Chapter 6: Entity-Relationship Model

Outline

Database Design Process

Modeling

Constraints

Weak Entity Sets

Reduction to Relation Schemas

Design Issues

Extended E-R Features

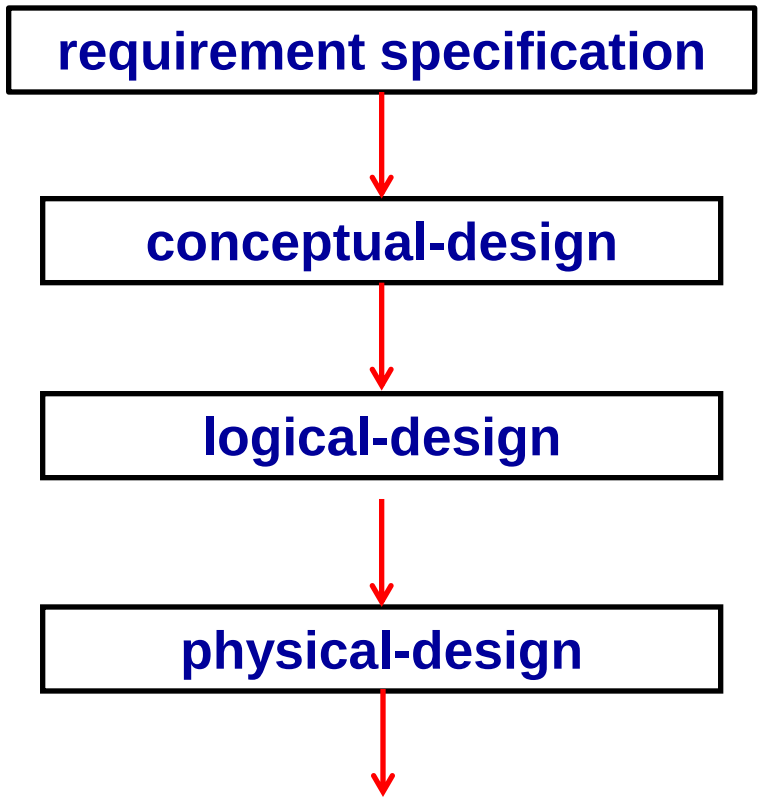
UML

University Schemas

Relational Schemas of a University Enterprise

classroom(<u>*building*</u>, <u>*room_number*</u>, *capacity*)
department(<u>*dept_name*</u>, *building*, *budget*)
course(<u>*course_id*</u>, *title*, *dept_name*, *credits*)
instructor(<u>*ID*</u>, *name*, *dept_name*, *salary*)
section(<u>*course_id*</u>, <u>*sec_id*</u>, <u>*semester*</u>, <u>*year*</u>, *building*, *room_number*, *time_slot_id*)
teaches(<u>*ID*</u>, <u>*course_id*</u>, <u>*sec_id*</u>, <u>*semester*</u>, <u>*year*</u>)
student(<u>*ID*</u>, *name*, *dept_name*, *tot_cred*)
takes(<u>*ID*</u>, <u>*course_id*</u>, <u>*sec_id*</u>, <u>*semester*</u>, <u>*year*</u>, *grade*)
advisor(<u>*s_ID*</u>, *i_ID*)
time_slot(<u>*time_slot_id*</u>, <u>*day*</u>, <u>*start_time*</u>, *end_time*)
prereq(<u>*course_id*</u>, <u>*prereq_id*</u>)

Database Design Process



specification of user requirement

E-R diagram

logical schema

physical schema

Design Phases

Initial phase（Requirement Specification） -- characterize fully the data needs of the prospective database users.

Second phase(Conceptual Design) -- 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 (Database Design)-- 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

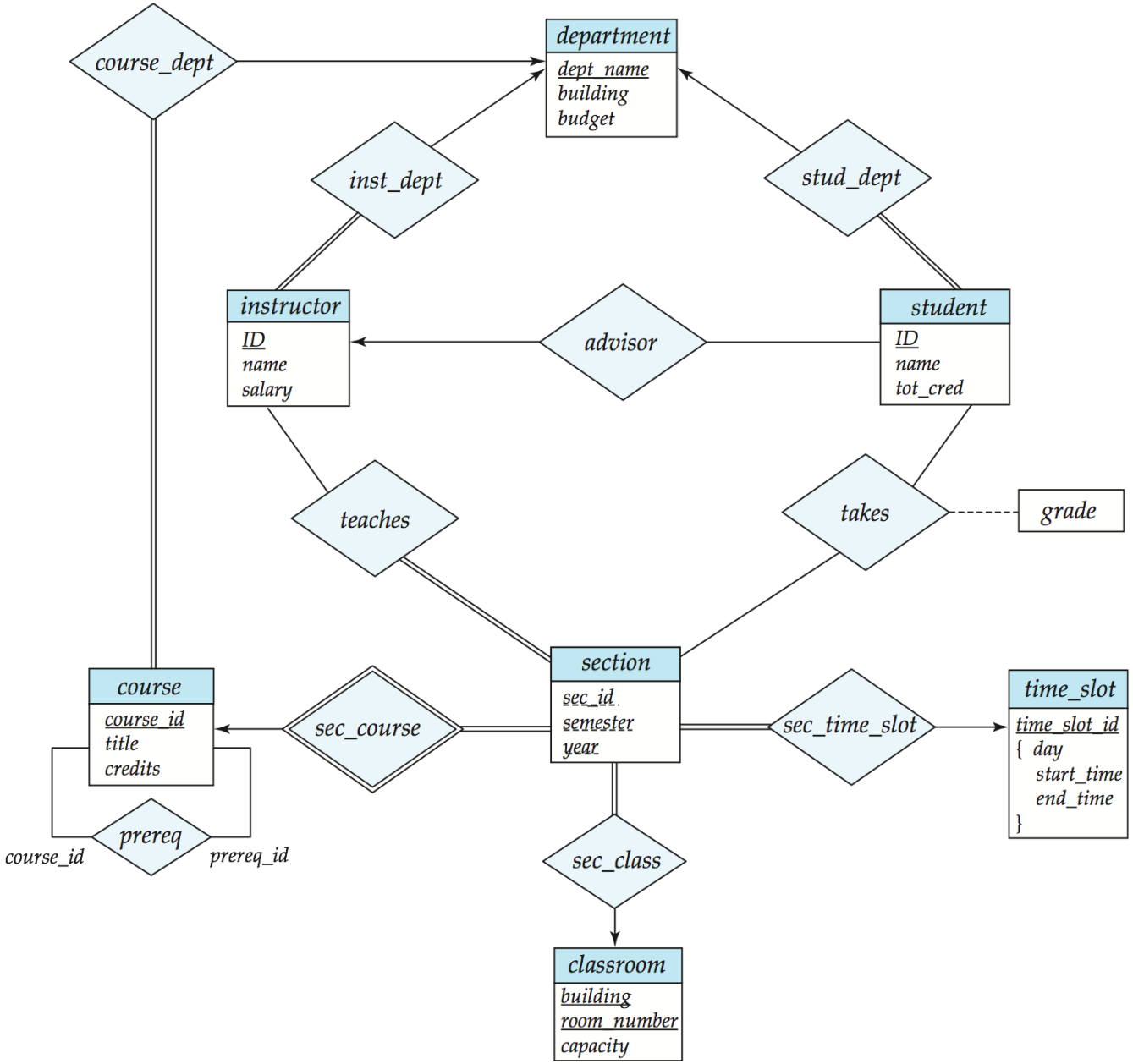
Design Alternatives

- 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.
- database design can be a challenging problem
  - **huge design space**

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

E-R Diagram for a University Enterprise



Outline of the ER Model

Database Modeling

A *database* can be modeled as:

a collection of entities,

relationship among entities.

An **entity** is an object that exists and is distinguishable from other objects.

Example: specific person, company, event, plant

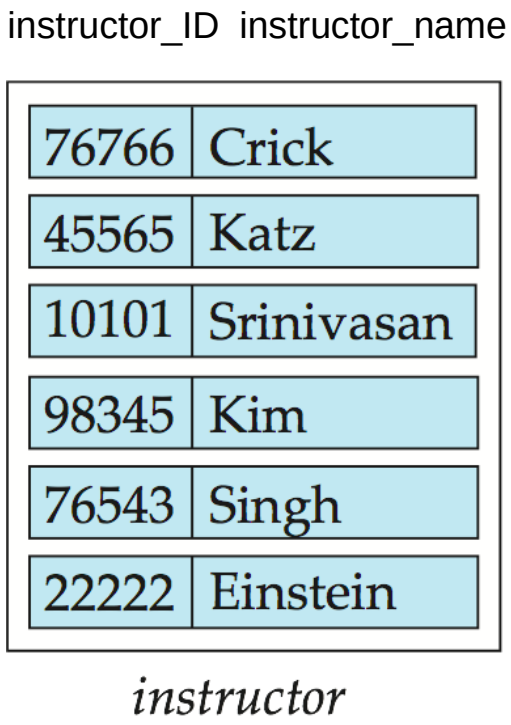
Entities have **attributes**

Example: people have *names* and *addresses*

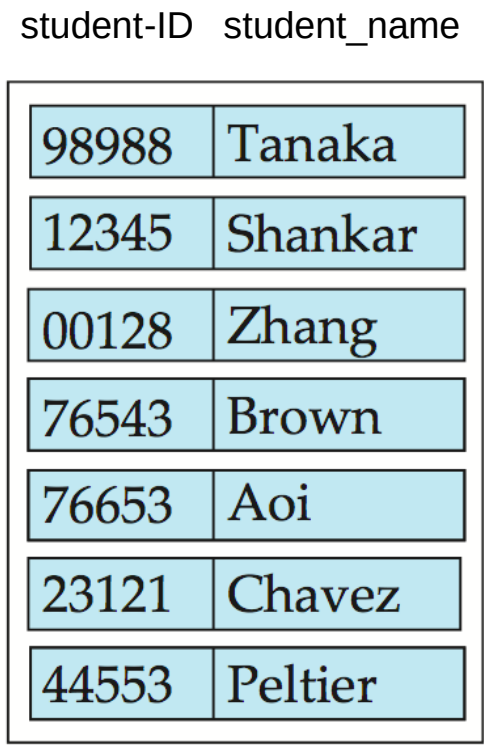
An **entity set** is a set of entities of the same type that share the same properties.

Example: set of all persons, companies, trees, holidays

Entity Sets *instructor* and *student*



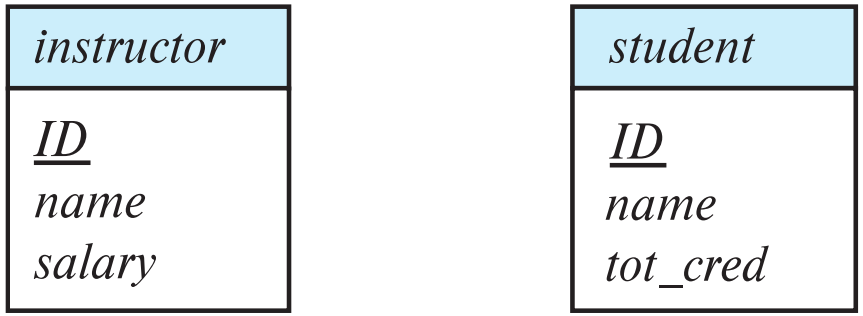
instructor



student

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



Relationship Sets

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

Example:

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

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

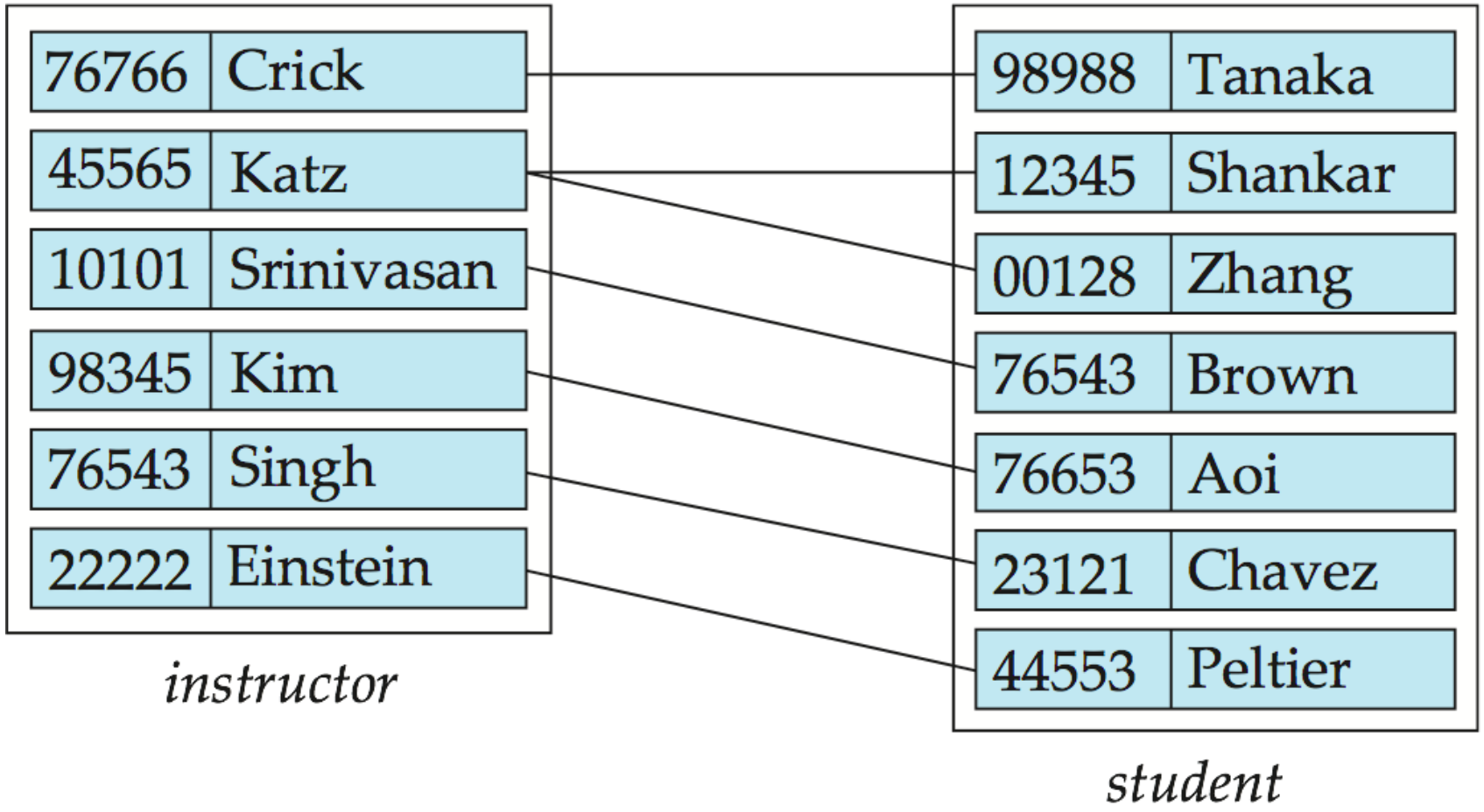
$$\{(e_1, e_2, \ldots e_n) \mid e_1 \in E_1, e_2 \in E_2, \ldots, e_n \in E_n\}$$

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

Example:

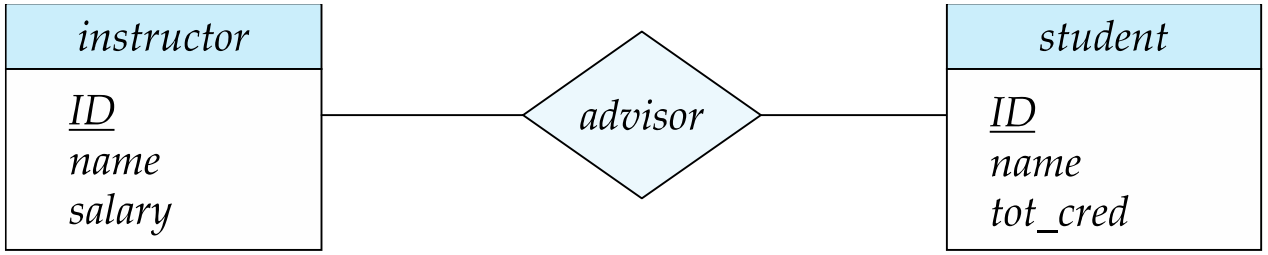
$(44553,22222) \in advisor$

Relationship Set *advisor*



Representing Relationship Sets in ER Diagrams

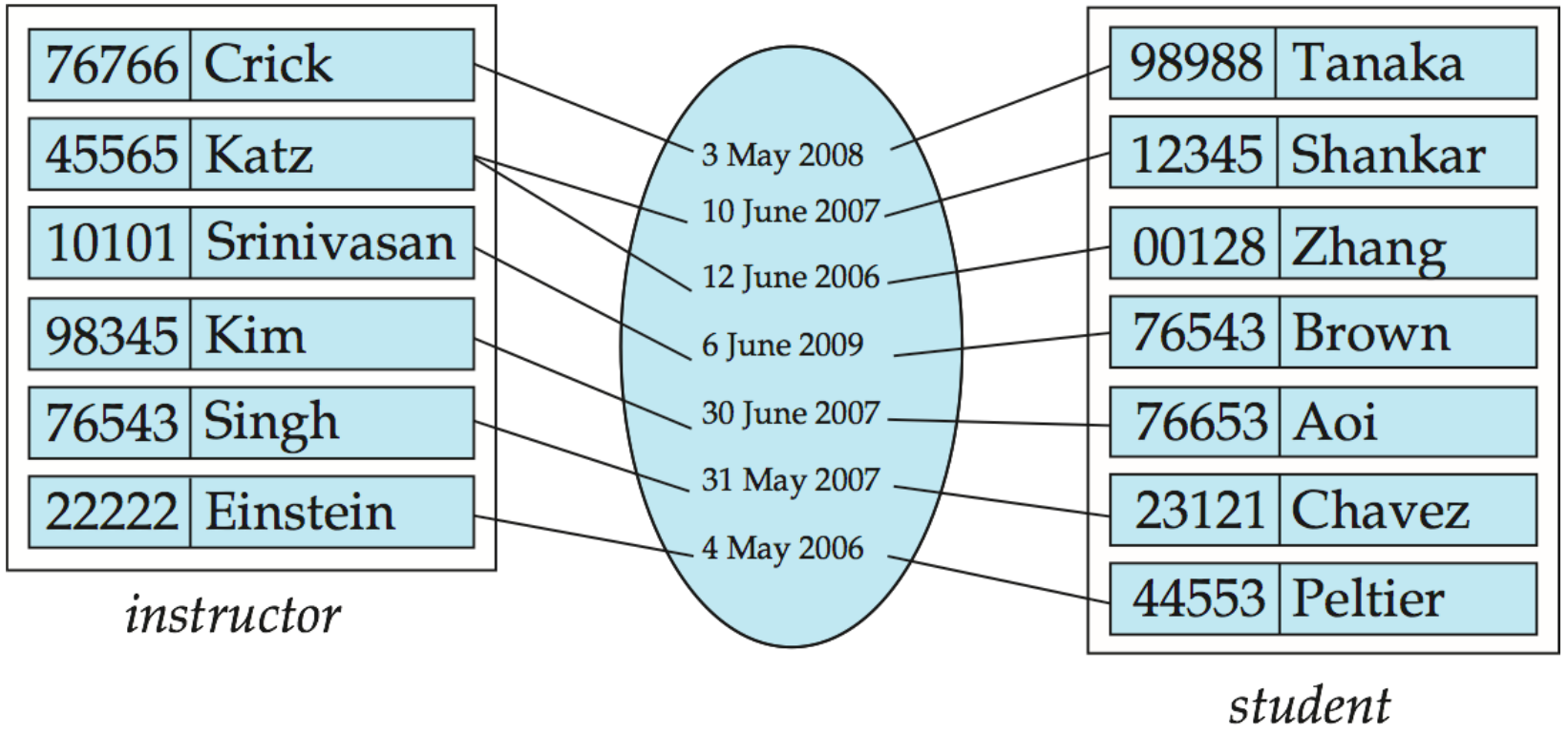
- Diamonds represent relationship sets.



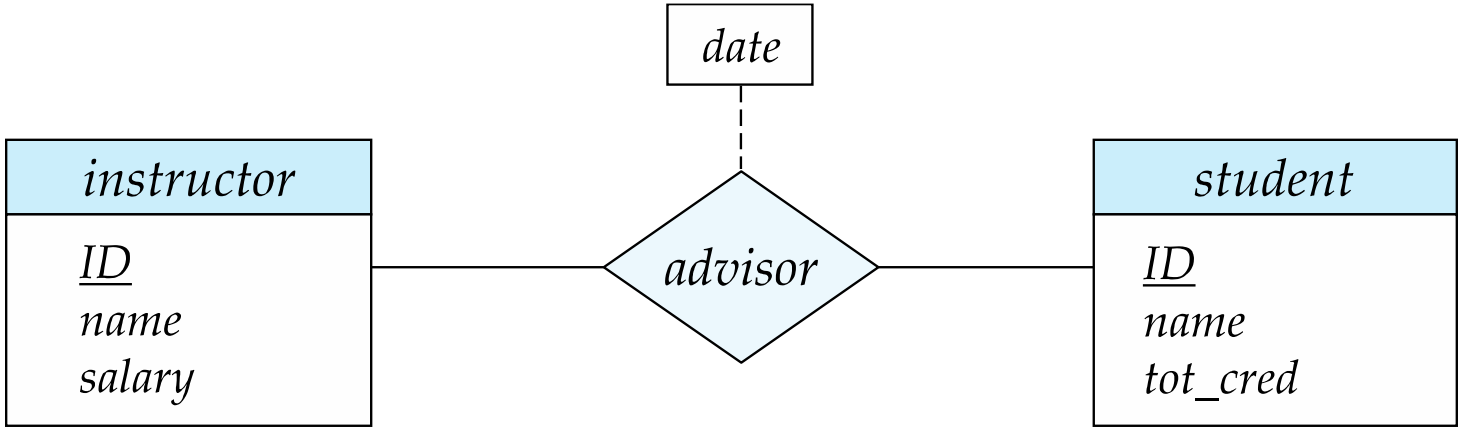
Relationship Sets with Attributes

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



Relationship Sets with Attributes

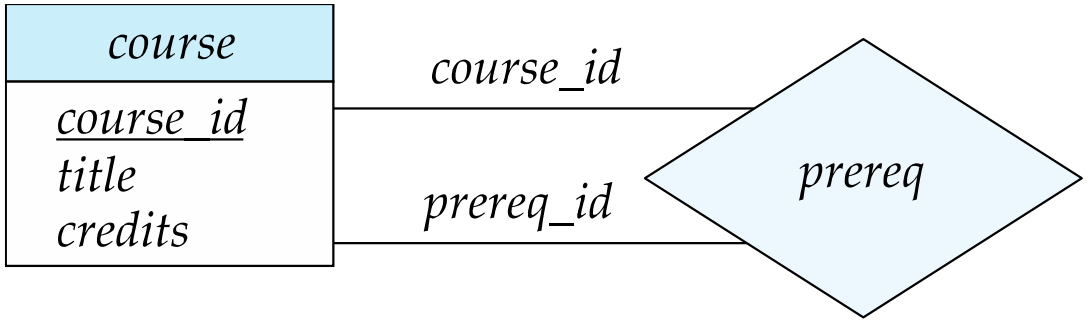


Roles

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**.



Degree(度) of a Relationship Set

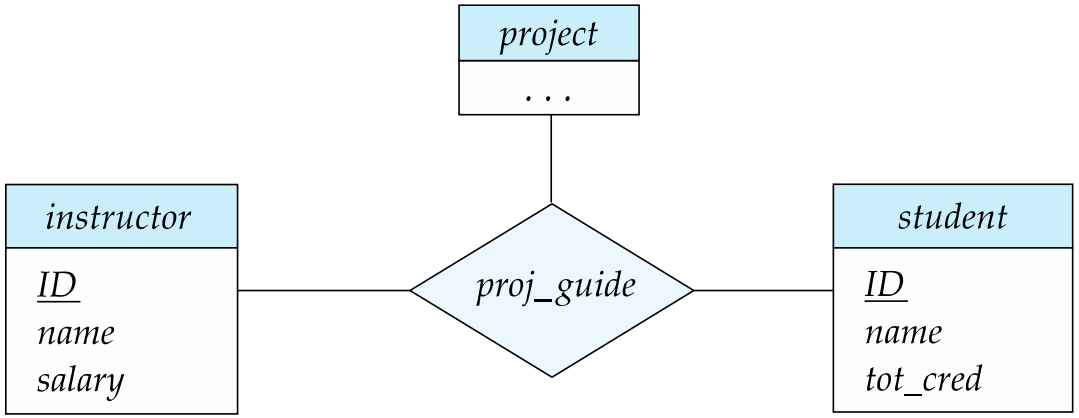
binary relationship（二元联系）

involve two entity sets (or degree two).

most relationship sets in a database system are binary.

There are occasions when it is more convenient to represent relationships as non-binary.

E-R Diagram with a Ternary Relationship



Attributes

An entity is represented by a set of attributes, that is descriptive properties possessed by all members of an entity set.

Example:

instructor = (*ID, name, street, city, salary*)
course= (*course_id, title, credits*)

Domain – the set of permitted values for each attribute

Attribute types:

Simple（简单）and **composite（复合）** attributes.

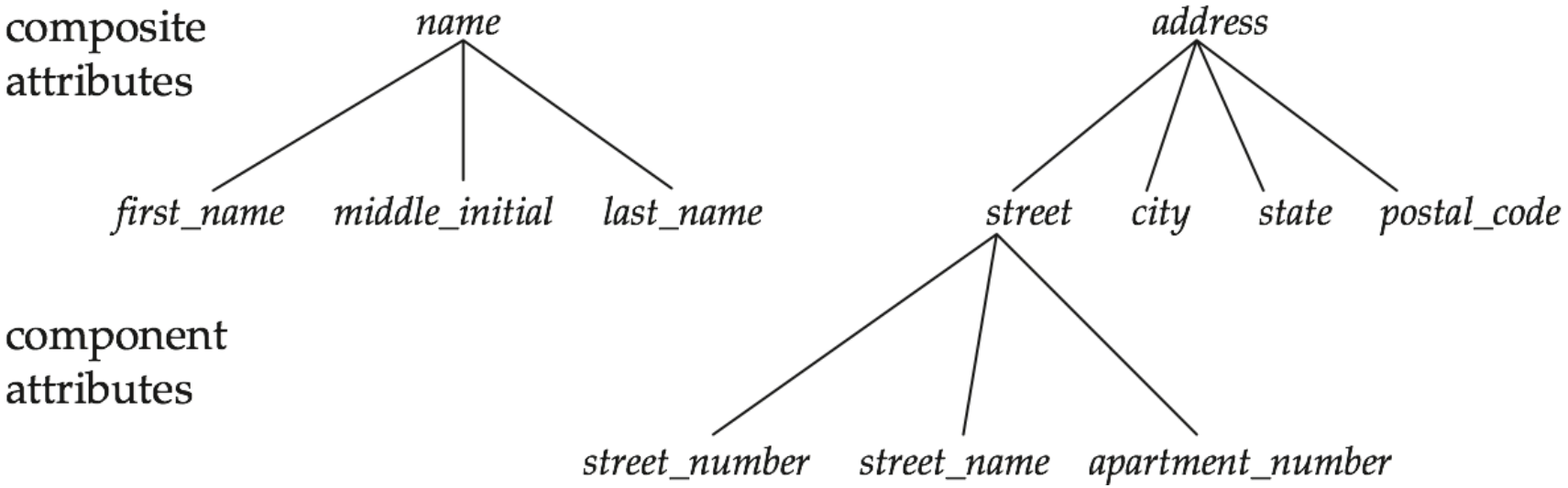
Single-valued（单值） and **multivalued（多值）** attributes

- Example: multivalued attribute: *phone_numbers*

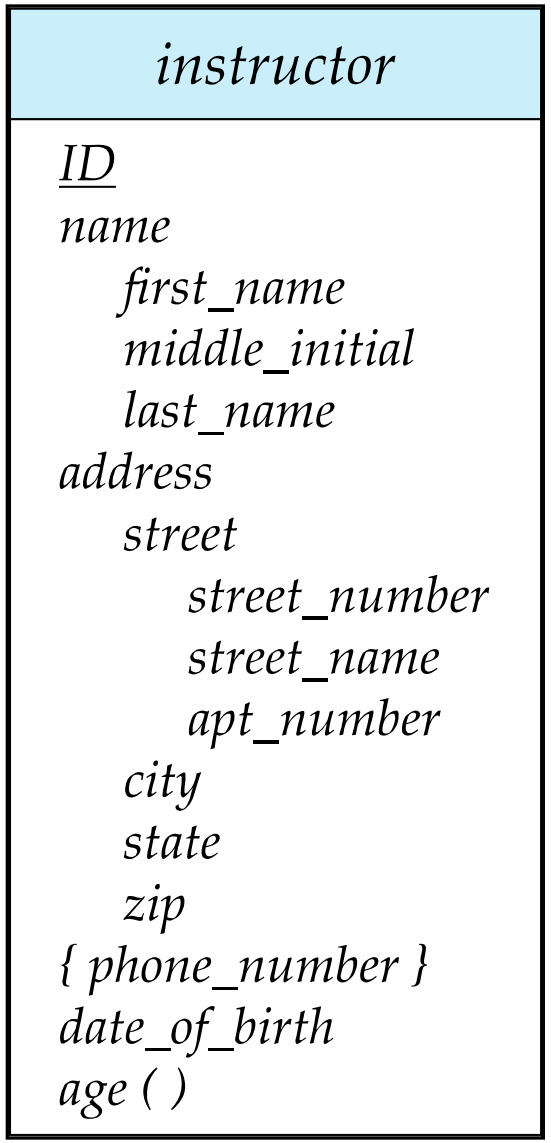
Derived（派生） attributes

- Can be computed from other attributes
- Example: age, given date_of_birth

Composite Attributes



Representing Complex Attributes in ER Diagram



Mapping Cardinality Constraints
（映射基数约束）

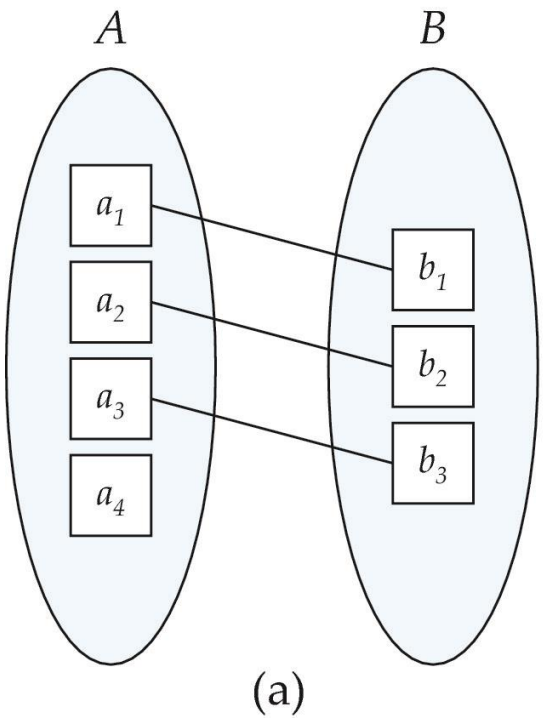
Express the number of entities to which another entity can be associated via a relationship set.

Most useful in describing binary relationship sets.

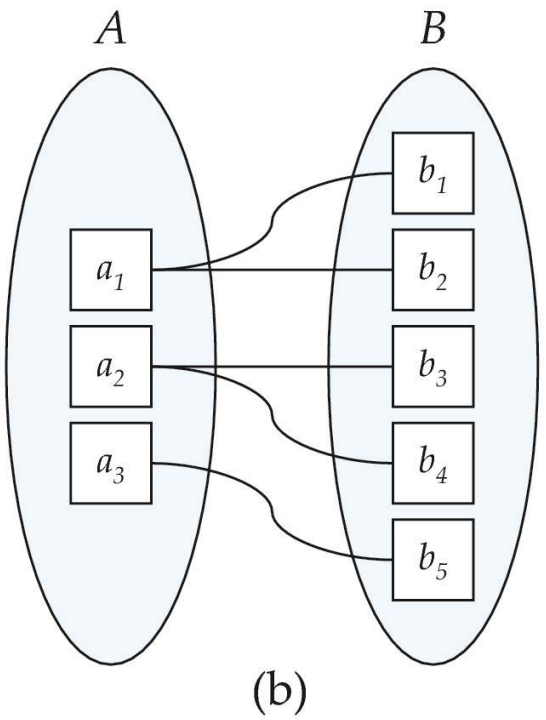
For a binary relationship set the mapping cardinality must be one of the following types:

- One to one
- One to many
- Many to one
- Many to many

Mapping Cardinalities



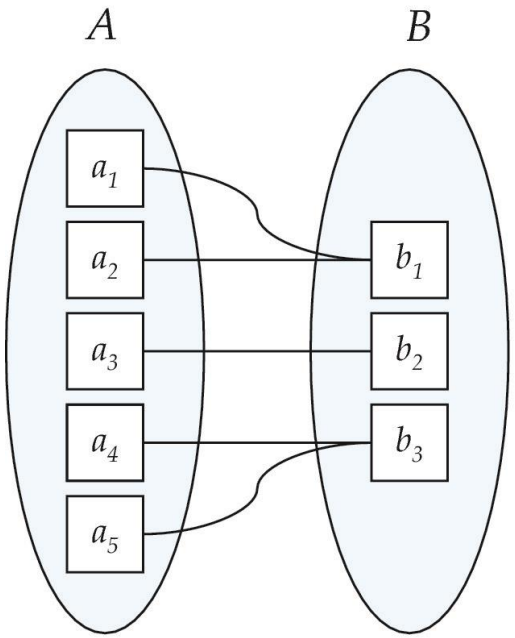
One to one



One to many

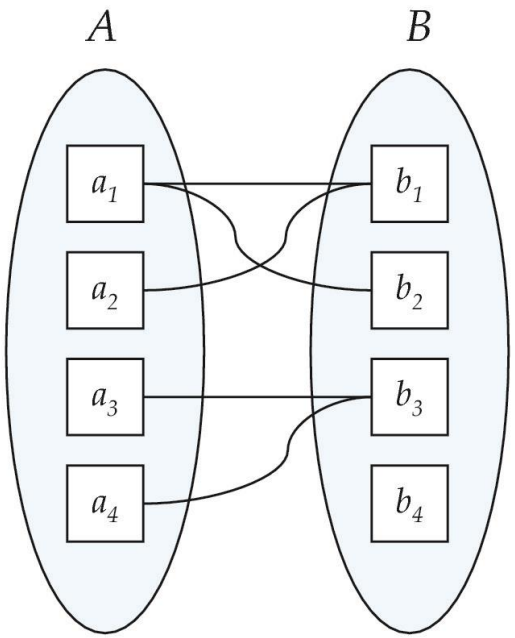
Note: Some elements in *A* and *B* may not be mapped to any elements in the other set

Mapping Cardinalities



(a)

Many to one



(b)

Many to many

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

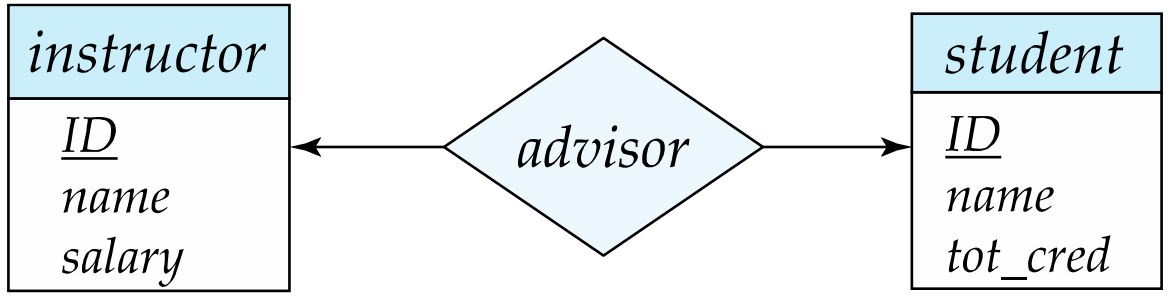
Representing Cardinality Constraints in ER Diagram

We express cardinality constraints by drawing either a directed line (→), signifying “one,” or an undirected line (—), signifying “many,” between the relationship set and the entity set.

One-to-one relationship between an *instructor* and a *student* :

A *student* is associated with at most one *instructor* via the relationship *advisor*

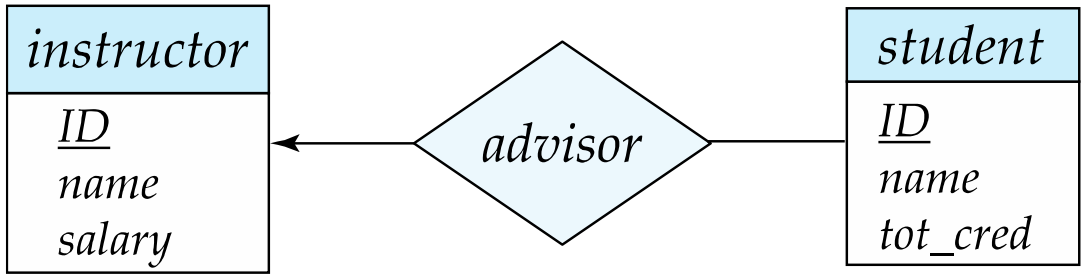
A *instructor* is associated with at most one *student* via the relationship *advisor*



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,

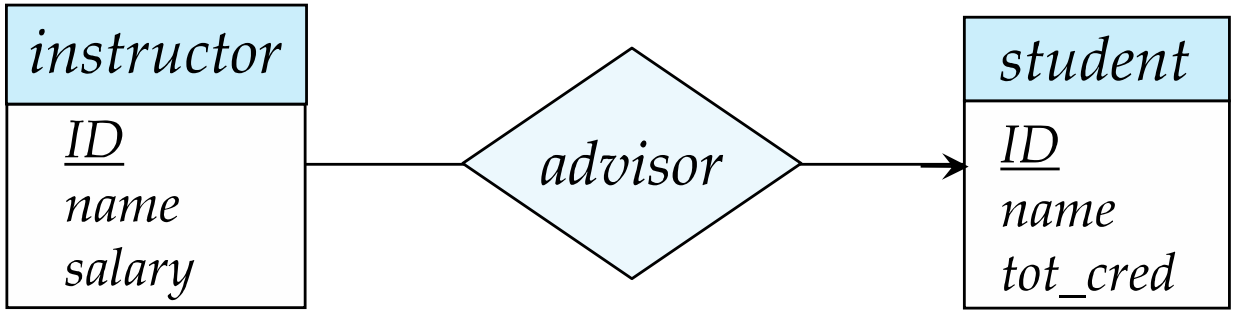


Many-to-One Relationships

In a many-to-one relationship between an *instructor* and a *student,*

an *instructor* is associated with at most one *student* via *advisor*,

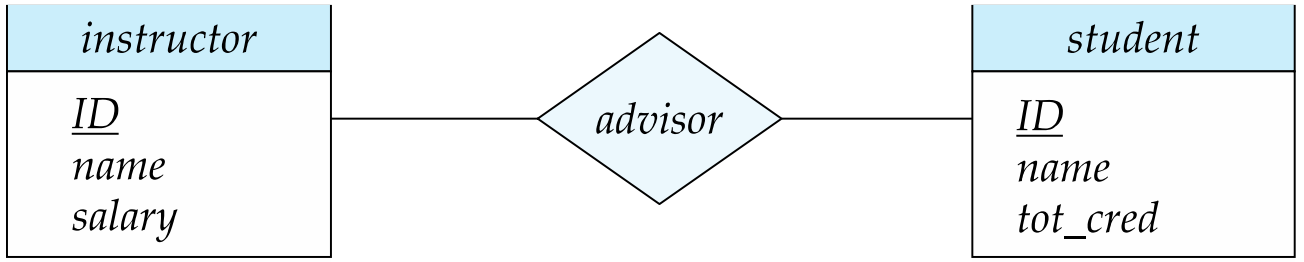
and a *student* is associated with several (including 0) *instructors* via *advisor*



Many-to-Many Relationship

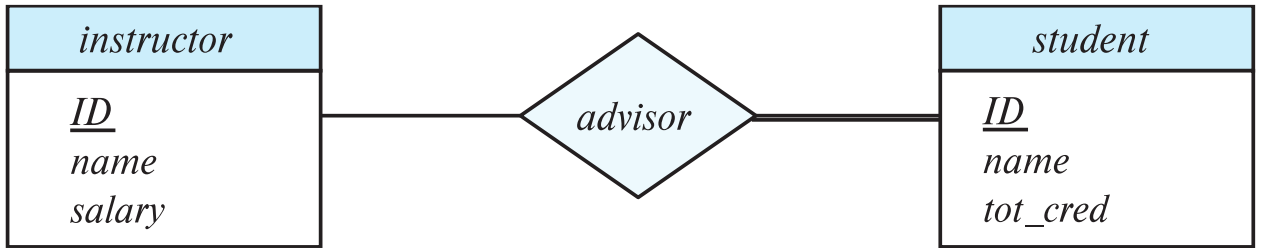
An *instructor* is associated with several (possibly 0) *students* via *advisor*

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



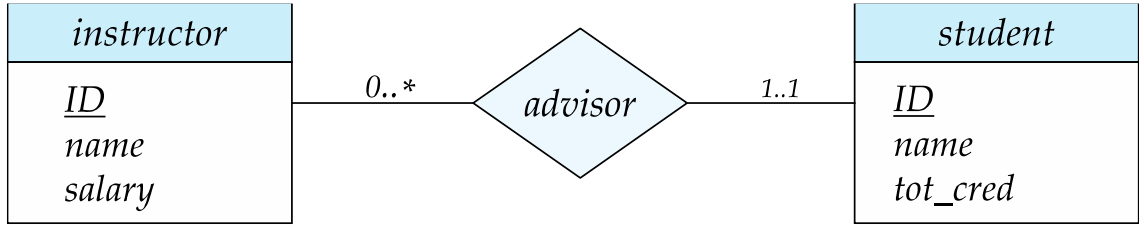
participation of *student* in *advisor* relation is total

  - 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

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



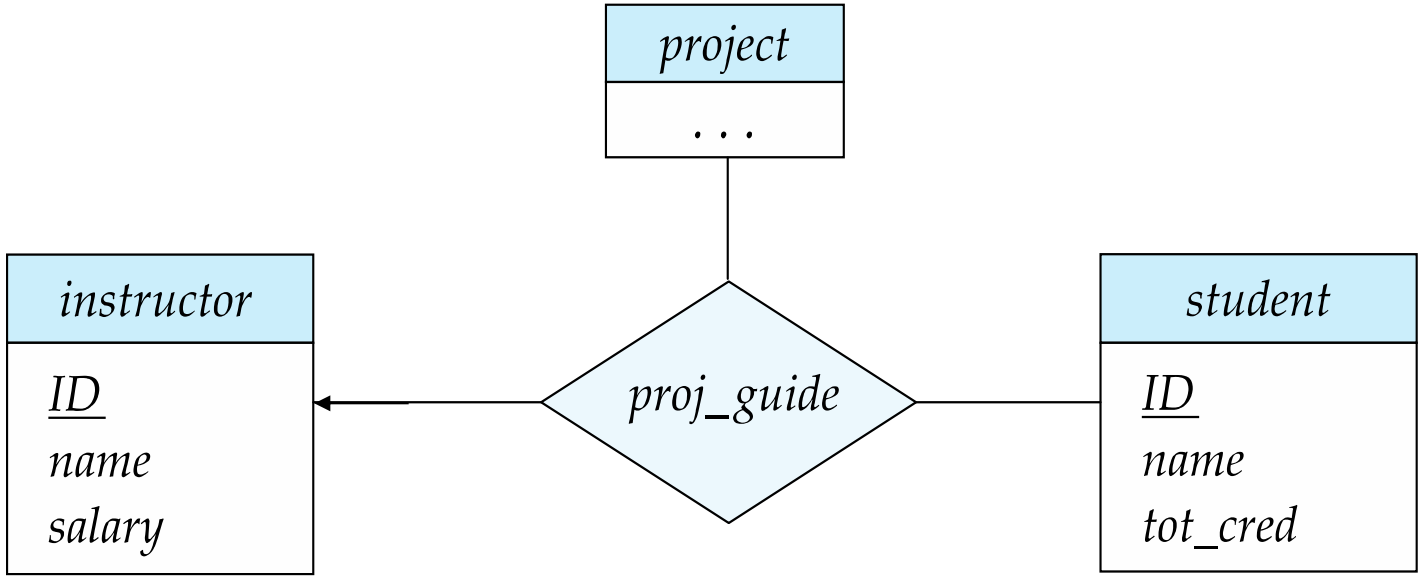
  - *Instructor* can advise 0 or more *students*. A *student* must have 1 *advisor (instructor)*; cannot have multiple *advisors(instructors).*

Cardinality Constraints on Ternary Relationship

We allow at most one arrow out of a ternary (or greater degree) relationship to indicate a cardinality constraint

For example, an arrow from *proj_guide* to *instructor* indicates each student has at most one guide for a project

To avoid confusion we outlaw(**禁止**) more than one arrow.



Primary Key

Primary keys provide a way to specify how entities and relations are distinguished. We will consider:

- Entity sets
- Relationship sets
- Weak entity sets

Primary key for Entity Sets

By definition, individual entities are distinct.

From database perspective, the differences among them must be expressed in terms of their attributes.

The values of the attribute values of an entity must be such that they can uniquely identify the entity.

> No two entities in an entity set are allowed to have exactly the same value for all attributes.

A key for an entity is a set of attributes that suffice to distinguish entities from each other

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.

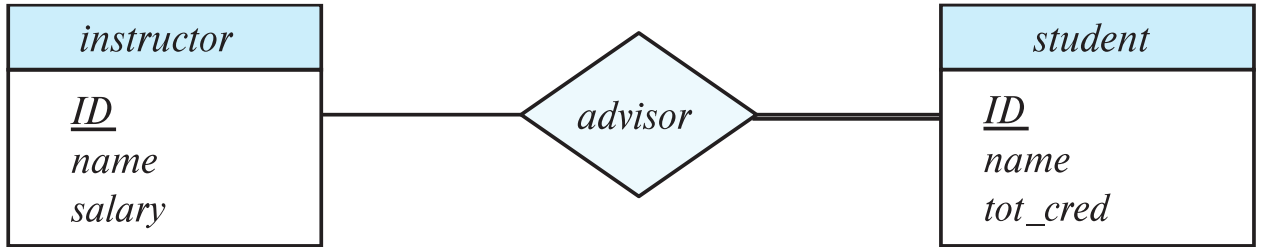
Let *R* be a relationship set involving entity sets E1, E2, .. En

The primary key for R is 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 the primary key of *R* also includes the attributes a1, a2, .., am

Example: relationship set “advisor”.

The primary key consists of *instructor.ID* and *student.ID*



The choice of the primary key for a relationship set depends on the mapping cardinality of the relationship set.

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.

Weak Entity Sets（弱实体集）

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（标识性实体集）**

> 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（标识性联系）** depicted using a double diamond

The **discriminator (分辨符**, *or partial key)* of a weak entity set is the set of attributes that distinguishes among all the entities of a weak entity set when the identifying entity they depend is known.

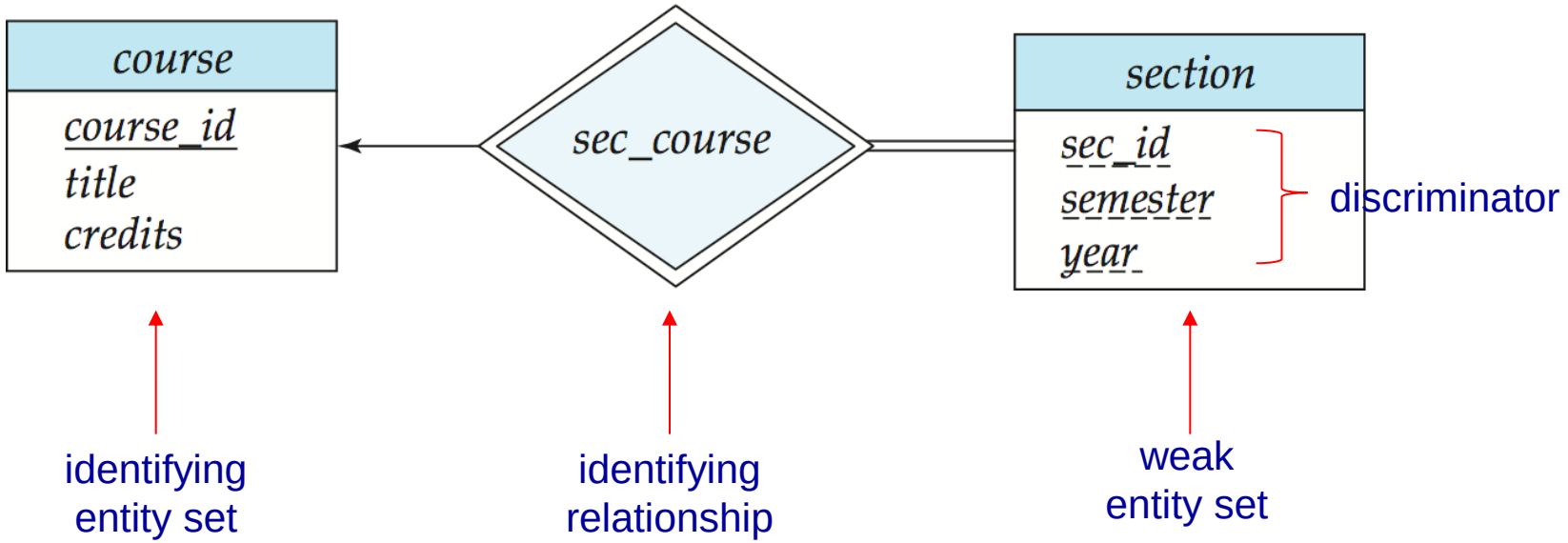
The primary key of a weak entity set is formed by the primary key of the strong entity set on which the weak entity set is existence dependent, plus the weak entity set's discriminator.

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.

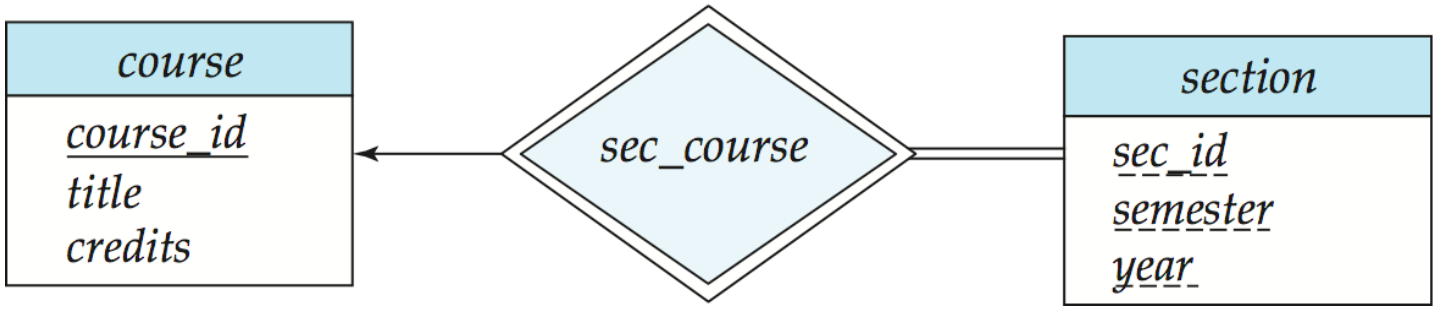
Primary key for *section* – (*course_id, sec_id, semester, year*)



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*



Redundant Attributes

Suppose we have entity sets:

student, with attributes: *ID*, *name, tot_cred*, *dept_name*

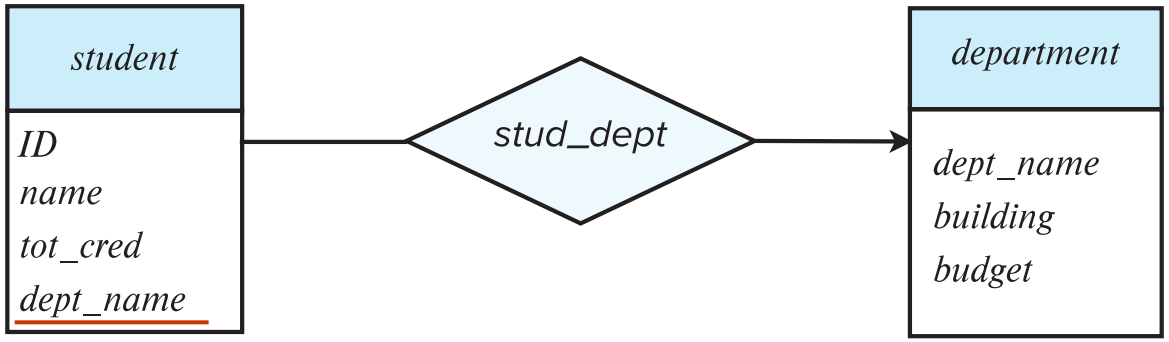
department, with attributes: *dept_name*, *building, budget*

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

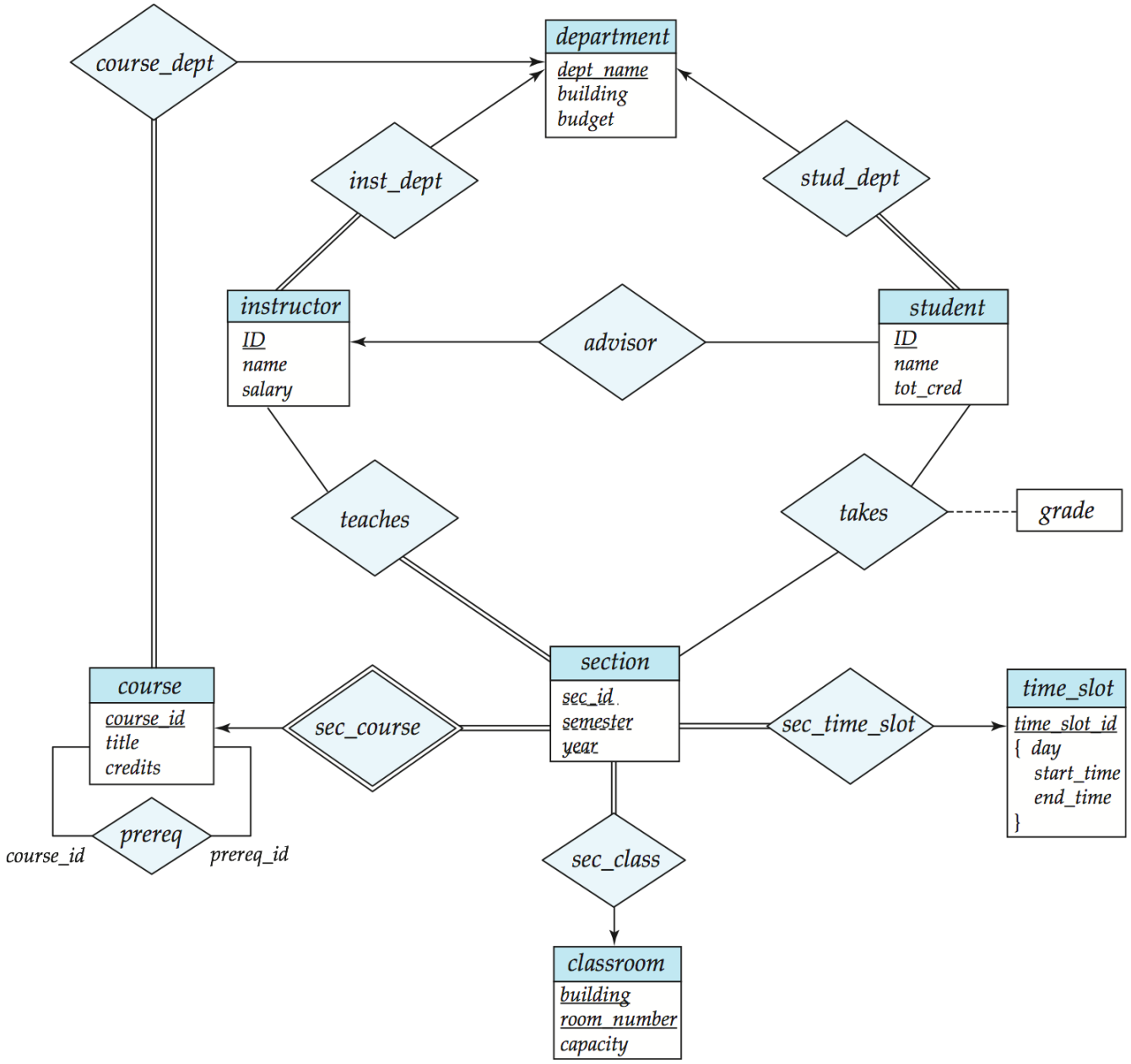
and needs to be removed.

BUT: when converting back to tables, in some cases the attribute gets reintroduced, as we will see later.



(a) Incorrect use of attribute

E-R Diagram for a University Enterprise



Reduction to Relational Schemas

Reduction to Relation Schemas

Entity sets and relationship sets can be expressed uniformly as *relation schemas* that represent the contents of the database.

A database which conforms to an E-R diagram can be represented by a collection of schemas.

For each entity set and relationship set there is a unique schema that is assigned the name of the corresponding entity set or relationship set.

Each schema has a number of columns (generally corresponding to attributes), which have unique names.

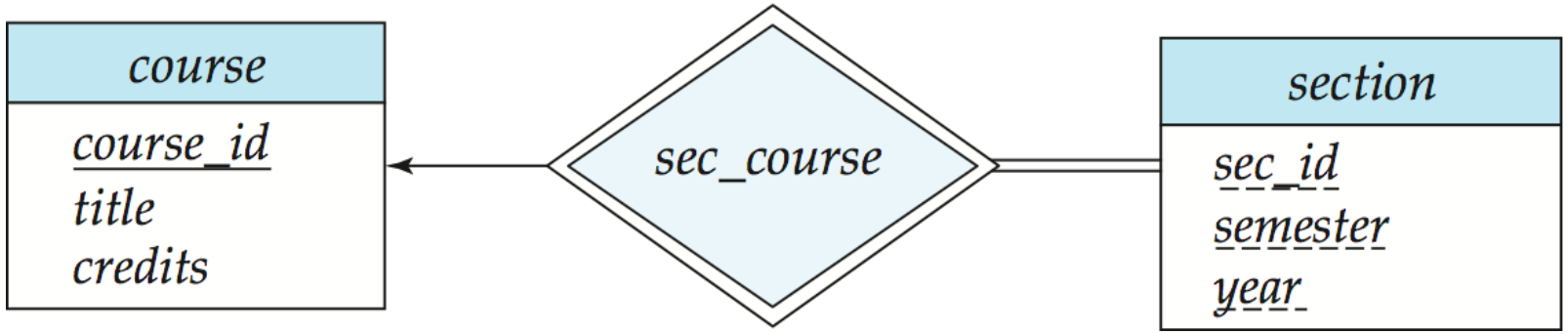
Representing Entity Sets With Simple Attributes

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

Primary key of the table is the union of the discriminator of the weak entity set and the primary key of the identifying strong entity set

section (course_id, sec_id, semester, year)

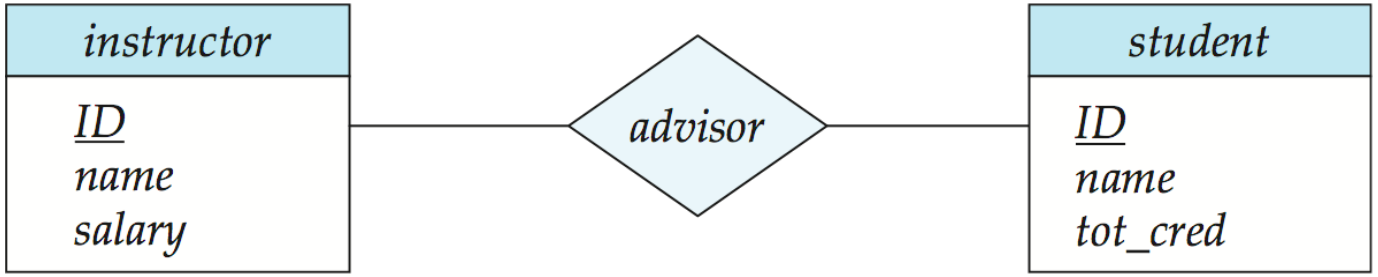


Representing Relationship Sets

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*

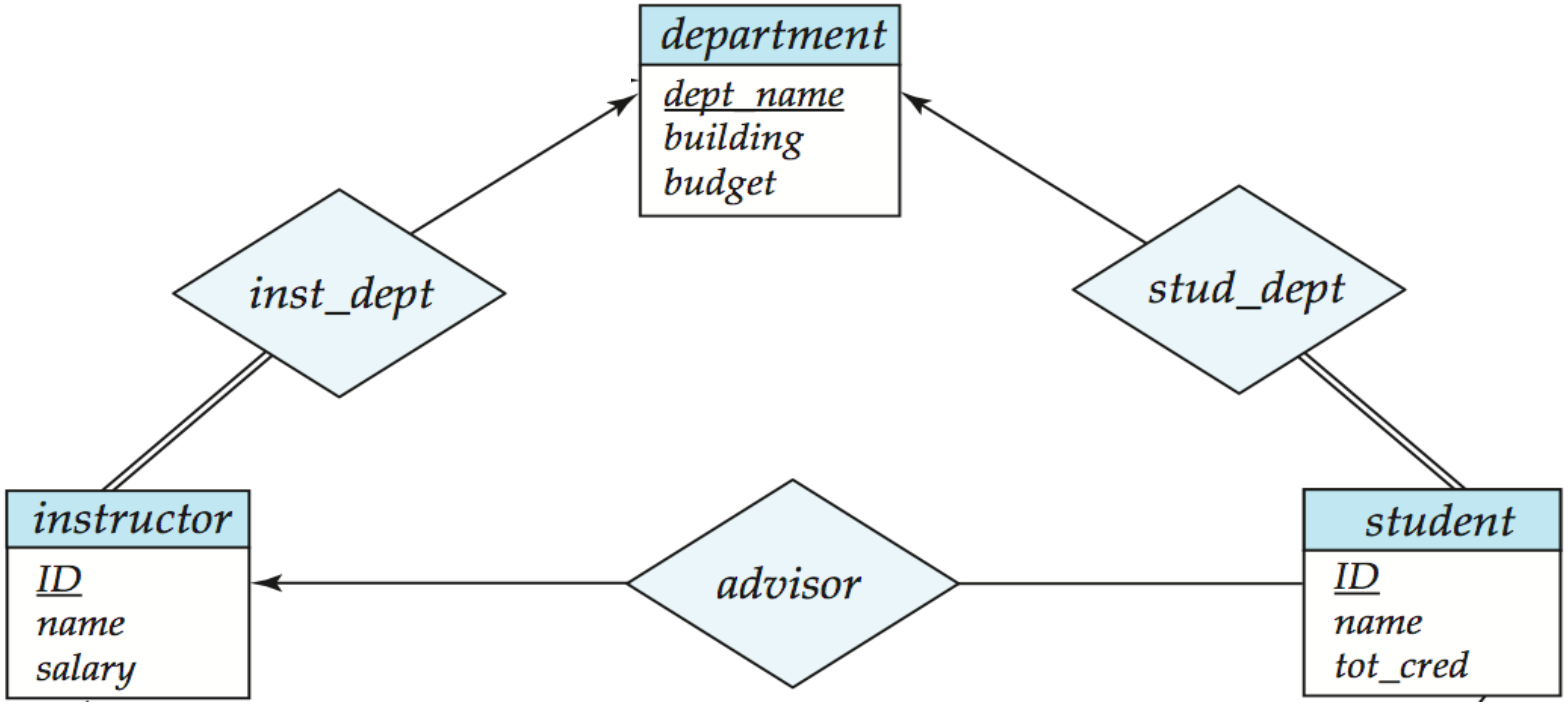
***advisor* = (*s_id, i_id*)**



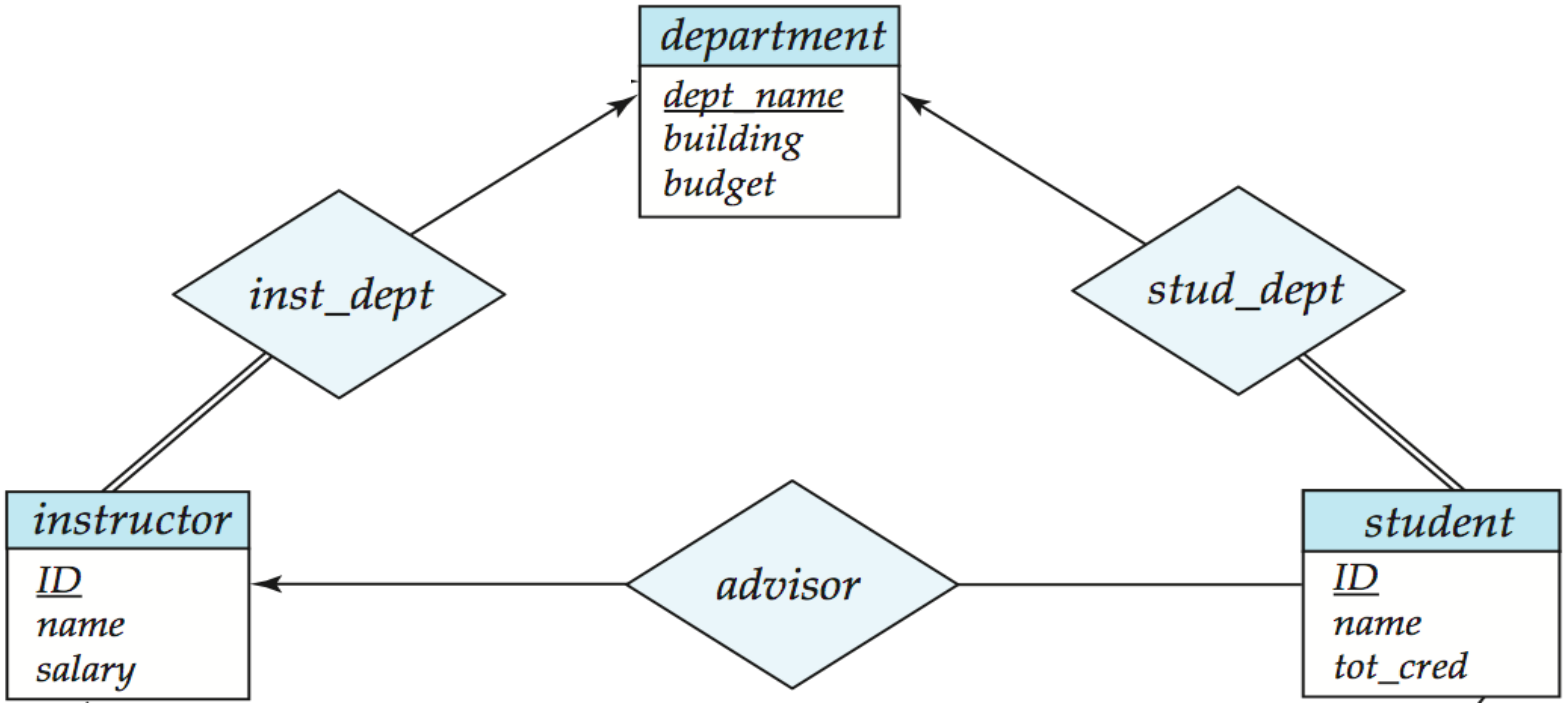
Redundancy of Schemas

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



department(<u>dept_name</u>, building, budget)
instructor(<u>ID</u>,name,salary)
inst_dept(<u>ID</u>, dept_name)

→

department(<u>dept_name</u>, building, budget)
instructor(<u>ID</u>, name, salary, **dept_name**)

Redundancy of Schemas (Cont.)

For one-to-one relationship sets, either side can be chosen to act as the “many” side

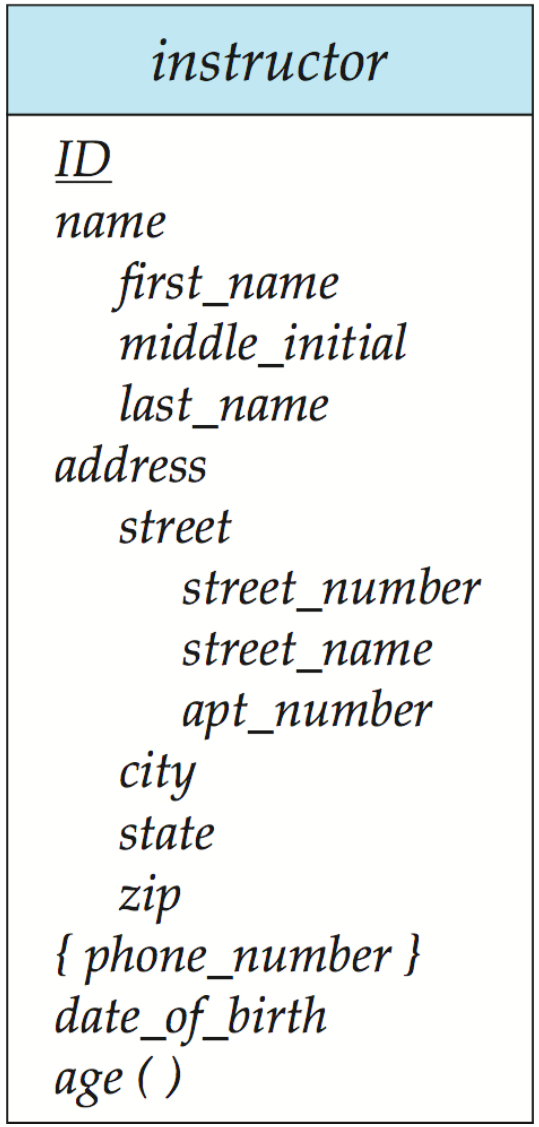
- That is, extra attribute can be added to either of the tables corresponding to the two entity sets

If participation is *partial* on the “many” side, replacing a schema by an extra attribute in the schema corresponding to the “many” side could result in null values

The schema corresponding to a relationship set linking a weak entity set to its identifying strong entity set is redundant.

- Example: The *section* schema already contains the attributes that would appear in the *sec_course* schema

Composite and Multivalued Attributes



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, age)

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)

  (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

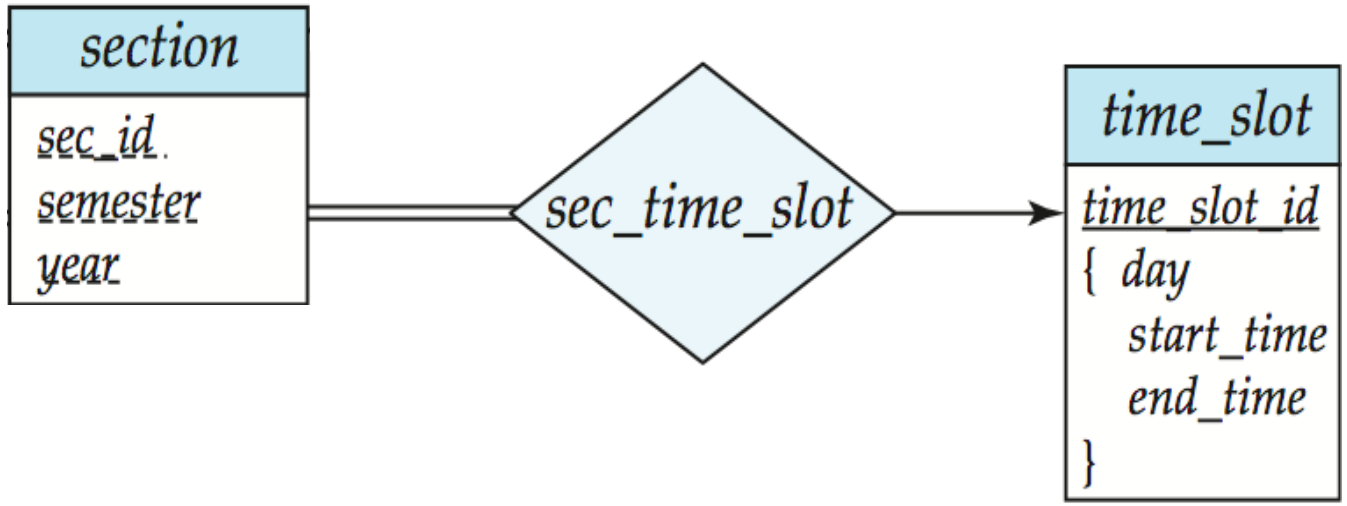
time_slot(time_slot_id)

time_slot_detail(time_slot_id, day, start_time, end_time)

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)

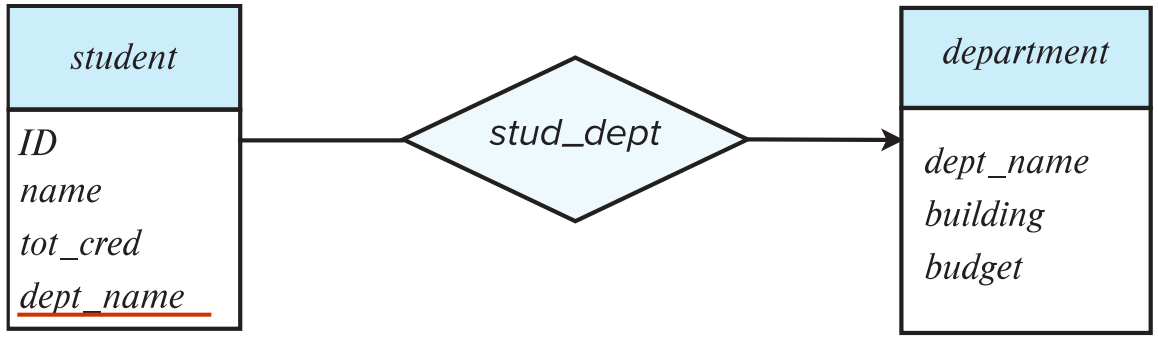
Caveat(警告): *time_slot* attribute of *section* (from *sec_time_slot*) cannot be a foreign key due to this optimization



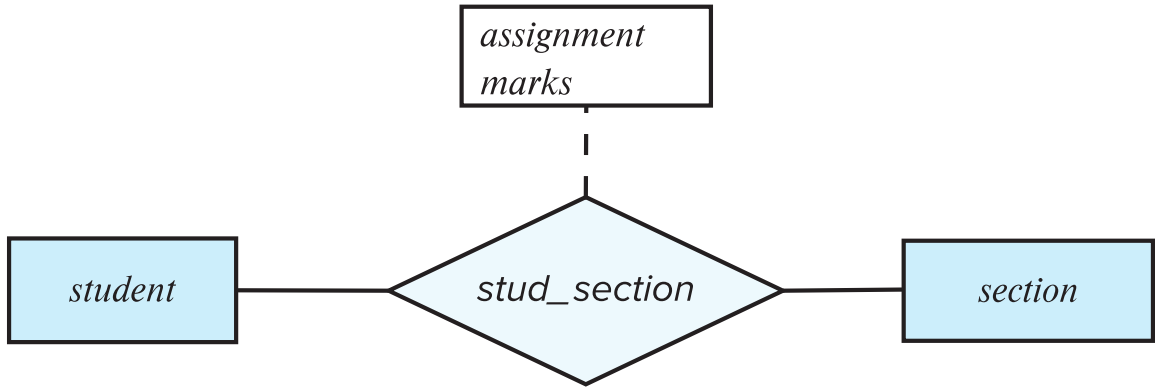
Design Issues

Common Mistakes in E-R Diagrams

- Example of erroneous E-R diagrams

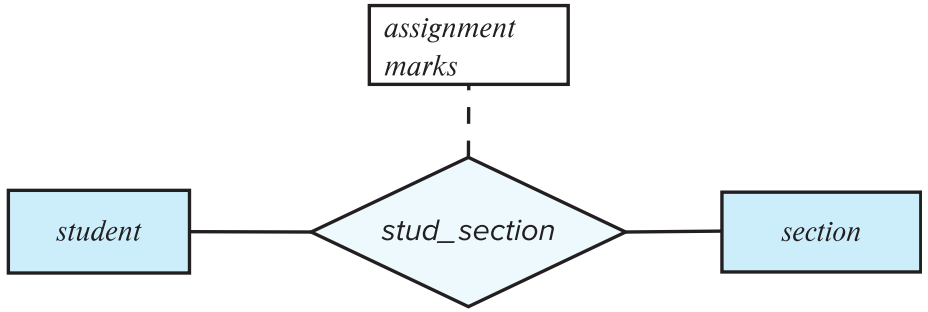


(a) Incorrect use of attribute

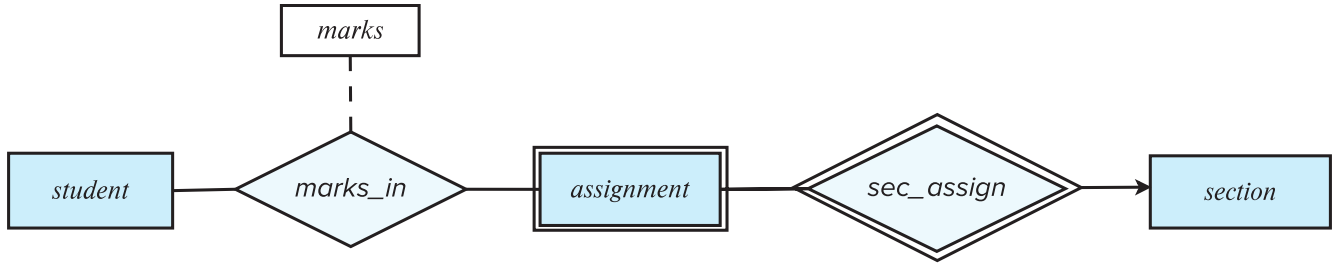


(b) Erroneous use of relationship attributes

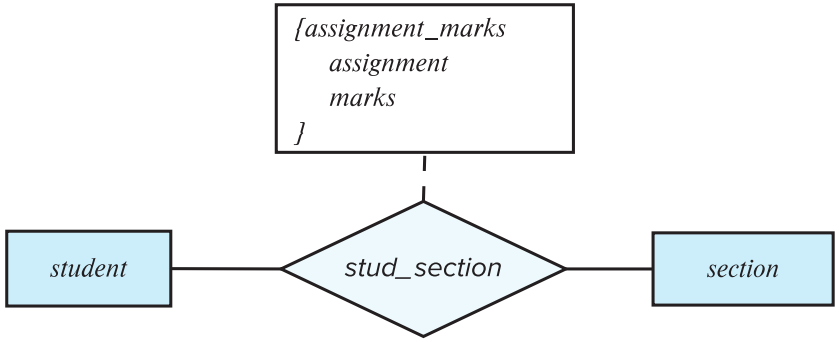
Common Mistakes in E-R Diagrams



(b) Erroneous use of relationship attributes



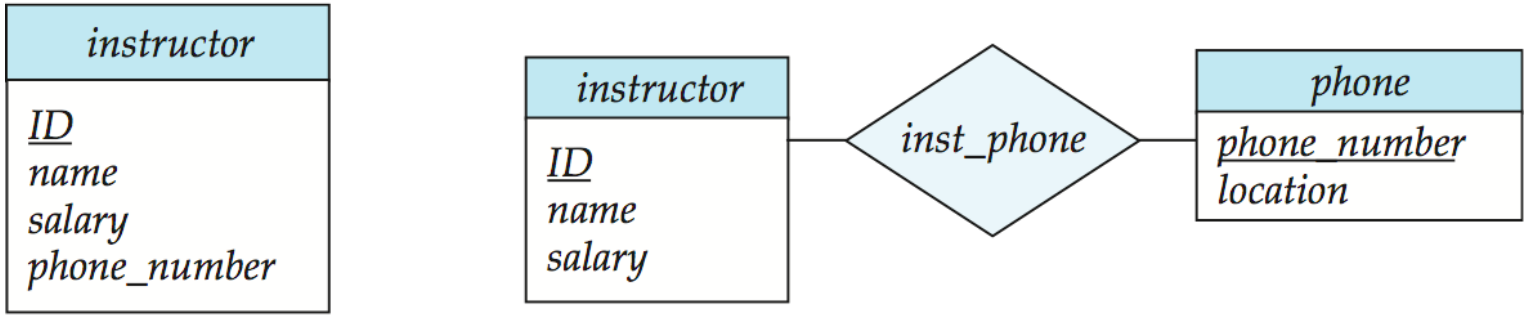
(c) Correct alternative to erroneous E-R diagram (b)



(d) Correct alternative to erroneous E-R diagram (b)

Design Issues

Use of entity sets vs. attributes

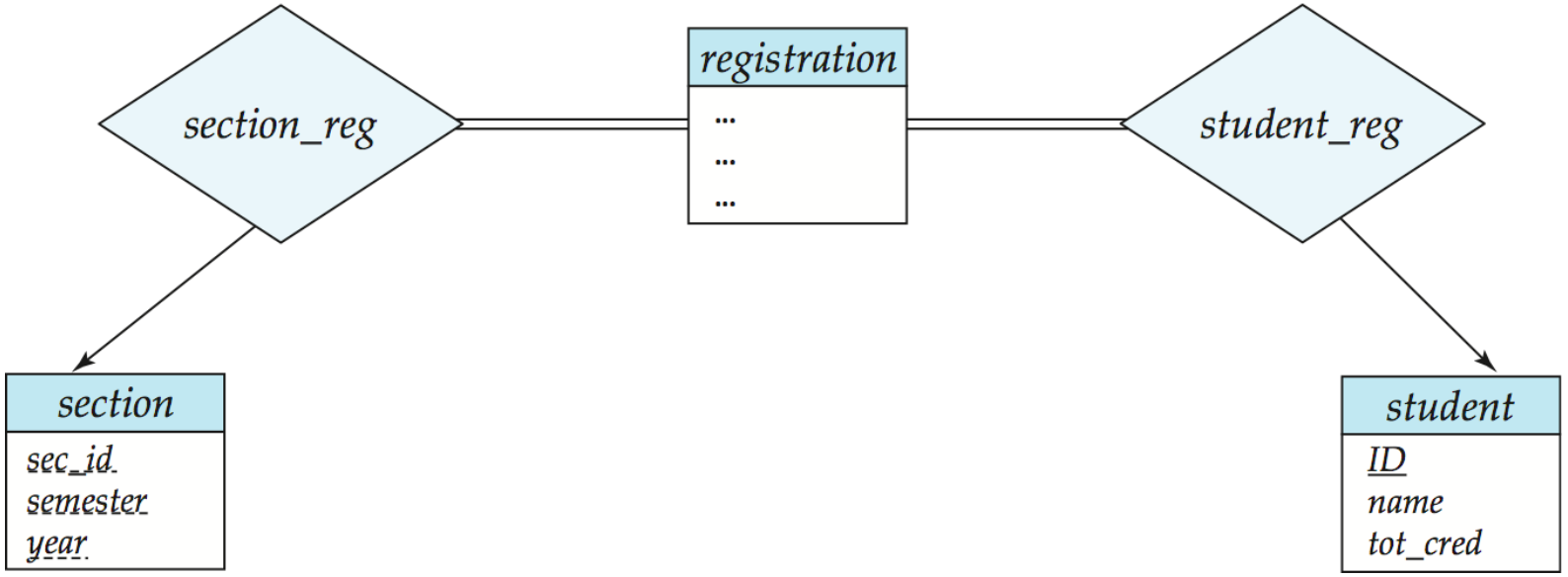


Use of phone as an entity allows extra information about phone numbers (plus multiple phone numbers)

Design Issues

Use of entity sets vs. relationship sets

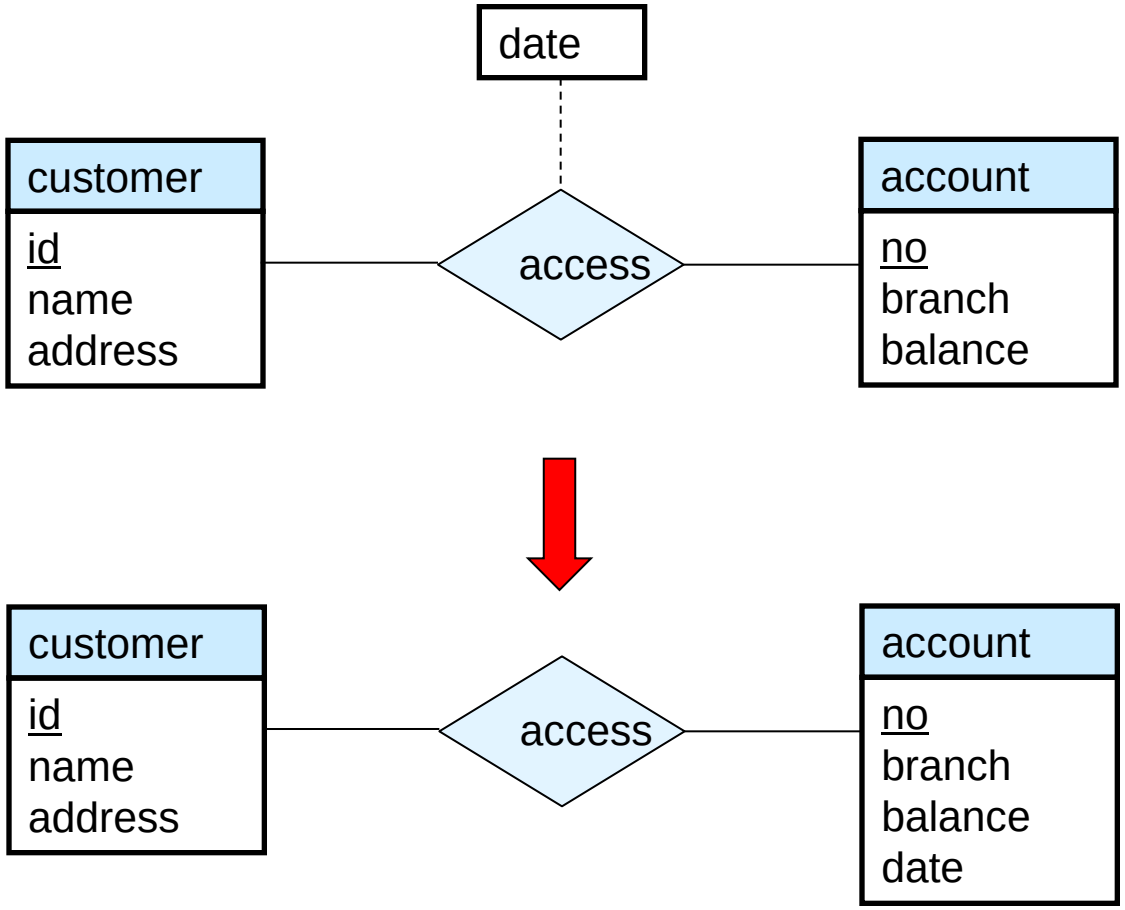
Possible guideline is to designate a relationship set to describe an action that occurs between entities



Design Issues

Placement of relationship attributes

e.g., attribute *date* as attribute of ***access*** or as attribute of ***account***



Binary Vs. Non-Binary Relationships

Binary versus n-ary relationship sets

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

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:

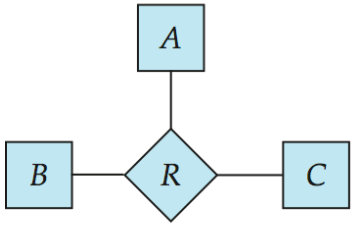
1. R_A, relating E and A
2. R_B, relating E and B
3. R_C, relating E and C

Create a special identifying attribute for E

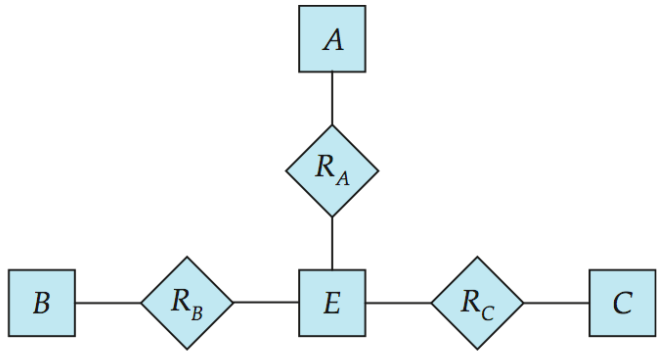
Add any attributes of R to E

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



(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*

- 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

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 or binary relationships.

The use of a strong or weak entity set.

Extended ER Features

Extended E-R Features: Specialization/Generalization

Specialization(特化)

Top-down design process; we designate subgroupings within an entity set that are distinctive from other entities in the set.

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.

Generalization（概化）

A bottom-up design process – combine a number of entity sets that share the same features into a higher-level entity set.

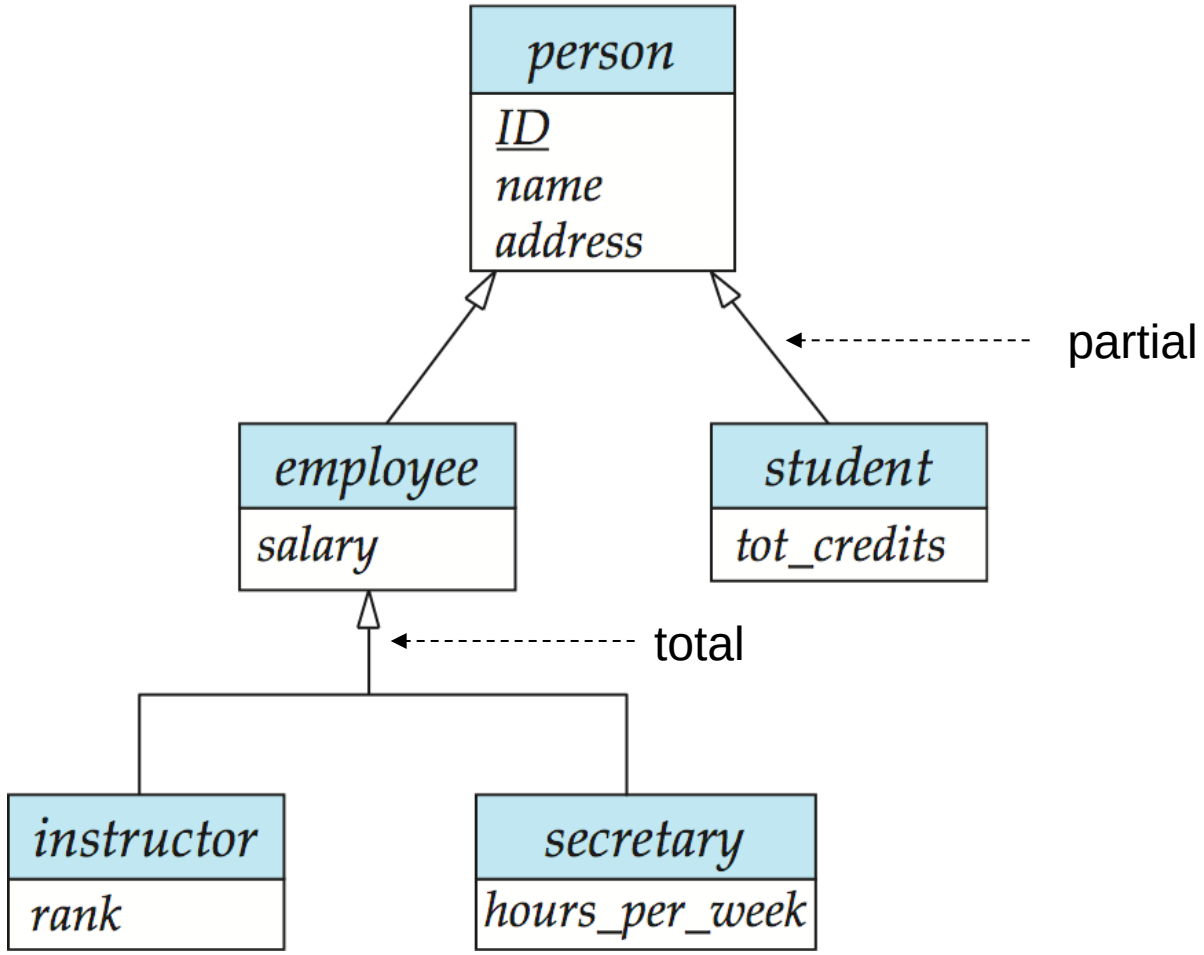
Specialization and generalization are simple inversions of each other; they are represented in an E-R diagram in the same way.

The terms specialization and generalization are used interchangeably.

Specialization Example

overlapping

disjoint



Design Constraints on a Specialization/Generalization

Constraint on whether or not entities may belong to more than one lower-level entity set within a single generalization.

Disjoint(不相交)

- an entity can belong to only one lower-level entity set
- Noted in E-R diagram by having multiple lower-level entity sets link to the same triangle

Overlapping（重叠）

- an entity can belong to more than one lower-level entity set

Design Constraints on a Specialization/Generalization (Cont.)

Completeness constraint（完全性約束）-- 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（全部）**: an entity must belong to one of the lower-level entity sets
- **Partial（部分）**: an entity need not belong to one of the lower-level entity sets

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

- 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

Representing Specialization as Schemas

Method 3:

Using single schema for all super /suber entity sets , with a type attribute to differentnate entities.

schema	attributes
person	*ID, name, street, city, **person_type**, tot_cred, salary)*

UML

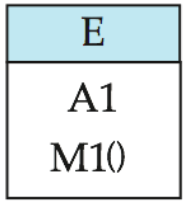
UML: Unified Modeling Language

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.

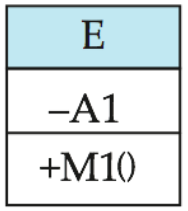
ER vs. UML Class Diagrams

ER Diagram Notation

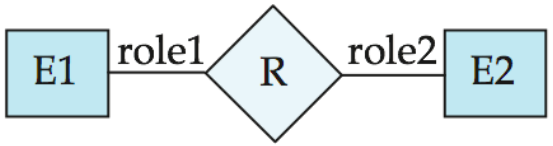


entity with attributes (simple, composite, multivalued, derived)

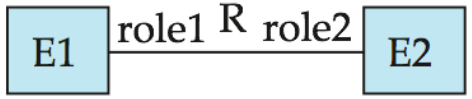
Equivalent in UML

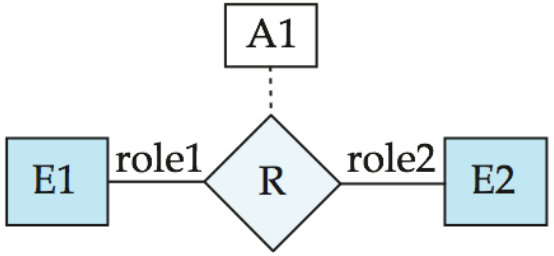


class with simple attributes and methods (attribute prefixes: + = public, – = private, # = protected)

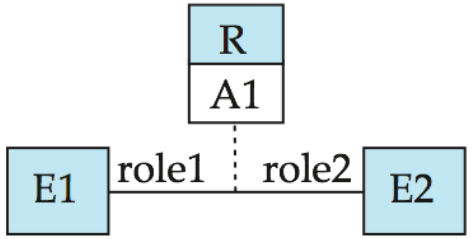


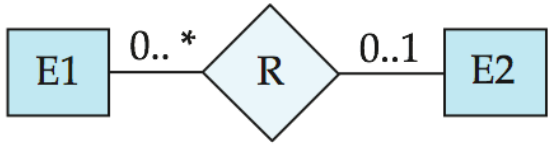
binary relationship



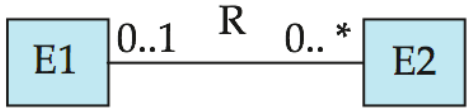


relationship attributes





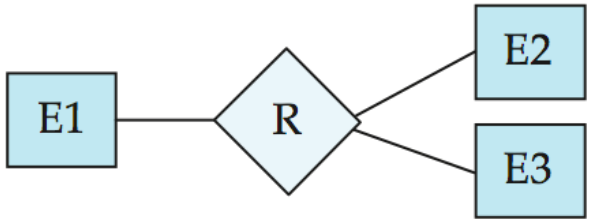
cardinality constraints



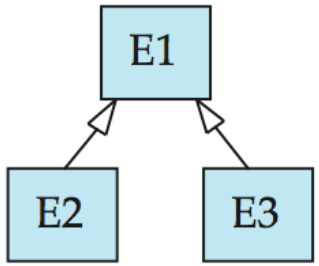
*Note reversal of position in cardinality constraint depiction

ER vs. UML Class Diagrams

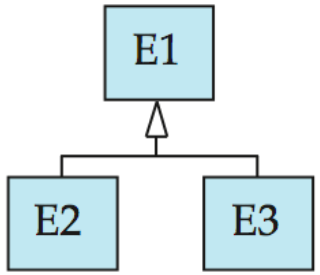
ER Diagram Notation



n-ary relationships

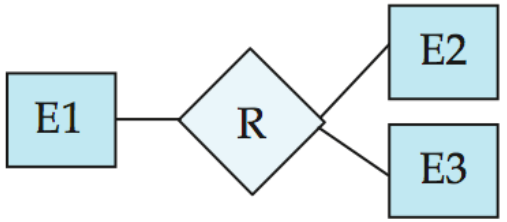


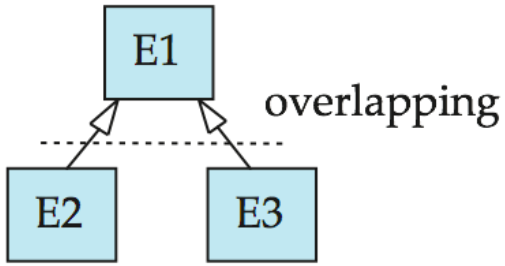
overlapping generalization



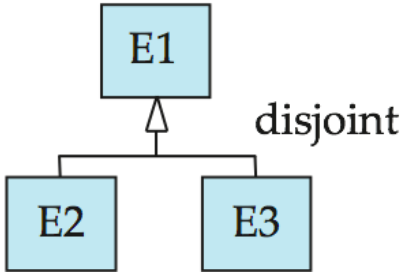
disjoint generalization

Equivalent in UML





overlapping



disjoint

*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 6