Database Design Using the Entity-Relationship Model

CENG315 INFORMATION MANAGEMENT

Design Phases

- How to design a database schema?
- Initial phase -- characterize the data needs of the prospective database users
- Second phase – conceptual schema
  - Choosing a data model
  - Translating these requirements into a conceptual schema of the database (by applying the concepts of the chosen data model)
  - A fully developed conceptual schema indicates the functional requirements of the enterprise
    - Users describe the kinds of operations 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
    - Mapping the high-level conceptual schema onto the implementation data model of the database system that will be used.
  - Physical Design – Deciding on the physical layout of the database
    - The physical features of the database are specified.
    - These features include the form of file organization and choice of index structures.

Outline of the Entity-Relationship (ER) Model

Entity Relationship Model

- 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
- The ER model also has an associated diagrammatic representation, the **ER diagram**, which can express the overall logical structure of a database graphically.

ER Model -- Database Modeling

- The ER data model employs three basic concepts:
  - **entity sets**
  - **relationship sets**
  - **attributes**

Entity Sets

- An **entity** is an object that exists and is distinguishable from other objects.
  - Example: specific person, company, event, plant
- An **entity set** is a set of entities of the same type that share the same properties.
  - Example: set of all persons, companies, holidays, trees

Entity Sets (Cont.)

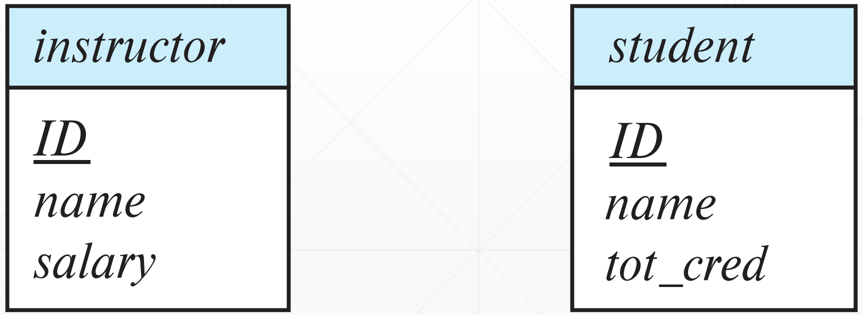
- An entity is represented by a set of **attributes**; i.e., descriptive properties possessed by all members of an entity set.
  - Example:

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

- A subset of the attributes form a **primary key** of the entity set; i.e., uniquely identifying each member of the set.

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



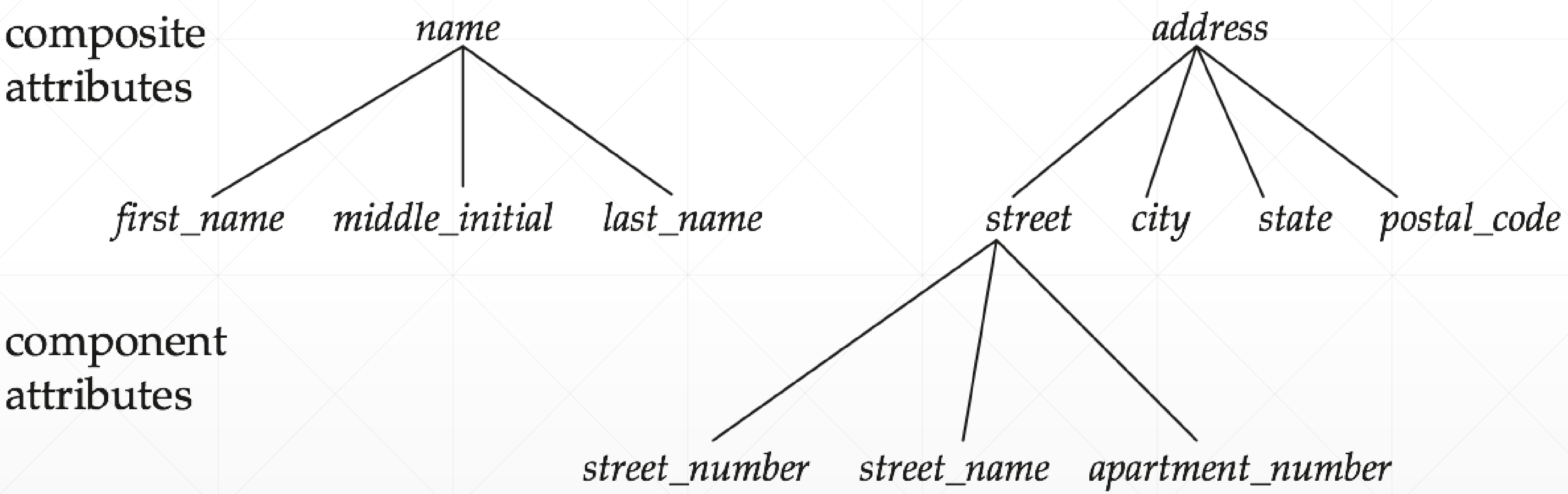
Domain of Attributes

- The set of permitted values for each attribute
  - Example: The domain of attribute *course_id* might be the set of all text strings of a certain length.
  - Example: The domain of attribute *semester* might be strings from the set {Fall, Winter, Spring, Summer}.

Complex Attributes

- Attribute types:
  - **Simple** and **composite** attributes
    - Composite attributes can be divided into subparts (other attributes).
  - **Single-valued** and **multivalued** attributes
    - Example: multivalued attribute: *phone_numbers*
  - **Derived** attributes
    - Can be computed from other attributes
    - Example: *age*, given *date_of_birth*

Composite Attributes



Representing Complex Attributes in ER Diagram

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

Relationship Sets

- A relationship is an association among several entities
  - Example:

44553 (Peltier) *student* entity	<u>*advisor*</u> relationship set	22222 (Einstein) *instructor* entity

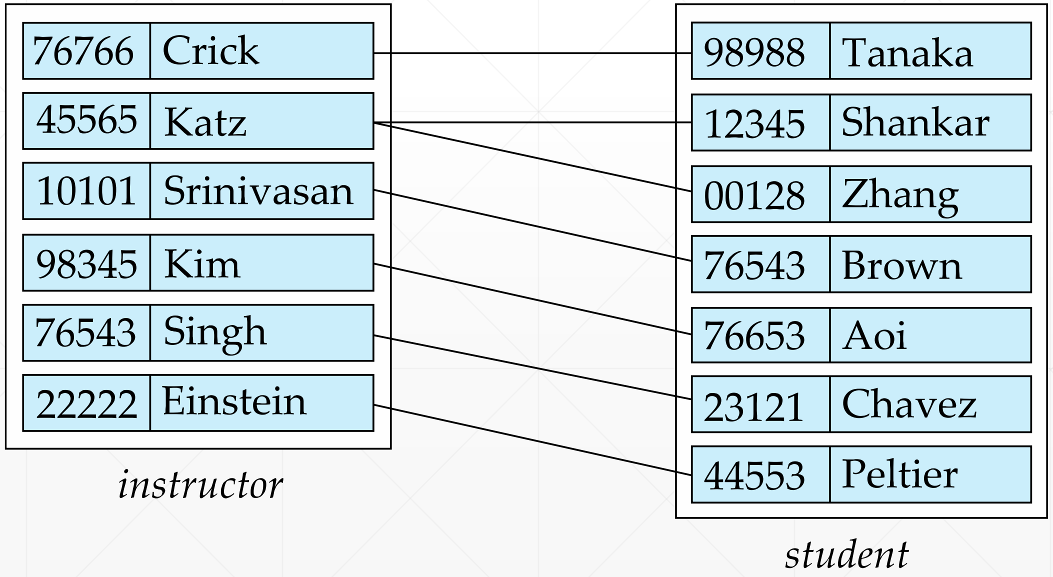
- A relationship set is a mathematical relation among n ≥ 2 entities, each taken from entity sets
  - $\{(e_1, e_2, \ldots e_n) \mid e_1 \in E_1, e_2 \in E_2, \ldots, e_n \in E_n\}$

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

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

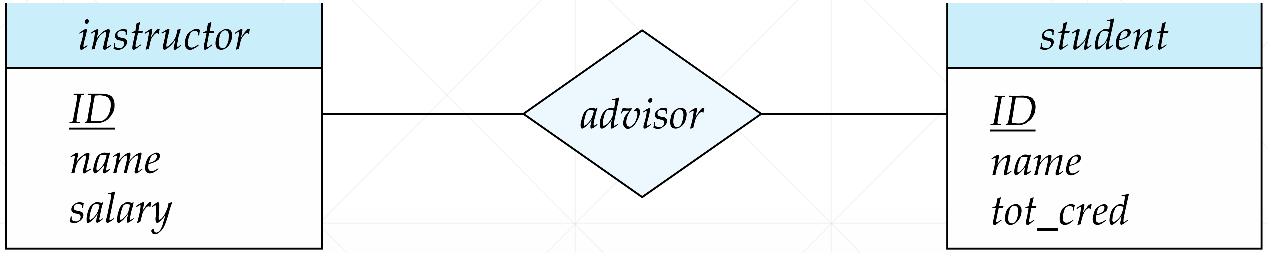
Relationship Sets (Cont.)

- 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.



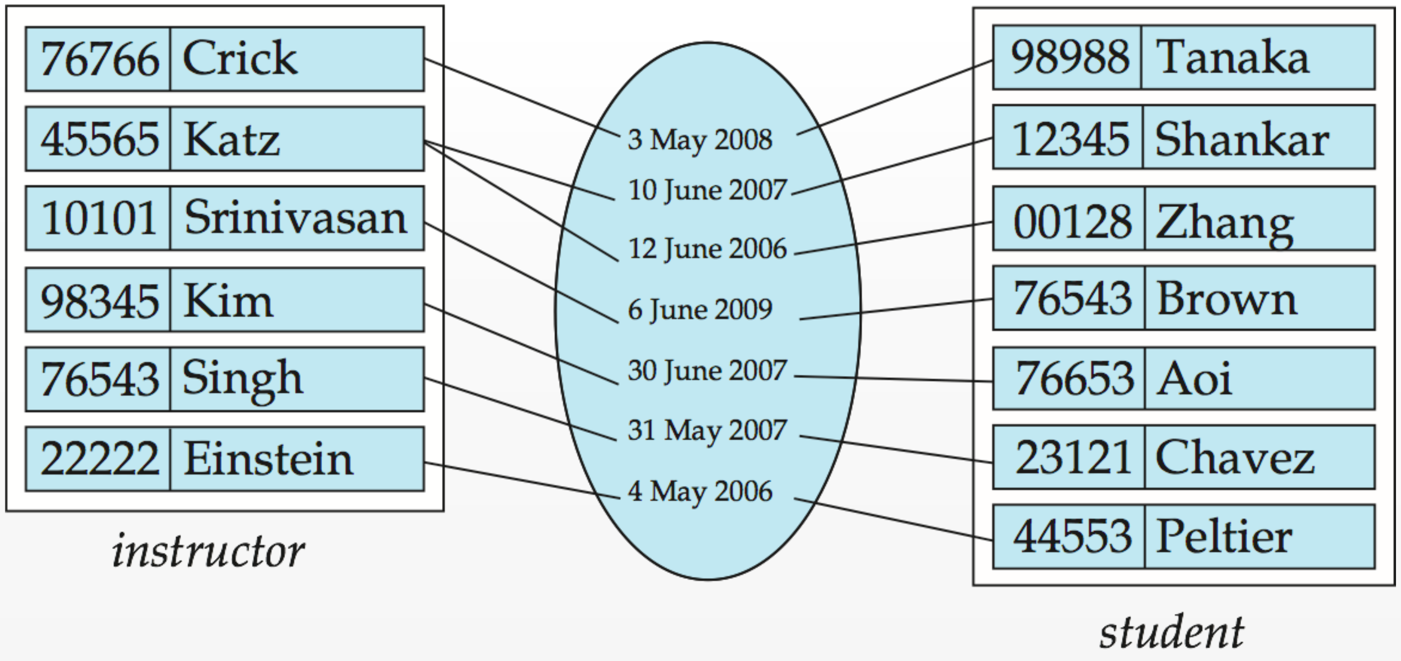
Representing Relationship Sets via ER Diagrams

- Diamonds represent relationship sets.

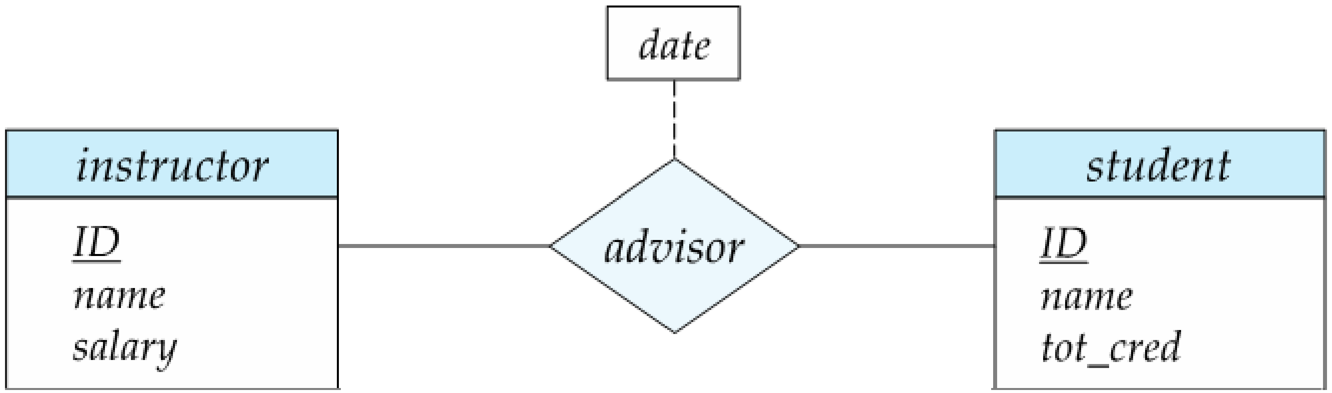


Relationship Sets with Attributes

- An attribute can also be associated with 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

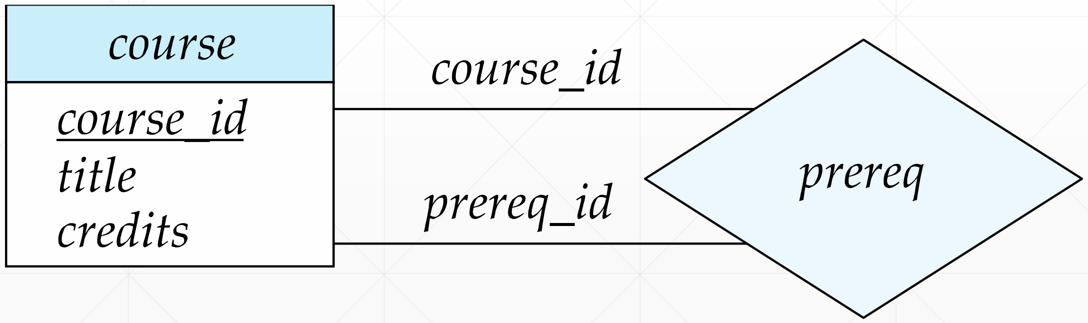


Relationship Sets with Attributes



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**.

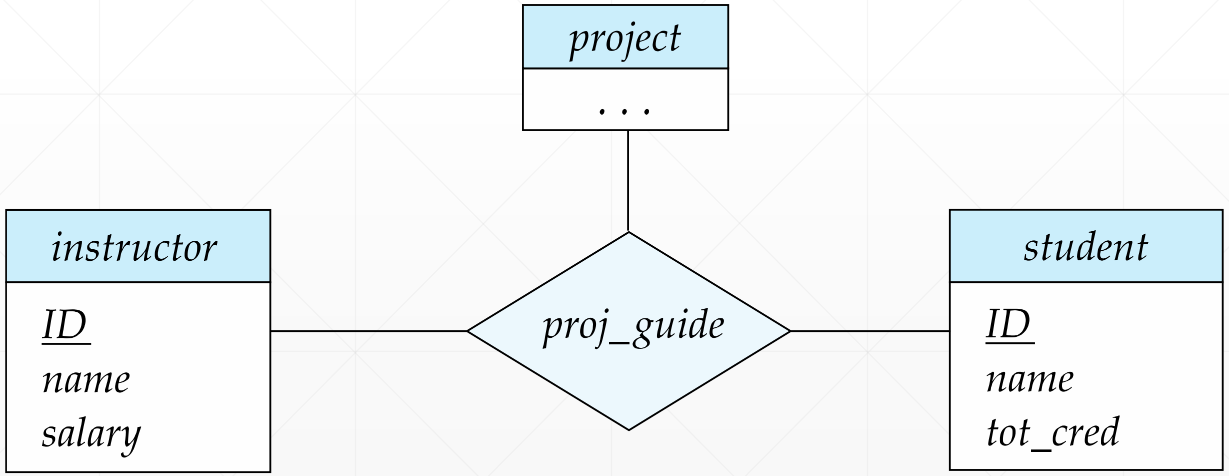


Degree of a Relationship Set

- Binary relationship
  - Involve two entity sets (or degree two).
  - Most relationship sets in a database system are binary.
- Relationships between more than two entity sets are rare.
  - Example: *students* work on research *projects* under the guidance of an *instructor*.
  - relationship *proj_guide* is a ternary relationship between *instructor, student,* and *project*

Non-binary Relationship Sets

- There are occasions when it is more convenient to represent relationships as non-binary.
- ER Diagram with a Ternary Relationship



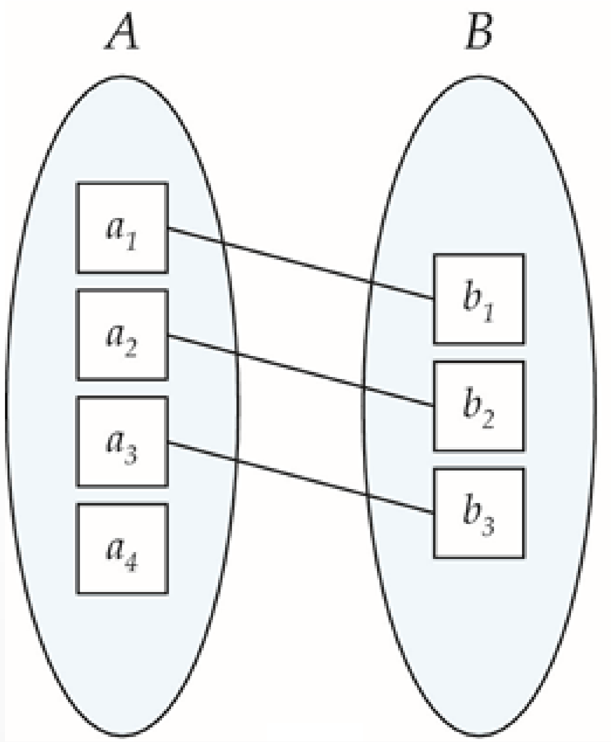
Mapping Cardinality Constraints

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

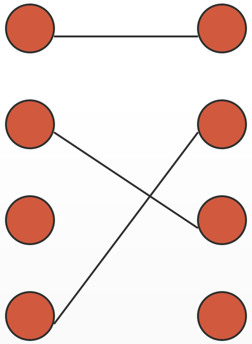
Representing Cardinality Constraints in ER Diagram

- 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.

Mapping Cardinalities: One to One



- An entity in *A* is associated with *<u>at most one</u>* entity in *B*, and an entity in *B* is associated with *<u>at most one</u>* entity in *A*.

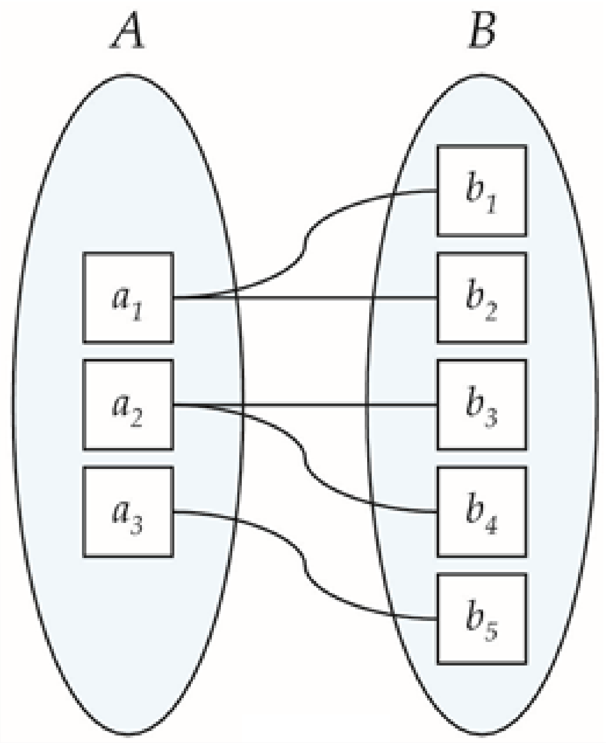


Note: Some elements in *A* and *B* may not be mapped to any elements in the other set

One-to-One Relationship

- One-to-one relationship between an *instructor* and a *student*:
  - An *instructor* may advise <u>*at most one*</u> *student,* and a *student* may have <u>*at most one*</u> *advisor.*

Mapping Cardinalities: One to Many



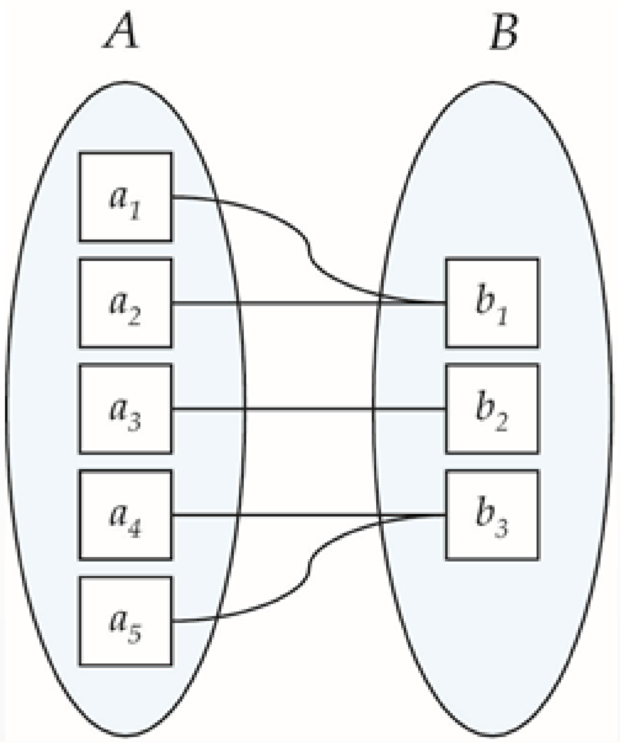
- An entity in *A* is associated with *any number* (zero or more) of entities in *B*. An entity in *B, however,* can be associated with *at most one* entity in *A*.

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

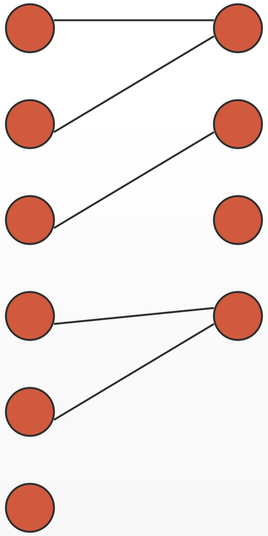
One-to-Many Relationship

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

Mapping Cardinalities: Many to One



- An entity in *A* is associated with *at most one* entity in *B*. An entity in *B*, however, can be associated with *any number* (zero or more) of entities in *A*.

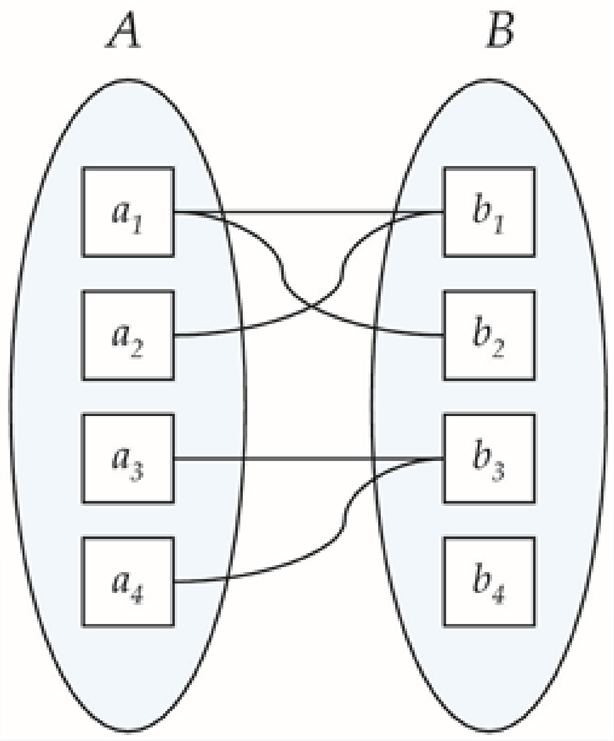


Note: Some elements in *A* and *B* may not be mapped to any elements in the other set

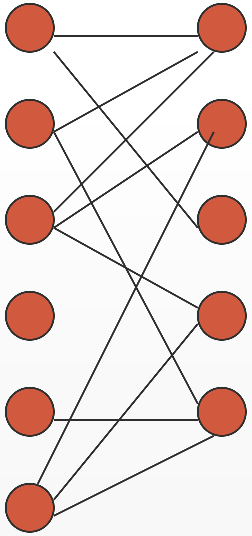
Many-to-One Relationships

- In a many-to-one relationship between an *instructor* and a *student,*
  - an *instructor* may advise *at most one* *student* via *advisor*,
  - but a *student* may have many *advisors* (including 0)

Mapping Cardinalities: Many to Many



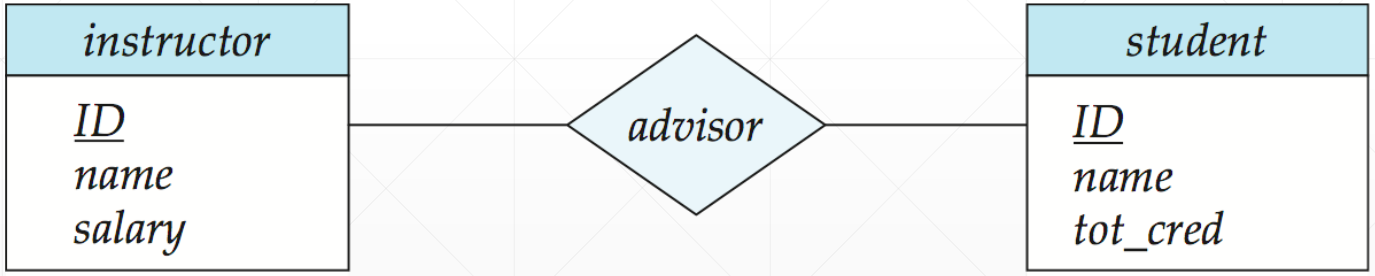
- An entity in *A* is associated with *__any number__* (zero or more) of entities in *B*, and an entity in *B* is associated with *__any number__* (zero or more) of entities in *A*.



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

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*



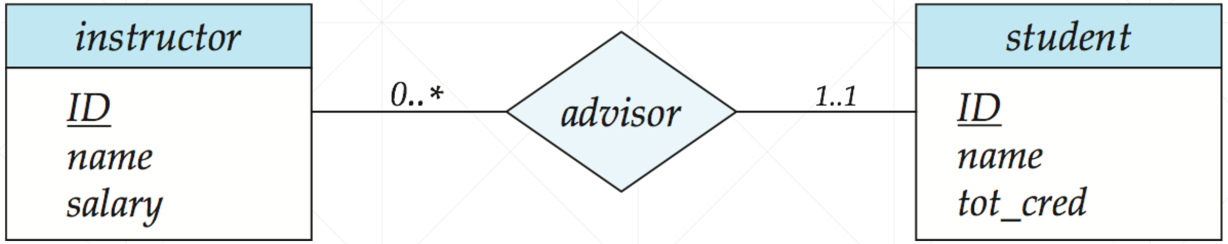
Total and Partial Participation

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

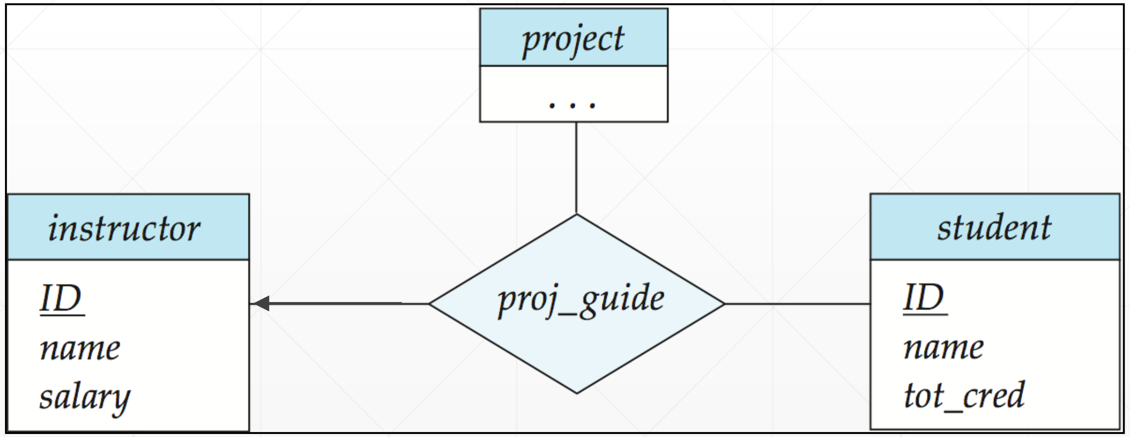


- An instructor can advise 0 or more students. A student must have 1 advisor; cannot have multiple advisors

- It is easy to misinterpret the 0.. ∗ on the left edge and think that the relationship advisor is many-to-one from instructor to student - this is exactly the reverse of the correct interpretation

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 *instructor* for a *project*



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

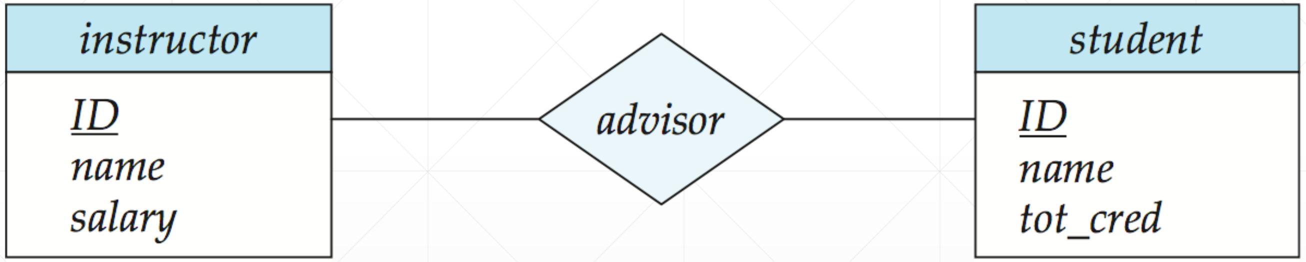
- By definition, individual entities are distinct.
- From database perspective, the differences among them must be expressed in terms of their attributes.
- The values of the attribute values of an entity must be such that they can uniquely identify the entity.
  - No two entities in an entity set are 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 E_1, E_2, .. E_n
  - The primary key for *R* is consists of the union of the primary keys of entity sets E_1, E_2, .. E_n
  - If the relationship set *R* has attributes a_1, a_2, .., a_m associated with it, then the primary key of *R* also includes the attributes a_1, a_2, .., a_m

Primary Key for Relationship Sets (Cont.)

- 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.

Choice of Primary Key for Binary Relationship

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

Weak Entity Sets

- A weak entity is an entity that cannot be uniquely identified by its attributes alone.
- 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.

Weak Entity Sets (Cont.)

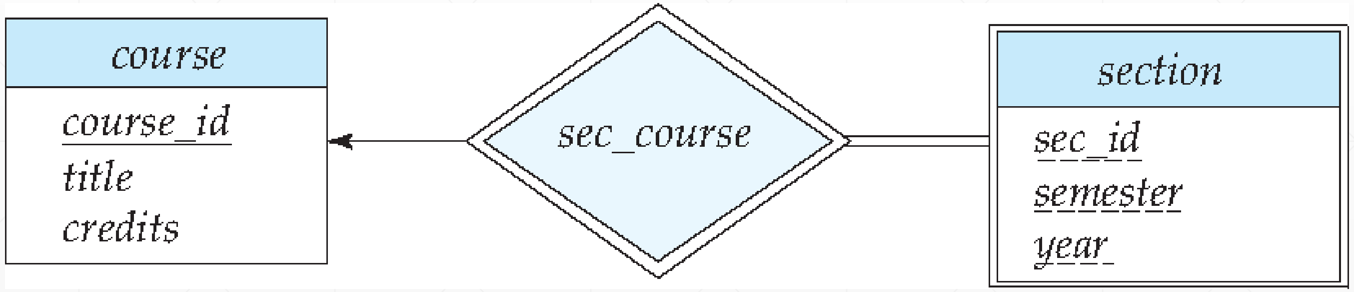
- An alternative way to deal with this redundancy is to not store the attribute *course_id* in the *section* entity and to only store the remaining attributes *section_id*, *year*, and *semester.*
  - However, the entity set *section* then does not have enough attributes to identify a particular *section* entity uniquely.
- To deal with this problem, we treat the relationship *sec_course* as a special relationship that provides extra information, in this case, the *course_id*, required to identify *section* entities uniquely.

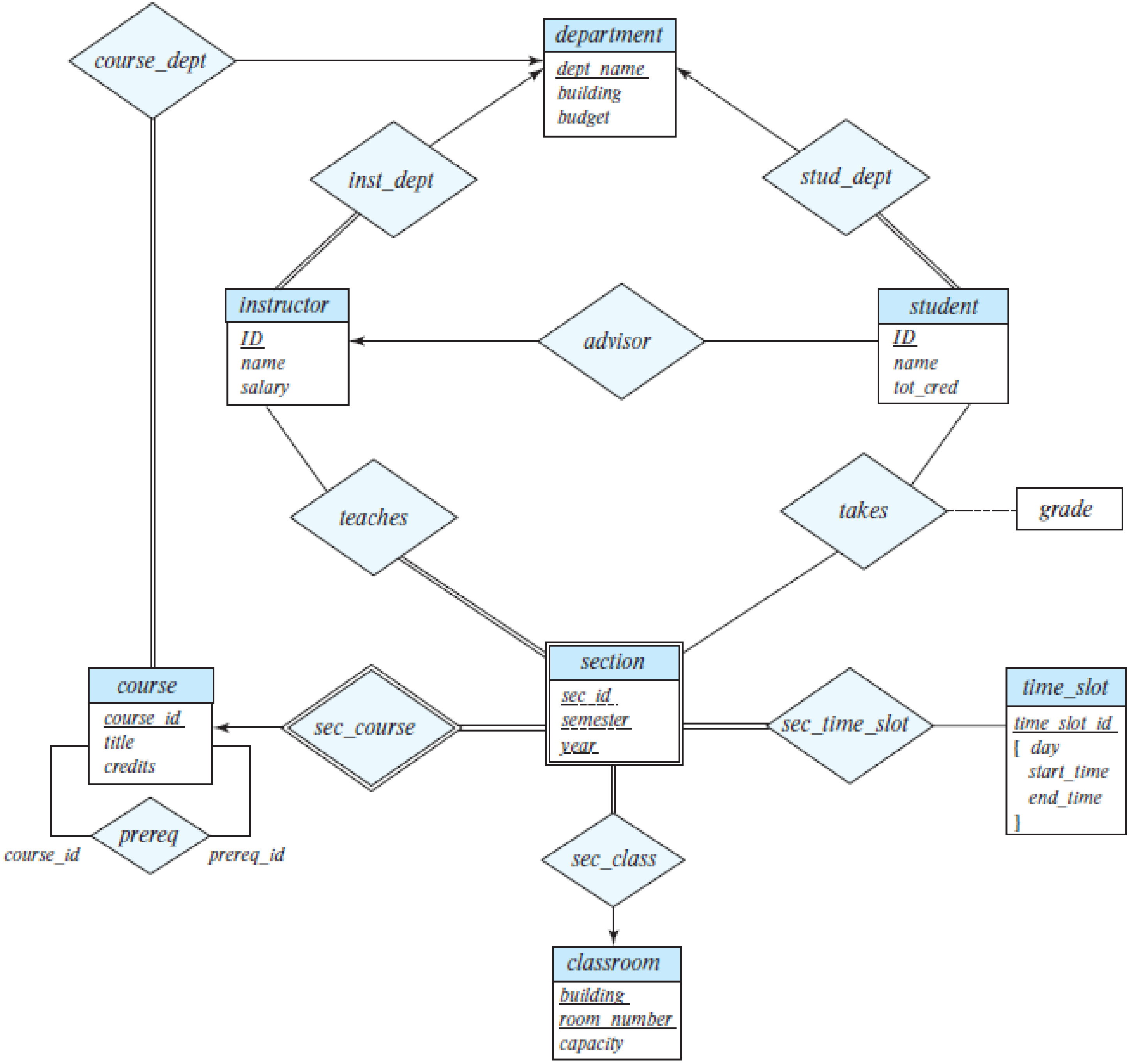
Weak Entity Sets (Cont.)

- A **weak entity set** is one whose existence is dependent on another entity, called its **identifying entity.**
- Instead of associating a primary key with a weak entity, we use the identifying entity, along with extra attributes called **discriminator** to uniquely identify a weak entity.
- An entity set that is not a weak entity set is termed a **strong 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.

Expressing Weak Entity Sets

- In ER 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*)





ER Diagram for a University Enterprise

Reduction to Relation Schemas

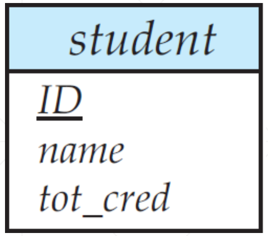
From ER Diagrams to Relational Schemas*

- To a first approximation, converting an ER design to a relational schema is straightforward:
  - Turn each entity set into a relation with the same set of attributes.
  - Replace a relationship set by a relation whose attributes are the keys for the connected entity sets (and any descriptive attributes of the relationship set).
- There are also several special situations that we need to deal with, including:
  - Weak entity sets cannot be translated straightforwardly to relations.
  - Sometimes, we do well to combine two relations.

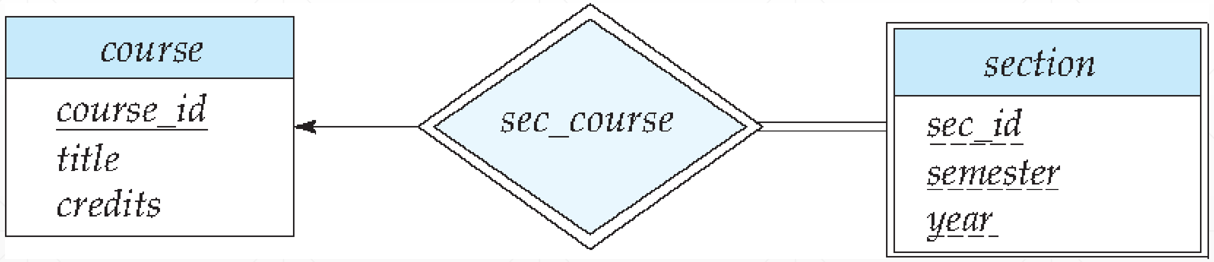
*Jeffrey D. Ullman and Jennifer Widom, “A First Course in Database Systems”, 3rd Ed., 2007.

Representing Entity Sets

- A strong entity set reduces to a schema with the same attributes
  - *student(<u>ID</u>, name, tot_cred)*



- A weak entity set becomes a table that includes a column for the primary key of the identifying strong entity set



  - *section(<u>course_id, sec_id, sem, year</u>)*

Representation of Entity Sets with Composite Attributes

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

- We handle composite attributes by creating a separate attribute for each of the component attributes, we do not create a separate attribute for the composite attribute itself.
  - Example: For the composite attribute name, the schema generated three attributes name_first_name, name_middle_initial and name_last_name
    - Prefix omitted if there is no ambiguity (*name_first_name* could be *first_name)*

Representation of Entity Sets with Composite Attributes (Cont.)

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

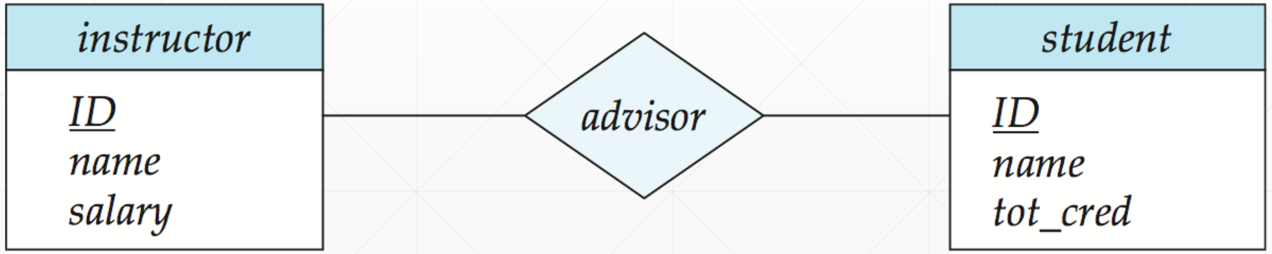
- Without including multivalued attributes, instructor schema is
  - *instructor(ID,*
first_name, middle_initial, last_name,
street_number, street_name,
apt_number, city, state, zip,
date_of_birth)

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 = (*<u>ID</u>*, *<u>phone_number</u>*)
- 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)

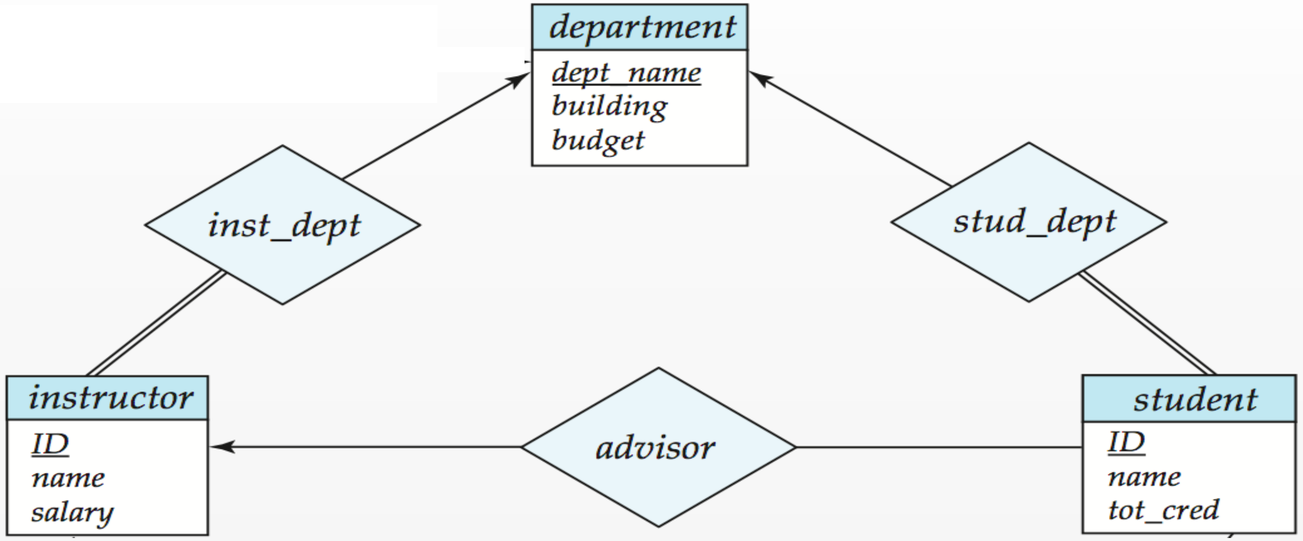
Representing Relationship Sets

- 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* = (*<u>s_id, i_id</u>*)



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*
  - *instructor* = (*ID*, *name*, *salary*, *dept_name*)



Redundancy of Schemas (Cont.)

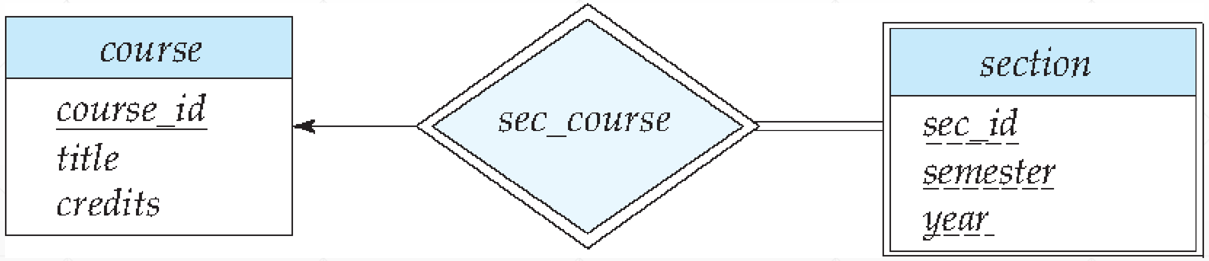
- For **one-to-one** relationship sets, either side can be chosen to act as the “many” side
  - That is, an extra attribute can be added to either of the tables corresponding to the two entity sets
- We can combine schemas even if the participation is partial by using null values.
  - In the previous example, if *inst_dept* were partial, then we would store null values for the *dept_name* attribute for those instructors who have no associated department.

Redundancy of Schemas (Cont.)

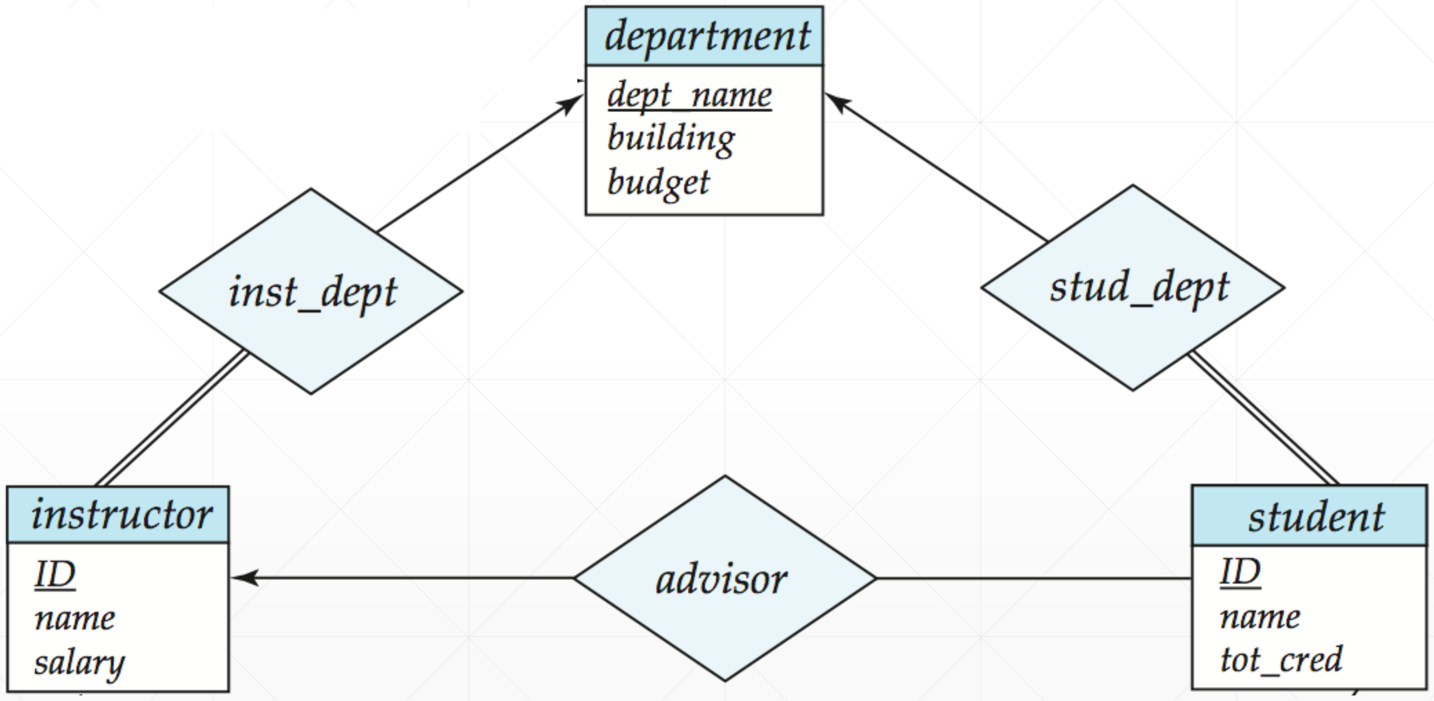
- Finally, we consider the foreign key constraints that would have appeared in the schema representing the relationship set.
  - There would have been foreign key constraints referencing each of the entity sets participating in the relationship set.
    - For example, *inst_de*pt has a foreign key constraint of the attribute *dept_name* referencing the department relation.
    - This foreign constraint is added to the instructor relation when the schema for *inst_dept* is merged into instructor.

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



Example: ERD to Relational Schema



Example: ERD to Relational Schema (Cont.)

- *instructor* (*ID*, *name*, *salary*)
- *student* (*ID*, *name*, *tot_cred*)
- *department* (*dept_name*, *building*, *budget*)
- *inst_dept* (*ID*, *dept_name*)
- *stud_dept* (*ID*, *dept_name*)
- *advisor* (*s_ID*, *i_ID*)

Example: ERD to Relational Schema (Cont.)

- *instructor* (*ID*, *name*, *salary*, *dept_name*)
- *student* (*ID*, *name*, *tot_cred*, *dept_name*)
- *department* (*dept_name*, *building*, *budget*)
- ~~*inst_dept* (*ID*, *dept_name*)~~
- ~~*stud_dept* (*ID*, *dept_name*)~~
- *advisor* (*s_ID*, *i_ID*)

Example: ERD to Relational Schema - Foreign Keys?

- *instructor* (*ID*, *name*, *salary*, *dept_name*)
- *student* (*ID*, *name*, *tot_cred*, *dept_name*)
- *department* (*dept_name*, *building*, *budget*)
- *advisor* (*s_ID*, *i_ID*)

Example: ERD to Relational Schema - Foreign Keys

- *instructor* = (*ID, name, salary, dept_name*)
  - *dept_name* referencing the *department* relation
- *student* (*ID*, *name*, *tot_cred, dept_name*)
  - *dept_name* referencing the *department* relation
- *department* (*dept_name*, *building*, *budget*)
- *advisor* = (*s_ID, i_ID*)
  - *s_ID* referencing the *student* relation
  - *i_ID* referencing the *instructor* relation

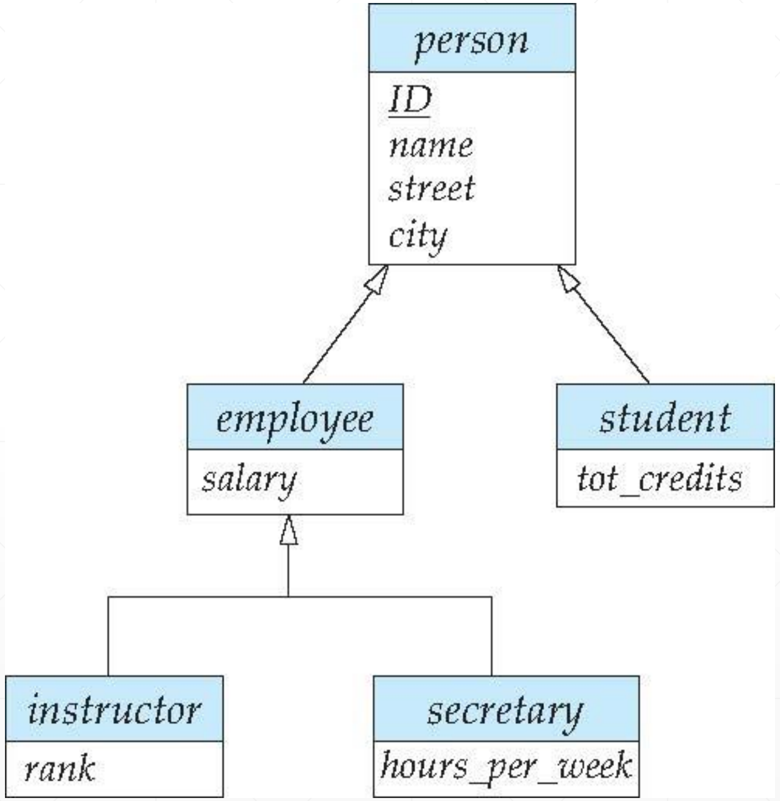
Extended ER Features

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.
- We refer to this relationship as the **ISA** relationship (e.g., *instructor* “is a” *person*).
- Depicted by a hollow arrow-head pointing from the specialized entity to the other entity.
- **Attribute inheritance** – a lower-level entity set inherits all the attributes and relationship participation of the higher-level entity set to which it is linked.

Specialization Example

- **Overlapping specialization**
  - An entity may belong to multiple specialized entity sets
  - Two separate arrows are used
  - Example: *employee* and *student*
- **Disjoint specialization**
  - An entity must belong to at most one specialized entity set
  - A single arrow is used
  - Example: *instructor* and *secretary*
- Total and partial



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 via 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

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 ER diagram in the same way.
- The terms specialization and generalization are used interchangeably.

Design Constraints on Specialization/Generalization

- **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.)

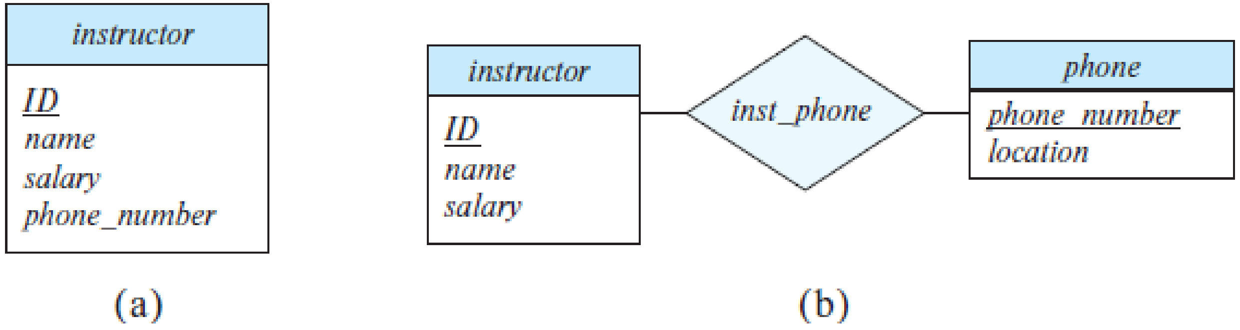
- Partial generalization is the default.
- We can specify total generalization in an ER diagram by adding the keyword **total** in the diagram and drawing a dashed line from the keyword to the corresponding hollow arrow-head to which it applies (for a total generalization), or to the set of hollow arrow-heads to which it applies (for an overlapping generalization).
- The university could create two specializations of *student*, namely *graduate* and *undergraduate*.
  - 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.

Design Issues

Some Clues

- Avoid redundancy.
- Limit the use of weak entity sets.
- Don’t use an entity set when an attribute will do.

Entities vs. Attributes



Alternatives for adding *phone* to the *instructor* entity set.

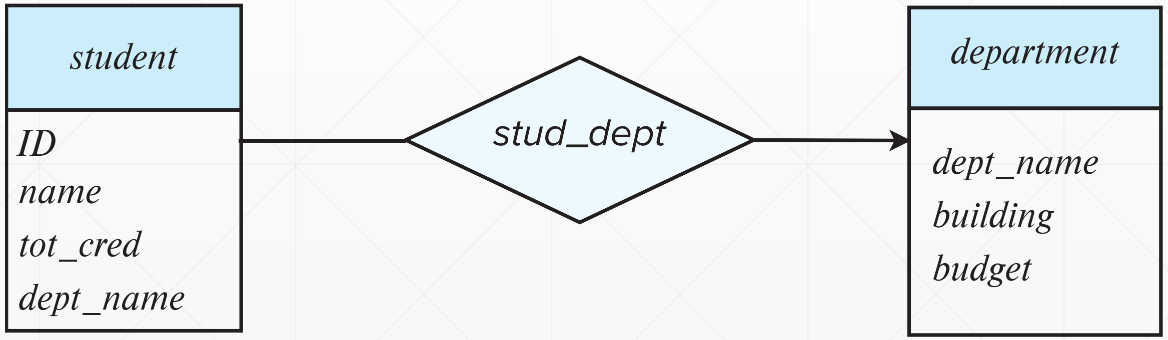
- Use of phone as an entity allows extra information about phone numbers (plus multiple phone numbers)
- In contrast, it would not be appropriate to treat the attribute name (of an instructor) as an entity; it is difficult to argue that name is an entity in its own right (in contrast to the phone).

Entities vs. Relationship Sets

- **Entities vs. Relationship Sets**
  - Possible guideline is to designate a relationship set to describe an action that occurs between entities
- **Placement of relationship attributes**
  - For example, or as attribute of *student*

Redundant Attributes

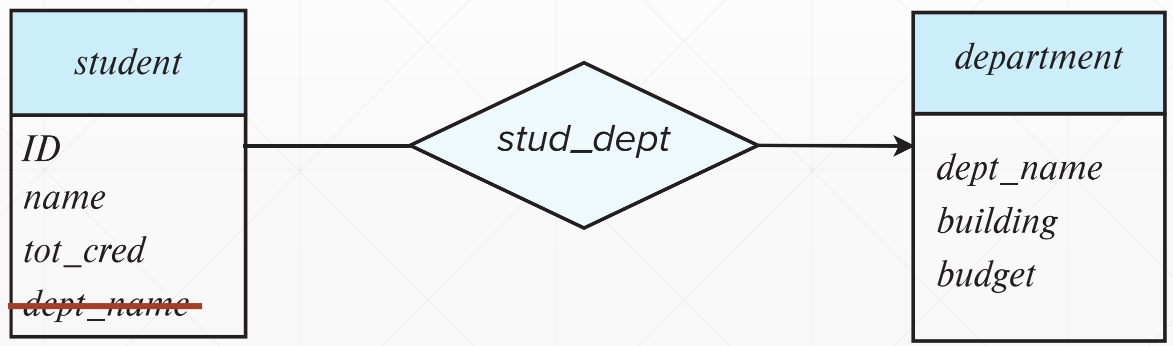
- Suppose we have entity sets:
  - *student*, with attributes: *ID*, *name, tot_cred*, *dept_name*
  - *department,* with attributes: *dept_name, building, budget*
- We model the fact that each student has an associated department using a relationship set *stud_dept*



(a) Incorrect use of attribute

Redundant Attributes (Cont.)

- The attribute *dept_name* in *student* below replicates information present in the relationship and is therefore redundant needs to be removed.

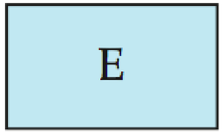


(a) Incorrect use of attribute

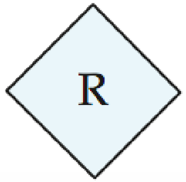
Binary Vs. Non-Binary Relationships

- Although it is possible to replace any non-binary (*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
  - For example, a ternary relationship *parents*, relating a child to his/her father and mother, is best replaced by two binary relationships, *father* and *mother*
    - Using two binary relationships allows partial information (e.g., only mother being known)
  - But there are some relationships that are naturally non-binary
    - Example: *proj_guide*

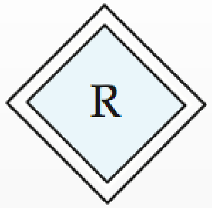
Summary of Symbols Used in ER Notation



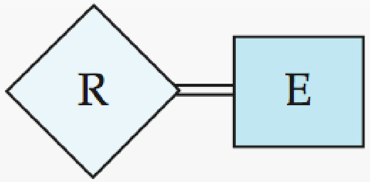
entity set



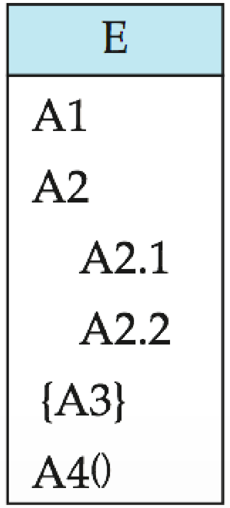
relationship set



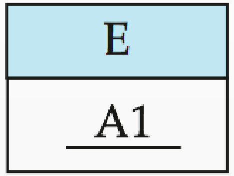
identifying relationship set for weak entity set



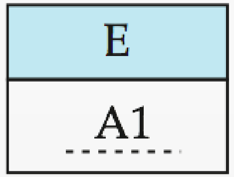
total participation of entity set in relationship



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

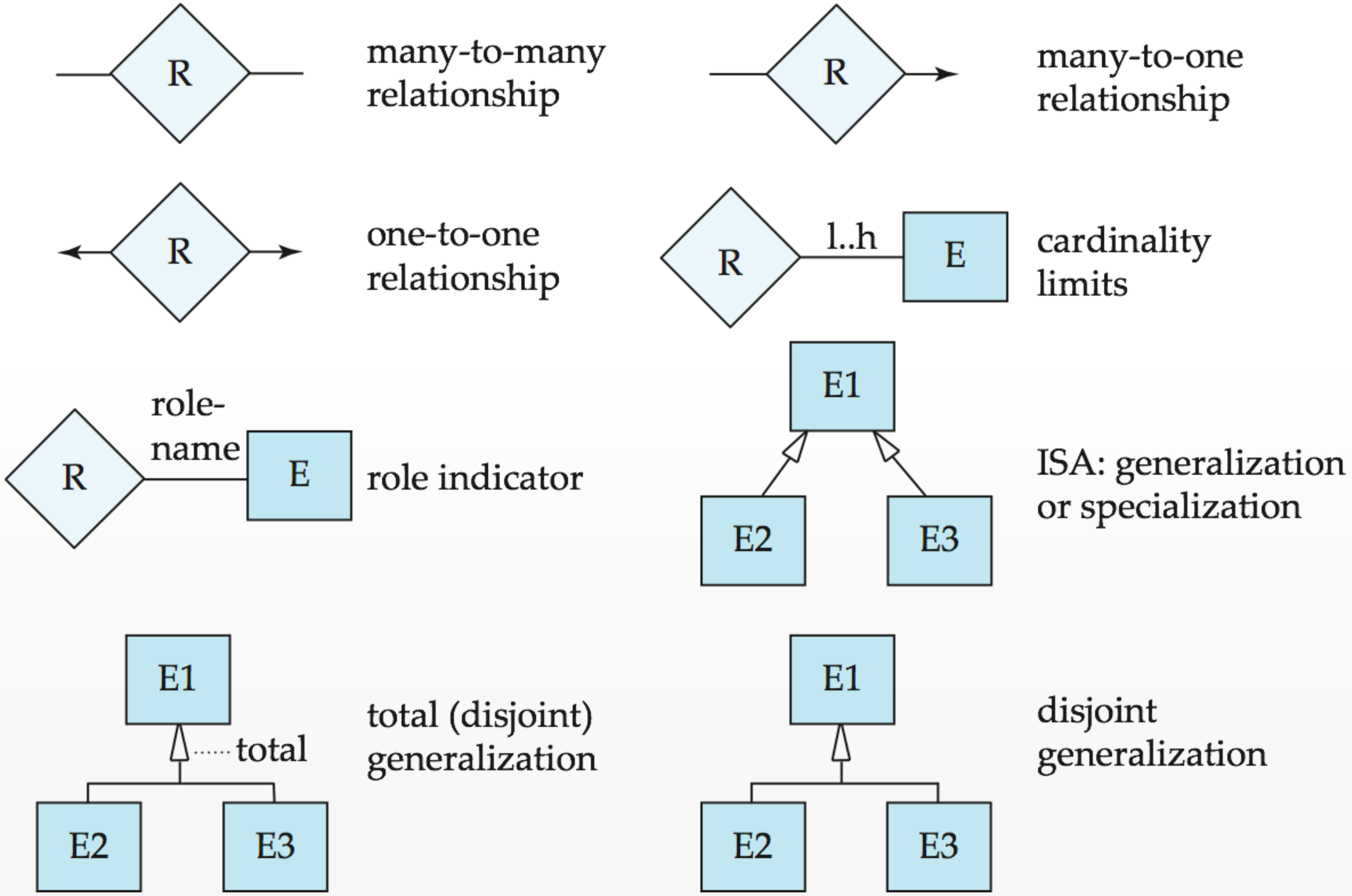


primary key



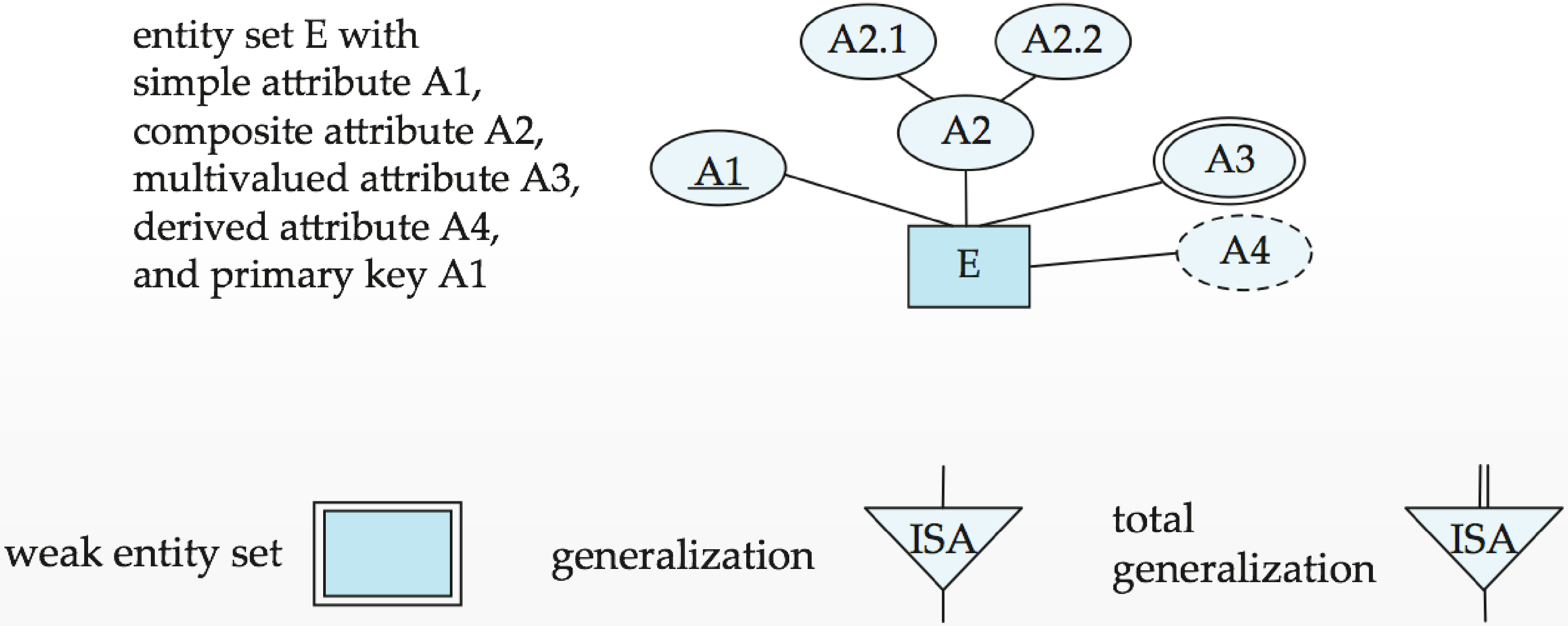
discriminating attribute of weak entity set

Symbols Used in ER Notation (Cont.)



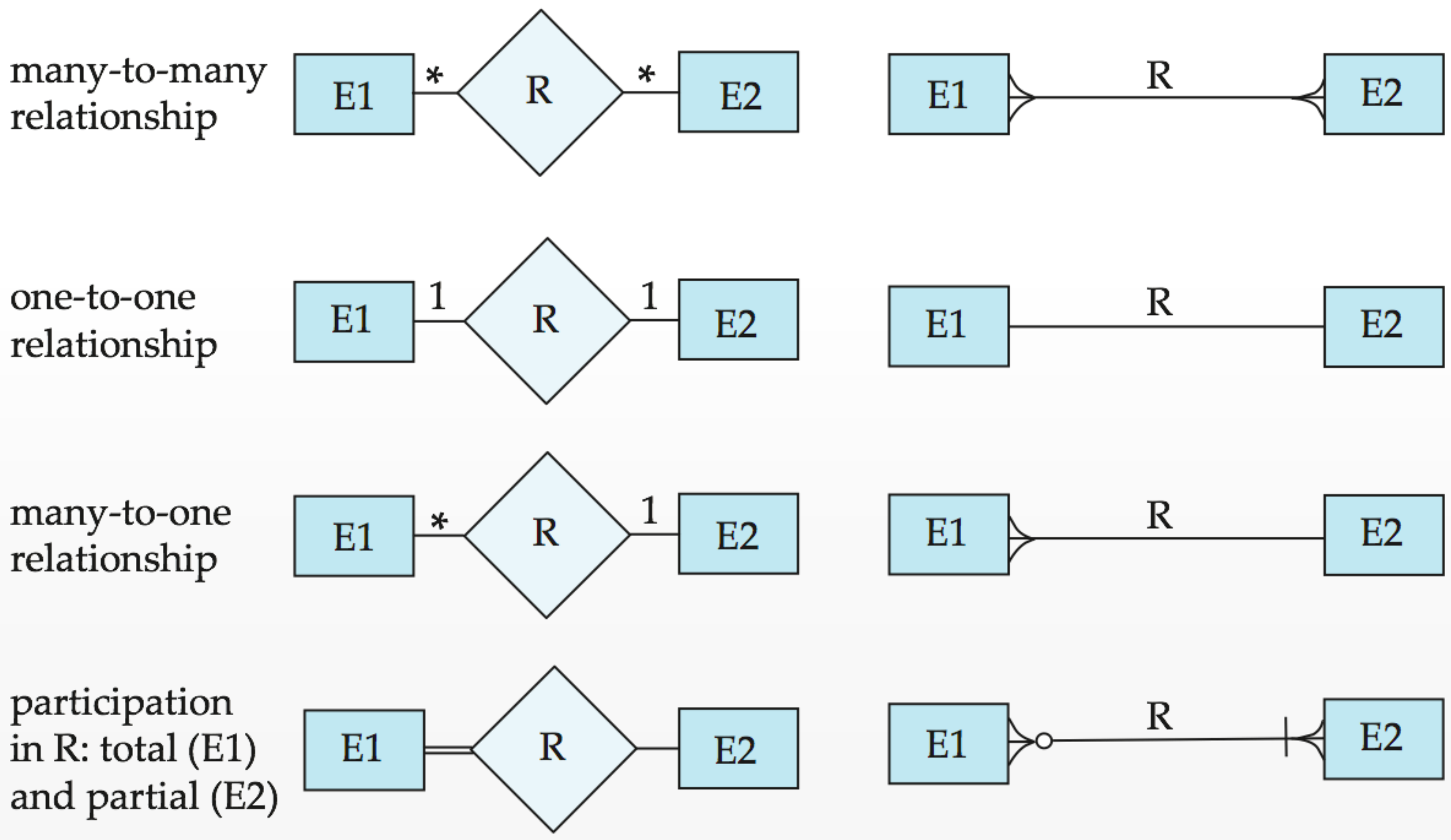
Alternative ER Notations

- Chen, IDE1FX, …



Alternative ER Notations (Cont.)

Chen **IDE1FX (Crows feet notation)**



References

- A. Silberschatz, HF. Korth, S. Sudarshan, Database System Concepts, 7th Ed., McGraw-Hill, 2019.
  - Chapter 6, https://www.db-book.com/db7/slides-dir/PPTX-dir/ch6.pptx (modified)
- A. Silberschatz, HF. Korth, S. Sudarshan, Database System Concepts, 6th Ed., McGraw-Hill, 2010.
  - Chapter 7, https://www.db-book.com/db6/slide-dir/PPT-dir/ch7.ppt (modified)