Modeling the World

ECE 356
University of Waterloo
Dr. Paul A.S. Ward

Acknowledgment: slides derived from Dr. Wojciech Golab

based on materials provided by
Silberschatz, Korth, and Sudarshan, copyright 2010
(source: www.db-book.com)

Learning Outcomes

- Building blocks of the entity relationship (ER) model:
  - entities and entity sets
  - relationships and relationship sets
  - constraints: cardinality, participation
  - primary keys
- ER diagrams

- Textbook sections (6th ed.): Chapter 7

Data in the Real World™

- A database organizes information about a particular project.
  - potentially large collection of interrelated data
  - set of programs to access the data
  - environment that is **convenient**, **efficient** and **reliable**
- Example applications:
  - Traditional IT
    - Banking – transactions
    - Airlines – reservations, schedules
    - Universities – registration, grades
    - Sales – customers, products, purchases
    - Online retailers – order tracking, customized recommendations
    - Manufacturing – production, inventory, orders, supply chain
    - Human resources – employee records, salaries, tax deductions
  - Sensor integration
    - Weather, electricity usage, location
  - Software status
    - Applications/versions installed

Data modeling

- To a first approximation, data can be modeled as:
  - a collection of entities (things)
  - relationship among entities.
- An **entity** is an object that exists and is distinguishable from other objects.
  - *e.g.*, a specific person, a specific telephone, a specific office
- Entities have properties (that we call **attributes**)
  - *e.g.*, a person has a name; a telephone has a location
- An **entity set** is a set of entities of the same type that share the same properties.
  - example: a set of UW ECE students

Entity Sets UW student and Telephone

- UW Student:
  - StudentID
  - Last Name
  - First Name and Initials
  - Degree Program
  - Term Admitted
  - UserID
  - E-Mail Address
  - …
- UW Telephone
  - Extension
  - Type
  - Office
  - …
  -
- Question: Should a telephone be an entity or an attribute? Why?

Relationship Sets

- A **relationship** is an association among two or more entities

  For example:

  Faculty Member 12345 *advises* Graduate Student 67890

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

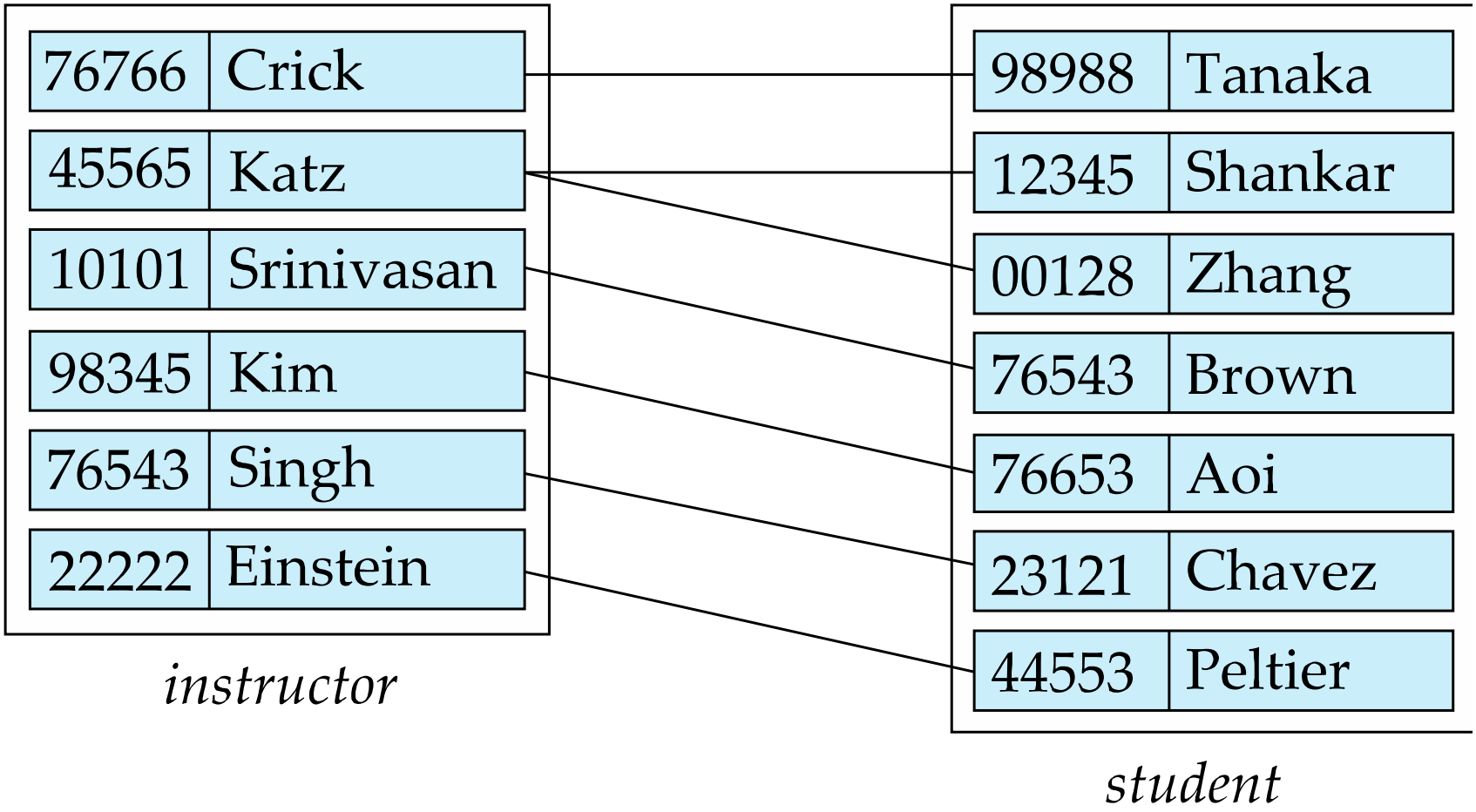
  $$\{(e_1, e_2, \ldots e_n) \mid e_1 \in E_1, e_2 \in E_2, \ldots, e_n \in E_n\}$$

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

  - Example:

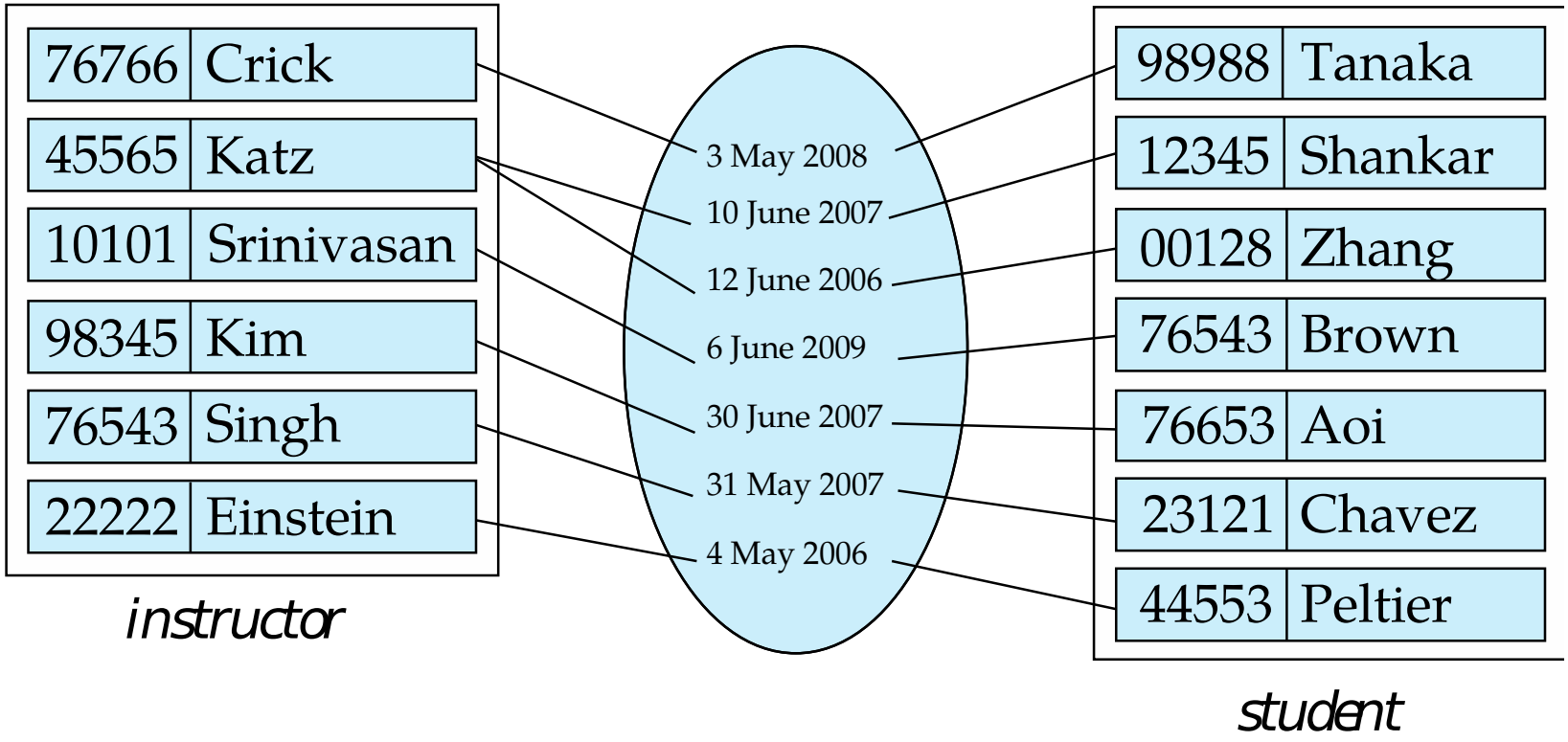
    $$(12345, 67890) \in advisor$$

Relationship Set *advisor*



Relationship Sets (Cont.)

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



Degree of a Relationship Set

- **binary relationship**
  - involves two entity sets (or degree two)
  - binary relationships are the simplest and work well in practice
  - relationships between more than two entity sets are less common, but exist
    - Example: *students* work on research *projects* under the guidance of an *instructor*.
  - Note that this is not the same thing as a binary relationship with an attribute
    - Think: what is the difference between these two cases?

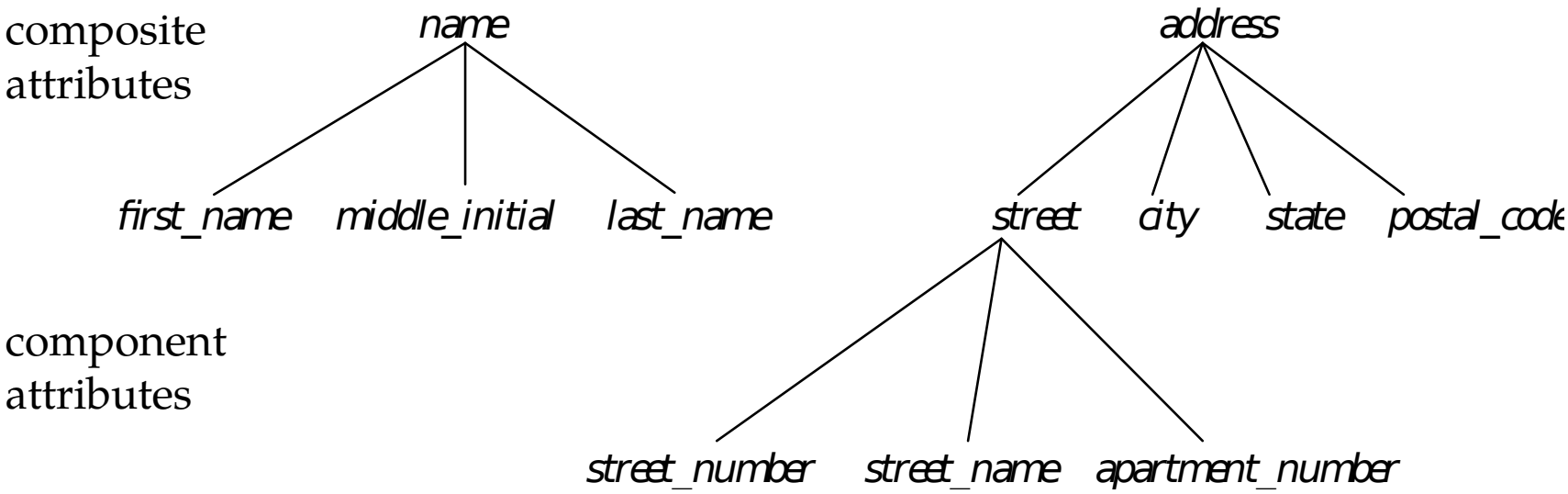
Attributes

- An entity is represented by a set of attributes, which are 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 an attribute
  - example: salary is a non-negative integer
- Attribute types:
  - **simple** versus **composite**
    - example of simple attribute: *GPA, age-in-years*
    - example of composite attribute: *name, address, birthdate, ...*
  - **single-valued** versus **multivalued**
    - example of single-valued attribute: *birthdate, course_ID*
    - example of multivalued attribute: *address, phone_numbers*

Composite Attributes

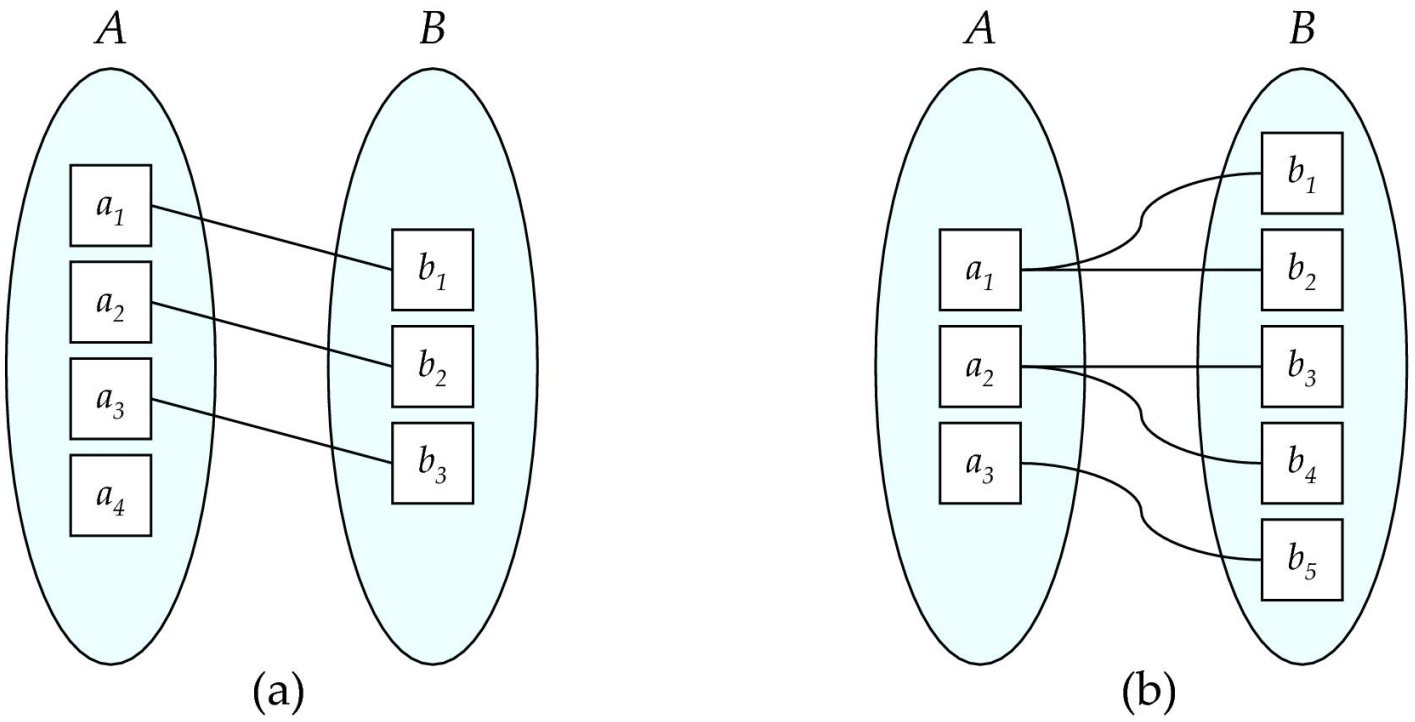
- A **composite attribute** comprises a collection of **component attributes**. Such attributes may be composed recursively, as shown below (*address* contains *street*, which contains *street_name*).



Mapping Cardinality Constraints

- **Cardinality constraints** express the number of entities to which another entity can be associated via a relationship set.
- The constraints reflect the general structure of the data rather than the structure in a specific data set.
  - example: *advisor* relationship (from instructor to student) is generally 1:N or N:N even though at a given time each instructor may have only one student
- They are most useful in describing binary relationship sets.
  - More precisely, there is NO standard method for expressing ternary (and beyond) relationship cardinality constraints
    - Why not? Why is this a hard problem?
- 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:N)

Mapping Cardinalities

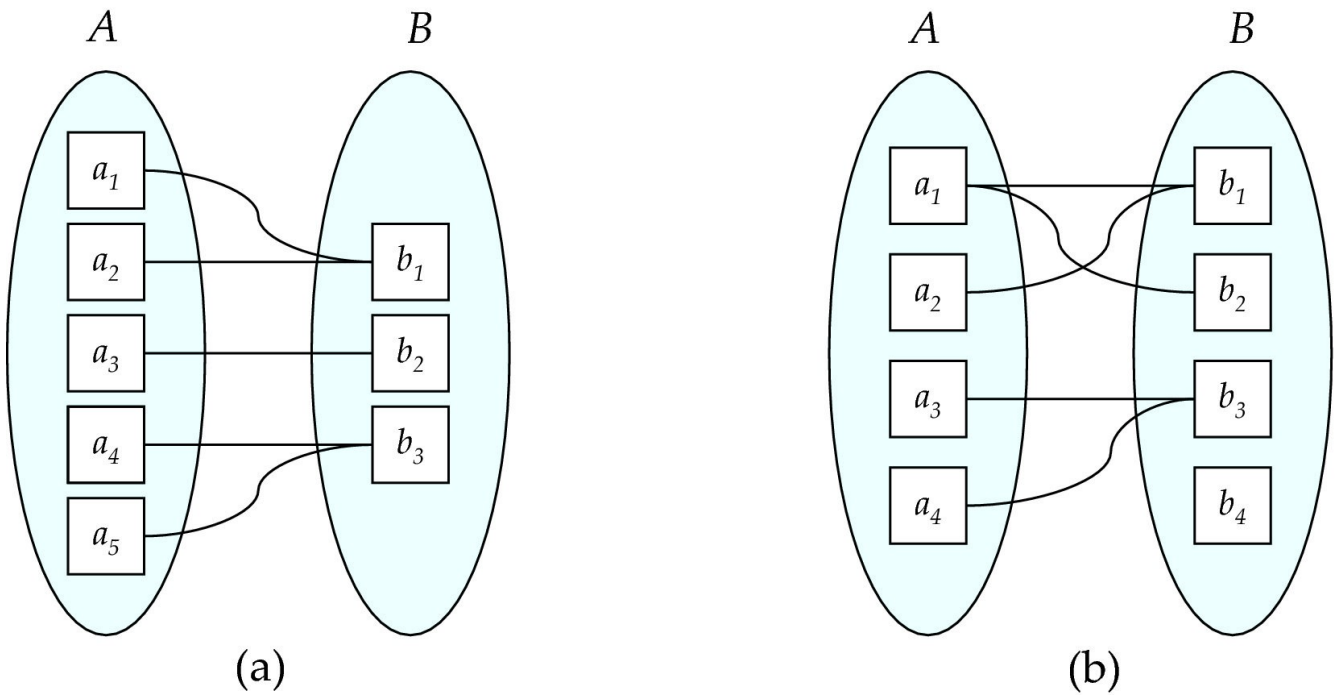


one to one

one to many

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

Mapping Cardinalities



many to one

many to many

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

Keys

- 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*

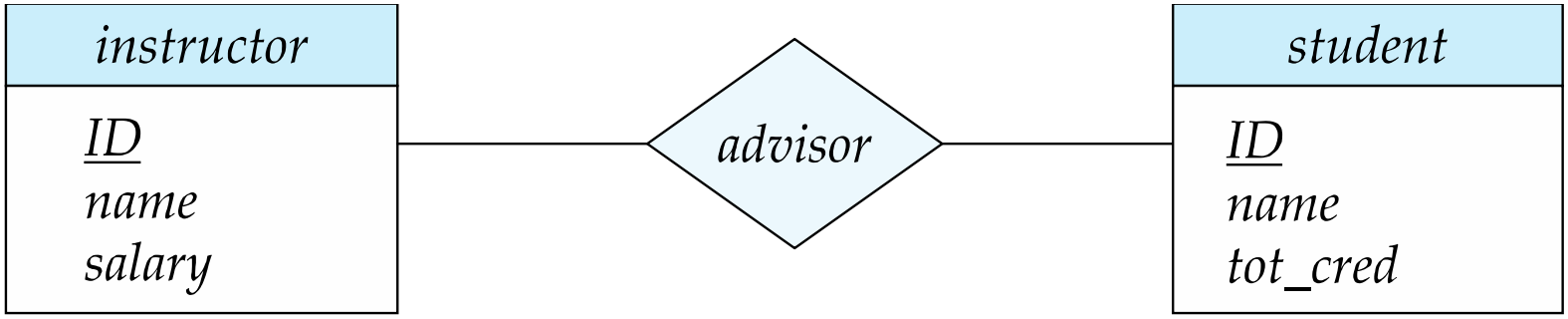
  Note: “minimal” in this context means that if we take out any attribute, we end up with something that is no longer a super key.

- Although several candidate keys may exist, one of the candidate keys is selected by the database designer to be the **primary key**.

Keys for Relationship Sets

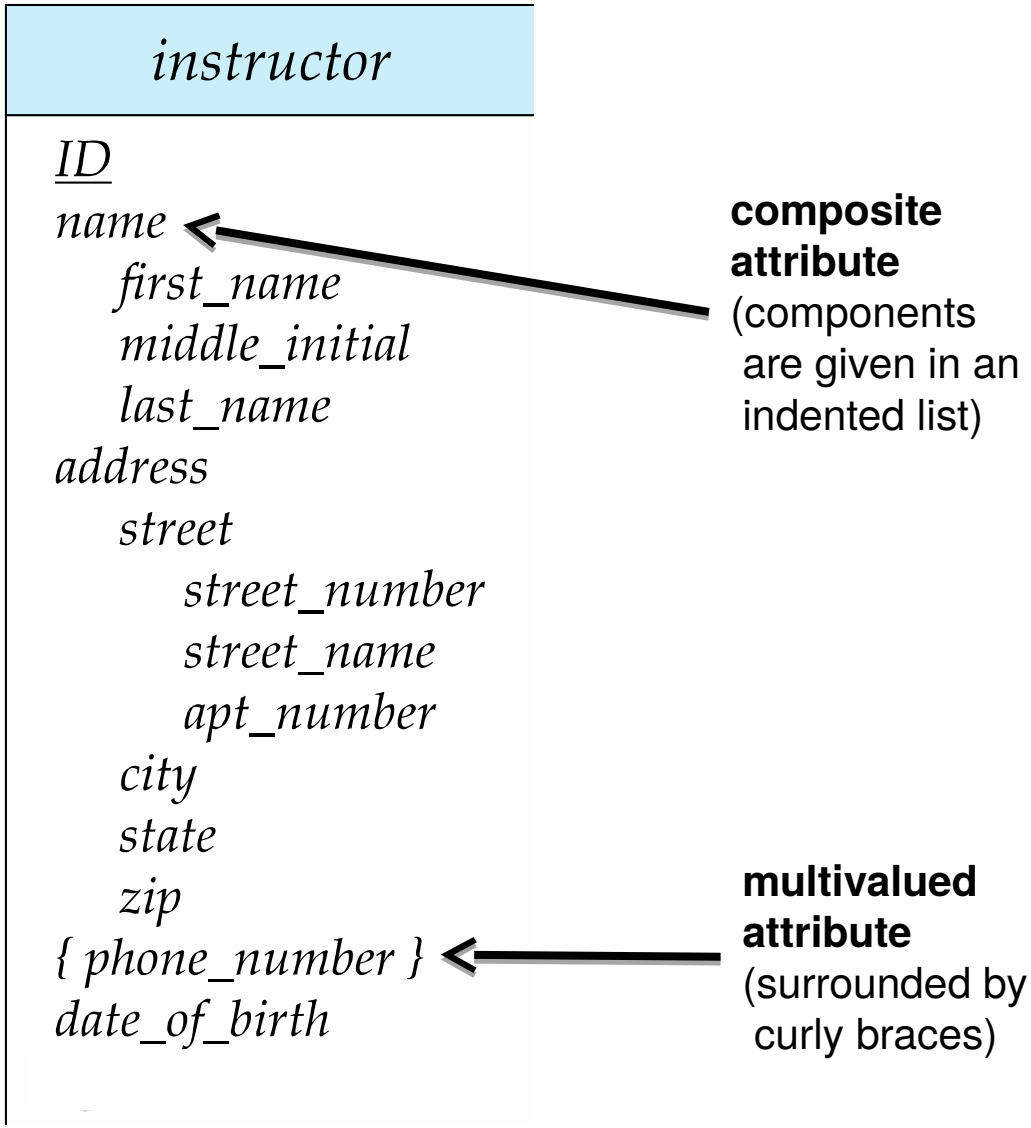
- The combination of primary keys of the participating entity sets forms a super key of a relationship set.
  - (*student.ID, instructor.ID*) is the super key of *advisor*
  - Note 1: **dotted notation** (*e.g.*, *student*.*ID* vs. *instructor*.*ID*) is used to resolve attribute name collisions
  - Note 2: **a pair of entities can participate at most once in a *particular* binary relationship set.**
    - Example: if we wish to track multiple meeting times between a student and his/her advisor, we cannot assume a distinct relationship for each meeting. Instead, we can add a multivalued descriptive attribute to the relationship set.
- Must consider the mapping cardinality of the relationship set when deciding what are the candidate keys.
- Need to consider semantics of relationship set in selecting the primary key if there is more than one candidate key.

E-R Diagrams

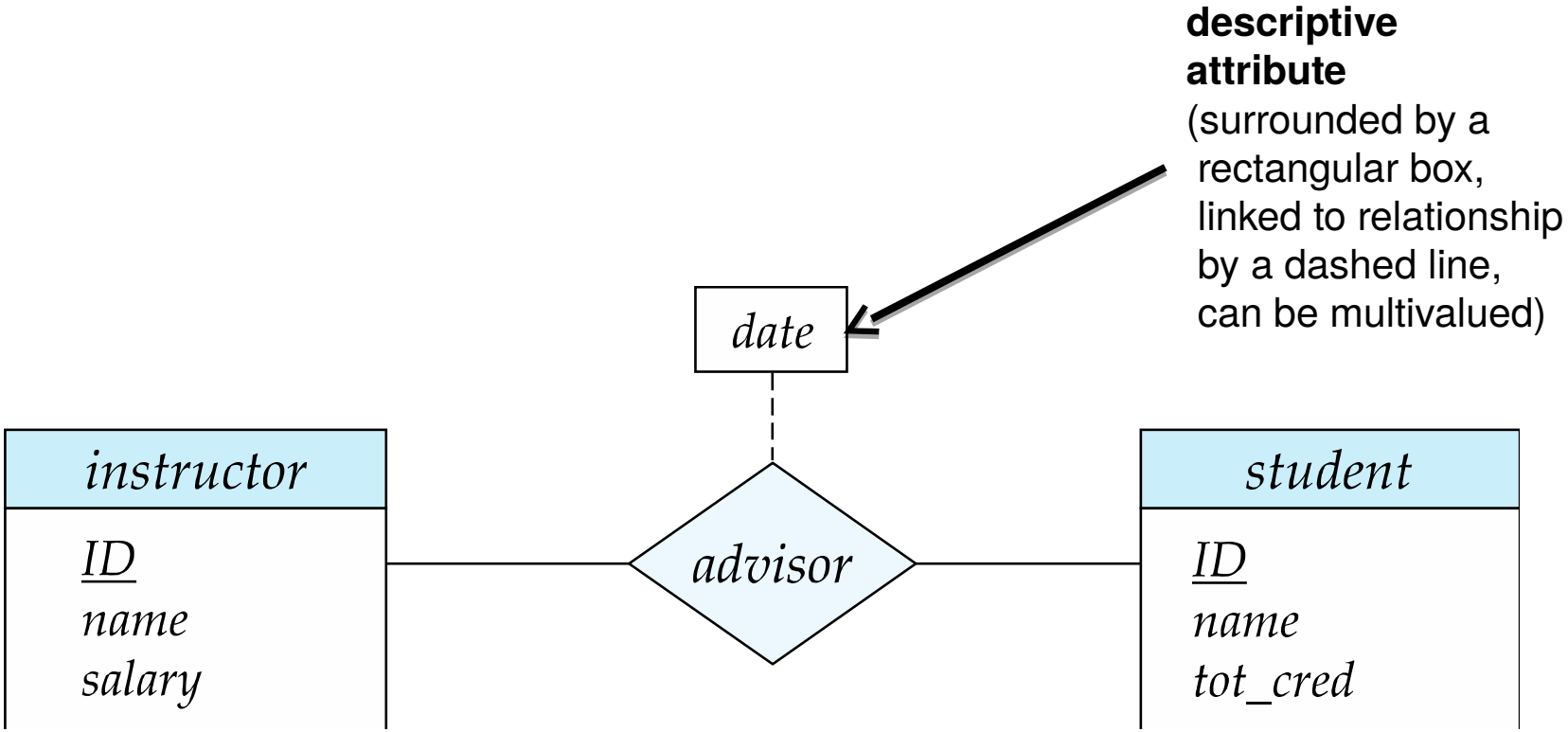


- Rectangles represent entity sets.
- Diamonds represent relationship sets.
- Attributes are listed inside entity rectangle.
- Underlined attributes are part of the primary key.

Entity Set with Composite & Multivalued Attributes

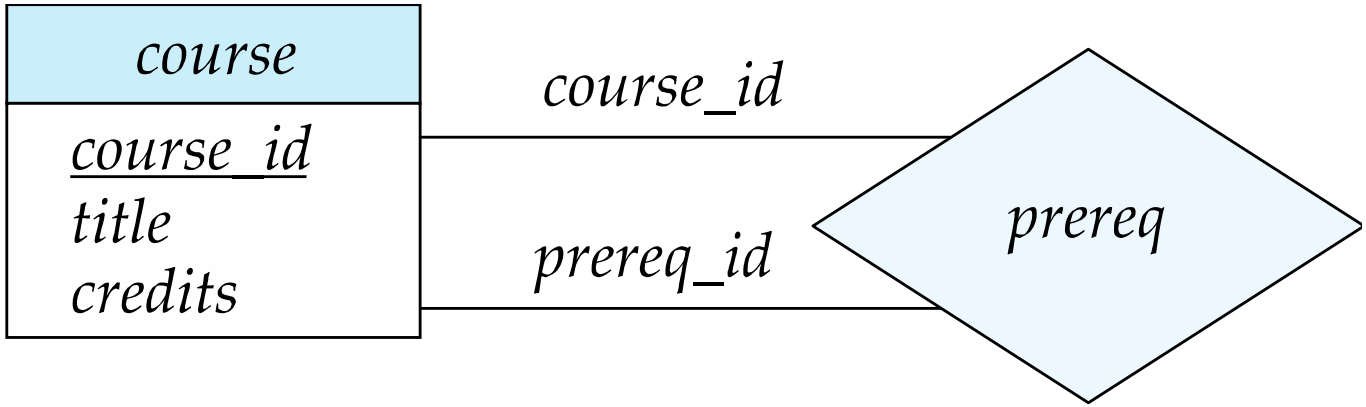


Relationship Set with Descriptive Attributes



Roles

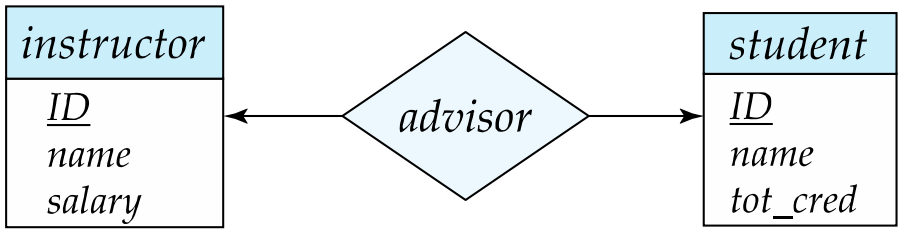
- The entity sets of a relationship need not be distinct.
- Each occurrence of an entity set plays a “role” in the relationship.
- The labels *“course_id”* and *“prereq_id”* below are called **roles**.



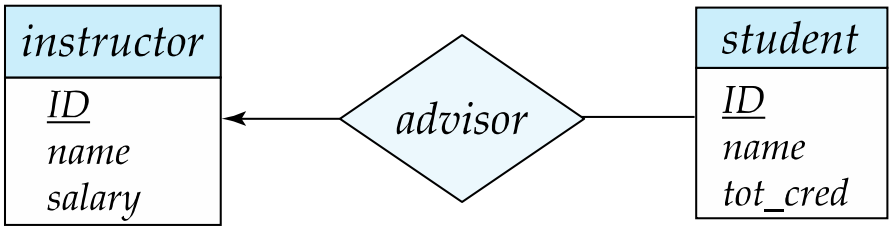
(*course_id.course_id, prereq_id.course_id*) is the super key of *prereq*

Cardinality Constraints

- 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.

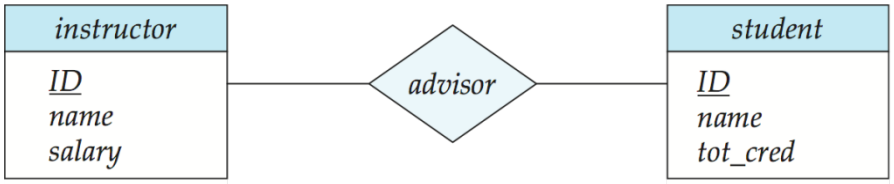


one to one
- at most one instructor per student
- at most one student per instructor



one to many
- at most one instructor per student

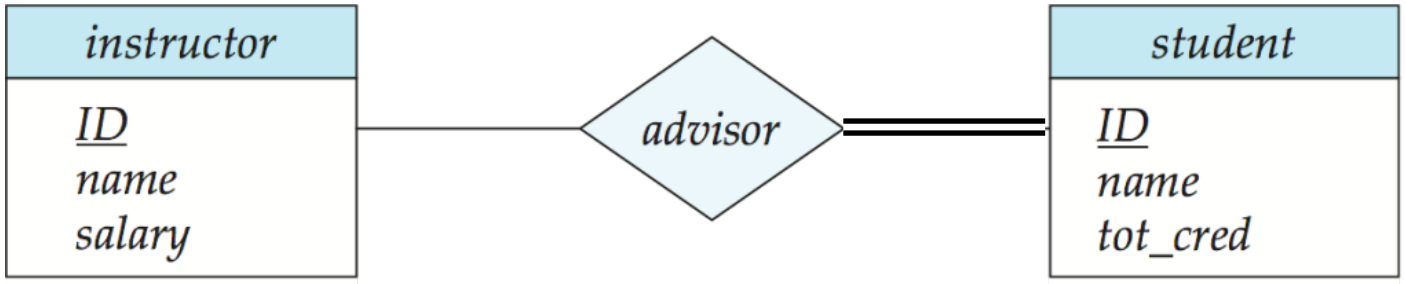
many to one
- at most one student per instructor



many to many

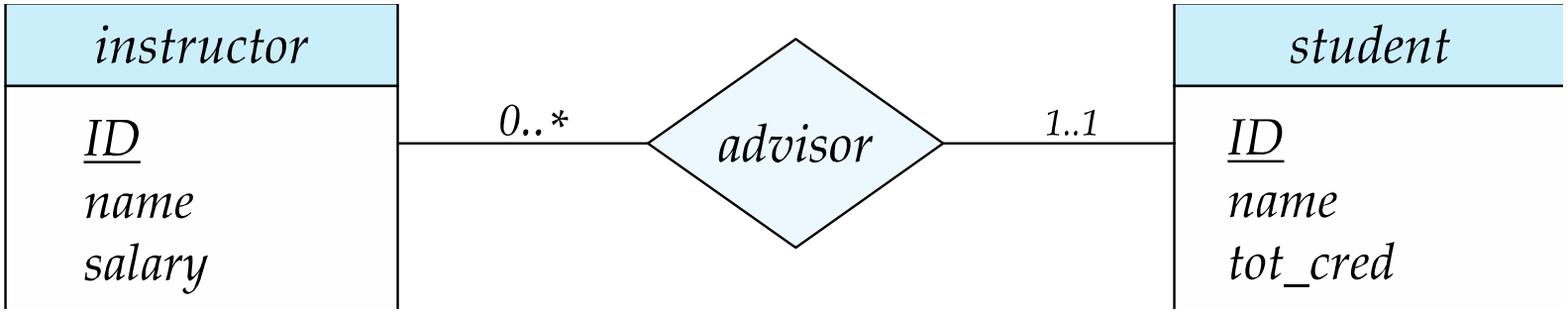
Participation of an Entity Set in a Relationship Set

- **Total participation** (indicated by double line): every entity in the entity set participates in at least one relationship in the relationship set.
- Example:
  - At Waterloo:
    - a graduate student may have multiple advisors
    - An advisor may have multiple graduate students
    - BUT, every graduate student must have *at least one* advisor.
    - However, an instructor is not required to advise any graduate students.



Preferred Notation for Cardinality Limits

- **Cardinality limits** can also express participation constraints.

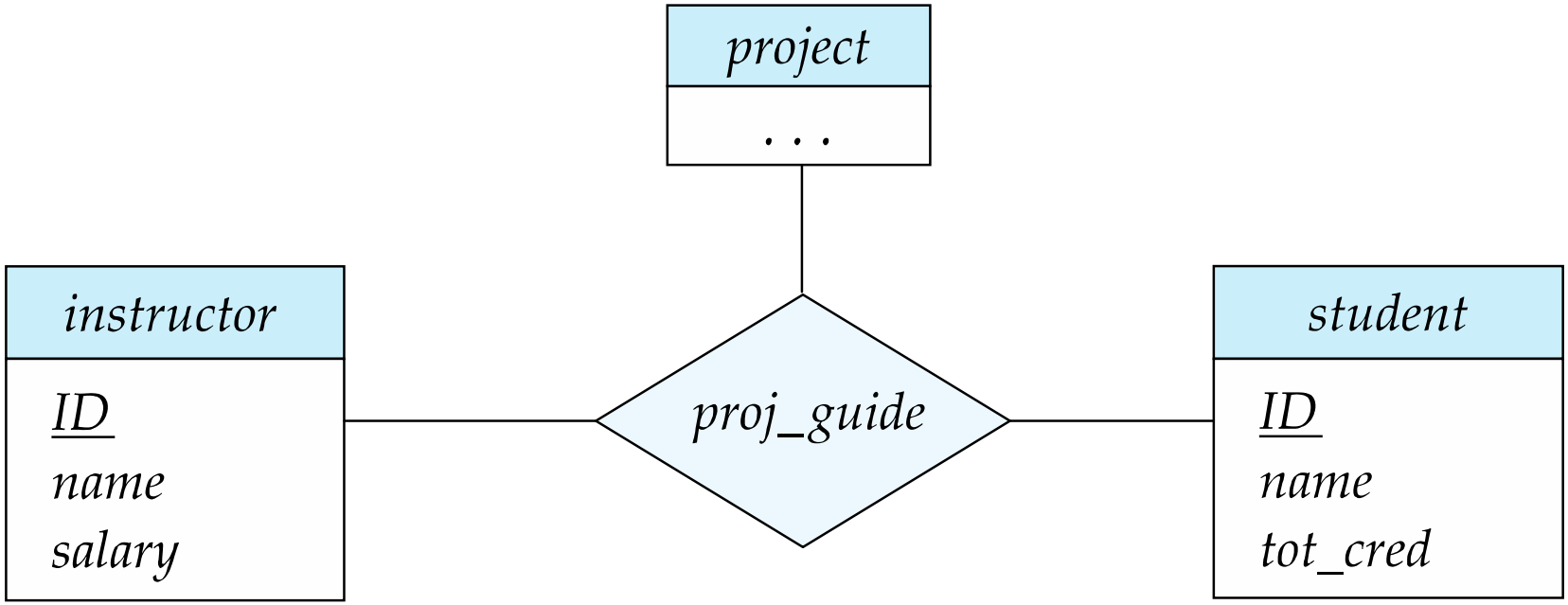


(Interpretation: every student must be advised by *exactly* one instructor, and an instructor may advise any number of students including possibly zero.)

is equivalent to

Why is the x..y notation preferred?

E-R Diagram with a Ternary Relationship



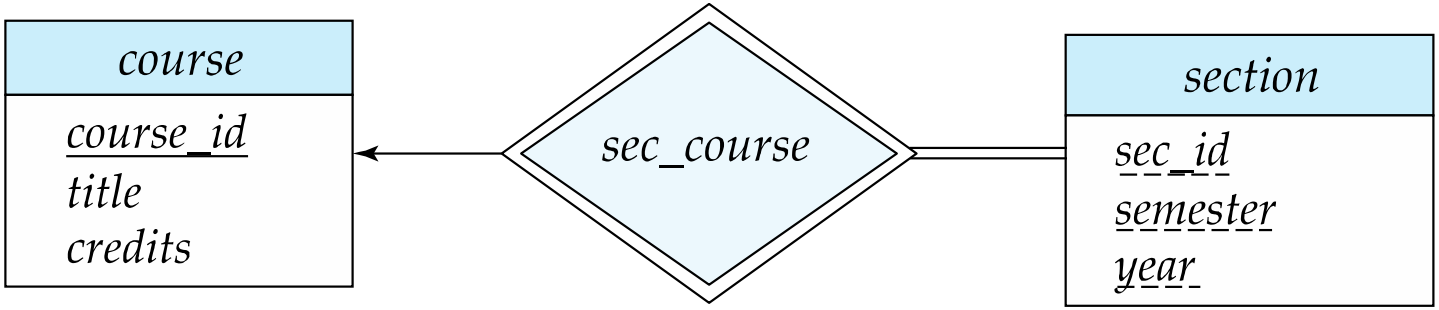
Question: What do cardinality constraints mean for a ternary relationship? Why are they difficult to express?

Weak Entity Sets

- The existence of some things (entities) is predicated on the existence of other things.
  - *e.g.*,
    - the offering of a course is predicated on the existence of that course
    - the sales figures for a car model is predicated on the existence of that car model
- If something is predicated on something else, it is necessarily the case that it cannot be uniquely specified without reference to the other entity
  - *e.g.*, sales figures alone are meaningless without reference to what they are sales figures for
- An entity set whose attributes do not provide a primary key is referred to as a **weak entity set**.

Weak Entity Sets

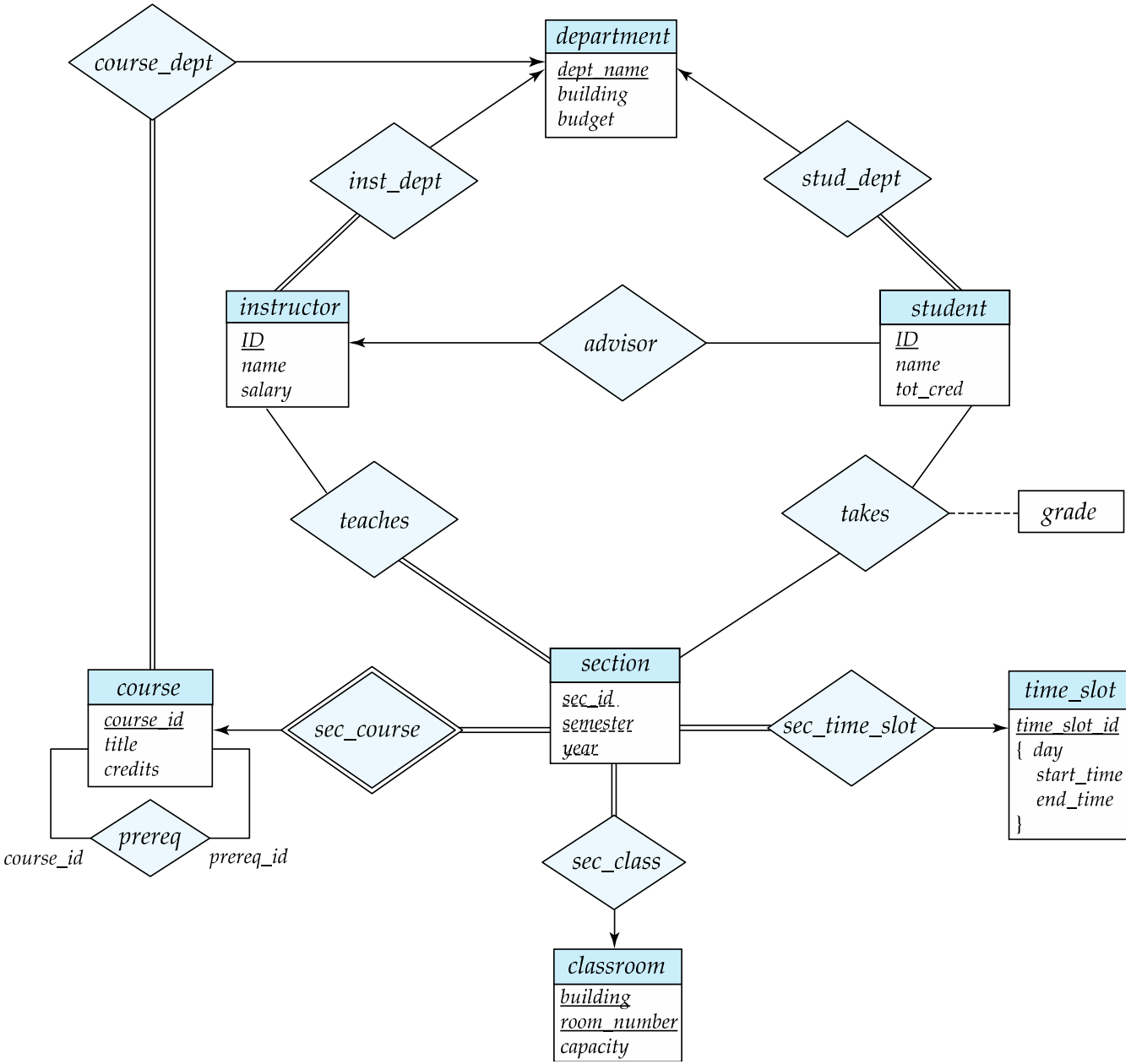
- We underline the discriminator of a weak entity set with a dashed line.
- We place the identifying relationship of a weak entity in a double diamond.
- Primary key for *section:* (*course_id, sec_id, semester, year*)
- The primary key of the weak entity is a super key of the identifying relationship.



Weak Entity Sets

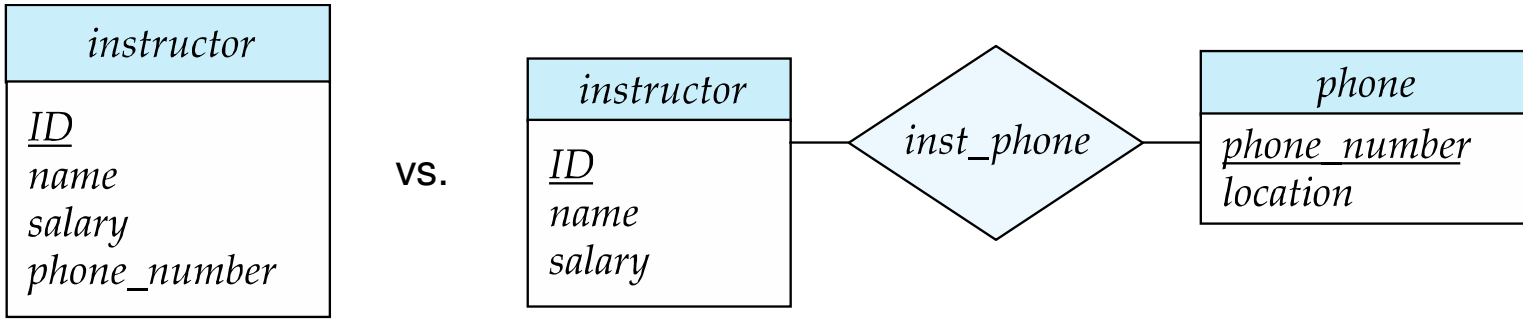
- The existence of a weak entity set depends on the existence of an **identifying entity set**.
  - The weak entity set must relate to the identifying entity set via a one-to-many relationship set from the identifying to the weak entity set, with total participation of the weak entity set.
  - **Identifying relationship** depicted using a double diamond.
- The **discriminator** (*or partial key)* of a weak entity set is the set of attributes that distinguishes a given weak entity among all the other entities that map to the same identifying entity.
  - If the lecture time was included in the section entity of the example, would it be part of the discriminator?
- The primary key of a weak entity set comprises the primary key of the identifying entity set plus the discriminator of the weak entity set. (Thus, the primary key of the weak entity set does exist but cannot be formed using only the attributes of the weak entity set.)

Example E-R Diagram for a University



Design Issues

Use of entity sets vs. attributes.

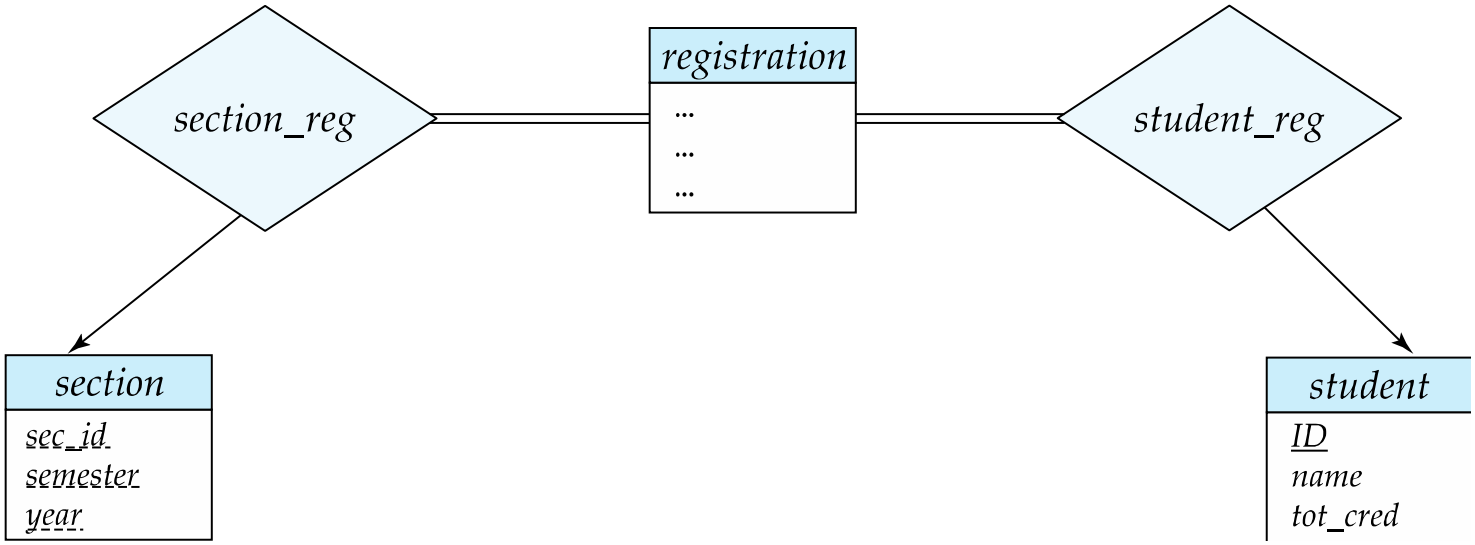


- Use of *phone* as an attribute is simpler.
- Use of *phone* as an entity is more expressive because it allows representing additional information about phone numbers, such as location (*e.g.*, home, office, or mobile).

Design Issues (Cont.)

Use of entity sets vs. relationship sets (actors vs. actions).

- Possible guideline is to use a relationship set to describe an action that occurs between entities.
- Example: *student takes section.*
  - If the registrar's office associates more complex information with each registration record, it might be best to replace the "takes" relationship set with a registration entity set as shown below:



Design Issues (Cont.)

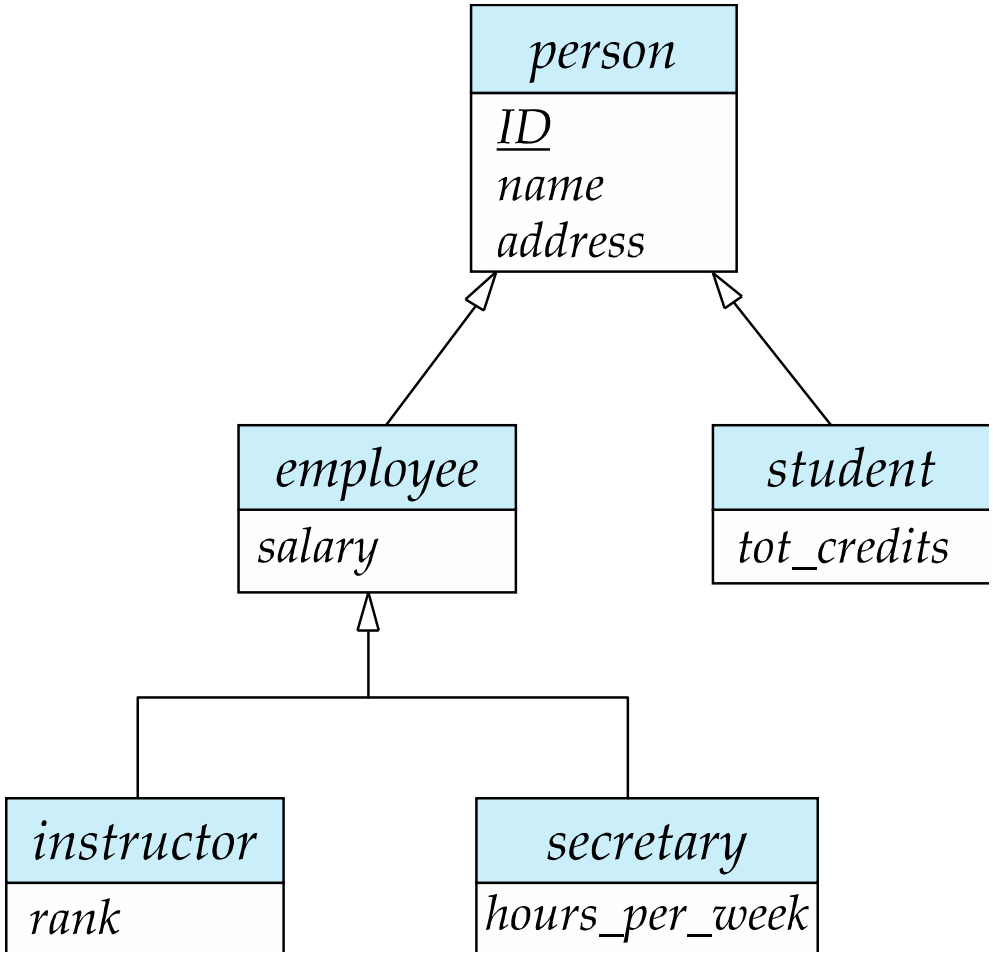
Binary versus n-ary relationship sets.

- Although it is possible to replace any non-binary (*n*-ary, for $n > 2$) relationship set by a collection of distinct binary relationship sets, an *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.
  - Example: a ternary relationship *parents*, relating a child to his/her father and mother, may be better replaced by two binary relationships, *father* and *mother*.
  - Using two binary relationships allows partial information (*e.g.*, only the mother being known).
  - Courses taken may be better as an n-ary relation (students are required to take a fixed number of courses) or a series of binary relations (core courses, technical electives, *etc*.)
- Nevertheless, there are some relationships that are inherently non-binary.
  - Example: *proj_guide* (see earlier slide).

Specialization and Generalization

- **Top-down design process** – designate sub-groupings within an entity set that are distinctive from other entities in the set.
- **A bottom-up design process** – combine a number of entity sets that share the same features into a higher-level entity set.
- The terms specialization and generalization are used interchangeably. Specialization and generalization are simple inversions of each other; they are represented in an ER diagram in the same way.
- **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

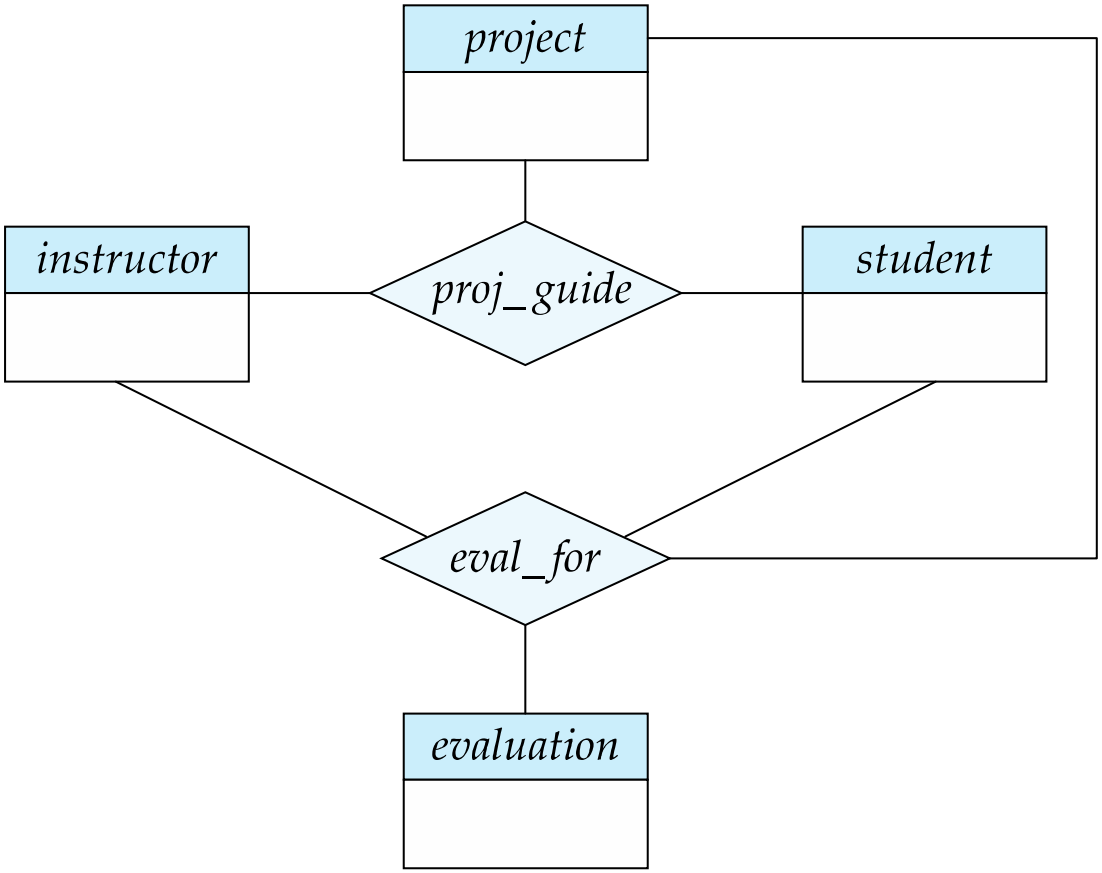


(**ordinary generalization is indicated by separate arrows from lower-level entities to higher-level entities,** a person can be both an employee and a student)

(**disjoint generalization is indicated by combined arrows from lower-level entities to higher-level entities,** an employee cannot be both an instructor and a secretary)

Aggregation

- Consider the ternary relationship *proj_guide*, discussed earlier.
- Suppose we want to record evaluations of a student by a guide on a project.

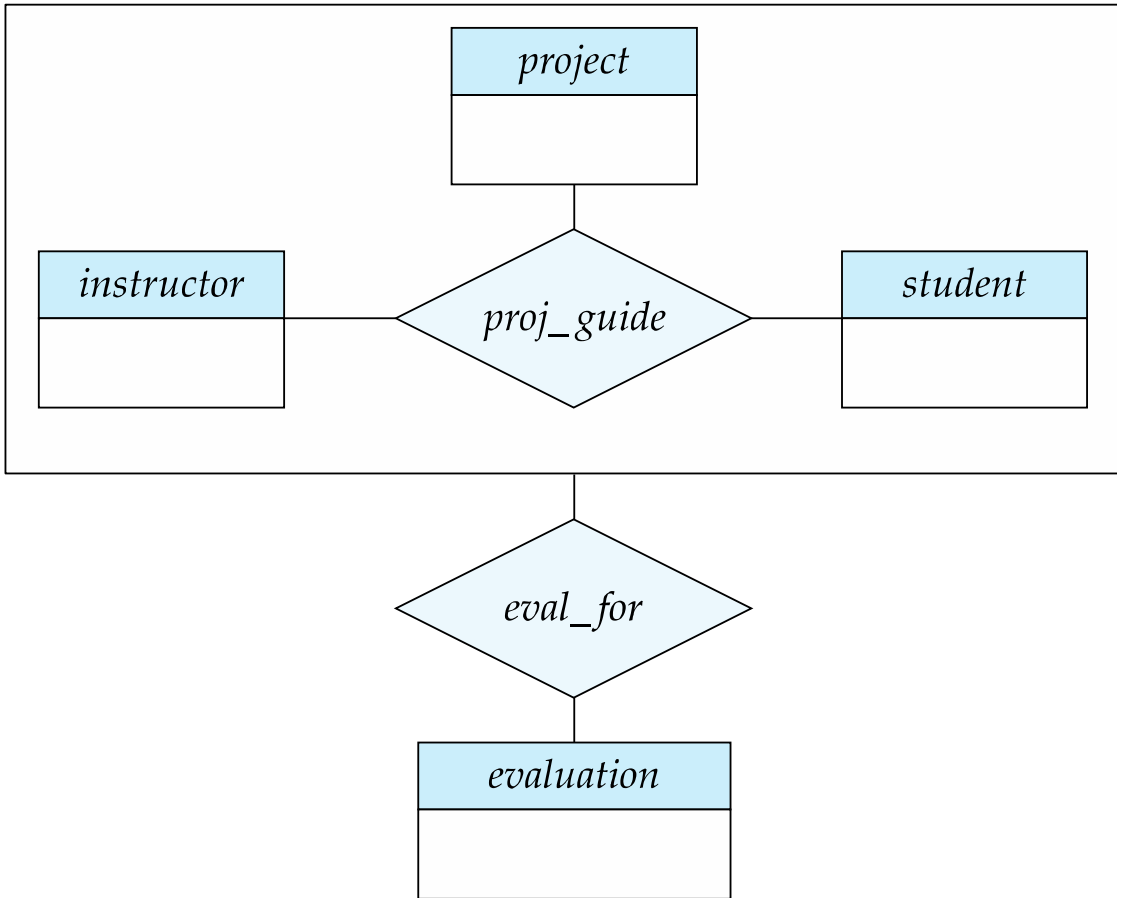


Aggregation (Cont.)

- Relationship sets *eval_for* and *proj_guide* represent overlapping information:
  - every *eval_for* relationship ties together a student, instructor, and a project, just like the *proj_guide* relationship
  - that said, we cannot discard *proj_guide* since some *proj_guide* relationships may not correspond to any *eval_for* relationships
- **Aggregation** – treat the entire relationship as an abstract entity.
  - removes redundancy
  - allows relationships between entities and relationships (or between relationships and relationships)

Aggregation (Cont.)

(**aggregate entity indicated by outer rectangle**)



Summary of ER Design Decisions

- The use of an attribute *vs*. 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 weak *vs*. ordinary entity set.
- The use of specialization/generalization, which contributes to modularity in the design.
- The use of aggregation, which allows us to treat the aggregate entity set as a single unit without concern for the details of its internal structure.

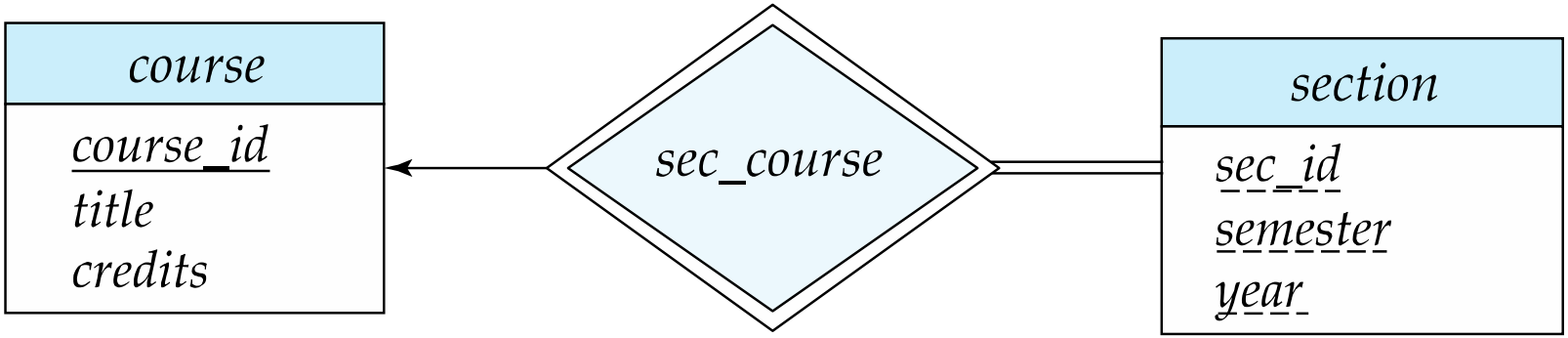
Reduction to Relation Schemas

- Entity sets and relationship sets can be expressed uniformly as *relation schemas*.
- A database that conforms to an ER diagram can be represented by a collection of schemas.
- For each entity set and relationship set there is (usually) a unique schema that is assigned the name of the corresponding entity set or relationship set.
- In special cases (*e.g.*, total participation) the schema for a relationship set is collapsed with the schema for one of the participating entity sets.
- The attributes of the schema reflects the attributes of the entity set or relationship set reduced to this schema.

Question: how do we translate cardinality and participation constraints in the ER model into equivalent constraints in the relational model?

Entity Sets With Simple Attributes

- **Simple attribute**: not composite or multi-valued.
- An ordinary (*i.e.*, non-weak) entity set with simple attributes reduces to a schema with the same attributes:
  student (*ID*, *name*, *tot_cred*)
- A weak entity set with simple attributes reduces to a schema that includes the primary key of the identifying entity set:
  section (*course_id*, *sec_id*, *sem*, *year*)

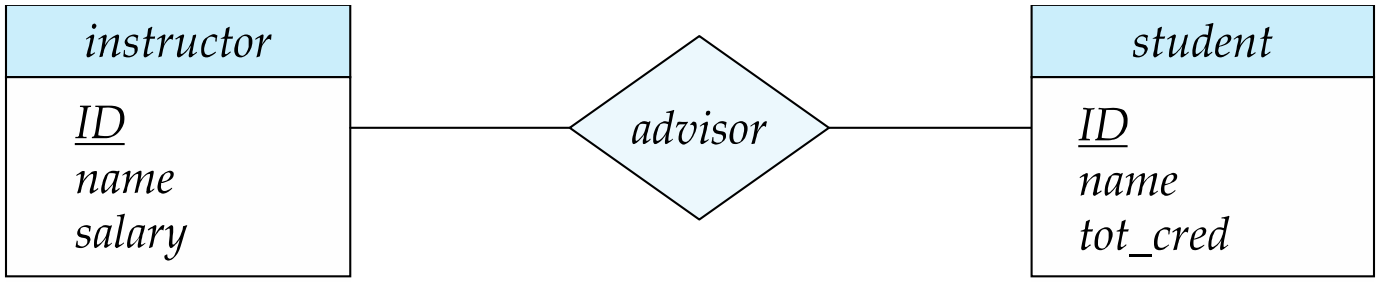


Representing Relationship Sets

- Any relationship set with simple descriptive attributes (or no descriptive attributes) can be represented as a schema with attributes for the primary keys of the participating entity sets, and for all the descriptive attributes.
- Example: schema for relationship set *advisor*

  advisor = (*s_ID*, *i_ID*)

  Note: we are free to choose new names for the primary keys of the participating entity sets, such as *s_ID* instead of *student.ID*



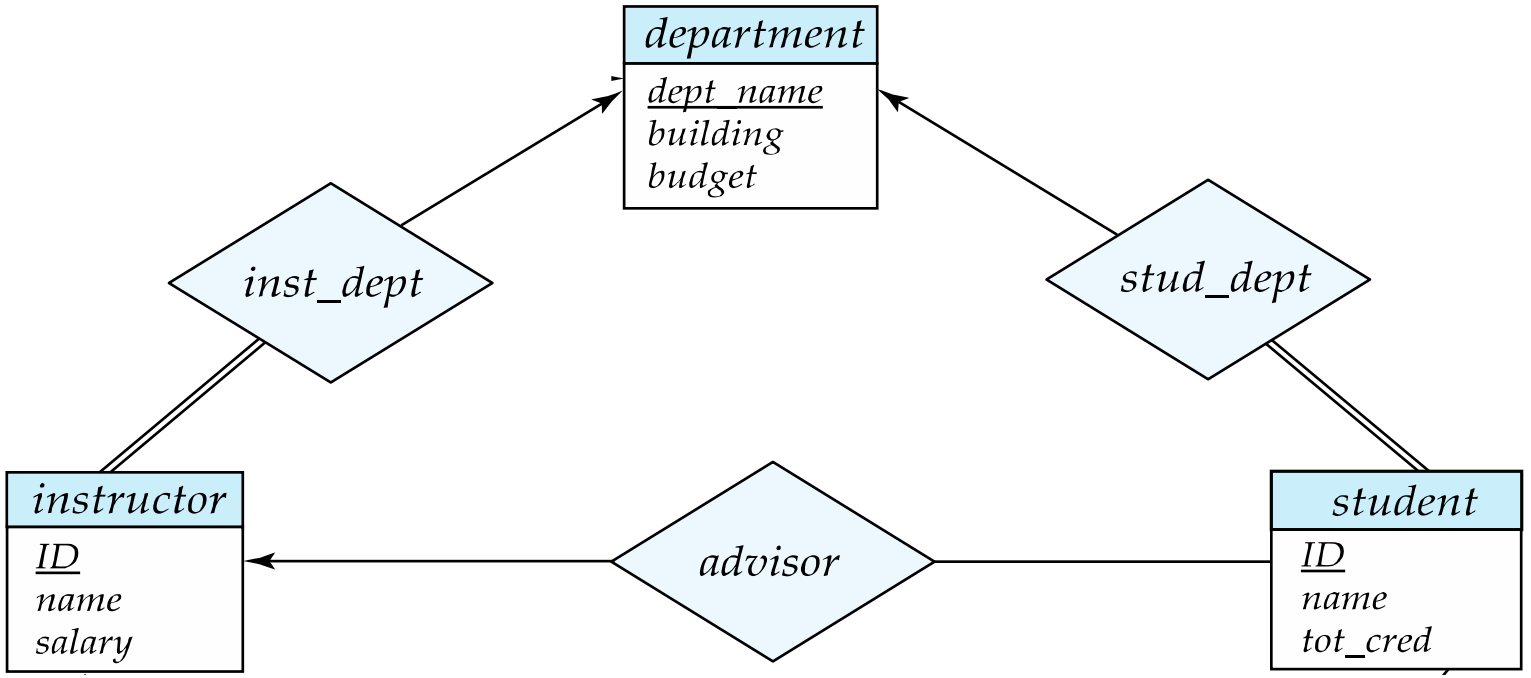
- Question: how do we correctly choose the primary key of the relation for the relationship?

Representing Relationship Sets

- Choosing primary keys for relations derived from binary relationship sets:
  - **many-to-many** – include attributes for the primary keys of both participating entity sets
    - *E.g.*: primary key is {*s_ID*, *i_ID*} for *advisor* relation on last slide
  - **one-to-many** – include only the attributes for the primary key of the "many" side
    - *E.g.*: one instructor to many students, primary key is {*s_ID*}
  - **many-to-one** – analogous to one-to-many
  - **one-to-one** – choose one of the participating entity sets (arbitrarily), and include only the attributes of that entity set
    - *E.g.*: one instructor to one student, primary key is {*s_ID*} or {*i_ID*}

Redundancy in Schemas

- Many-to-one and one-to-many relationship sets that are total on the many-side can also be represented by adding an extra attribute to the "many" side, containing the primary key of the "one" side.
- Example: Instead of creating a separate schema for relationship set *inst_dept*, add an attribute *dept_name* to the schema for *instructor*.
- Always do this for the identifying relationship of weak entity!



Composite and Multivalued Attributes

instructor

ID
name
first_name
middle_initial
last_name
address
street
street_number
street_name
apt_number
city
state
zip
{ phone_number }
date_of_birth
age ( )

- 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*
    - note: omit the prefix (in this case "*name_*") if there is no ambiguity
- Ignoring multivalued attributes, the flattened instructor schema is
  - *instructor*(*ID, first_name, middle_initial, last_name, street_number, street_name, apt_number, city, state, zip_code, date_of_birth*)

Composite and Multivalued Attributes

- A multivalued attribute *M* of an entity *E* is represented by a separate schema *EM.*
  - Schema *EM* has attributes corresponding to the primary key of *E* and an attribute corresponding to multivalued attribute *M.*
  - The primary key of *EM* comprises both attributes.
  - 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)

Specialization via Schemas

- Method 1:
  - Form a schema for the higher-level entity first.
  - Then form a schema for each lower-level entity set. Include the primary key of the 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.
    - Materialized views help with this

Specialization via Schemas (Cont.)

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

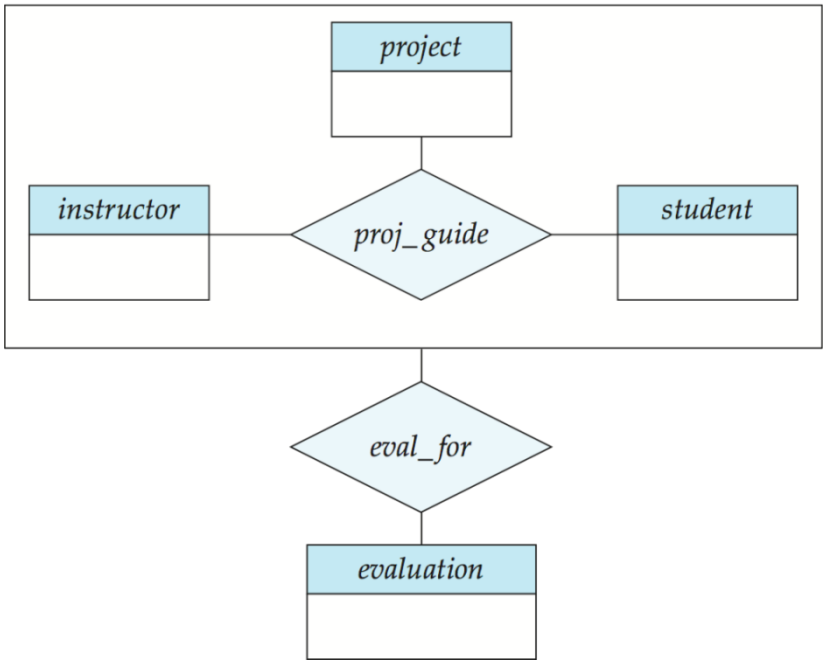
schema	attributes
person	*ID, name, street, city*
student	*ID, name, street, city, tot_cred*
employee	*ID, name, street, city, salary*

  - If specialization is total, the schema for the generalized entity set (*person*) is not required to store information.
    - Relation *person* can be defined as a "view" relation containing the union of student and employee.
    - An explicit schema may still be needed for foreign key constraints
      - Why?
  - Drawback: *name, street* and *city* may be stored redundantly for people who are both students and employees.
    - *i.e.*, whenever specialization is NOT total, redundancy will occur

Schemas Corresponding to Aggregation

- To represent the aggregation of a relationship, simply create a schema for that relationship using the technique described earlier.
- The schema for *eval_for* comprises the primary key of the schema for *proj_guide*, the primary key for *evaluation*, as well as any descriptive attributes of *eval_for*.

  eval_for (*i_ID, p_ID, s_ID, evaluation_ID*)



Question:
Does this make the relation for *proj_guide* redundant?

Design Problem

Consider the problem of providing electricity to consumers, where production must be matched in real time to demand, and billing charges vary during the day and at different seasons of the year. Homes with a smart meter can record their electricity-consumption amount on an hourly basis, which may then be provided to the electricity utility. Consumption will be affected not only by time of day and year, but also by the weather (notably, the outdoor temperature, but also wind, precipitation, etc., and the house type (detached, semi, townhouse, apartment, ...).

(a) Create an entity-relationship diagram to model this energy production, consumption, and billing situation. Your diagram may follow any of the notational conventions described in the lecture notes, in class, or the course textbook, but should be consistent in its usage. Include all relevant entities and relationships, cardinality constraints, and participation constraints. Indicate primary keys of entities by underlining them.

(b) Translate your ER model into a relational model. You do not need to give specific create table commands. However, you should identify the following constraints: foreign keys, any attributes which be unique in a relation, and any attributes which should not be NULL.