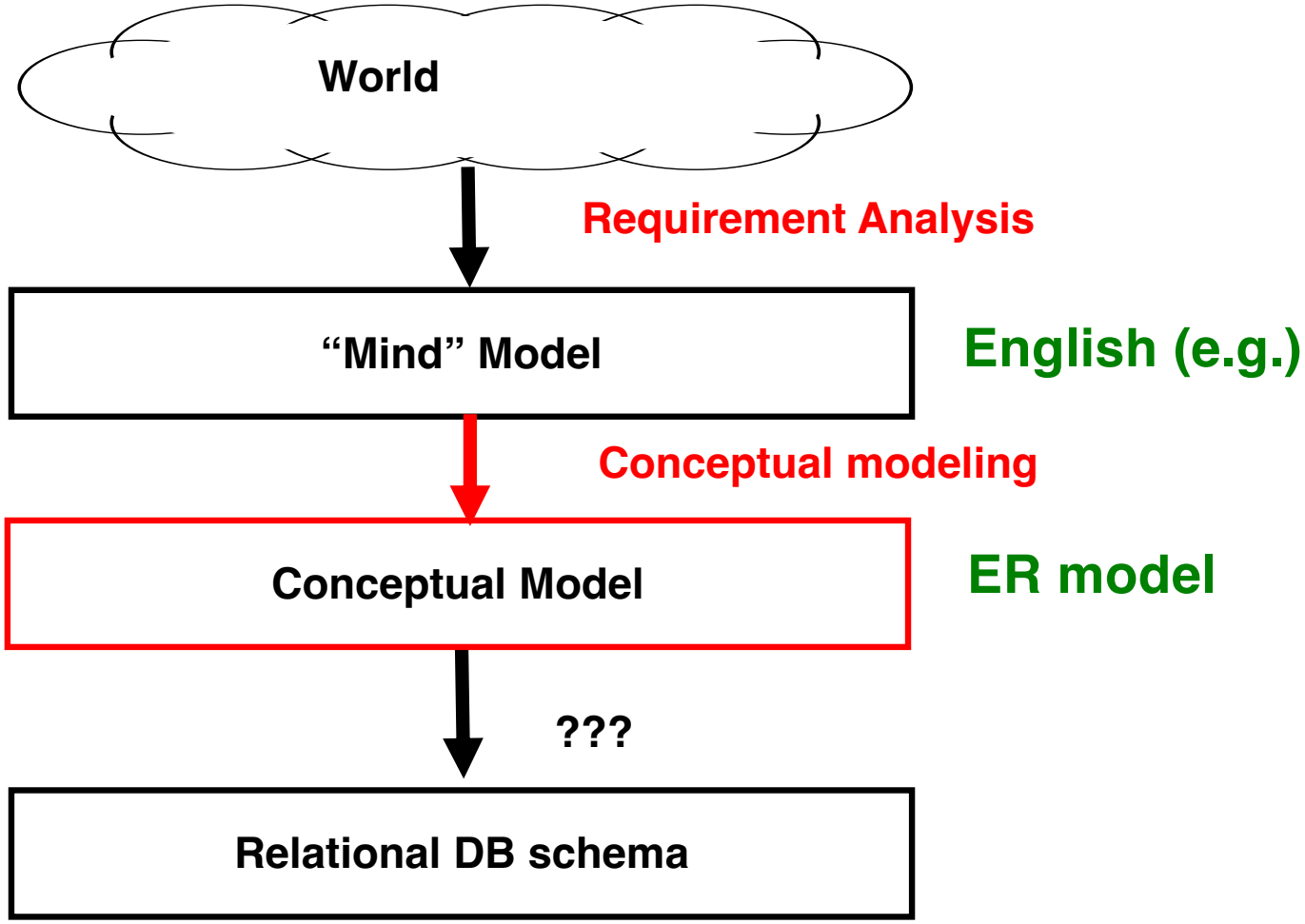
Database Design

First: need to develop a “mind”-model based on a requirement analysis



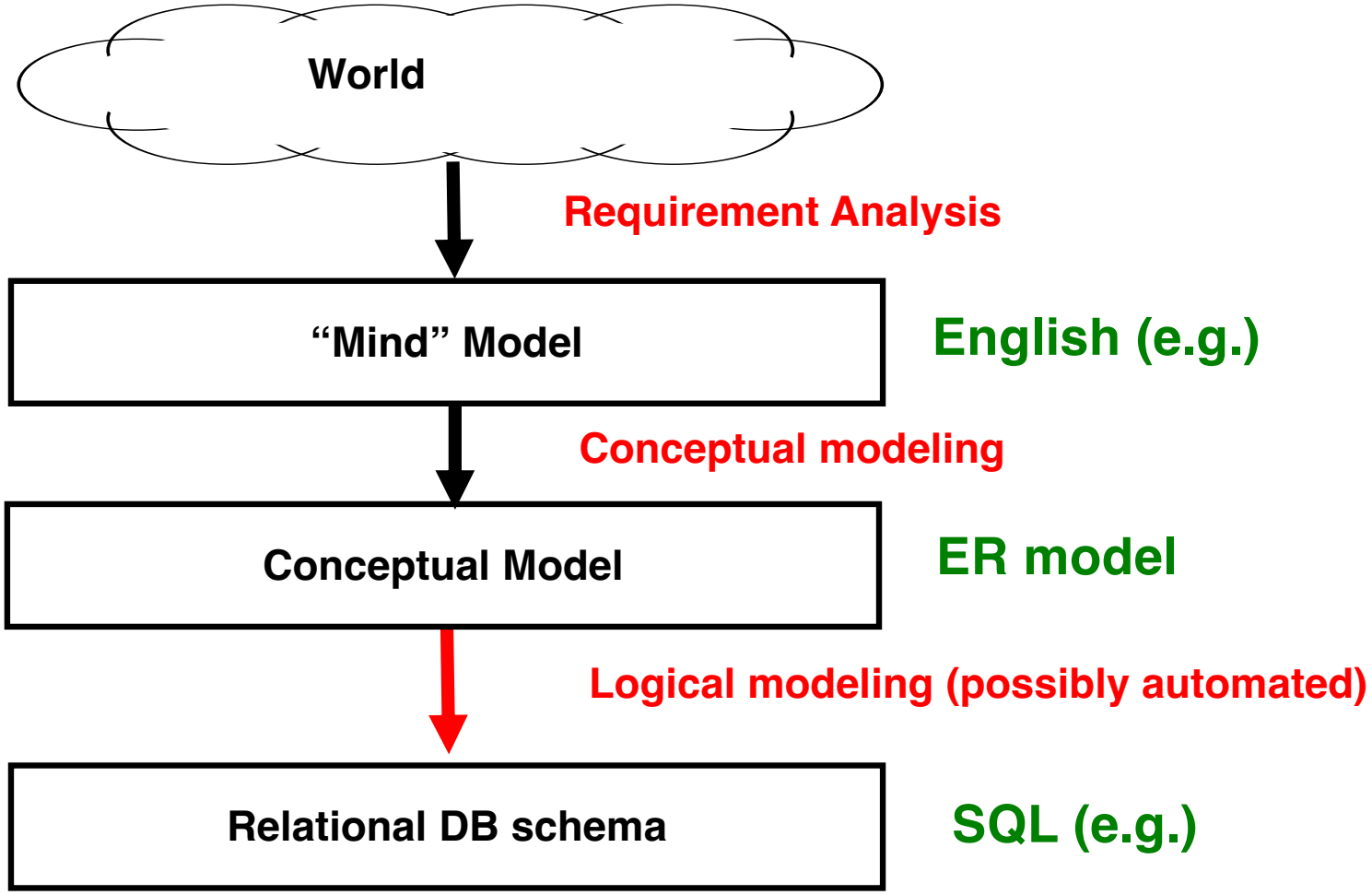
Database Design

Second: Formalize this model by developing a conceptual model



Database Design

Third: Logical and physical modeling



Design Phases

- Initial phase -- characterize fully the data needs of the prospective database users.
  - Specification of user requirements
- Second phase -- choosing a data model
  - Applying the concepts of the chosen data model
  - Translating these requirements into a conceptual schema of the database.
  - A fully developed conceptual schema indicates the functional requirements of the enterprise.
    - Describe the kinds of operations (or transactions) that will be performed on the data.

Design Phases (Cont.)

- Final Phase -- Moving from an abstract data model to the implementation of the database
  - Logical Design – Deciding on the database schema.
    - Database design requires that we find a “good” collection of relation schemas.
      - Business decision – What attributes should we record in the database?
      - Computer Science decision – What relation schemas should we have and how should the attributes be distributed among the various relation schemas?
  - Physical Design – Deciding on the physical layout of the database

Requirement Analysis Example Zoo

- The zoo stores information about animals, cages, and zoo keepers.
- Animals are of a certain species and have a name. For each animal, we want to record its weight and age.
- Each cage is located in a section of the zoo. Cages can house animals, but there may be cages that are currently empty. Cages have a size in square meter.
- Zoo keepers are identified by their social security number. We store a first name and last name for each zoo keeper. Zoo keepers are assigned to cages they have to take care of (clean, …). Each cage that is not empty has a zoo keeper assigned to it. A zoo keeper can take care of several cages. Each zoo keeper takes care of at least one cage.

Design Alternatives

- In designing a database schema, we must ensure that we avoid two major pitfalls:
  - **Redundancy**: a bad design may result in repeat information.
    - Redundant representation of information may lead to data inconsistency among the various copies of information.
  - **Incompleteness**: a bad design may make certain aspects of the enterprise difficult or impossible to model.
- Avoiding bad designs is not enough. There may be a large number of good designs from which we must choose.

Design Approaches

- **Entity Relationship** Model (covered in this chapter)
  - Models an enterprise as a collection of ***entities*** and ***relationships***
    - Entity: a “thing” or “object” in the enterprise that is distinguishable from other objects
      - Described by a set of ***attributes***
    - Relationship: an association among several entities
  - Represented diagrammatically by an *entity-relationship diagram:*
- **Normalization** Theory (Chapter 7)
  - Formalize what designs are bad, and test for them
  - Study the process of creating good designs using a broader set of constraints

Modeling – ER model

- 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 set of relationships of the same type. It is a mathematical relation among $n \geq 2$ entities, each taken from entity sets

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

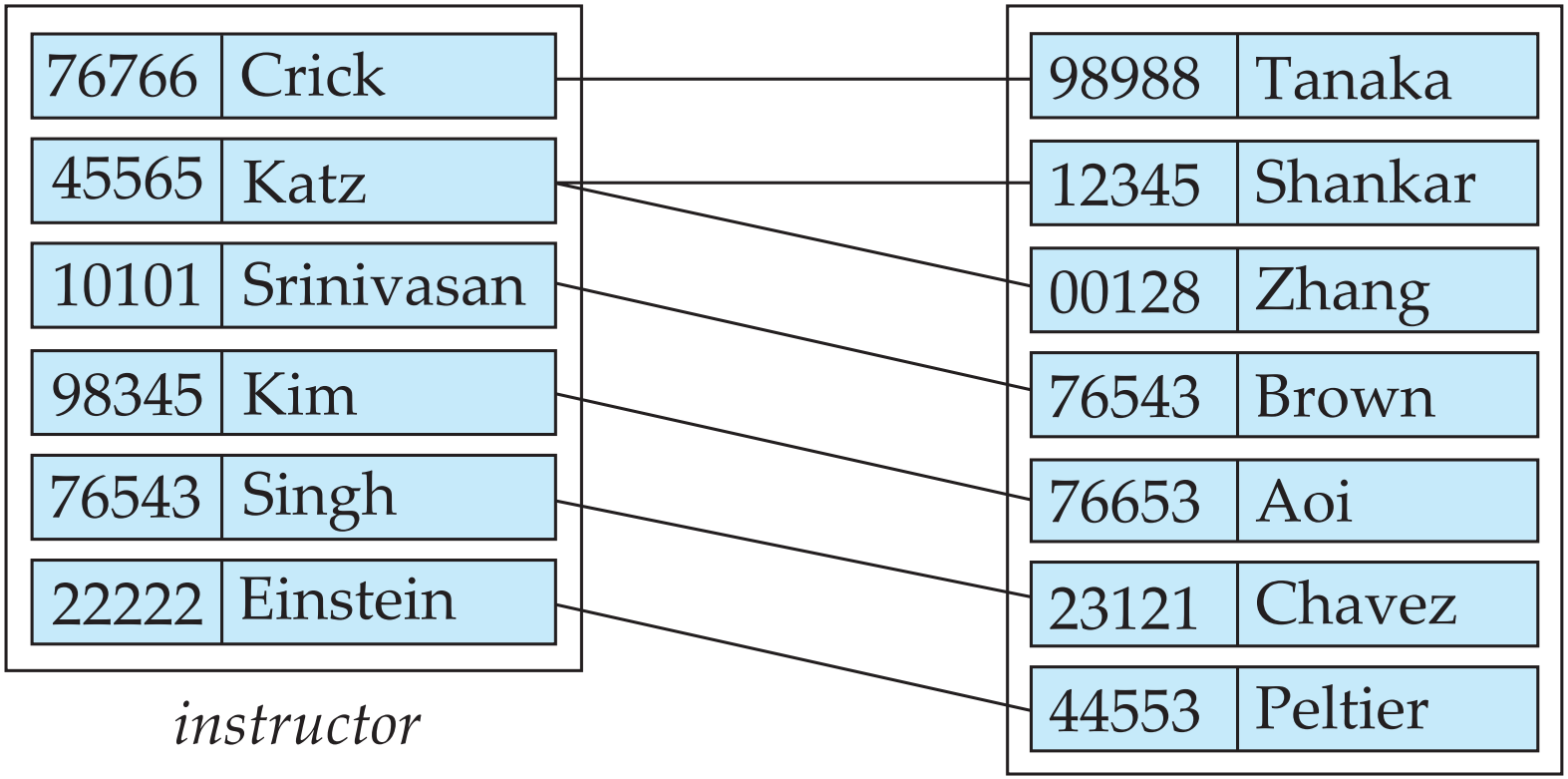
  where $(e_1, e_2, \ldots, e_n)$ is a relationship

  - Example:

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

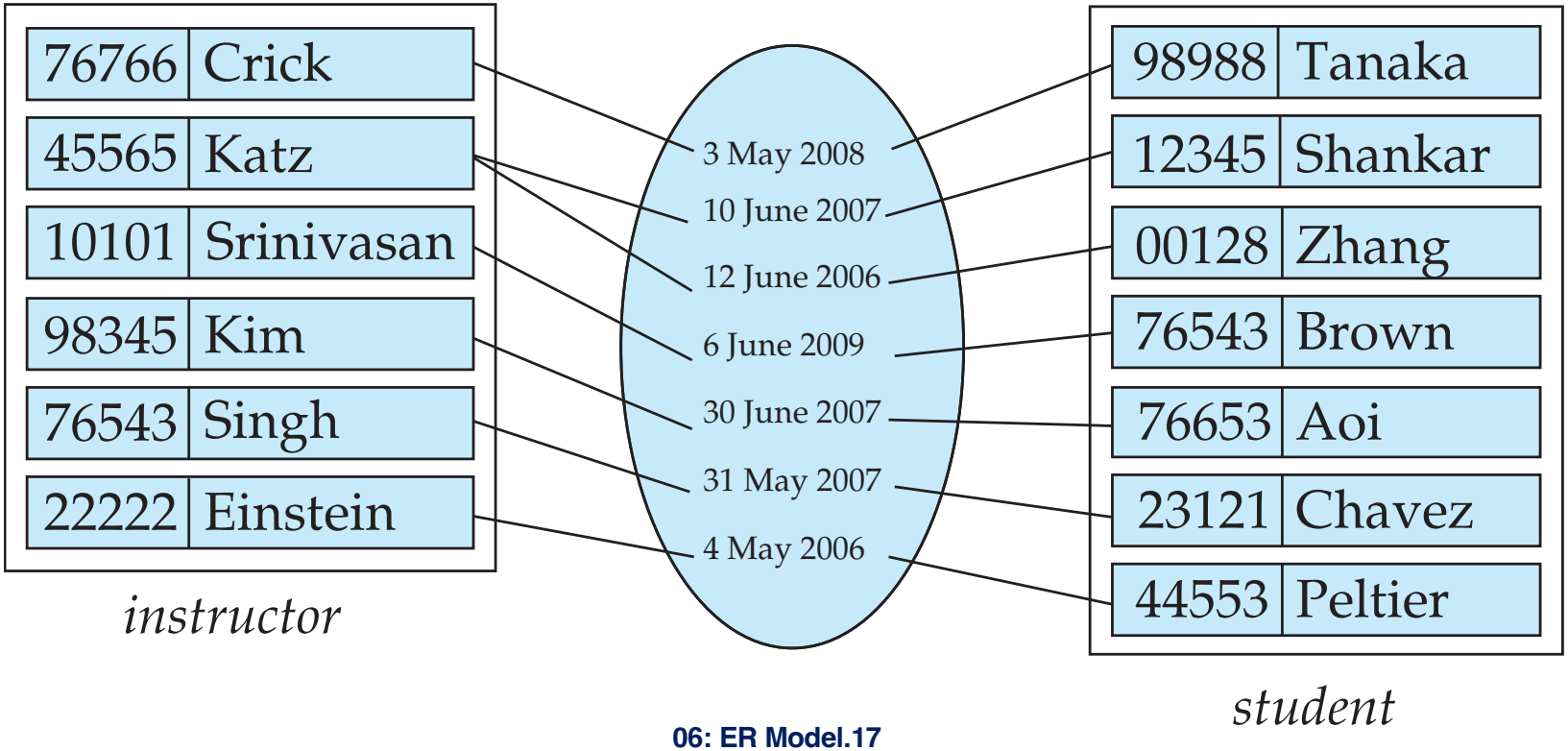
Relationship Set *advisor*

- Example: we define the relationship set *advisor* to denote the associations between students and the instructors who act as their advisors.
- Pictorially, we draw a line between related entities.



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



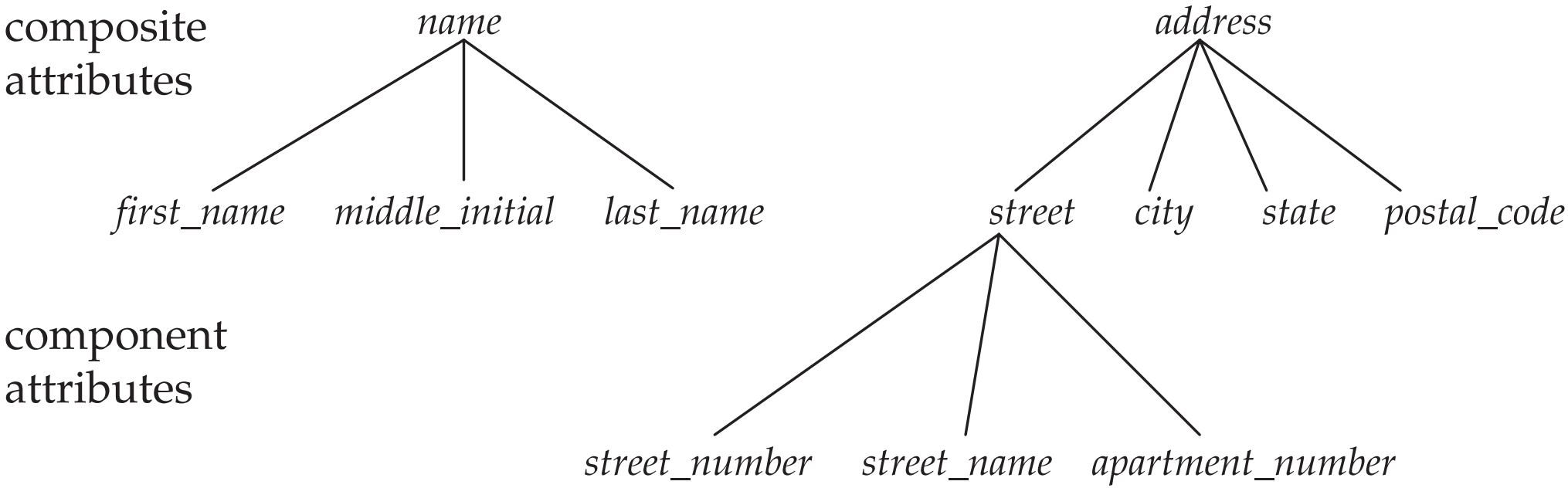
Degree of a Relationship Set

- **Binary relationship**
  - involve two entity sets (or degree two).
- 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

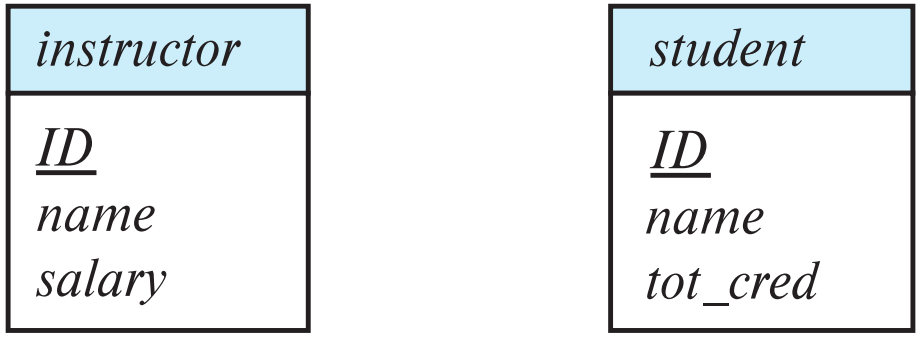
- An entity is represented by a set of attributes, that are descriptive properties possessed by all members of an entity set.
  - *instructor* = (*ID, name, street, city, salary*)
    course= (*course_id, title, credits*)
- **Domain** – the set of permitted values for each attribute
- Attribute types:
  - **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, number of students advising

Composite Attributes

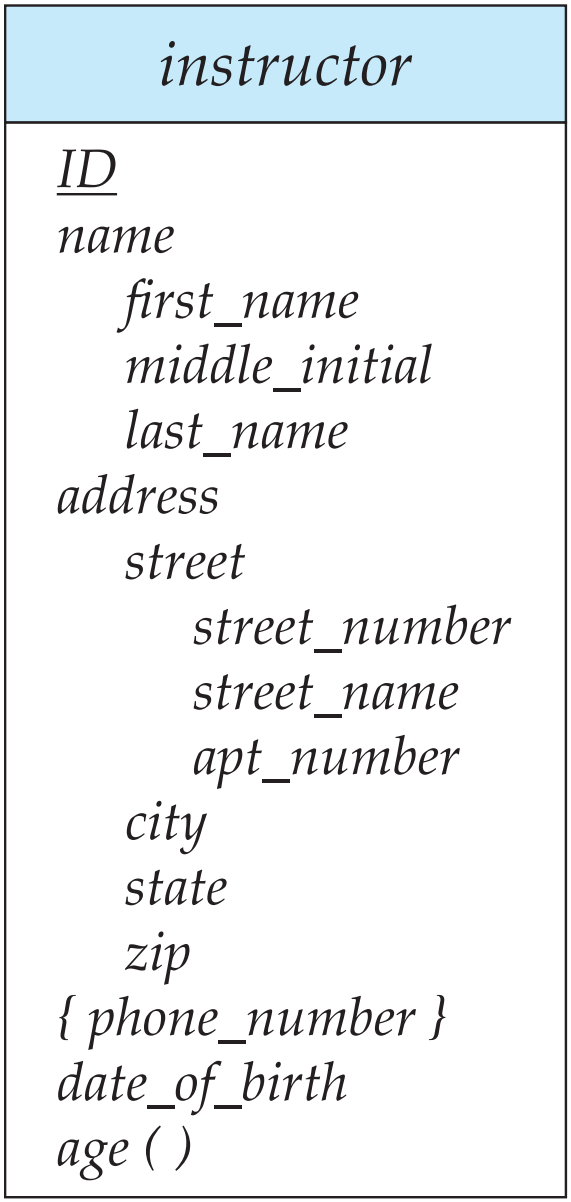


Representing Entity sets in ER Diagram

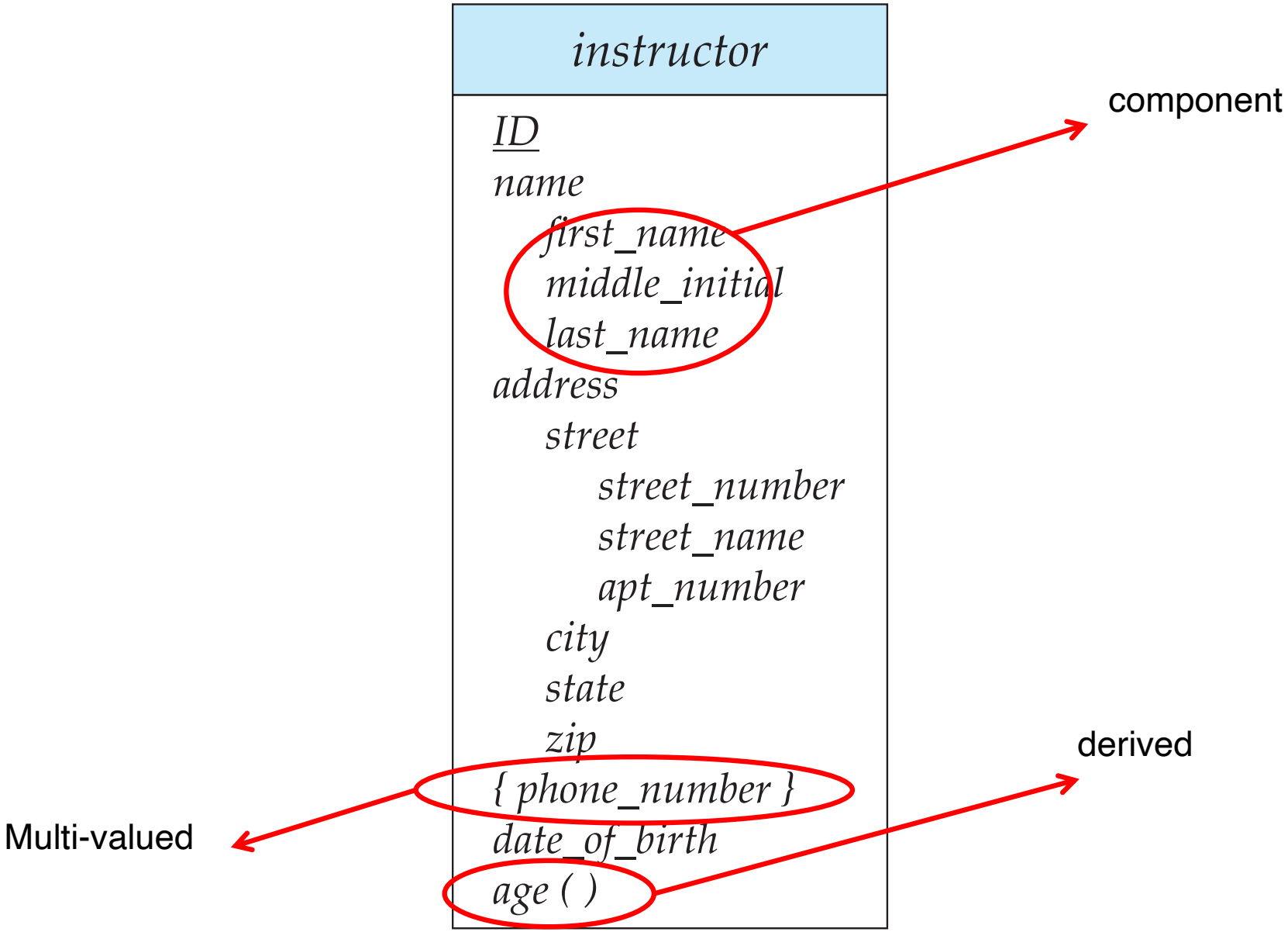
- Entity sets can be represented graphically as follows:
  - Rectangles represent entity sets.
  - Attributes listed inside entity rectangle
  - Underline indicates primary key attributes



Entity With Composite, Multivalued, and Derived Attributes



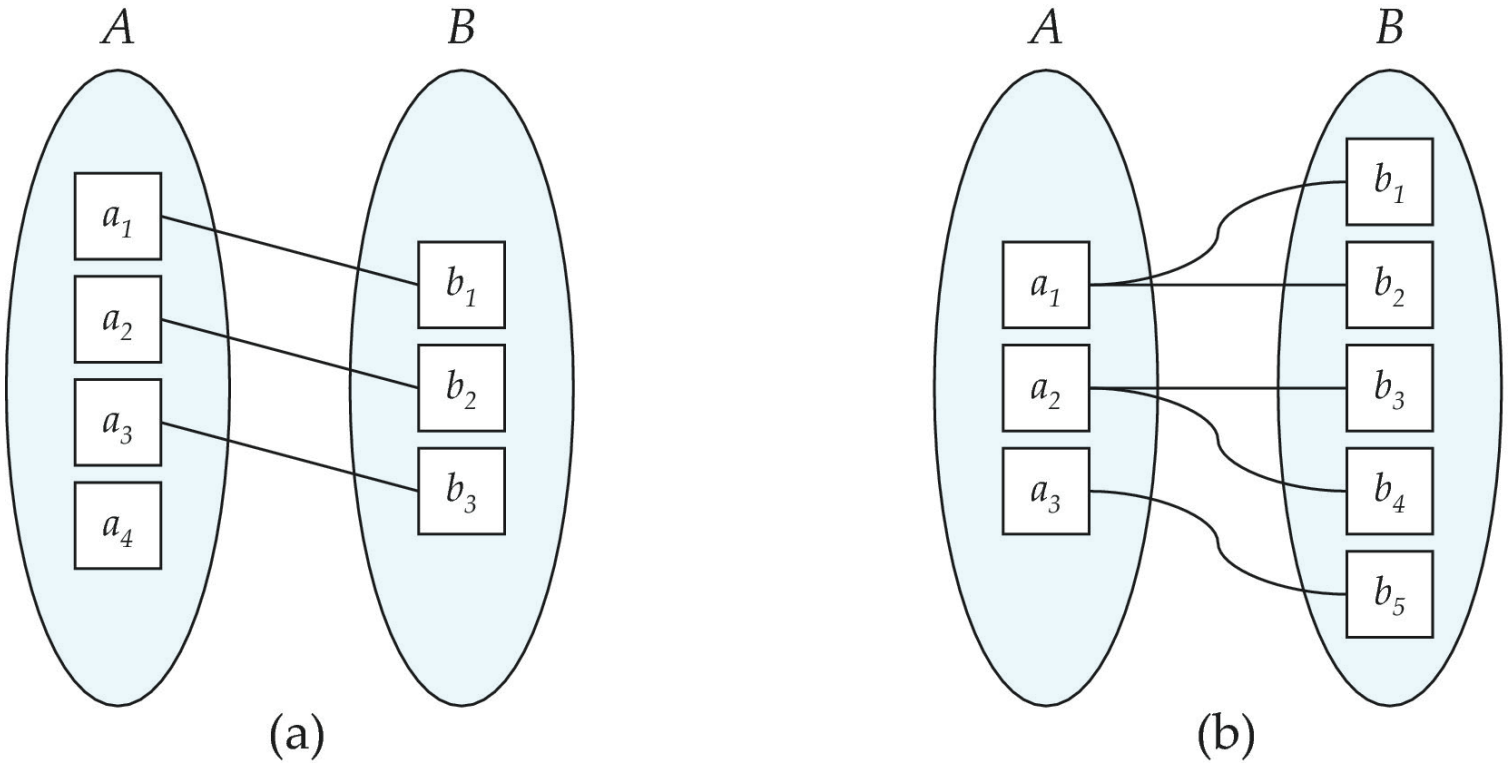
Entity With Composite, Multivalued, and Derived Attributes



Mapping Cardinality Constraints

- An E-R enterprise schema may define certain constraints to which the contents of a database must conform.
- Express the number of entities to which another entity can be associated via a relationship set.
- For a binary relationship set, the mapping cardinality must be one of the following types:
  - One to one (1-1)
  - One to many (1-N)
  - Many to one (N-1)
  - Many to many (N-M)

Mapping Cardinalities Example

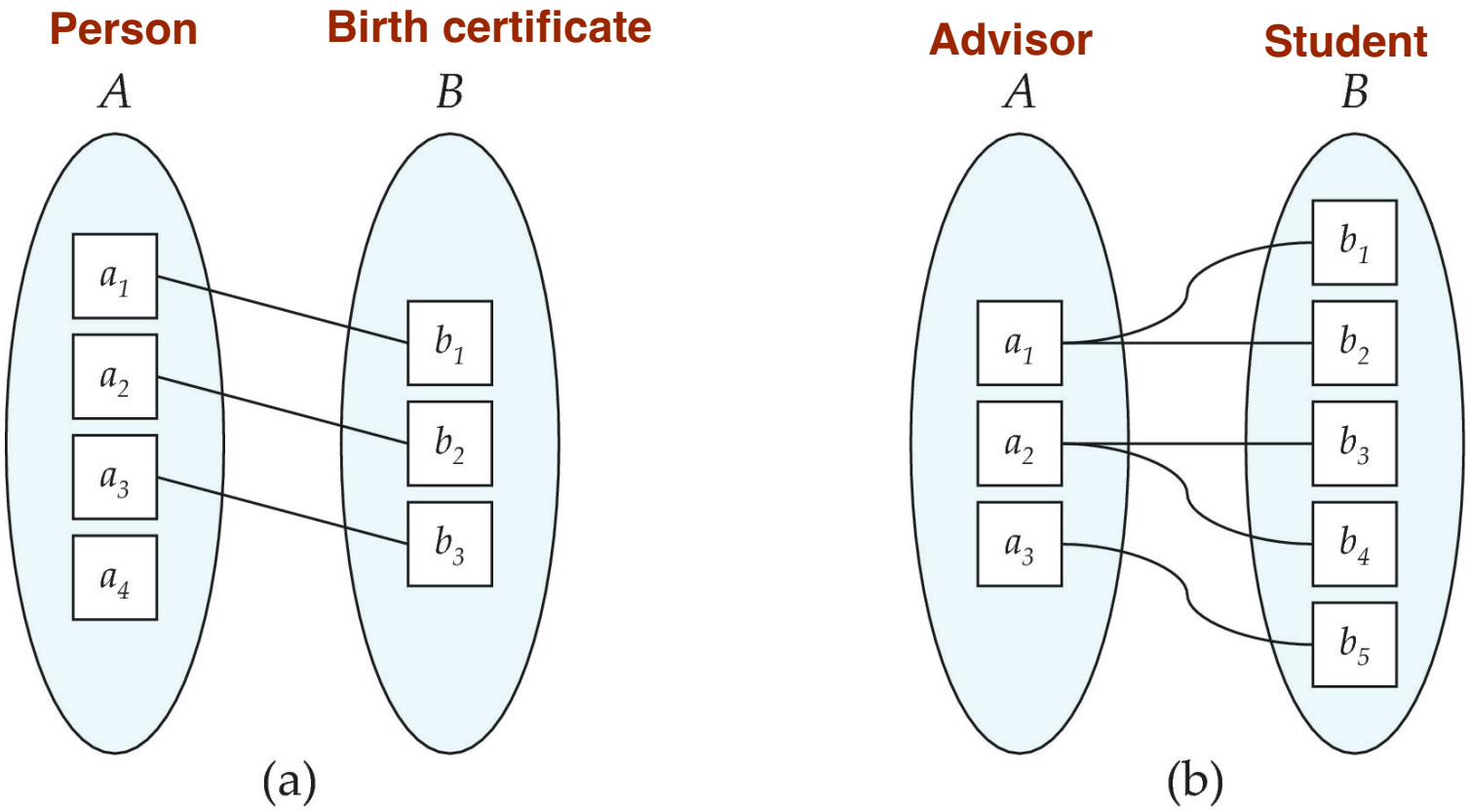


One to one

One to many

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

Mapping Cardinalities Example

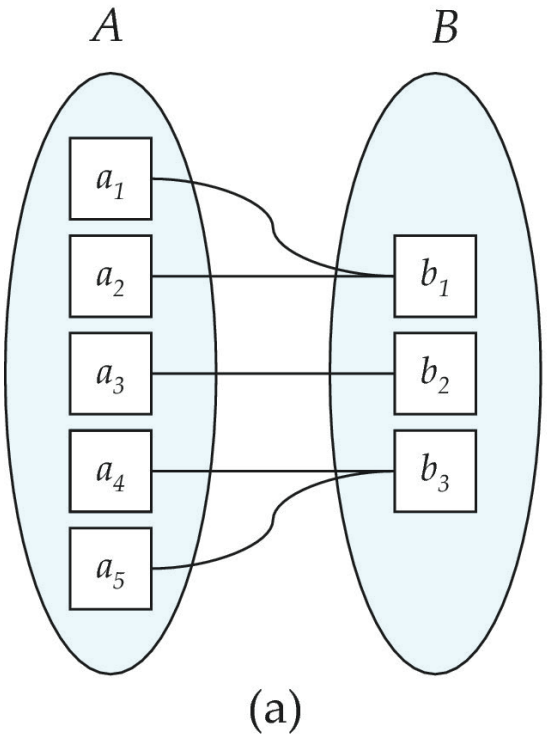


One to one

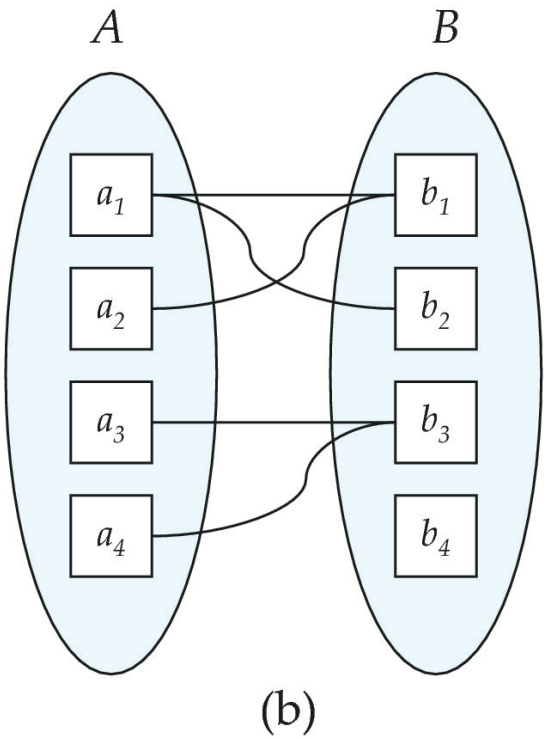
One to many

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

Mapping Cardinalities Example



Many to one

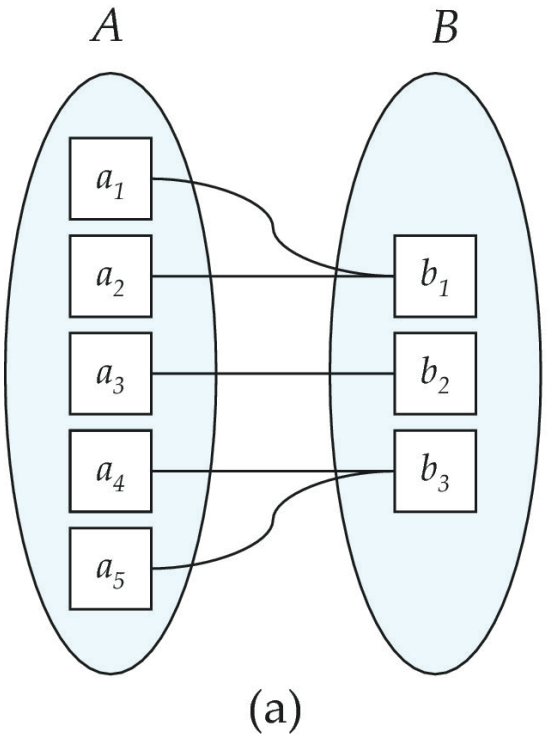


Many to many

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

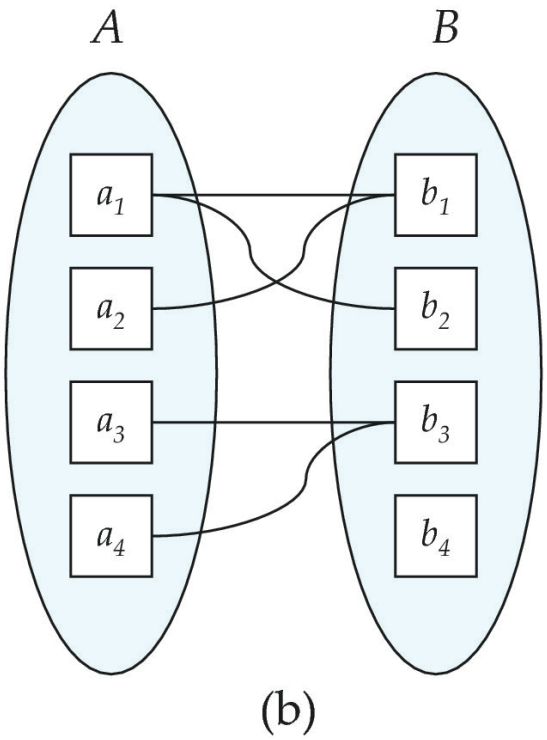
Mapping Cardinalities Example

Employee **Department**



Many to one

Student **Course**



Many to many

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

Mapping Cardinality Constraints Cont.

- What if we allow some elements to not be mapped to another element?
  - E.g., 0:1 – 1
- For a binary relationship set, the mapping cardinality must be one of the following types:
  - **1-1**
    - 1-1
    - 0:1-1
    - 1-0:1
    - 0:1-0:1
  - **1-N**
    - 0:1-N
    - 0:1-0:N
    - 1-N
    - 1-0:N
  - **N-1**
    - N-1
    - N-0:1
    - 0:N-1
    - 0:N-0:1
  - **N-M**
    - N-M
    - N-0:M
    - 0:N-M
    - 0:N-0:M

Keys

- We must have a way to specify how entities within a given entity set are distinguished.
- Individual entities are distinct; from a database perspective, however, the differences among them must be expressed in terms of their attributes.
- Therefore, the values of the attribute values of an entity must be such that they can uniquely identify the entity. In other words, no two entities in an entity set are allowed to have exactly the same value for all attributes.

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**.

Note: Basically the same as for the relational model

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 entities 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 or model meeting as a separate entity
- Must consider the mapping cardinality of the relationship set when deciding what are the candidate keys
- Need to consider the semantics of the relationship set in selecting the *primary key* in case of more than one candidate key

Keys for Relationship Sets Cont.

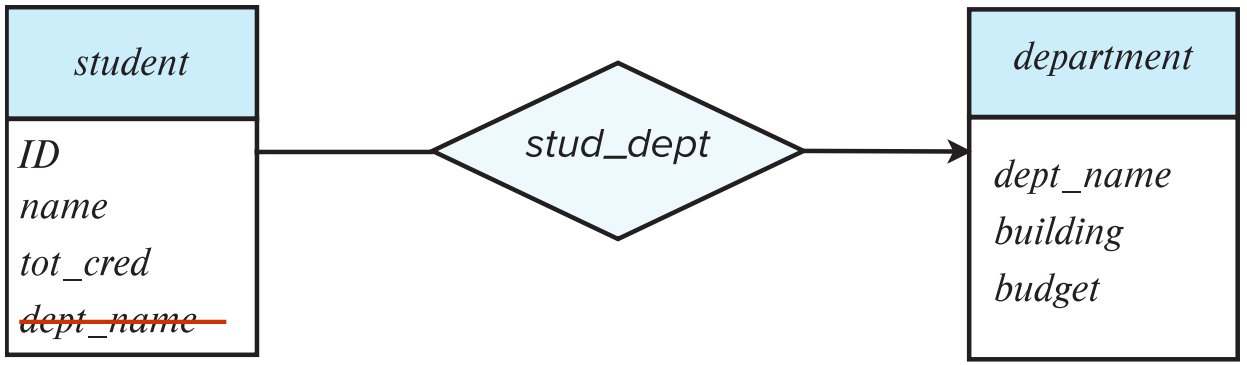
- The choice of the primary key for a relationship set depends on the mapping cardinality of the relationship set.
  - 1-1: both primary keys are candidate keys
    - Example: **hasBc**: (Person-Birthcertificate)
  - N-1: the N side is the candidate key
    - Example: **worksFor**: (Instructor-Department)
  - N-M: the combination of both primary keys
    - Example: **takes**: (Student-Course)

Redundant Attributes

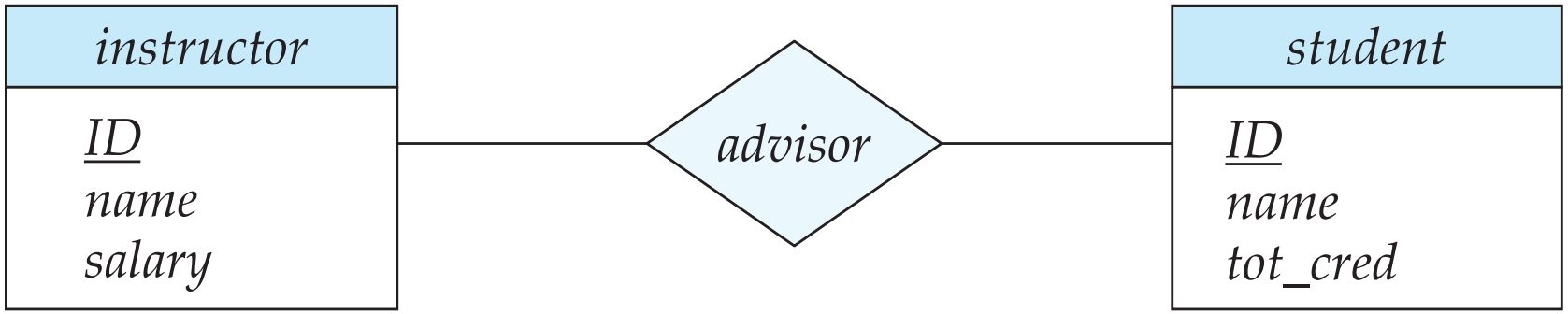
- When we design a database using the E-R model, we usually start by **identifying those entity sets** that should be included.
- Once the entity sets are decided upon, we must choose the **appropriate attributes**.
- These attributes are supposed to represent the various values we want to capture in the database.
- Once the entities and their corresponding attributes are chosen, the **relationship sets** among the various entities are formed.
- "These relationship sets may result in a situation where attributes in the various entity sets are redundant and need to be removed from the original entity sets".

Redundant Attributes

- Suppose we have entity sets:
  - *student*, with attributes: *ID*, *name, tot_cred*, *dept_name*
  - *department,* with attributes: *dept_name, building, budget*
- We model the fact that each student has an associated department using a relationship set *stud_dept*
- The attribute *dept_name* in *student* below replicates information present in the relationship and is therefore **redundant**
- BUT: when converting back to tables, in some cases the attribute gets reintroduced, as we will see later.

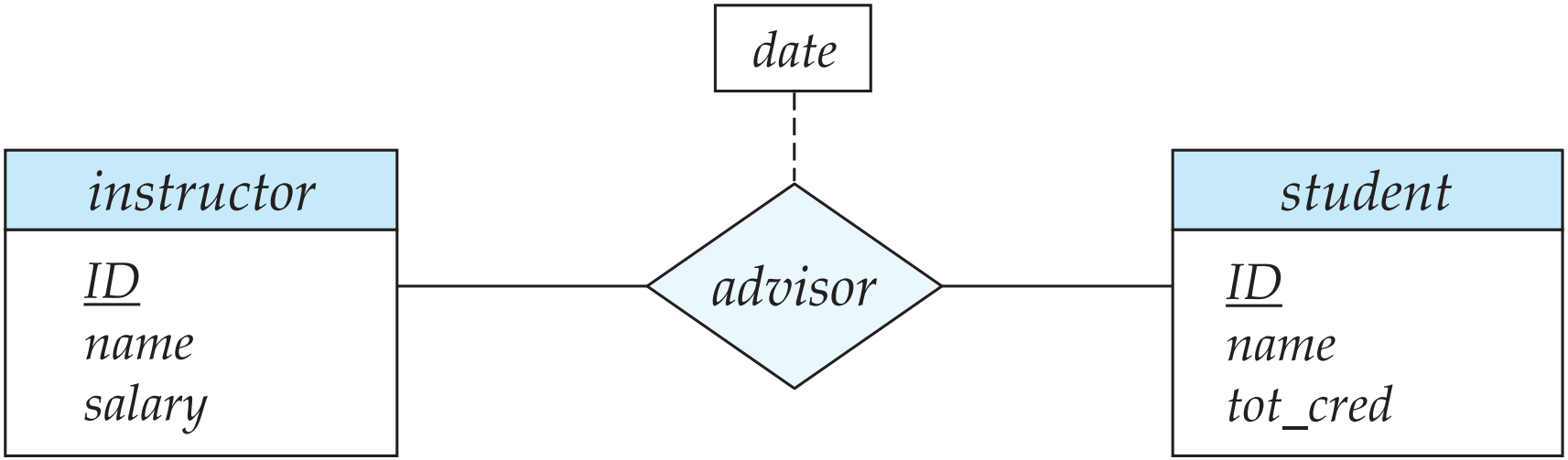


E-R Diagrams



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

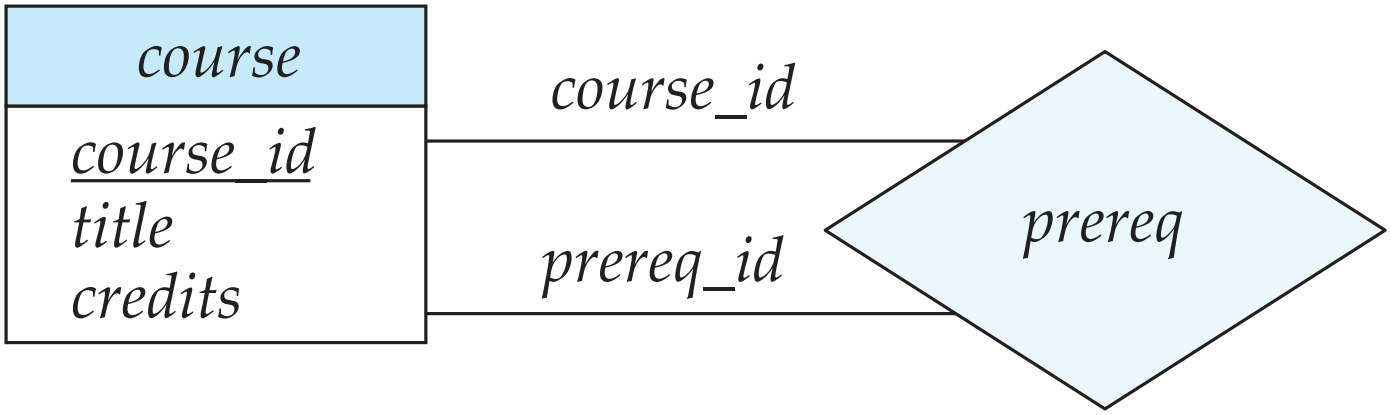
Relationship Sets with Attributes



- If a relationship set has some attributes associated with it, then we enclose the attributes in a rectangle and link the rectangle with **a dashed line** to the diamond representing that relationship set.
- For example, we have the *date* **descriptive attribute** attached to the relationship set advisor to specify the date on which an instructor became the advisor.

Roles

- When entity sets of a relationship are not distinct
  - Each occurrence of an entity set plays a "role" in the relationship
  - The same entity set participates in a relationship set more than once in different roles
- The labels "*course_id*" and "*prereq_id*" are called **roles**.



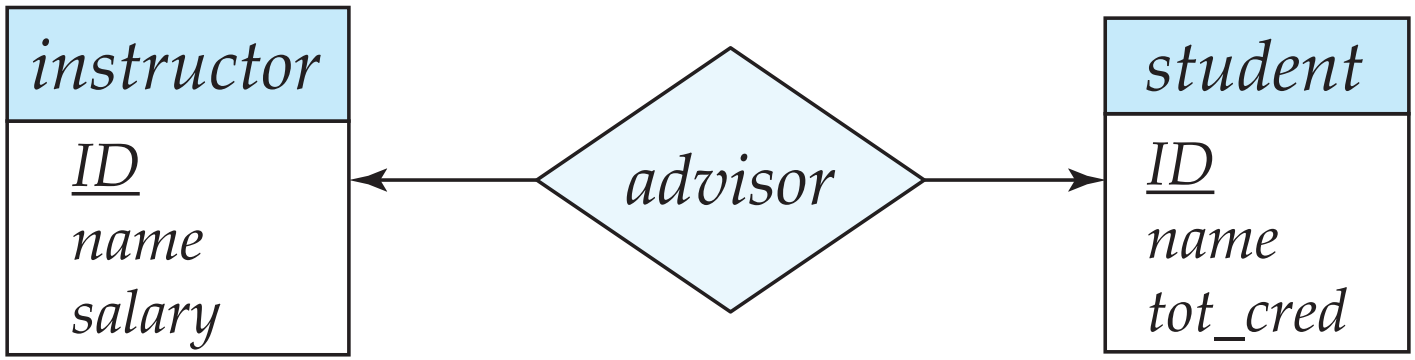
Cardinality Constraints

- 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

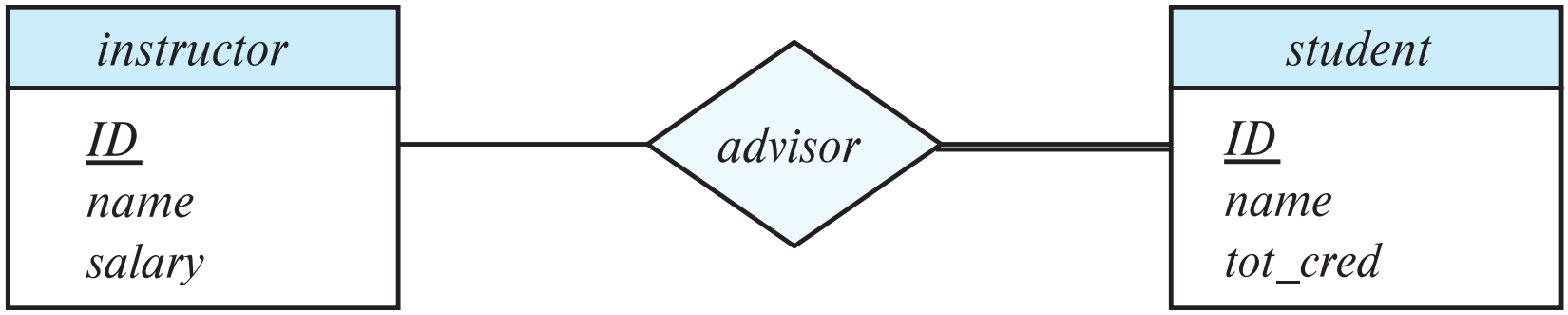
- In a **many-to-one** relationship between an *instructor* and a *student,*
  - an instructor is associated with at most one student via *advisor*,
  - and a student is associated with several (including 0) instructors via *advisor*

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*

Total and Partial Participation

- **Total participation** (indicated by double line): every entity in the entity set participates in at least one relationship in the relationship set



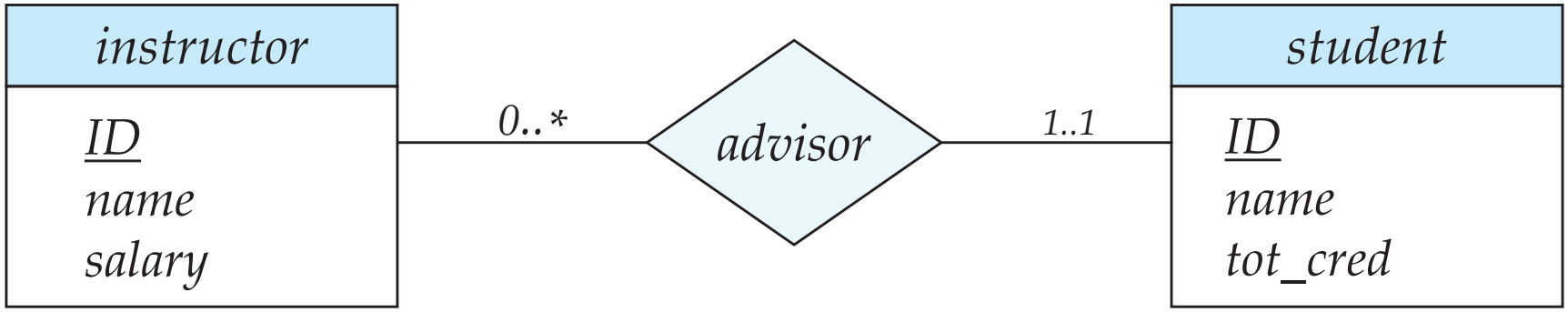
participation of *student* in *advisor* relation is total

  - 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.

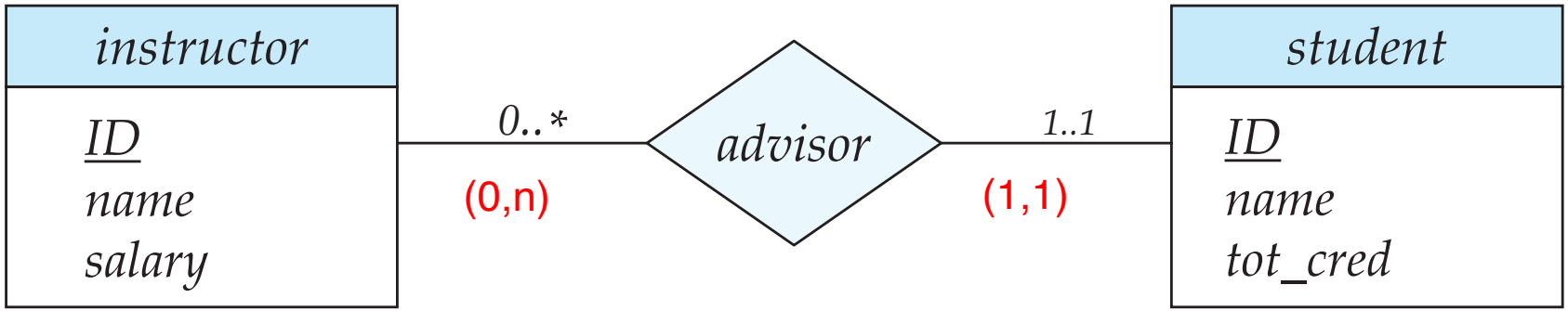
Alternative Notation for Cardinality Limits

- 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



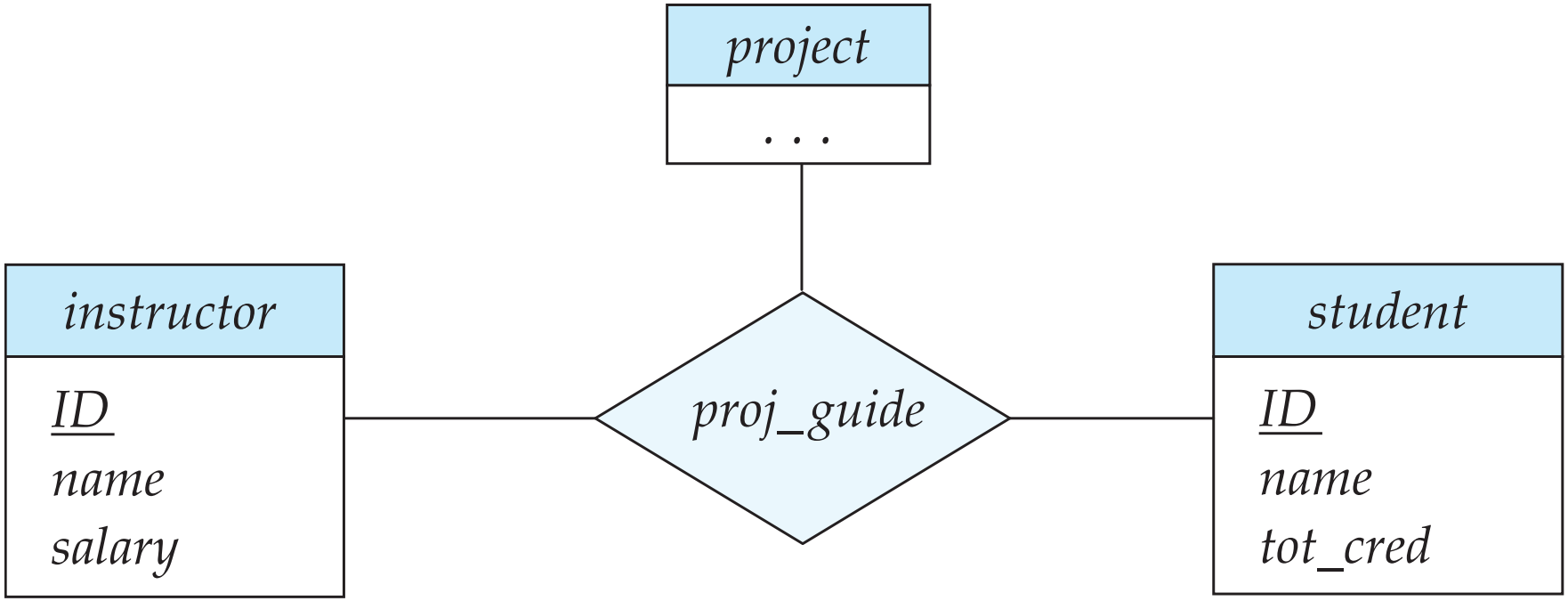
Alternative Notation for Cardinality Limits

- Alternative notation



E-R Diagram with a Ternary Relationship

- We can specify some types of many-to-one relationships in the case of non-binary relationship sets.
- Nonbinary relationship sets can be specified easily in an E-R diagram.



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
- Better to use cardinality constraints such as (0,n)

Weak Entity Sets

- An entity set that does not have a primary key is referred to as a **weak entity set**.
- An entity set that is not a weak entity set is termed a **strong entity set**.
- 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.)

- The existence of a weak entity set depends on the existence of a **identifying entity set**
  - Every weak entity must be associated with an identifying entity; that is, the weak entity set is said to be **existence dependent** on the identifying entity set.
  - The identifying entity set is said to **own** the weak entity set that it identifies.
- The relationship associating the weak entity set with the identifying entity set is called the **identifying relationship**.
  - Identifying relationship depicted using a double diamond
  - It must relate to the identifying entity set via a total, one-to-many relationship set from the identifying to the weak entity set.
  - The identifying relationship set should not have any descriptive attributes.

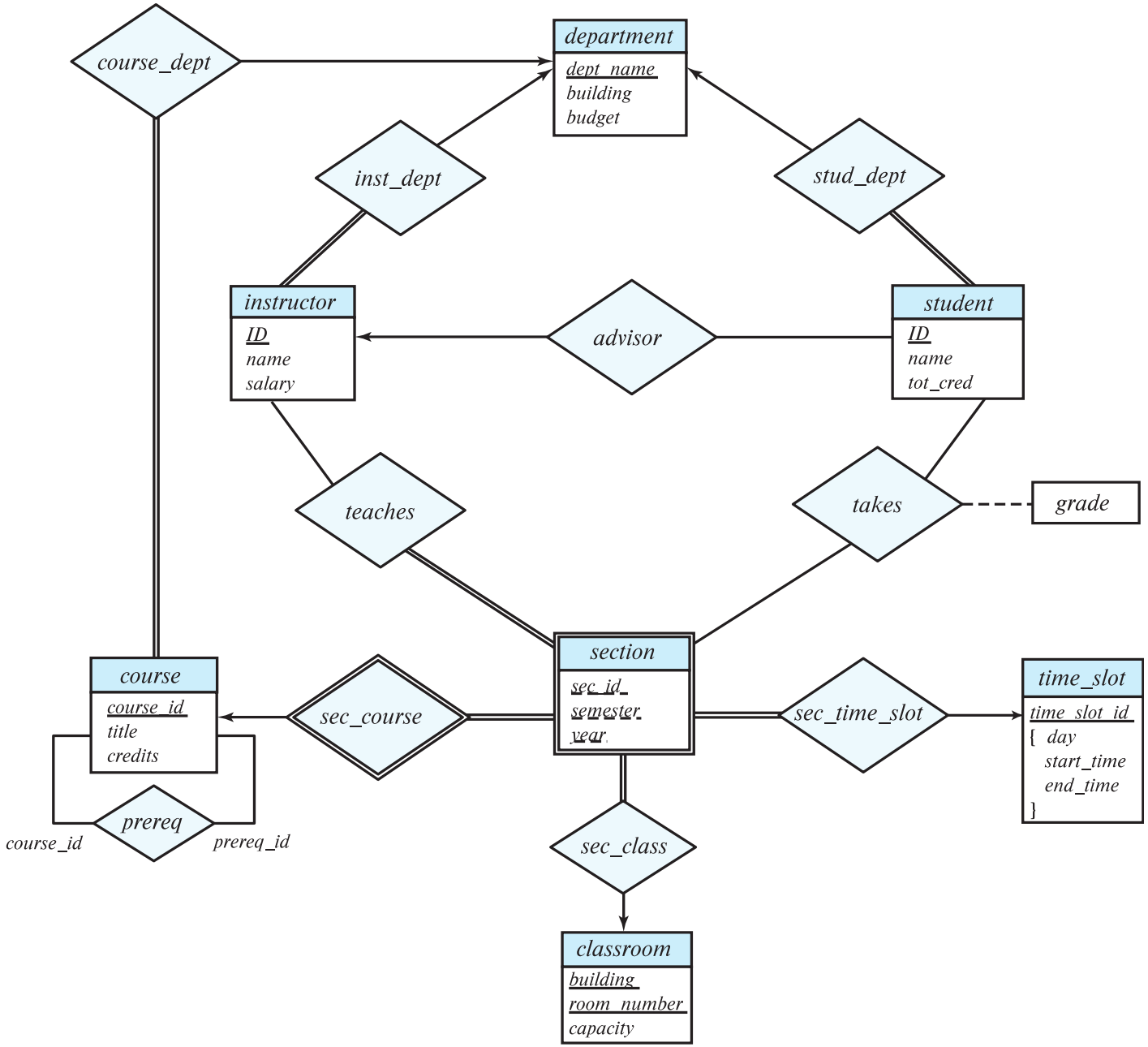
Expressing Weak Entity Sets

- In E-R diagrams, a weak entity set is depicted via a double rectangle.
- We underline the discriminator of a weak entity set with a dashed line.
- The relationship set connecting the weak entity set to the identifying strong entity set is depicted by a double diamond.
- Primary key for *section* – (*course_id, sec_id, semester, year*)

Weak Entity Sets

- Consider a *section* entity, which is uniquely identified by a *course_id*, *semester, year*, and *sec_id*.
- Clearly, section entities are related to course entities. Suppose we create a relationship set *sec_course* between entity sets *section* and *course*.
- Note that the information in *sec_course* is redundant since *section* already has an attribute *course_id*, which identifies the course with which the section is related.
- One option to deal with this redundancy is to **get rid of the relationship *sec_course***; however, by doing so the relationship between *section* and *course* becomes implicit in an attribute, which is not desirable.

E-R Diagram for a University Enterprise



Reduction to Relation 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

- A strong entity set reduces to a schema with the same attributes

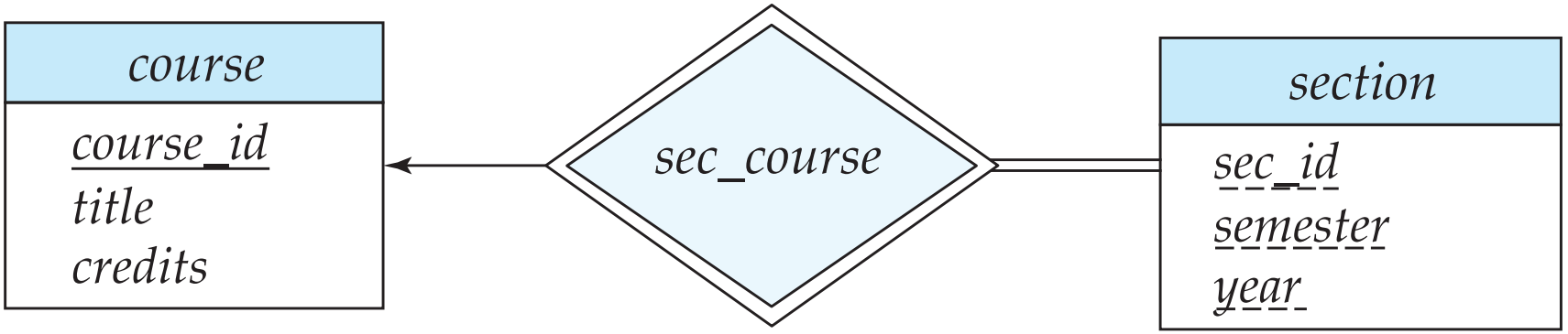
  student(ID, name, tot_cred)

  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)

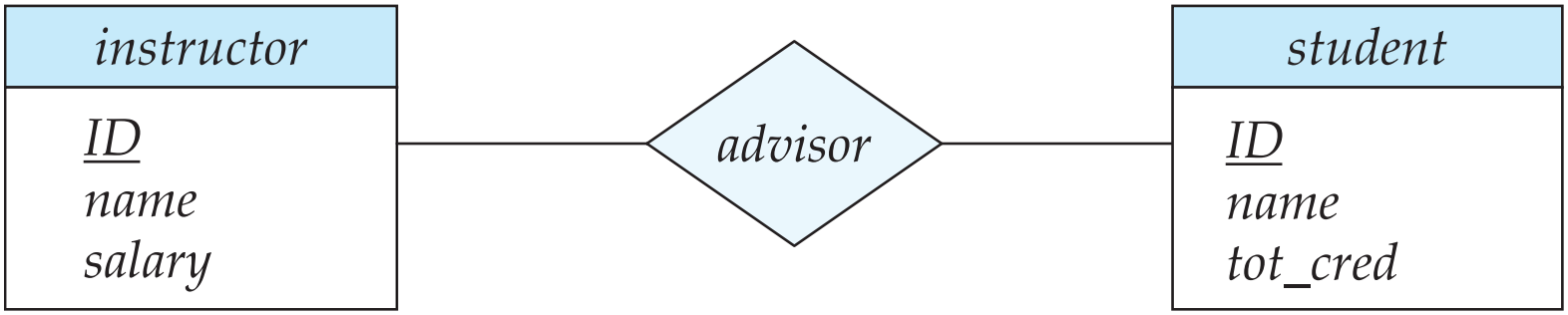
- Example



Representing Relationship Sets

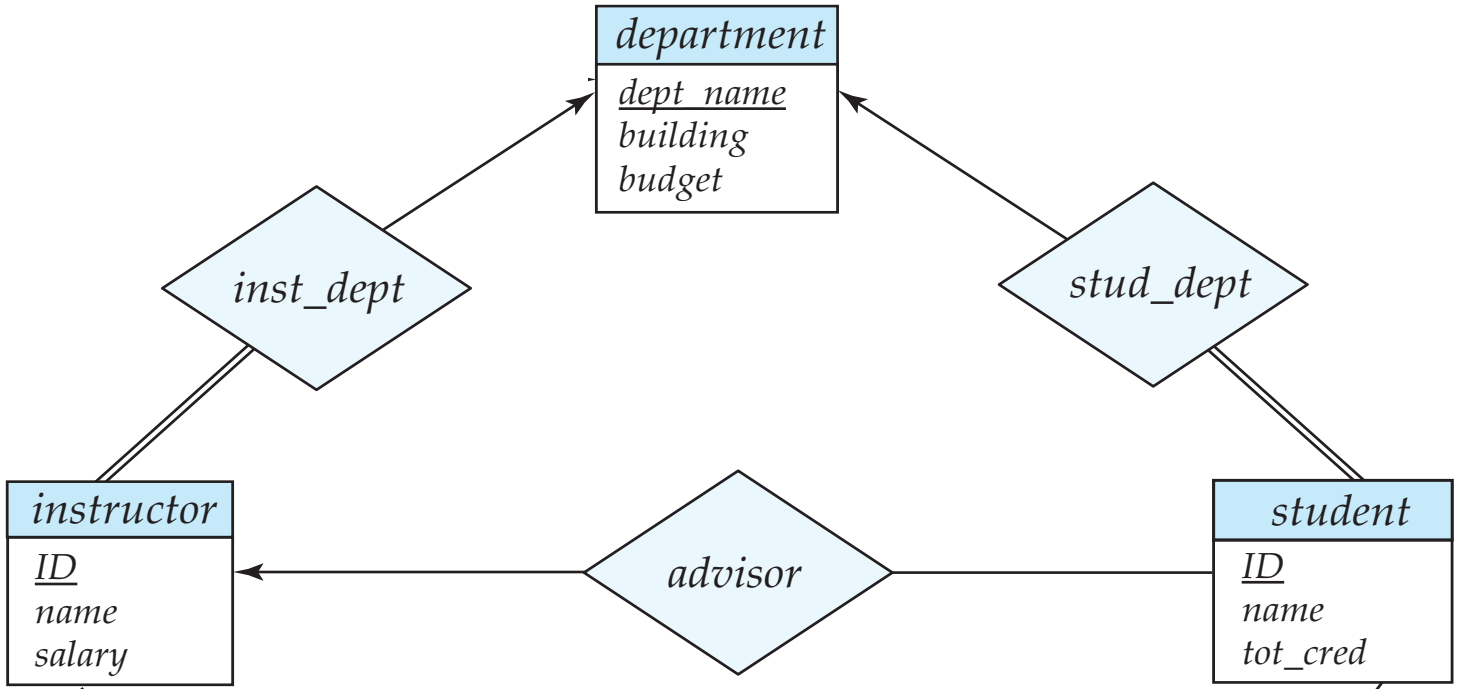
- A **many-to-many relationship set is represented as a schema** with attributes for the primary keys of the two participating entity sets, and any descriptive attributes of the relationship set.
- Example: schema for relationship set *advisor*

  advisor = (*s_id, i_id*)



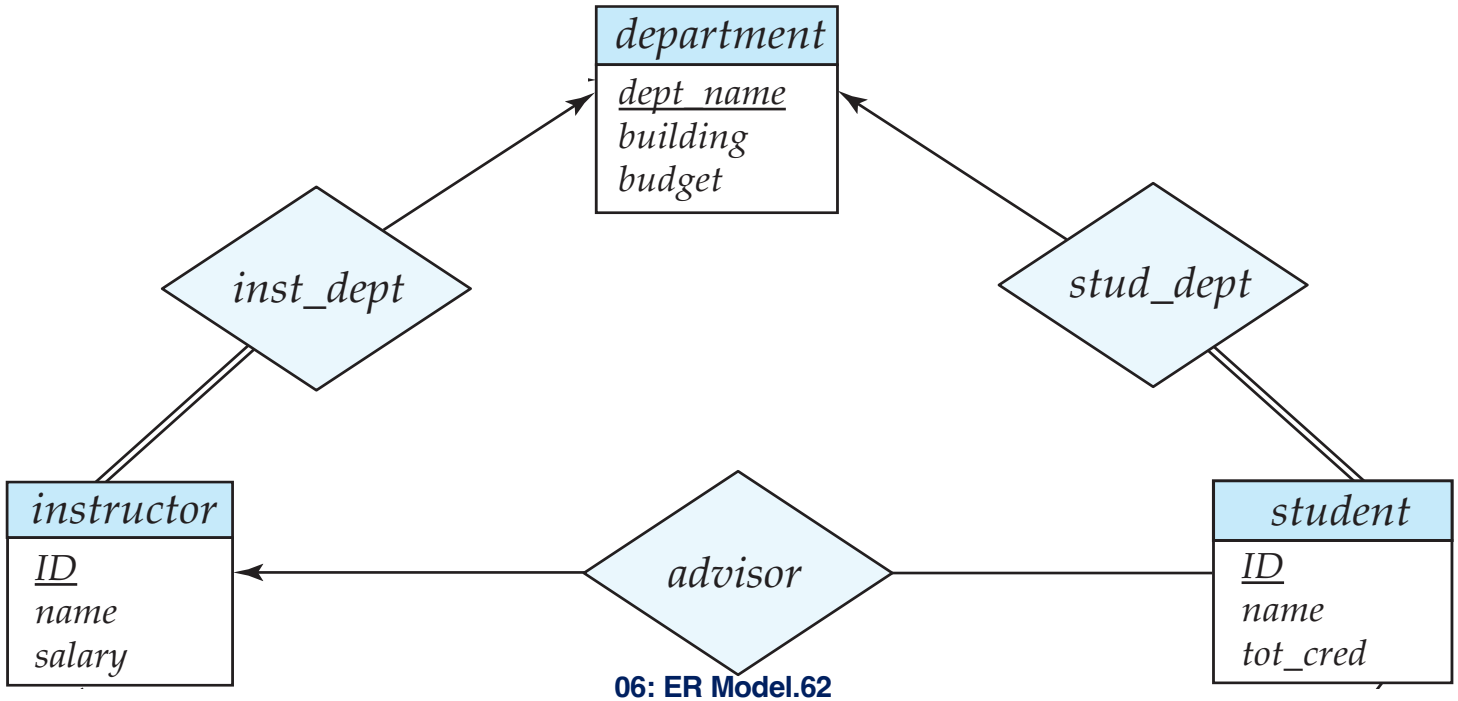
Redundancy of Schemas

- 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*
- Example



Redundancy of Schemas (Cont.)

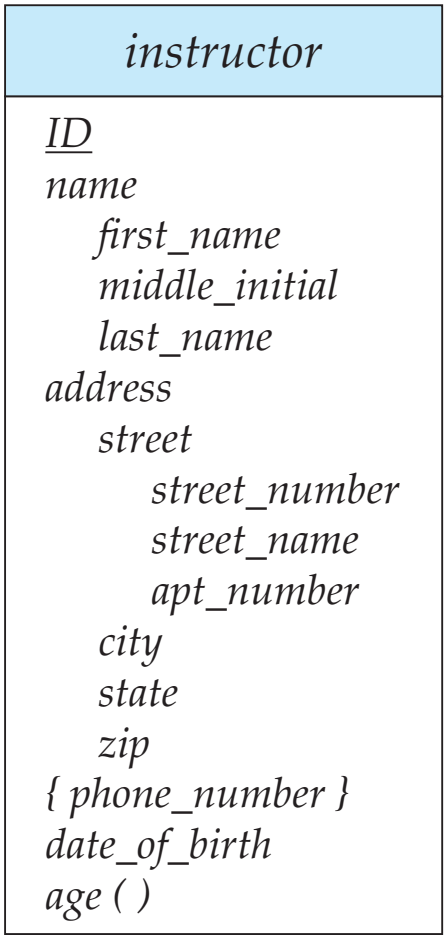
- If participation is *partial* on the "many" side, replacing a schema by an extra attribute in the schema corresponding to the "many" side could result in null values
- For one-to-one relationship sets, either side can be chosen to act as the "many" side
  - That is, an extra attribute can be added to either of the tables corresponding to the two entity sets



Redundancy of Schemas (Cont.)

- 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

Representation of Entity Sets with Composite Attributes



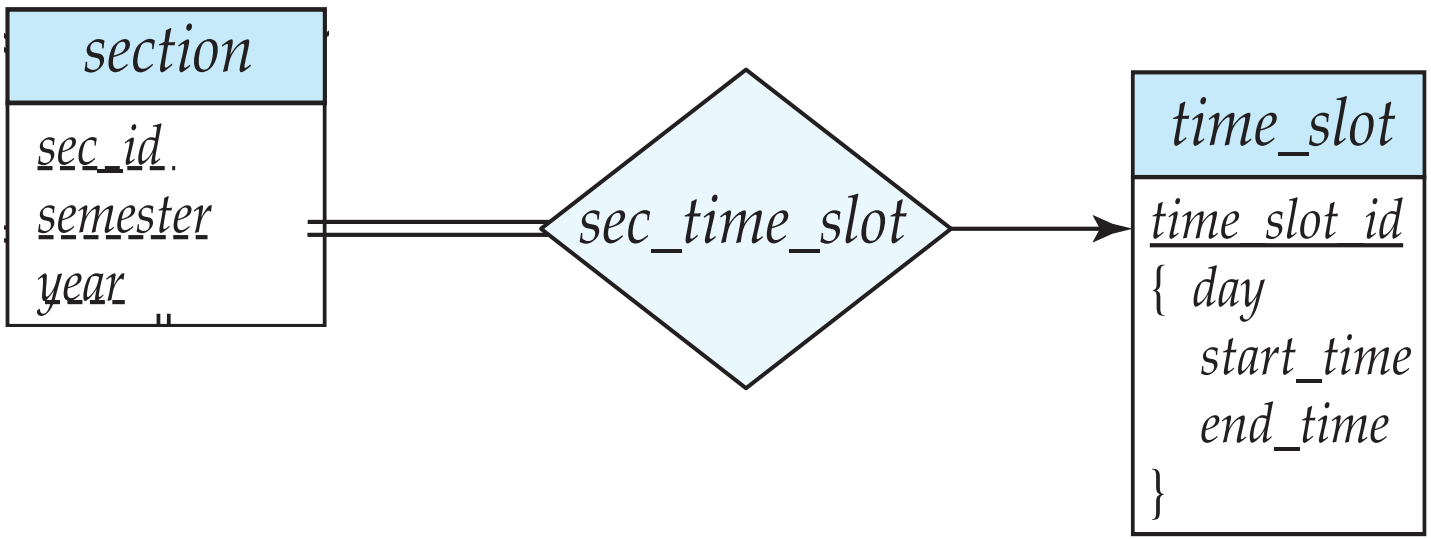
- Composite attributes are ***flattened out*** by creating a separate attribute for each component attribute
  - Example: given entity set *instructor* with composite attribute *name* with component attributes *first_name* and *last_name* the schema corresponding to the entity set has two attributes *name_first_name* and *name_last_name*
    - Prefix omitted if there is no ambiguity (*name_first_name* could be *first_name)*
- 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)

Representation of Entity Sets with 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)

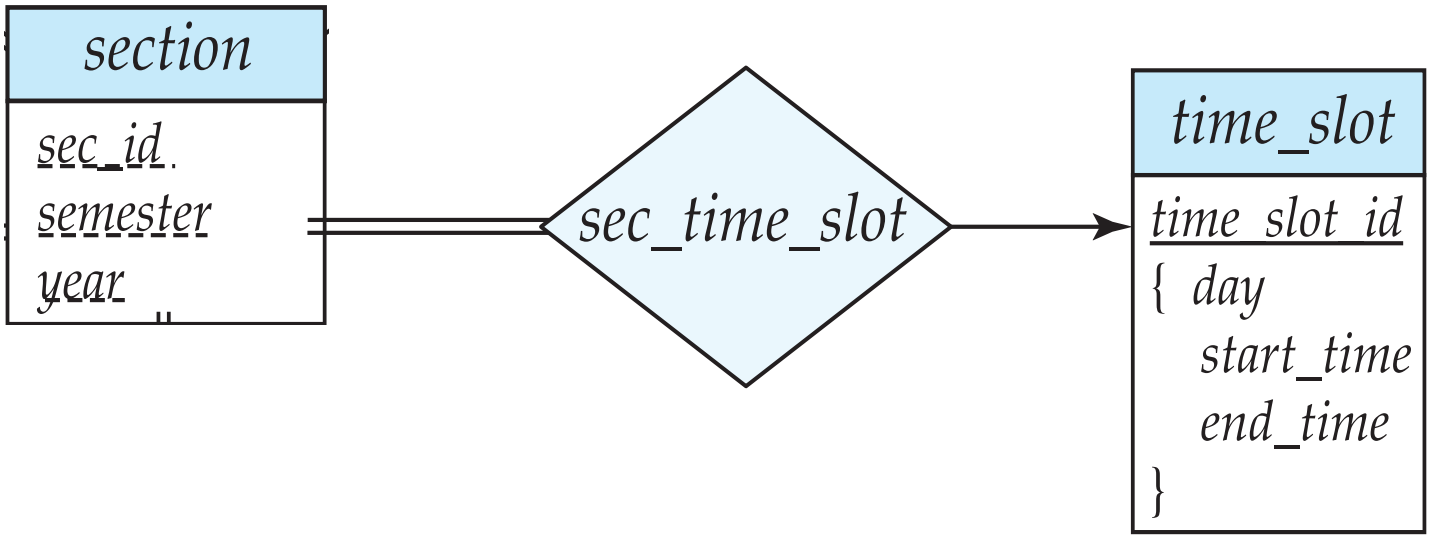
Multivalued Attributes (Cont.)

- **Special case**: entity *time_slot* has only one attribute other than the primary-key attribute, and that attribute is multivalued



Multivalued Attributes (Cont.)

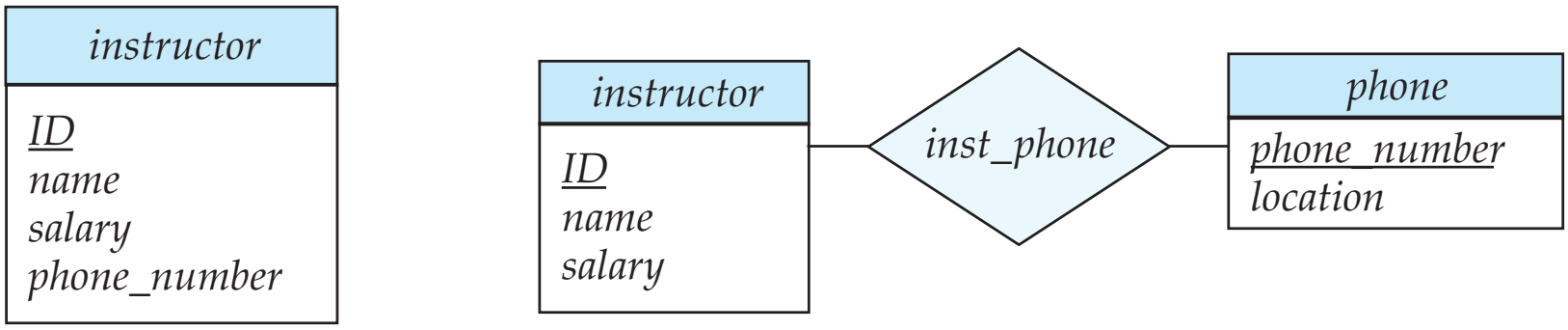
- **Special case**: entity *time_slot* has only one attribute other than the primary-key attribute, and that attribute is multivalued
  - Optimization: Don't create the relation corresponding to the entity, just create the one corresponding to the multivalued attribute
  - *time_slot*(*time_slot_id, 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

Design Issues

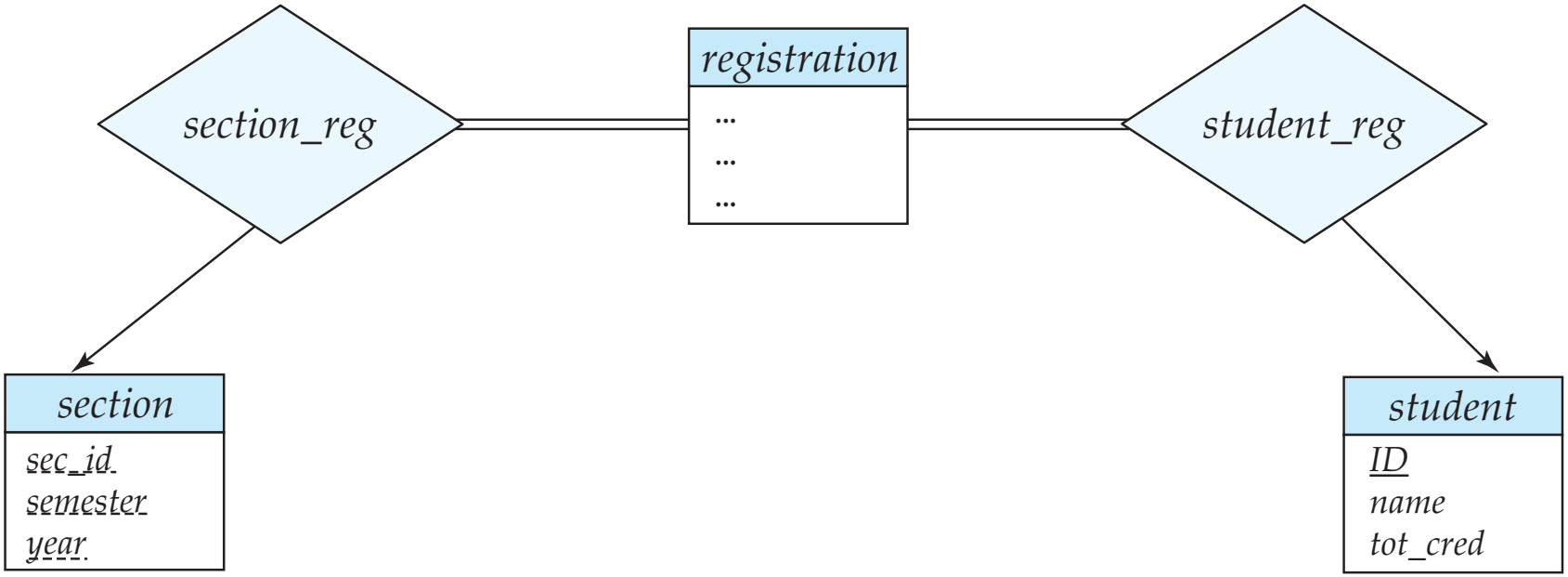
- **Use of entity sets vs. attributes**



- Designing phone as an entity allow for primary key constraints for phone
- Designing phone as an entity allow phone numbers to be used in relationships with other entities (e.g., student)
- Use of phone as an entity allows extra information about phone numbers

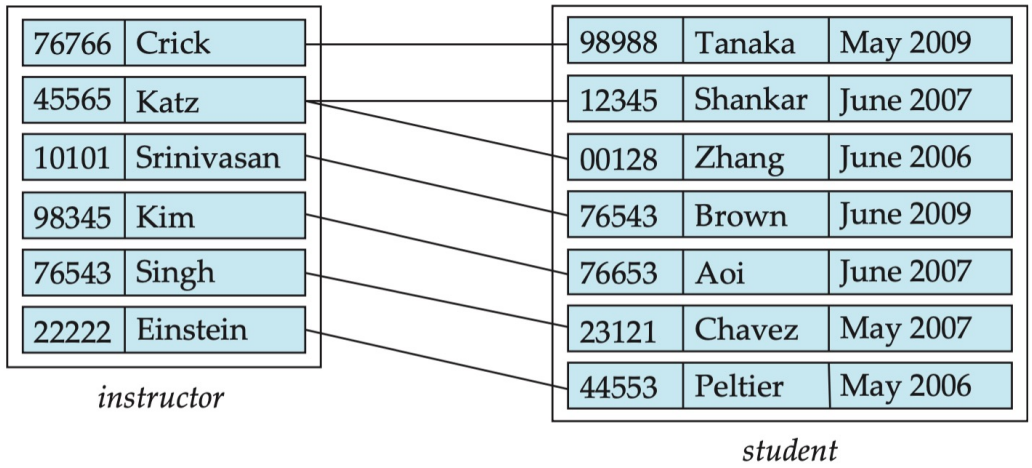
Design Issues

- **Use of entity sets vs. relationship sets**
  - Possible guideline is to designate a relationship set to describe an action that occurs between entities
  - Possible hint: the relationship only relates entities, but does not have an existence by itself. E.g., hasAddress: (department-address)



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 + an artificial entity set, 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*
  - Does not work for *N-M relationships!*



Binary Vs. Non-Binary Relationships

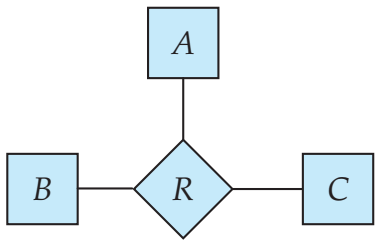
- Some relationships that appear to be non-binary may be better represented using binary relationships
  - E.g., A ternary relationship *parents*, relating a child to his/her father and mother, is best replaced by two binary relationships, *father* and *mother*
    - Using two binary relationships allows partial information (e.g., only mother being know)
  - But there are some relationships that are naturally non-binary, e.g., *proj_guide*
    - *Instructor Katz works on projects A and B with students Shankar and Zhang*
    - *we would not be able to record that Katz works on project A with student Shankar and works on project B with student Zhang*

Converting Non-Binary Relationships to Binary Form

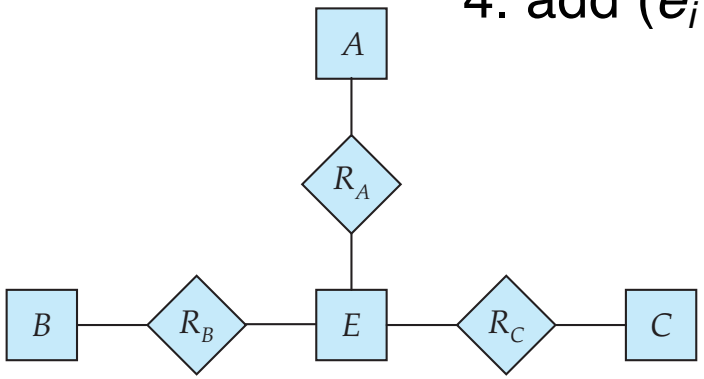
- In general, any non-binary relationship can be represented using binary relationships by creating an artificial entity set.
  - Replace *R* between entity sets A, B and C by an entity set *E*, and three relationship sets:

    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*
  - Create a new entity e_i in 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



(a)



(b)

Converting Non-Binary Relationships: Is the New Entity Set E Necessary?

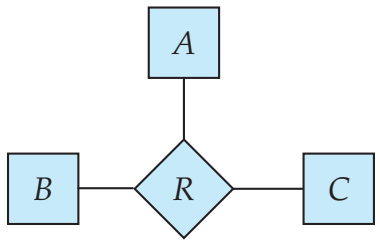
- Yes, because a non-binary relationship stores more information than any number of binary relationships

- *Consider R = order, A = supplier, B = item, C = customer*

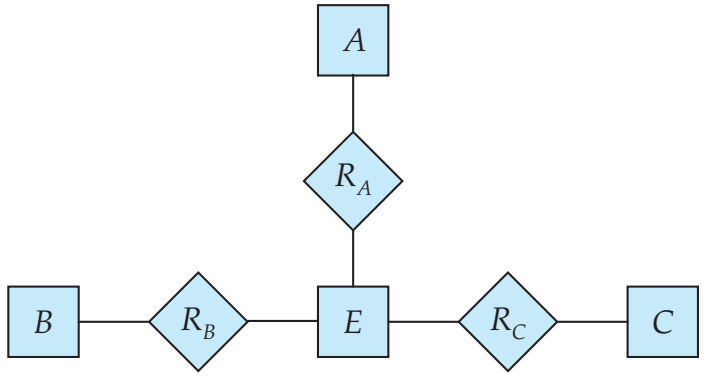
(Gunnar, chainsaw, Bob) – Bob ordered a chainsaw from Gunnar

→ *(Gunnar, chainsaw), (chainsaw, Bob), (Gunnar, Bob)*

Gunnar supplies chainsaws, Bob ordered a chainsaw, Bob ordered something from Gunnar. E.g., we do not know what Bob ordered from Gunnar.



(a)



(b)

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*
    - *Consider a constraint that says that R is many-to-one from A, B to C; that is, each pair of entities from A and B is associated with at most one C entity. This constraint cannot be expressed by using cardinality constraints on the relationship sets* R_A, R_B, *and* R_C.

ER-model to Relational Summary

- ***Rule 1)*** *Strong entity E*
  - *Create relation with attributes of E*
  - *Primary key is equal to the PK of E*
- ***Rule 2)*** *Weak entity W identified by E through relationship R*
  - *Create relation with attributes of W and R and PK(E).*
  - *Set PK to discriminator attributes combined with PK(E). PK(E) is a foreign key to E.*
- ***Rule 3)*** *Binary relationship R between A and B: one-to-one*
  - *If no side is total add PK of A to as foreign key in B or the other way around. Add any attributes of the relationship R to A respective B.*
  - *If one side is total add PK of the other-side as foreign key. Add any attributes of the relationship R to the total side.*
  - *If both sides are total merge the two relations into a new relation E and choose either PK(A) or PK(B) as the new PK. Add any attributes of the relationship R to the new relation E.*

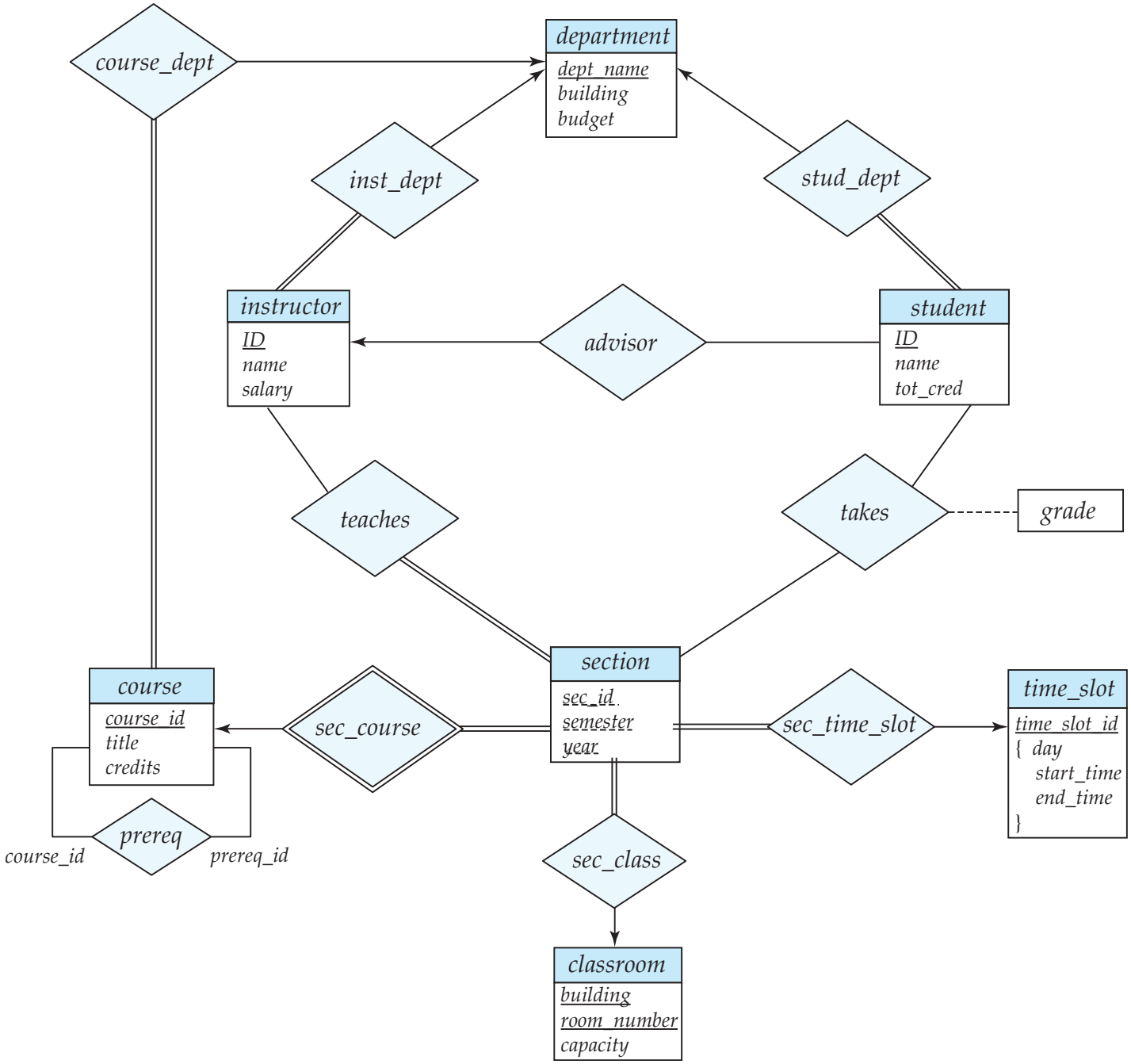
ER-model to Relational Summary (Cont.)

- ***Rule 4)*** *Binary relationship R between A and B: one-to-many / many-to-one*
  - *Add PK of the "one" side as foreign key to the "many" side.*
  - *Add any attributes of the relationship R to the "many" side.*
- ***Rule 5)*** *Binary relationship R between A and B: many-to-many*
  - *Create a new relation R.*
  - *Add PK's of A and B as attributes + plus all attributes of R.*
  - *The primary key of the relationship is PK(A) + PK(B). The PK attributes of A/B form a foreign key to A/B*
- ***Rule 6)*** *N-ary relationship R between E_1 … E_n*
  - *Create a new relation.*
  - *Add all the PK's of E_1 … E_n. Add all attributes of R to the new relation.*
  - *The primary key or R is PK(E_1) … PK(E_n). Each PK(E_i) is a foreign key to the corresponding relation.*

ER-model to Relational Summary (Cont.)

- ***Rule 7)*** *Entity E with multi-valued attribute A*
  - *Create new relation. Add A and PK(E) as attributes.*
  - *PK is all attributes. PK(E) is a foreign key.*

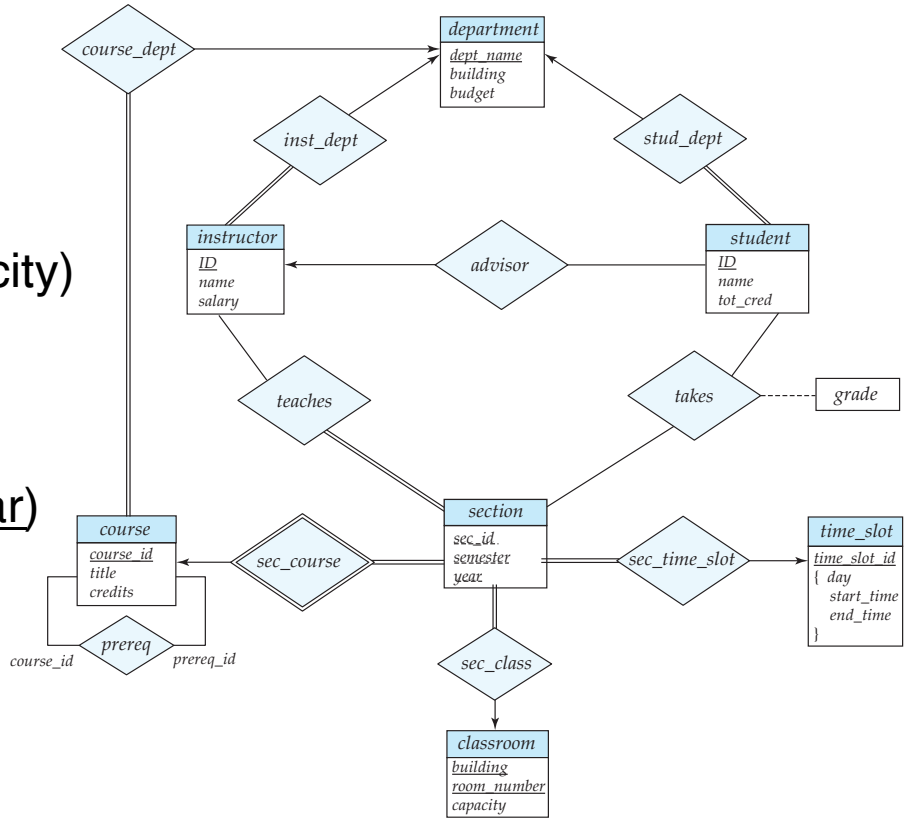
E-R Diagram for a University Enterprise



Translate the University ER-Model

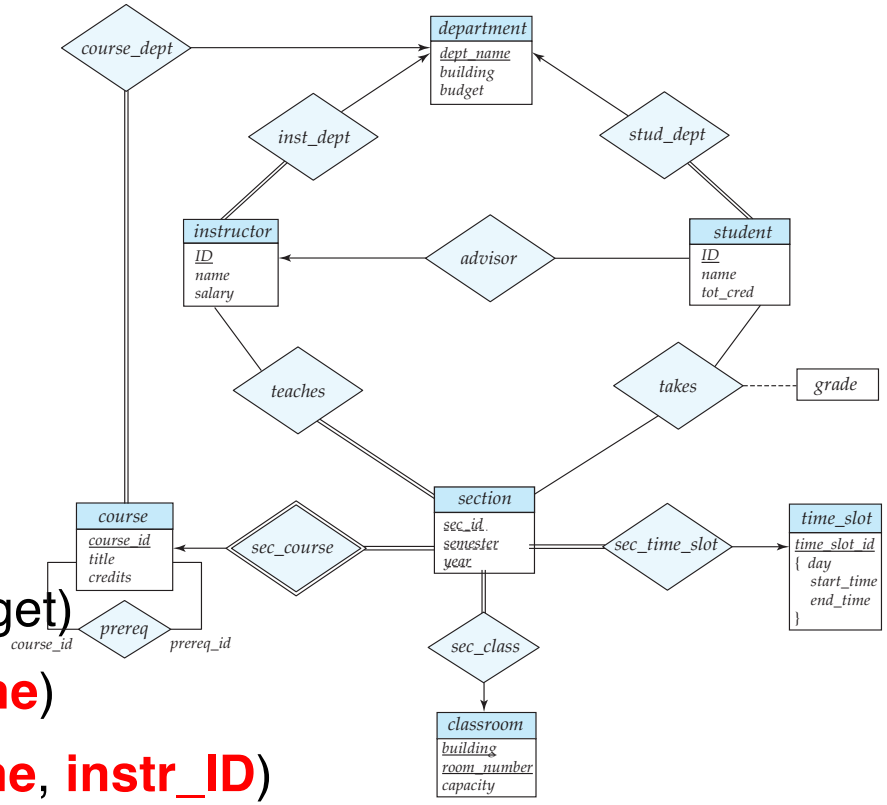
- **Rule 1) Strong Entities**
  - **department**(dept_name, building, budget)
  - **instructor**(ID, name, salary)
  - **student**(ID, name, tot_cred)
  - **course**(course_id, title, credits)
  - **time_slot**(time_slot_id)
  - **classroom**(building,room_number, capacity)

- **Rule 2) Weak Entities**
  - **section**(course_id, sec_id, semester, year)



Translate the University ER-Model

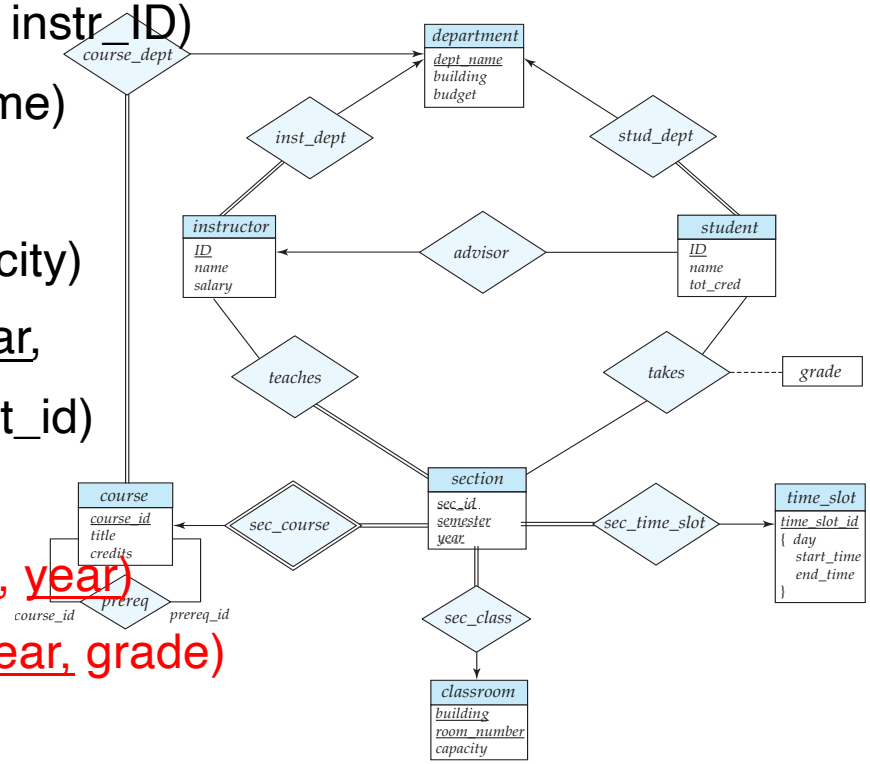
- **Rule 3) Relationships one-to-one**
  - None exist



- **Rule 4) Relationships one-to-many**
  - **department**(dept_name, building, budget)
  - **instructor**(ID, name, salary, **dept_name**)
  - **student**(ID, name, tot_cred, **dept_name**, **instr_ID**)
  - **course**(course_id, title, credits, **dept_name**)
  - **time_slot**(time_slot_id)
  - **classroom**(building,room_number, capacity)
  - **section**(course_id, sec_id, semester, year, **room_building**, **room_number**, **time_slot_id**)

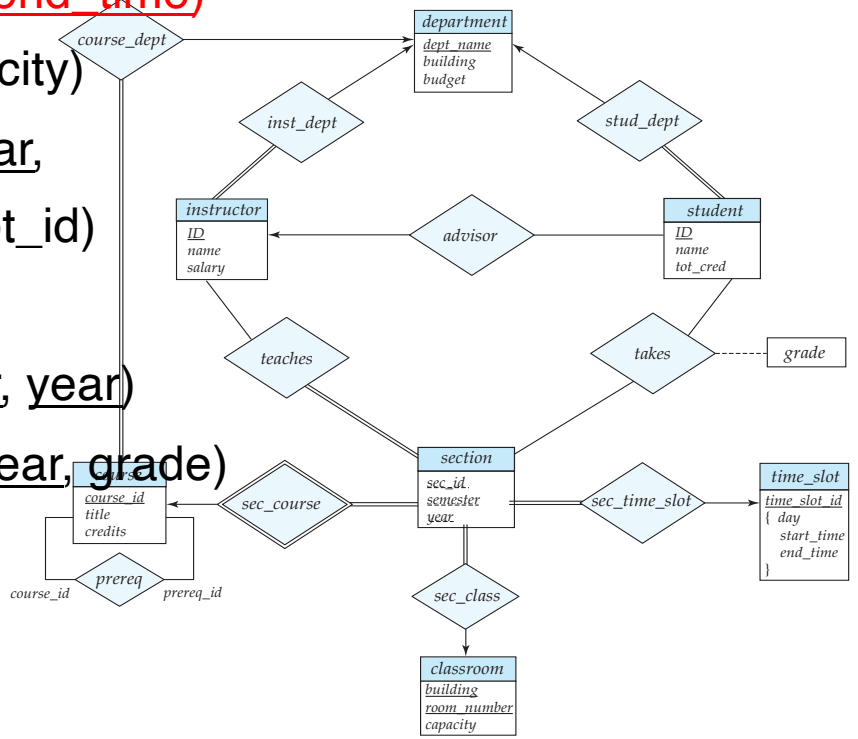
Translate the University ER-Model

- **Rule 5) Relationships many-to-many**
  - **department**(dept_name, building, budget)
  - **instructor**(ID, name, salary, dept_name)
  - **student**(ID, name, tot_cred, dept_name, instr_ID)
  - **course**(course_id, title, credits, dept_name)
  - **time_slot**(time_slot_id)
  - **classroom**(building,room_number, capacity)
  - **section**(course_id, sec_id, semester, year,
    room_building, room_number, time_slot_id)
  - **prereq**(course_id, prereq_id)
  - **teaches**(ID, course_id, sec_id, semester, year)
  - **takes**(ID, course_id, sec_id, semester, year, grade)
- **Rule 6) N-ary Relationships**
  - none exist



Translate the University ER-Model

- **Rule 7) Multivalued attributes**
  - **department**(dept_name, building, budget)
  - **instructor**(ID, name, salary, dept_name)
  - **student**(ID, name, tot_cred, dept_name, instr_ID)
  - **course**(course_id, title, credits, dept_name)
  - **time_slot**(time_slot_id)
  - **time_slot_day**(time_slot_id, start_time, end_time)
  - **classroom**(building,room_number, capacity)
  - **section**(course_id, sec_id, semester, year,
    room_building, room_number, time_slot_id)
  - **prereq**(course_id, prereq_id)
  - **teaches**(ID, course_id, sec_id, semester, year)
  - **takes**(ID, course_id, sec_id, semester, year, grade)



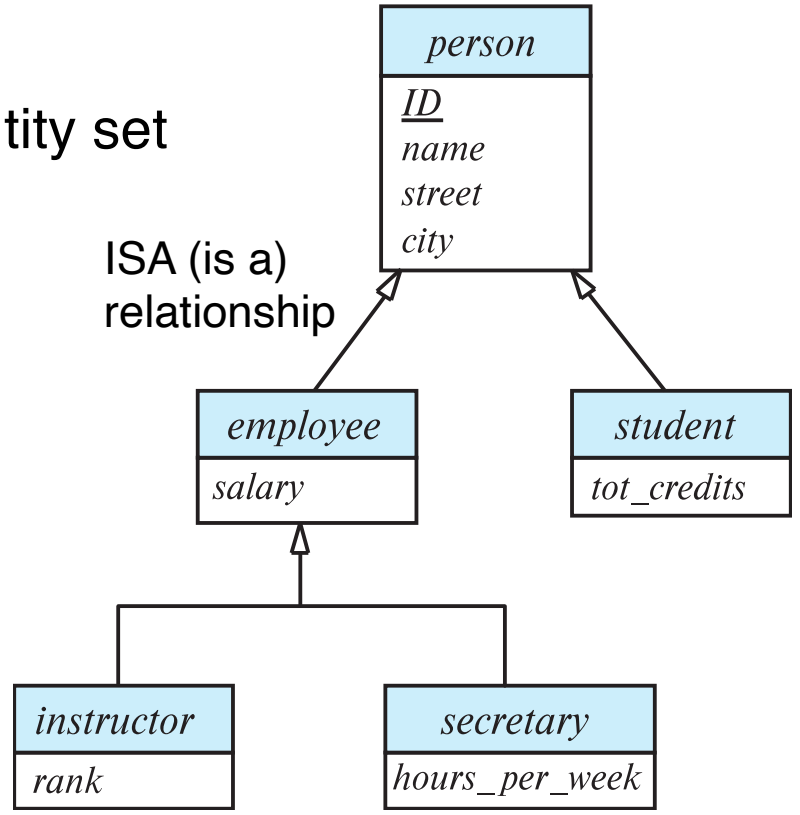
Extended E-R Features

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

- The way we depict specialization in an E-R diagram depends on whether an entity may belong to multiple specialized entity sets or if it must belong to at most one specialized entity set.
- **Overlapping:** multiple specialized entity sets
  - *employee* and *student*
  - Two separate arrows
- **Disjoint:** at most one specialized entity set
  - *instructor* and *secretary*
  - Single arrow



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

  - Drawback: *name, street* and *city* may be stored redundantly for people who are both students and employees

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.
- Similarities between entities
  - instructor entity set with attributes instructor *id*, *instructor name*, *instructor salary*, and *rank*
  - secretary entity set with attributes *secretary_id, secretary_name, secretary_salary*, and *hours_per_week*

Completeness constraint

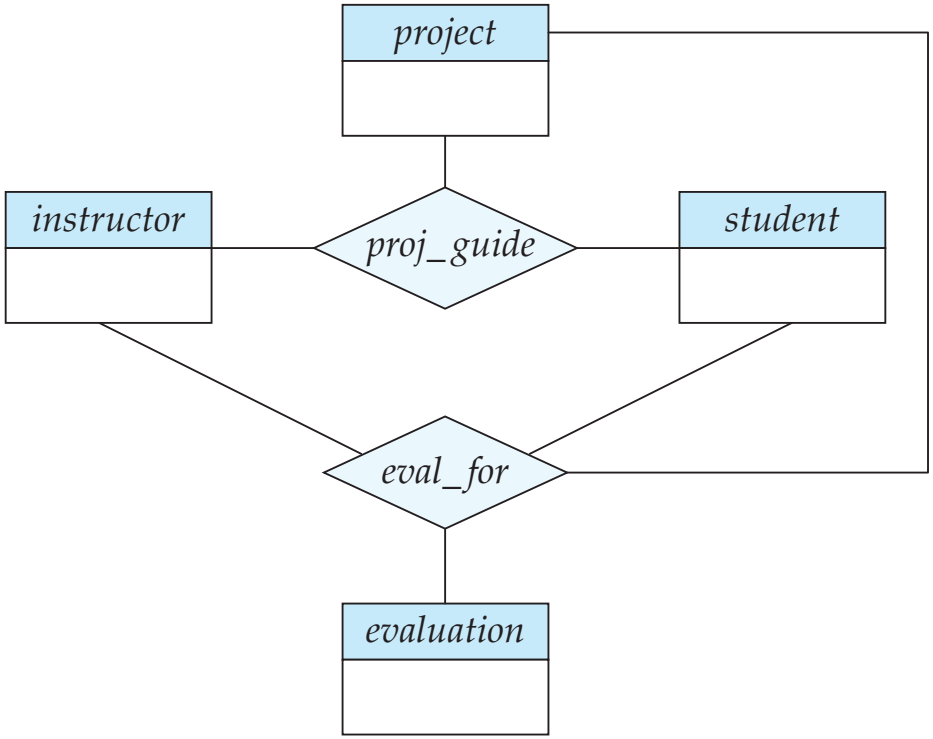
- **Completeness constraint** -- specifies whether or not an entity in the higher-level entity set must belong to at least one of the lower-level entity sets within a generalization.
  - **total**: an entity must belong to one of the lower-level entity sets
  - **partial**: an entity need not belong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

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

Aggregation

- One limitation of the E-R model is that it cannot express relationships among relationships.
- 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

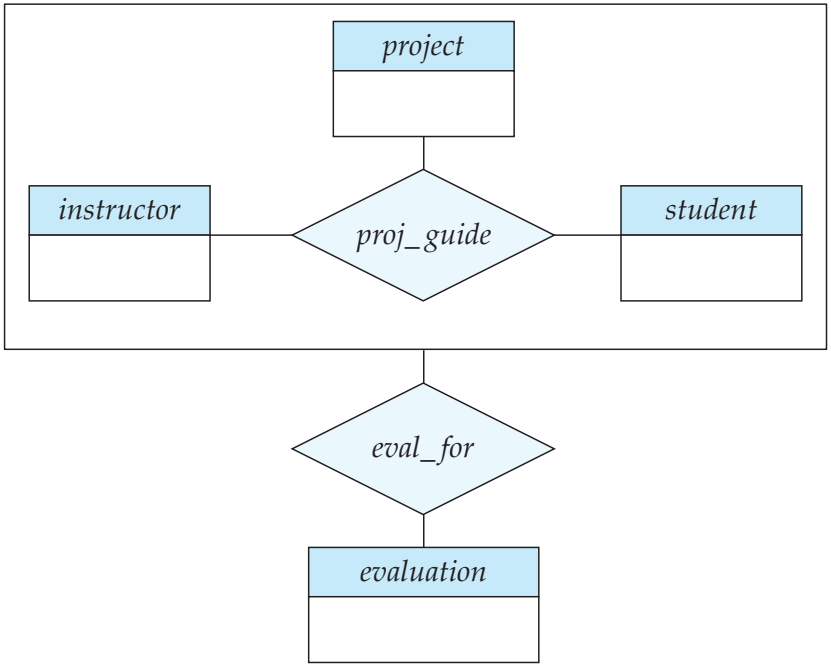


Aggregation (Cont.)

- 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.)

- Eliminate this redundancy via *aggregation* 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



Reduction to Relational Schemas

- To represent aggregation, create a schema containing
  - Primary key of the aggregated relationship,
  - The primary key of the associated entity set
  - Any descriptive attributes
- In our example:
  - The schema *eval_for* is:

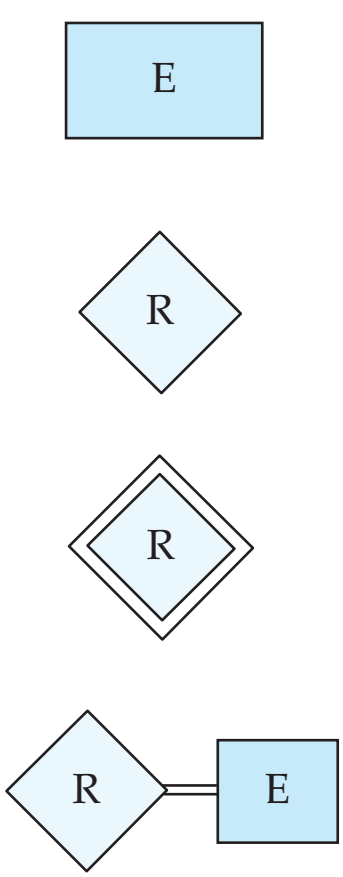
    eval_for (*s_ID, project_id, i_ID, evaluation_id*)
  - The schema *proj_guide* is redundant.

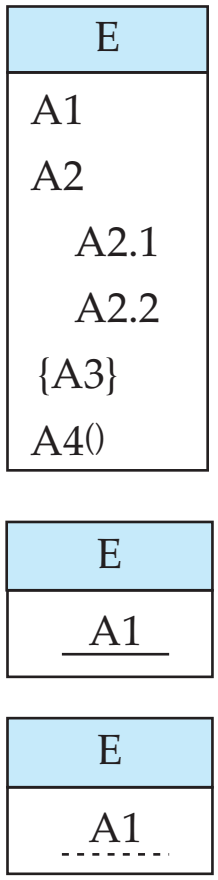
E-R Design Decisions

- 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.

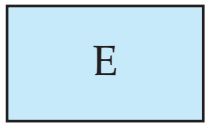
Alternatives

Summary of Symbols Used in E-R Notation

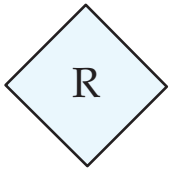




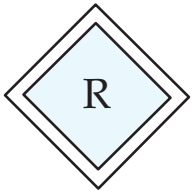
Summary of Symbols Used in E-R Notation



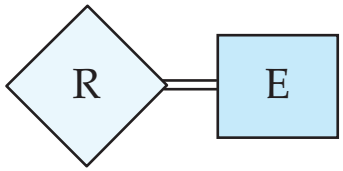
entity set



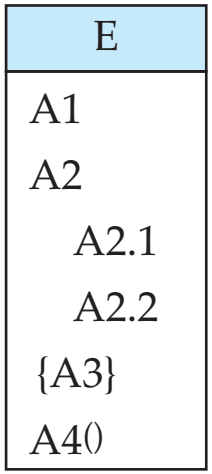
relationship set



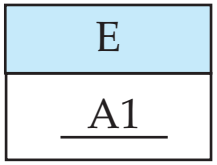
identifying relationship set for weak entity set



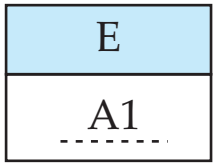
total participation of entity set in relationship



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

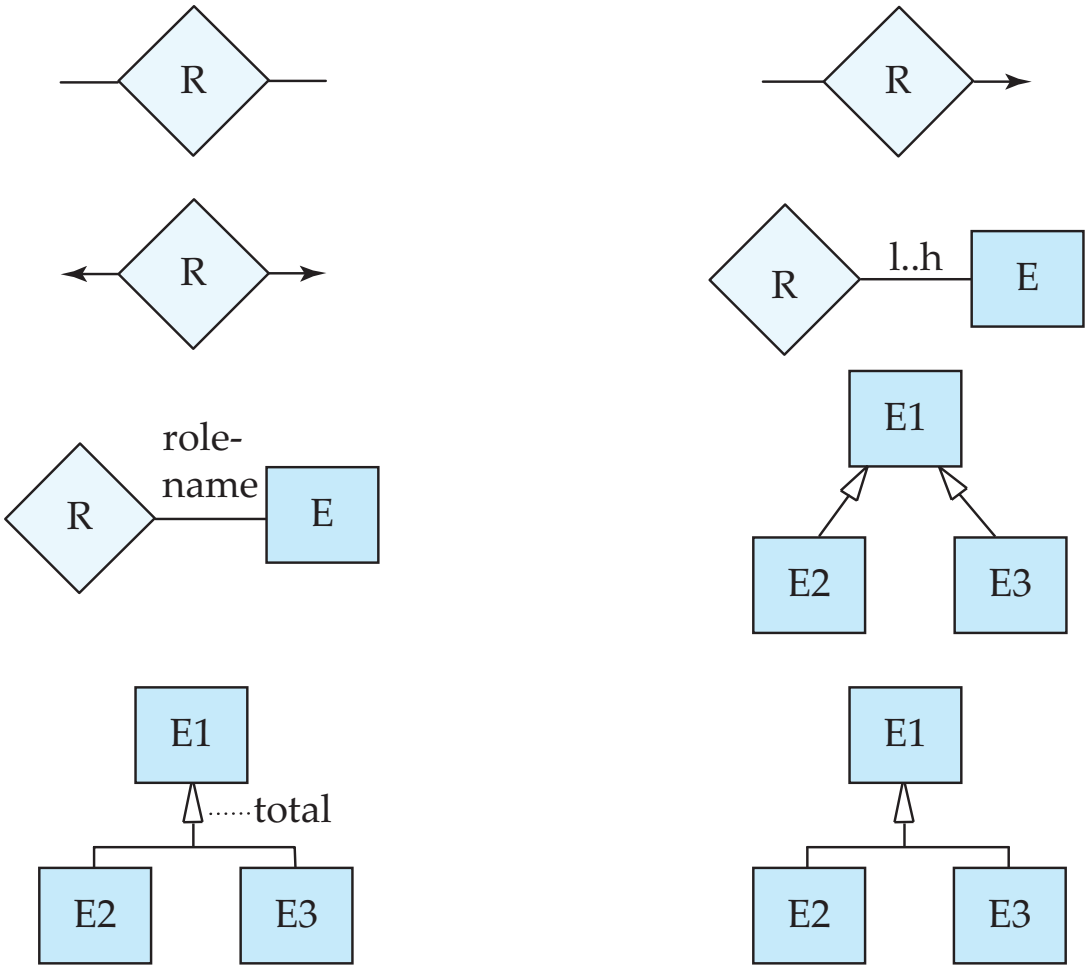


primary key

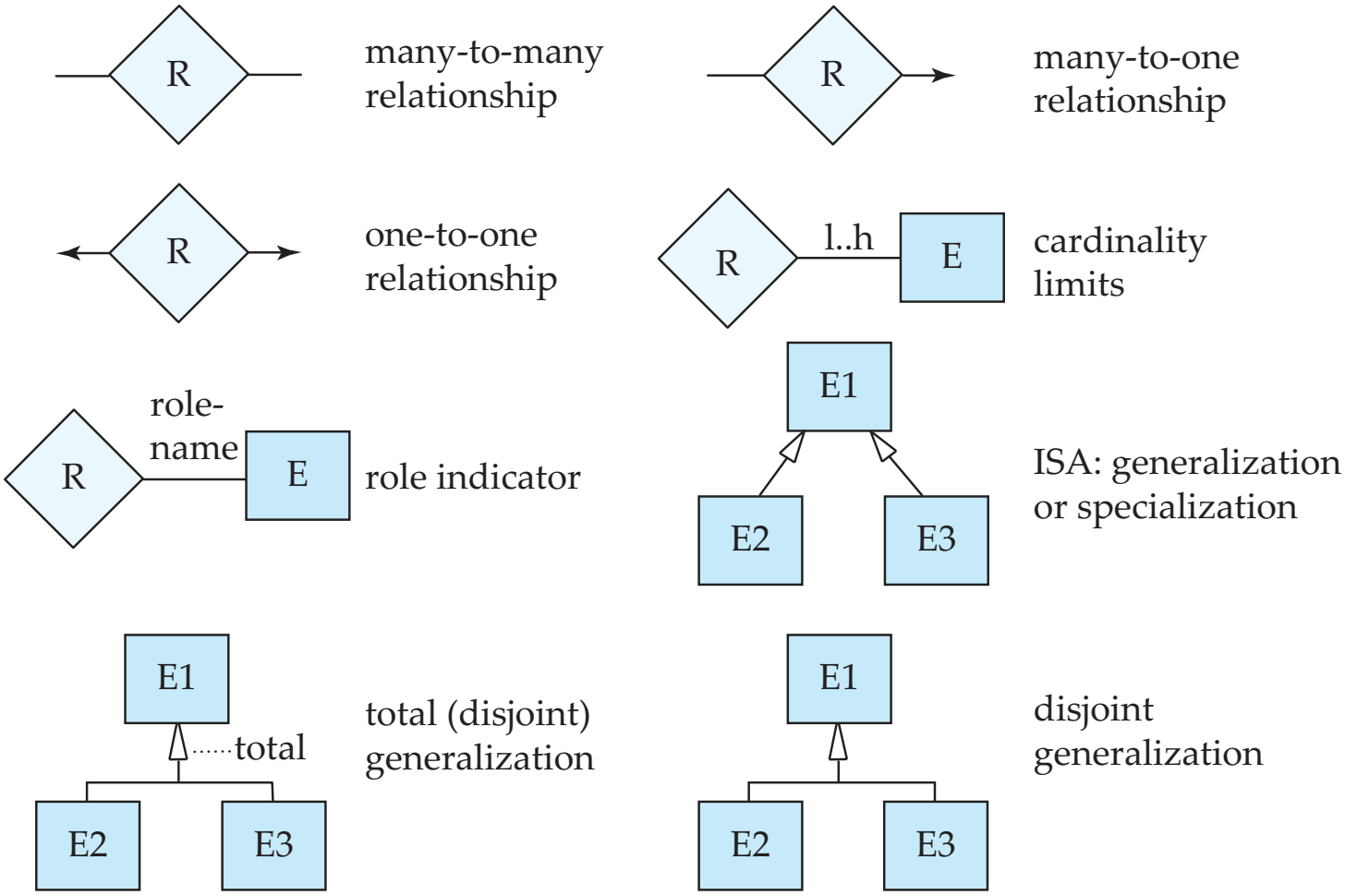


discriminating attribute of weak entity set

Symbols Used in E-R Notation (Cont.)



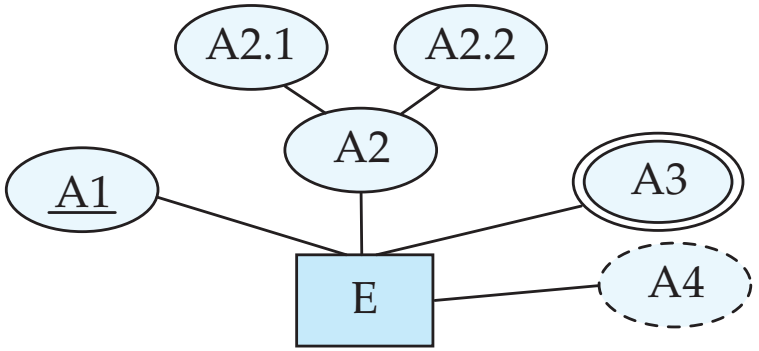
Symbols Used in E-R Notation (Cont.)



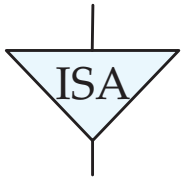
Alternative ER Notations

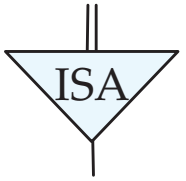
- Chen, IDEF1X, …

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

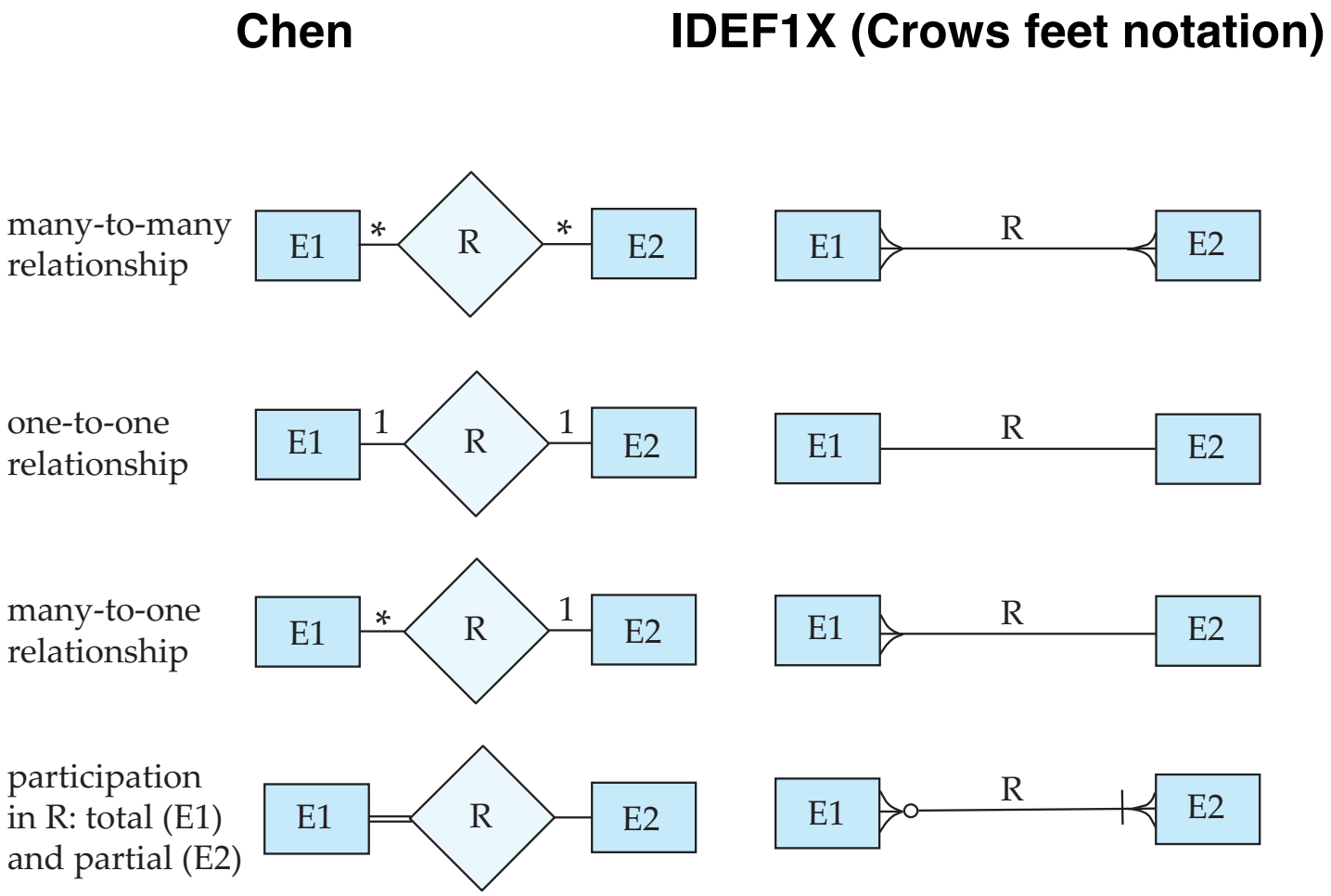


weak entity set 

generalization 

total
generalization 

Alternative ER Notations

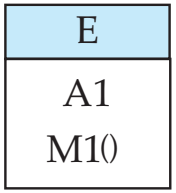


UML

- **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.

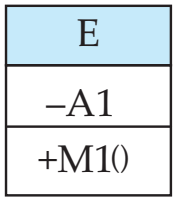
ER vs. UML Class Diagrams

ER Diagram Notation

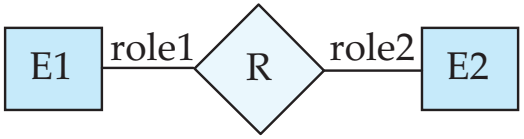


entity with attributes (simple, composite, multivalued, derived)

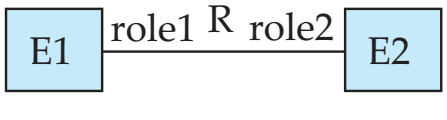
Equivalent in UML

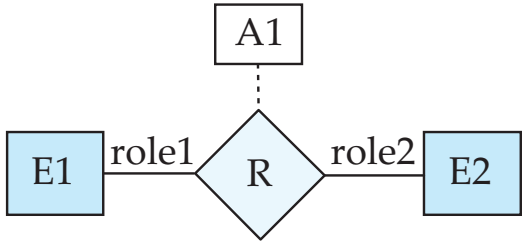


class with simple attributes and methods (attribute prefixes: + = public, – = private, # = protected)

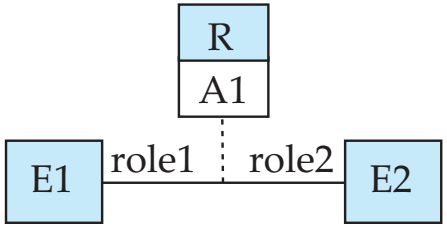


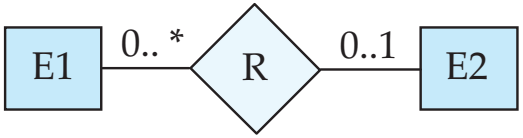
binary relationship



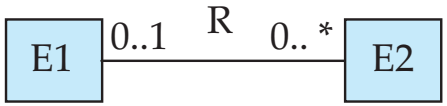


relationship attributes



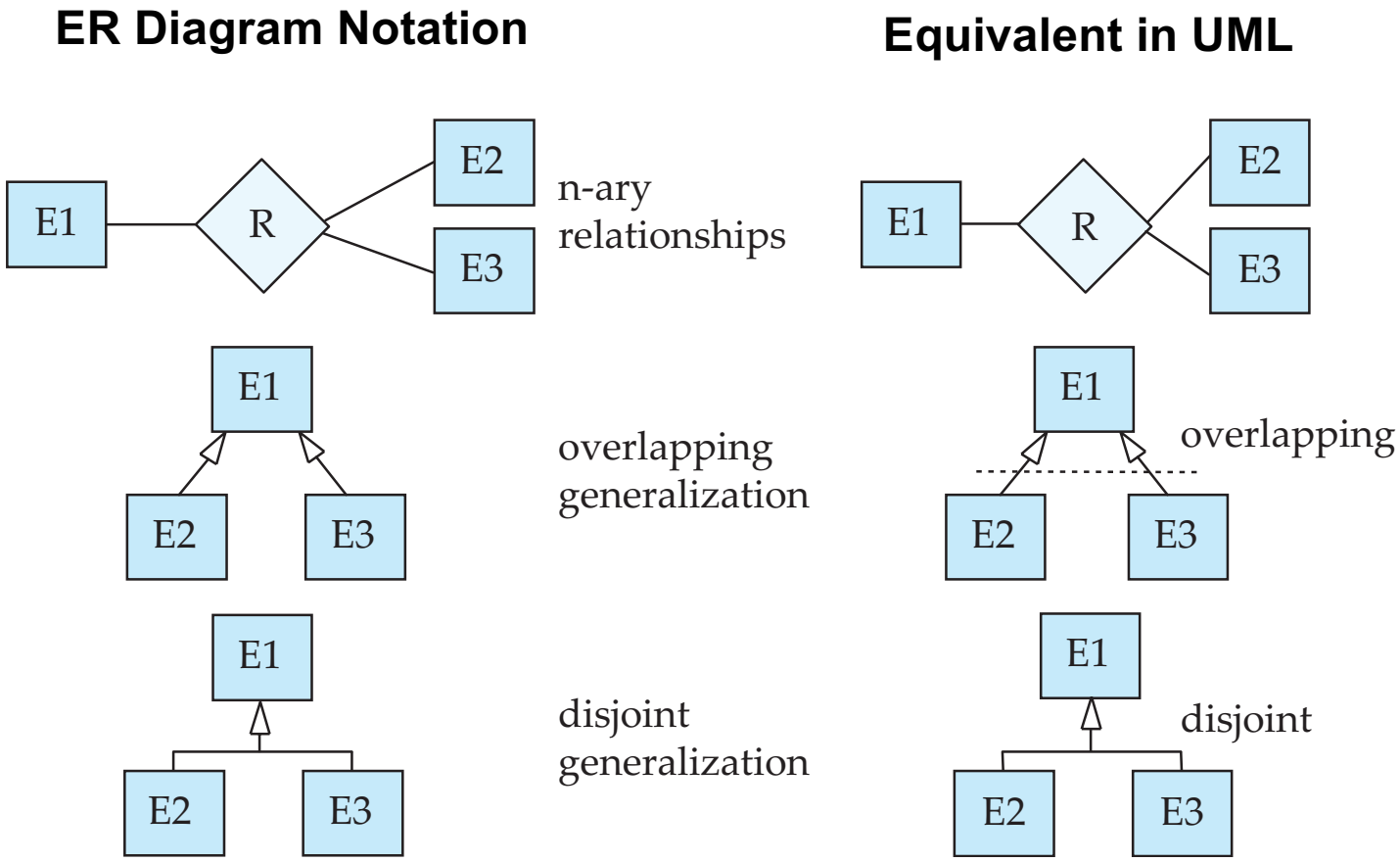


cardinality constraints



* Note reversal of position in cardinality constraint depiction

ER vs. UML Class Diagrams

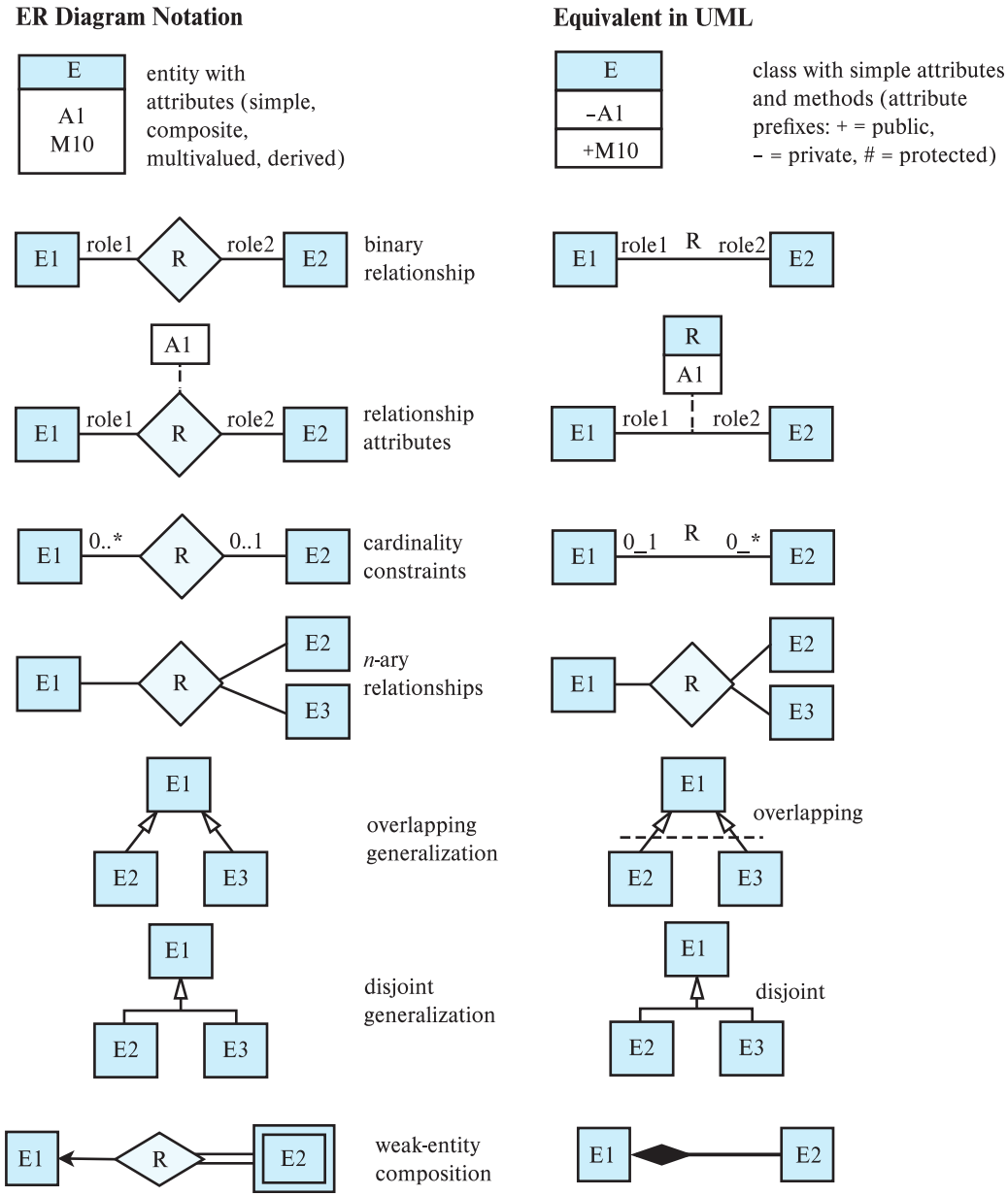


* Generalization can use merged or separate arrows independent of disjoint/overlapping

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.

ER vs. UML Class Diagrams



Recap

- ER-model
  - Entities
    - Strong
    - Weak
  - Attributes
    - Simple vs. Composite
    - Single-valued vs. Multi-valued
  - Relationships
    - Degree (binary vs. N-ary)
  - Cardinality constraints
  - Specialization/Generalization
    - Total vs. partial
    - Disjoint vs. overlapping
  - Aggregation

Corresponding Reading Materials

- Database System Concepts 6th & 7th Edition
  - Chapter 4 & 5