Database System

Lecture 13-15:
Entity-Relationship Model

Dr. Momtazi
momtazi@aut.ac.ir

Based on the slides of the course book

Outline

- **Design Process**
- E-R Data Modeling
- E-R Constraints
- Removing Redundancy
- E-R Diagram
- Reduction to Relation Schemas
- Redundancy of Schemas
- Extended E-R Features
- Design Issues
- Alternative Notions of Data Modeling

Database Design

- The task of creating a database application involves
  - Design of the database schema
  - Design of the programs that access and update the data
  - Design of a security scheme to control access to data

Design Phases

- Characterizing the data needs of the prospective database users
- Choosing a data model
- By applying the concepts of the chosen data model, translating the requirements into a conceptual schema of the database
- Reviewing the schema to ensure it meets functional requirements
  - A fully developed conceptual schema also indicates the functional requirements of the enterprise
  - In a “specification of functional requirements”, users describe the kinds of operations (or transactions) that will be performed on the data

Design Phases

The process of moving from an abstract data model to the implementation of the database proceeds in two final design phases.

- 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 Approaches

- Entity Relationship Model (current 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 8)
  - Formalize what designs are bad, and test for them

Outline

- Design Process
- **E-R Data Modeling**
- E-R Constraints
- Removing Redundancy
- E-R Diagram
- Reduction to Relation Schemas
- Redundancy of Schemas
- Extended E-R Features
- Design Issues
- Alternative Notions of Data Modeling

ER model -- Database Modeling

- The ER data model was developed to facilitate database design by allowing specification of an enterprise schema that represents the overall logical structure of a database.

- The ER model is very useful in mapping the meanings and interactions of real-world enterprises onto a conceptual schema. Because of this usefulness, many database-design tools draw on concepts from the ER model.

Major issues

- In designing a database schema, we must ensure that we avoid two major pitfalls:
  - **Redundancy**: A bad design may repeat information.

  Example: if we store the course identifier and title of a course with each course offering, the title would be stored redundantly with each course offering. It would suffice to store only the course identifier with each course offering, and to associate the title with the course identifier only once, in a course entity

  - **Incompleteness**: A bad design may make certain aspects of the enterprise difficult or impossible to model.

  Example: if we only have entities corresponding to sections, without having an entity corresponding to courses and repeat all of the course information once for each section, it would then be impossible to represent information about a new course, unless that course is offered.

ER model -- Database Modeling

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

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

Entity

- An **entity** is an object that exists and is distinguishable from other objects.
  - Example: instructors, students, departments, courses
- An **entity set** is a set of entities of the same type that share the same properties.
  - Example: set of all instructors, students, departments, courses

Attributes

- 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, street, city, 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.

Entity Sets -- *instructor* and *student*

instructor_ID instructor_name

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

instructor

student-ID student_name

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

student

Relationship

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

  Example: An instructor advising a student, a student taking an offered course, an instructor teaching an offered course

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

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

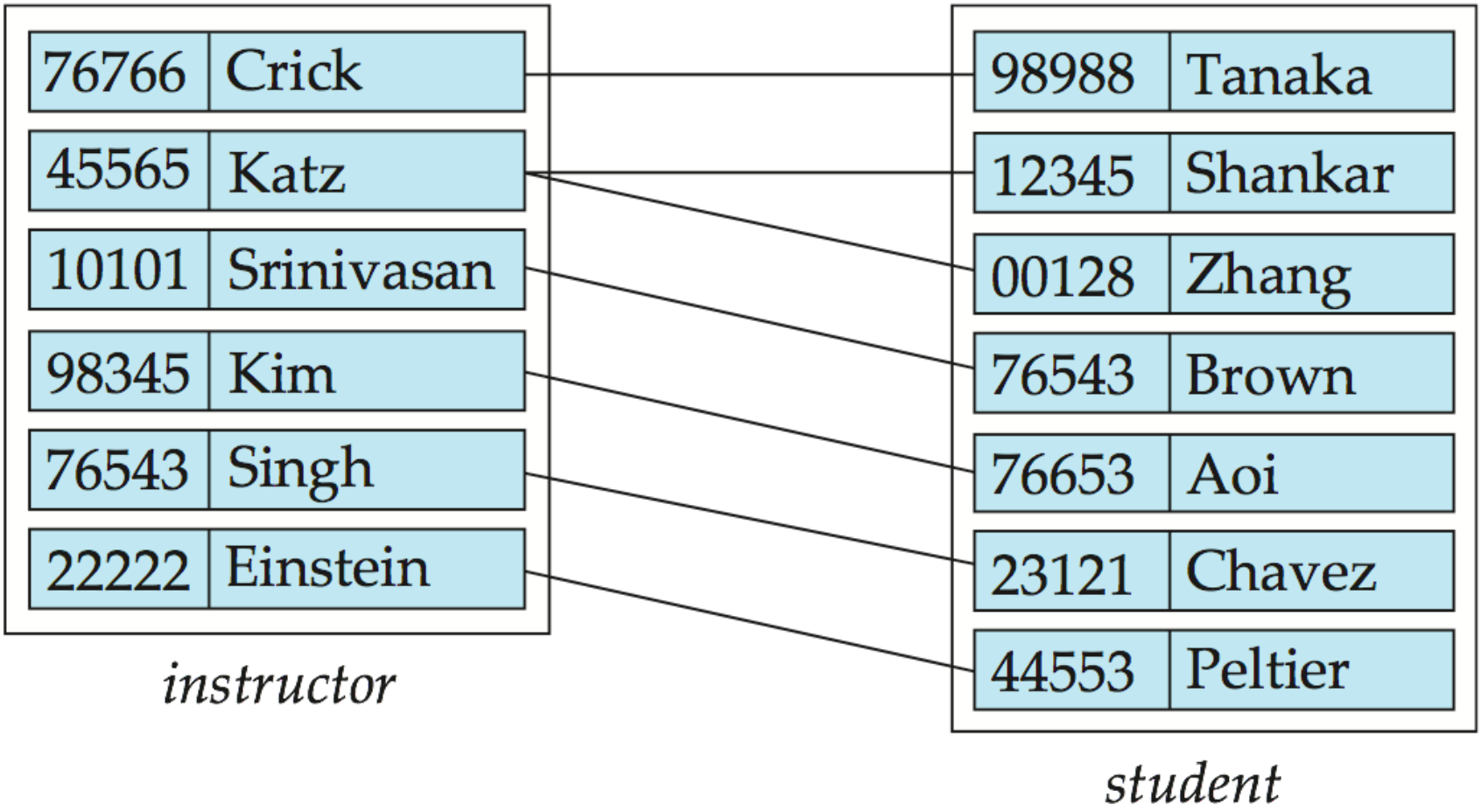
  $$\{(e_1, e_2, \ldots e_n) \mid e_1 \in E_1, e_2 \in E_2, \ldots, e_n \in E_n\}$$

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

  - Example:

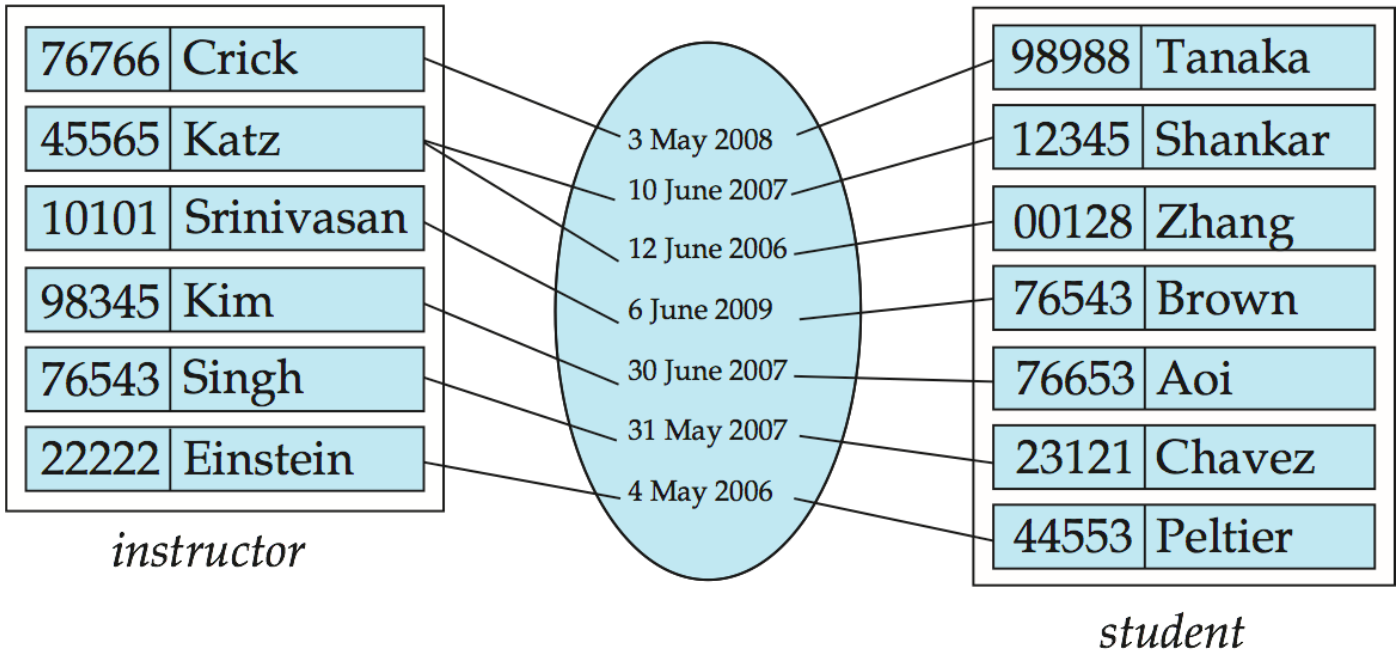
    $(44553,22222) \in advisor$

Relationship Set *advisor*



Descriptive Attributes

- An attribute can also be associated with a relationship set.
- Example: the *advisor* relationship set between entity sets *instructor* and *student* may have the attribute *date* which tracks when the student started being associated with the advisor



Degree of a Relationship Set

- Binary relationship
  - involve two entity sets (or degree two).
  - most relationship sets in a database system are binary.
  - Example: Advisor relationship between instructor and student

- Relationships involving more than two entity sets
  - Example: *students* work on research *projects* under the guidance of an *instructor*.
  - relationship *proj_guide* is a ternary relationship between *instructor, student,* and *project*

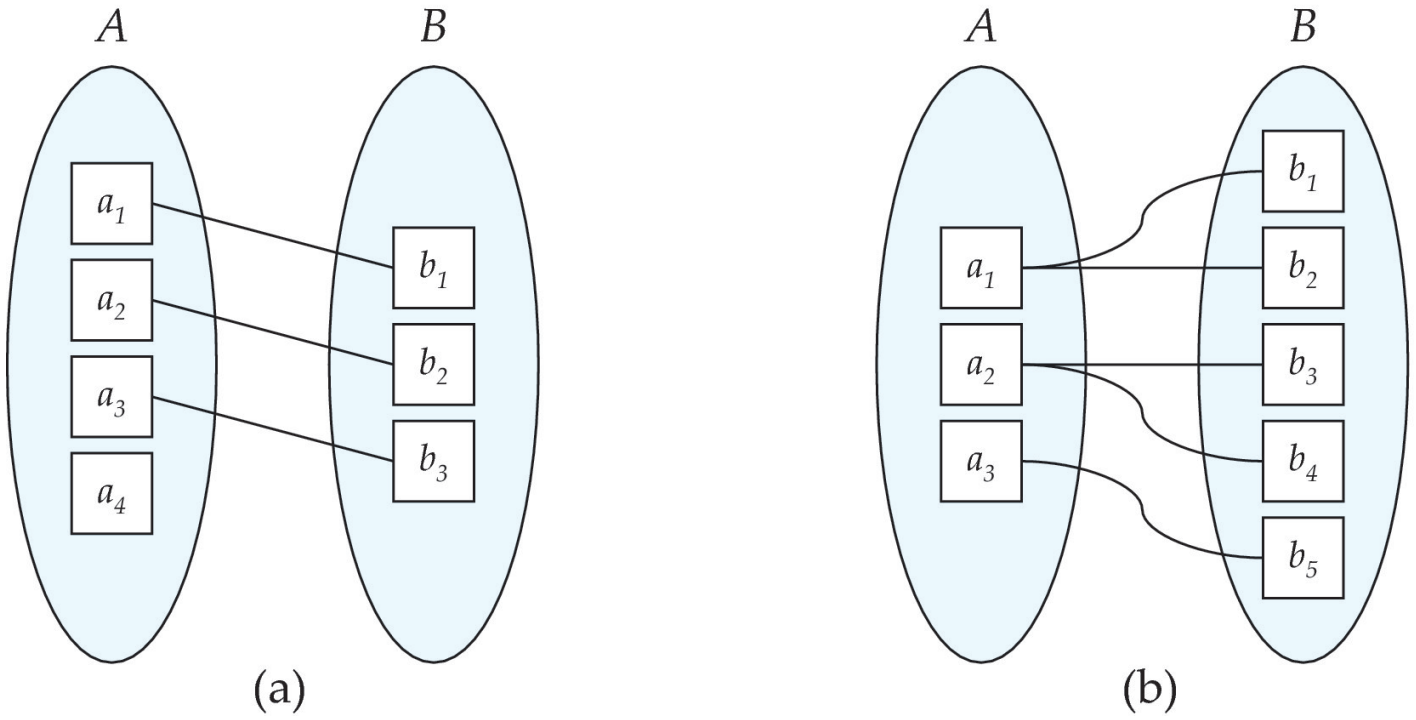
Outline

- Design Process
- E-R Data Modeling
- **E-R Constraints**
- Removing Redundancy
- E-R Diagram
- Reduction to Relation Schemas
- Redundancy of Schemas
- Extended E-R Features
- Design Issues
- Alternative Notions of Data Modeling

Mapping Cardinality Constraints

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

Mapping Cardinalities

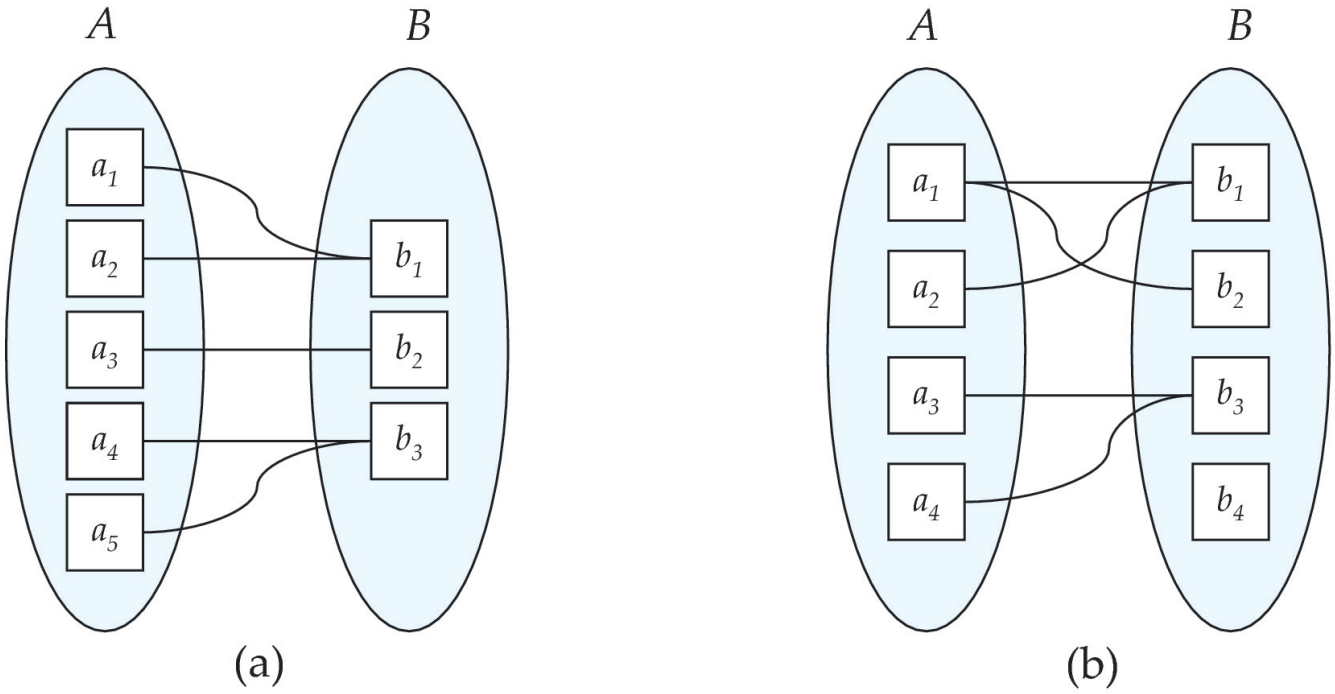


One to one

One to many

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

Mapping Cardinalities



Many to one

Many to many

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

Total and Partial Participation

- Total participation: every entity in the entity set participates in at least one relationship in the relationship set
  - Example: participation of *student* in *advisor r*elation is total
    - every *student* must have an associated instructor

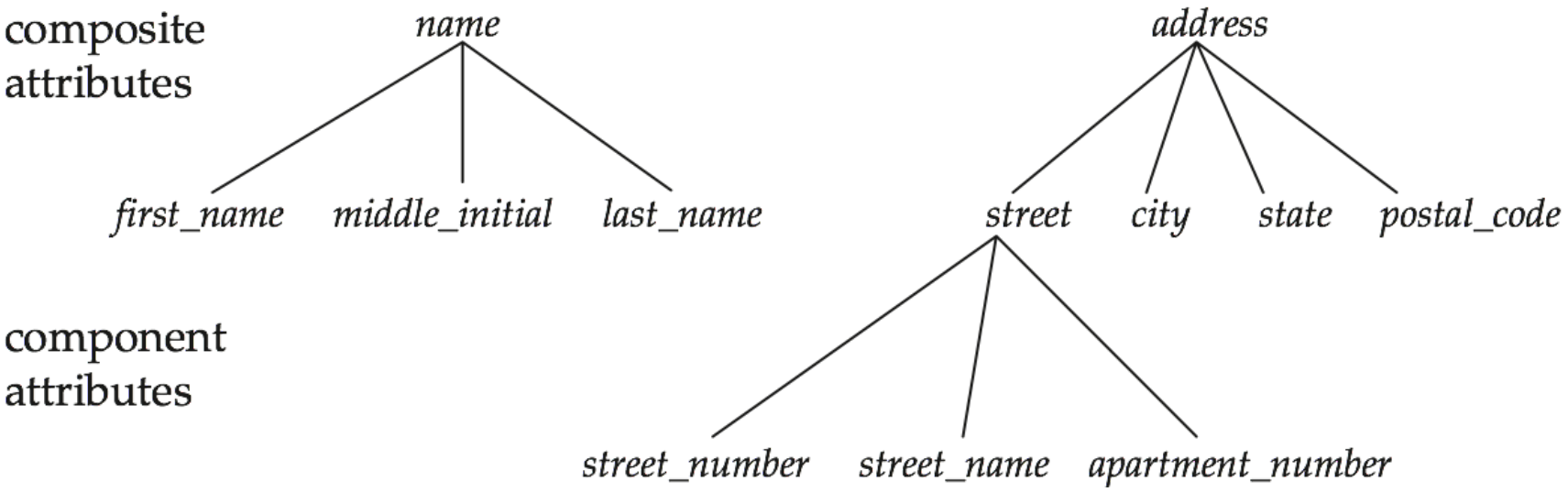
- Partial participation: some entities may not participate in any relationship in the relationship set
  - Example: participation of *instructor* in *advisor* is partial
    - *An instructor need not advise any* students
    - *It is possible that only some of the instructor entities are related to the* student *entity set through the advisor relationship*

Complex Attributes

- Attribute types:
  - **Simple** and **composite** attributes.
  - **Single-valued** and **multivalued** attributes
    - Example: multivalued attribute: *phone_numbers*
  - **Derived** attributes
    - Can be computed from other attributes
    - Example: age, given date_of_birth

- **Domain** – the set of permitted values for each attribute

Composite Attributes



Designing Attributes

- Once the entity sets are decided upon, we most choose the appropriate attributes.
- These attributes are supposed to represent the various values we want to capture in the database.
- The choice of what attributes to include is up to the designer, who has a good understanding of the structure of the enterprise.

- Example: for the instructor entity set,
  - we included the attributes ID, name, dept name, and salary.
  - We can also add the attributes phone number, office number, home_page, etc.

Outline

- Design Process
- E-R Data Modeling
- E-R Constraints
- **Removing Redundancy**
- E-R Diagram
- Reduction to Relation Schemas
- Redundancy of Schemas
- Extended E-R Features
- Design Issues
- Alternative Notions of Data Modeling

Redundant Attributes

- Suppose we have entity sets: *instructor* and *department*
- We should model the fact that each instructor has an associated department
- Two possible approaches:
  - Using the attribute *dept_name* appears in instructor.
    - *instructor: ID*, *name*, *dept_name, salary*
    - *department: dept_name, building, budget*
  - Using a relationship set *inst_dept*
    - *instructor: ID*, *name*, *salary*
    - *department: dept_name, building, budget*
    - *inst_dept: ID, dept_name*

    (the attribute *dept_name* in *instructor* is redundant and should be removed)

Weak Entity Sets

- Suppose we want to model the fact that each course is offered within one or more sections
- Two possible approaches:
  - Using the attribute *course_id* appears in *section*
  - Using the relationship set *sec_course* between entity sets *section* and *course*
    - *course_id* is redundant information appeared in *sec_course,* 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 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; different courses may share the same *section_id*, year, and semester.

Weak Entity Sets

- A **weak entity set** is one whose existence is dependent on another entity, called its **identifying entity**;
  - Example:
    - weak entity: *section*
    - identifying entity: *course*

- 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.**

Weak Entity Sets

- Every weak entity must be associated with an identifying entity
  - The weak entity set is said to be **existence dependent** on the identifying entity set.
  - The identifying entity set is said to **own** the weak entity set that it identifies.
  - The relationship associating the weak entity set with the identifying entity set is called the **identifying relationship**.

List of Entity Sets and their Attributes

- **classroom**: with attributes (*building*, *room_number*, *capacity*).
- **department**: with attributes (*dept_name*, *building*, *budget*).
- **course**: with attributes (*course_id*, *title*, *credits*).
- **instructor**: with attributes (*ID*, *name*, *salary*).
- **section:** with attributes (*course_id*, *sec_id*, *semester*, *year*).
- **student**: with attributes (*ID*, *name*, *tot_cred*).
- **time_slot**: with attributes (*time_slot_id*, {(*day*, *start_time*, *end_time*) }).

List of Relationships

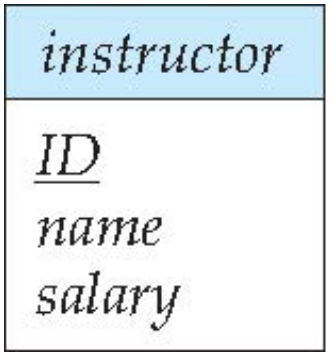
- **inst_dept**: relating instructors with departments.
- **stud_dept**: relating students with departments.
- **teaches**: relating instructors with sections.
- **takes**: relating students with sections, with a descriptive attribute *grade*.
- **course_dept**: relating courses with departments.
- **sec_course**: relating sections with courses.
- **sec_class**: relating sections with classrooms.
- **sec_time_slot**: relating sections with time slots.
- **advisor**: relating students with instructors.
- **prereq**: relating courses with prerequisite courses.

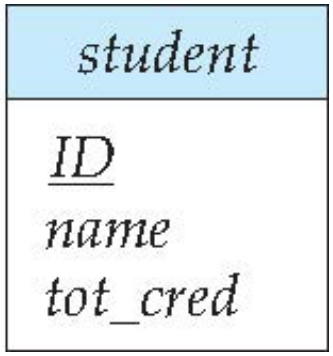
Outline

- Design Process
- E-R Data Modeling
- E-R Constraints
- Removing Redundancy
- **E-R Diagram**
- Reduction to Relation Schemas
- Redundancy of Schemas
- Extended E-R Features
- Design Issues
- Alternative Notions of Data Modeling

Entity Sets

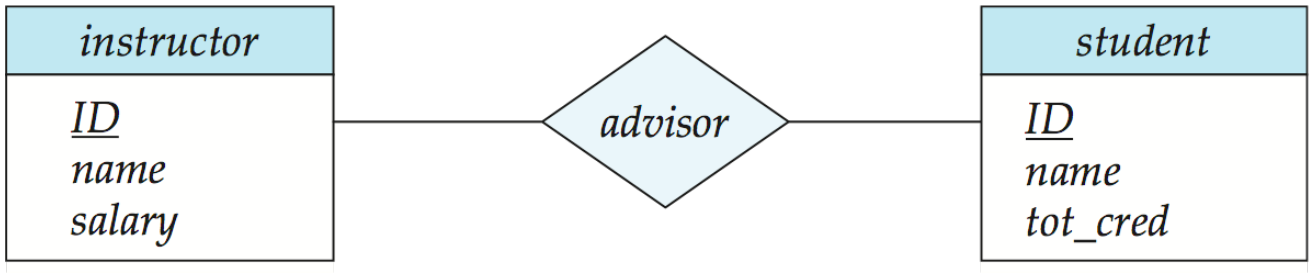
- Entities can be represented graphically as follows:
  - Rectangles represent entity sets.
  - Attributes listed inside entity rectangle
  - Underline indicates primary key attributes



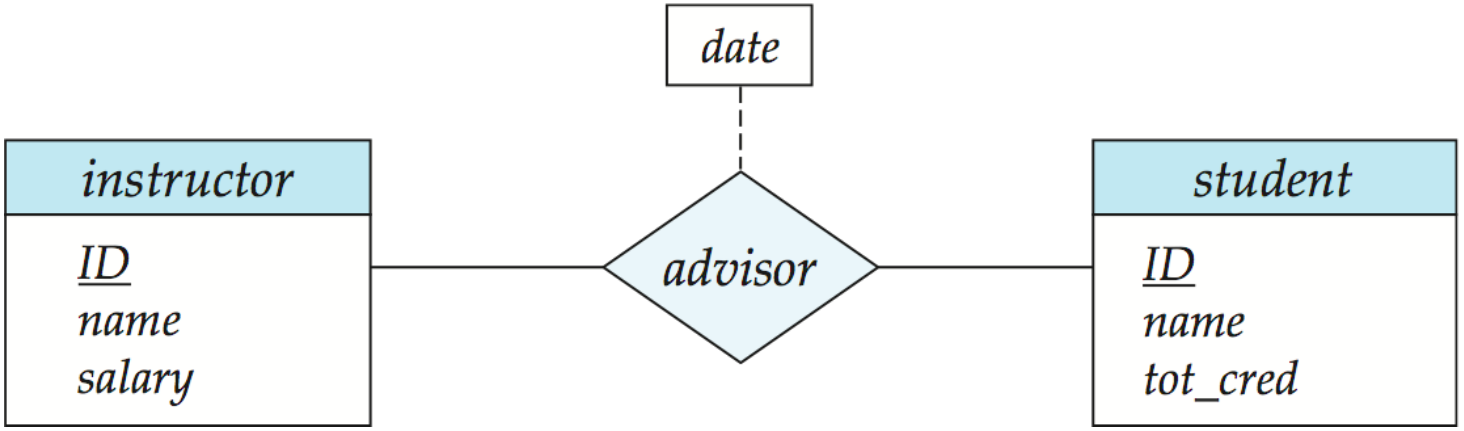


Relationship Sets

- Diamonds represent relationship sets.



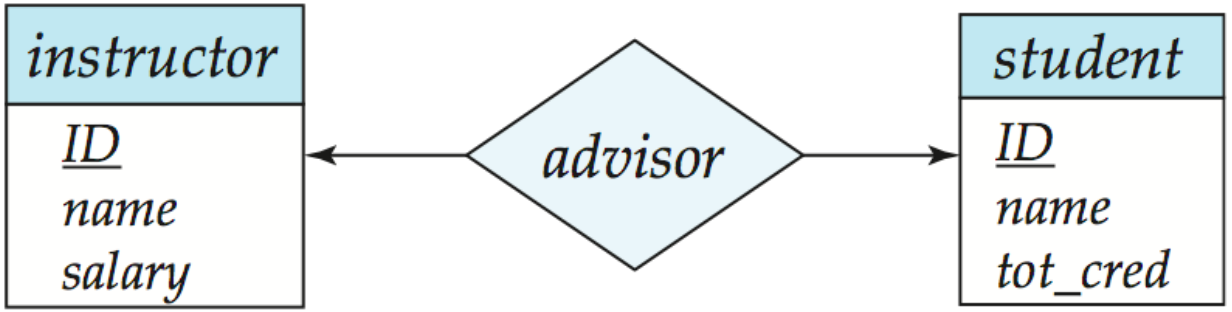
Relationship Sets with Attributes



Cardinality Constraints

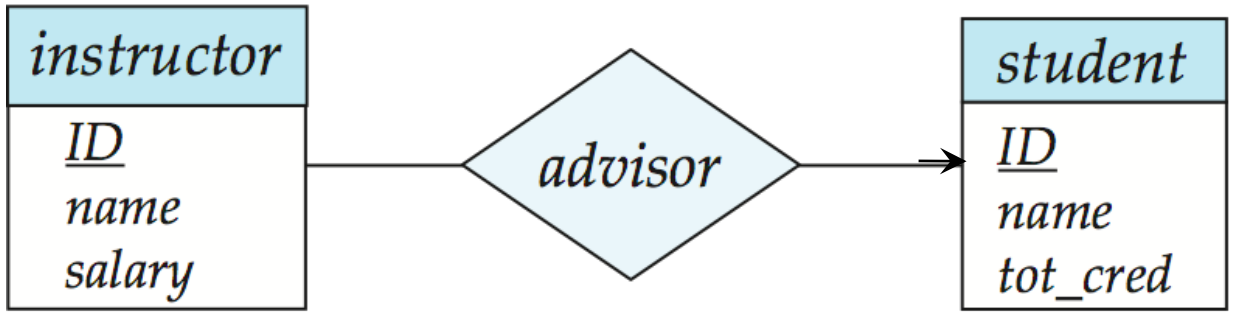
- The cardinality constraints are expressed by drawing either
  - a directed line (→), signifying “one” or
  - an undirected line (—), signifying “many”

- One-to-one relationship between an *instructor* and a *student* :
  - A student is associated with at most one *instructor* via the relationship *advisor*
  - An *instructor* is associated with at most one *student* via *advisor*



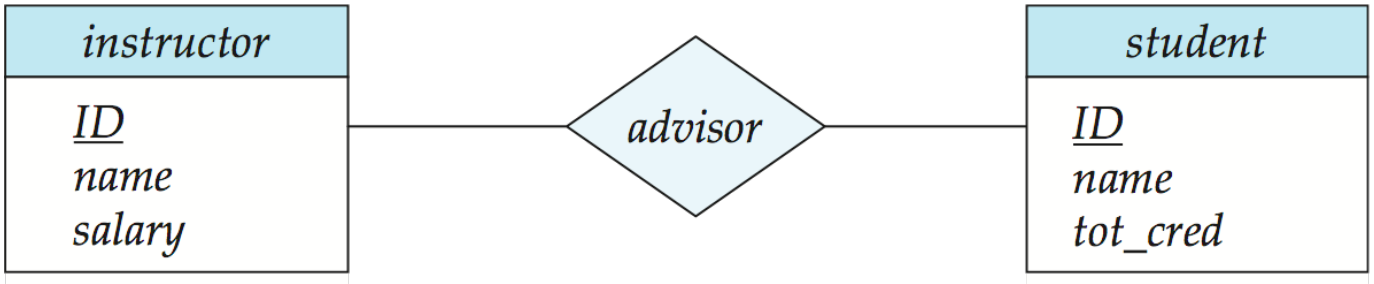
Many-to-One Relationships

- 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*



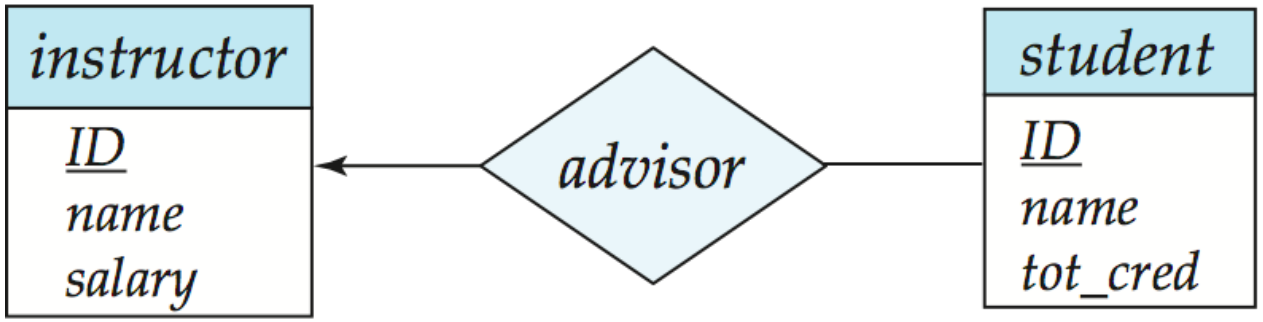
Many-to-Many Relationship

- many-to-many relationship between an *instructor* and a *student:*
  - An instructor is associated with several (possibly 0) students via *advisor*
  - A student is associated with several (possibly 0) instructors via *advisor*



One-to-Many Relationship

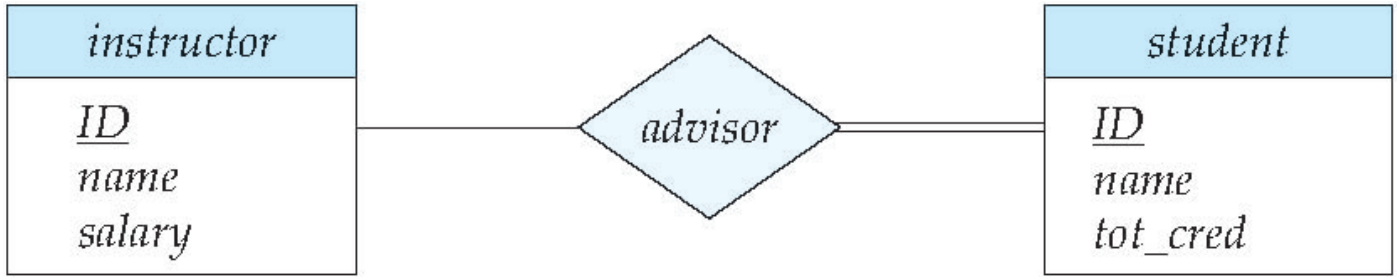
- one-to-many relationship between an *instructor* and a *student*
  - an instructor is associated with several (including 0) students via *advisor* (an instructor may advise many students)
  - a student is associated with at most one instructor via *advisor* (a student may have at most one advisor)



Total and Partial Participation

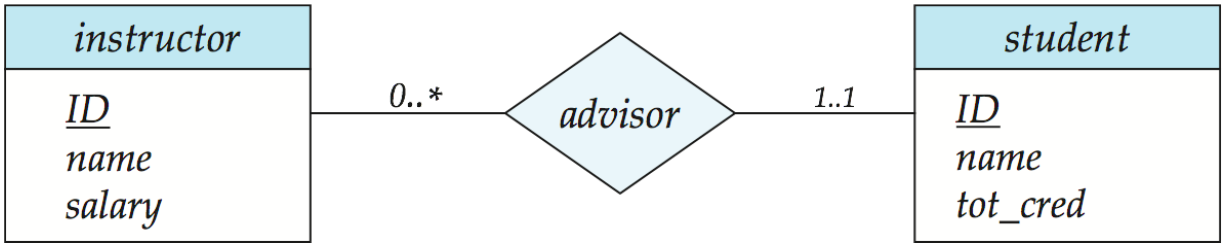
- Total participation (indicated by double line): every entity in the entity set participates in at least one relationship in the relationship set
  - every *student* must have an associated instructor

- Partial participation: some entities may not participate in any relationship in the relationship set
  - participation of *instructor* in *advisor* is partial



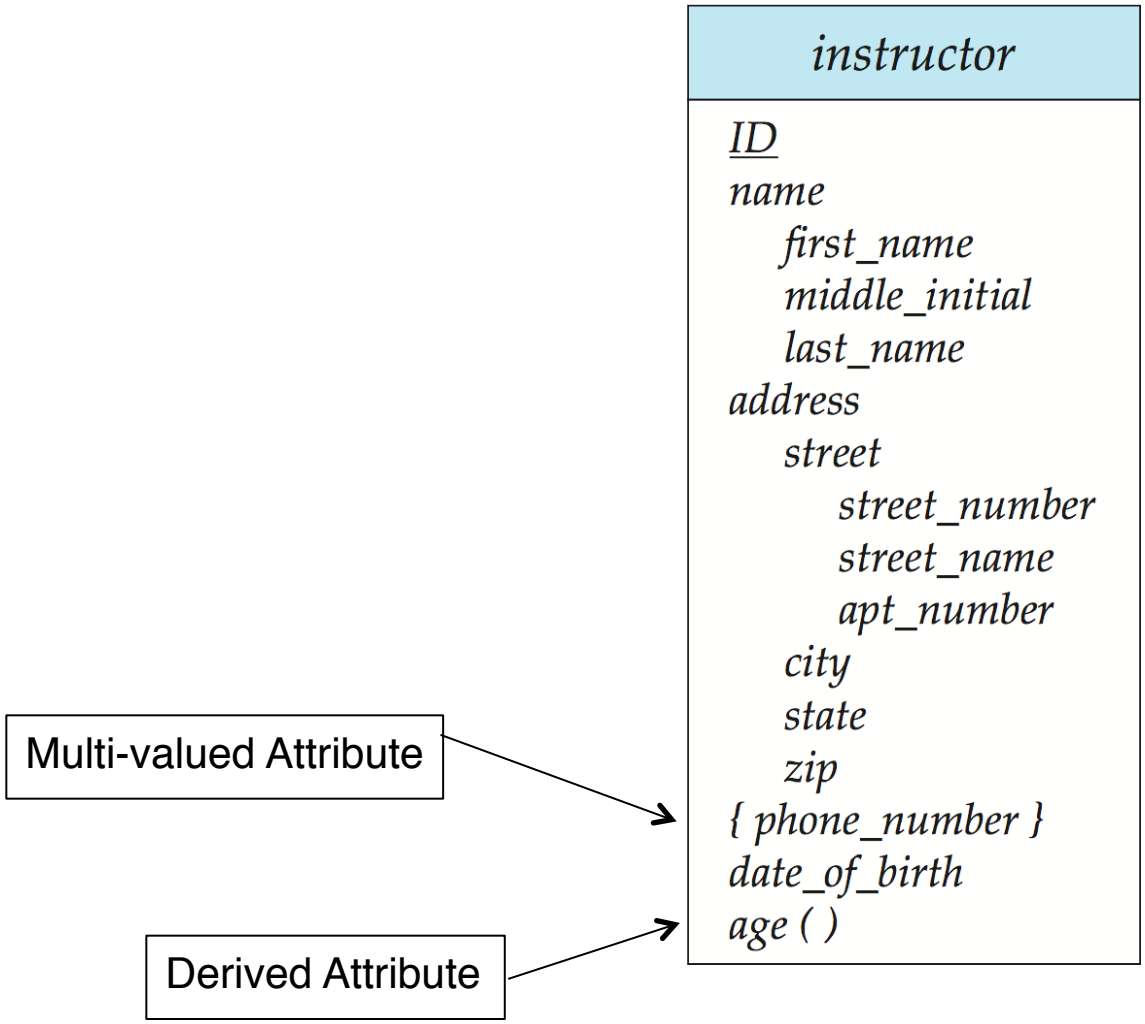
Notation for Expressing More Complex Constraints

- 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.



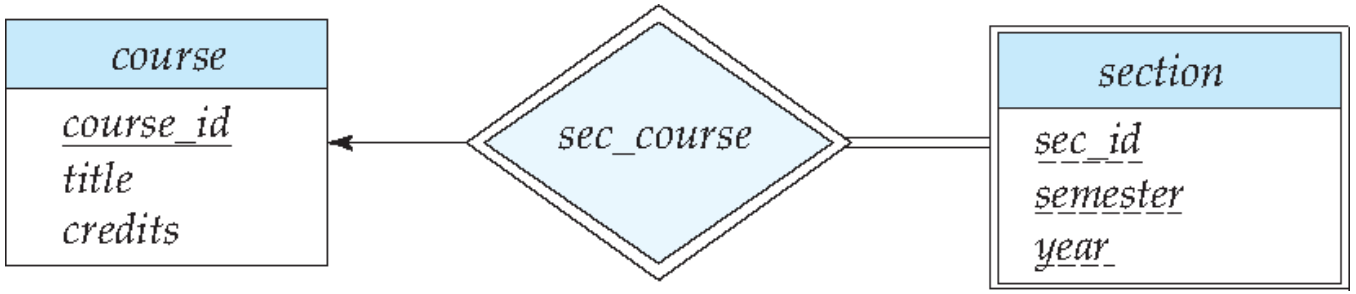
Instructor can advise 0 or more students. A student must have 1 advisor; cannot have multiple advisors

Notation to Express Entity with Complex Attributes



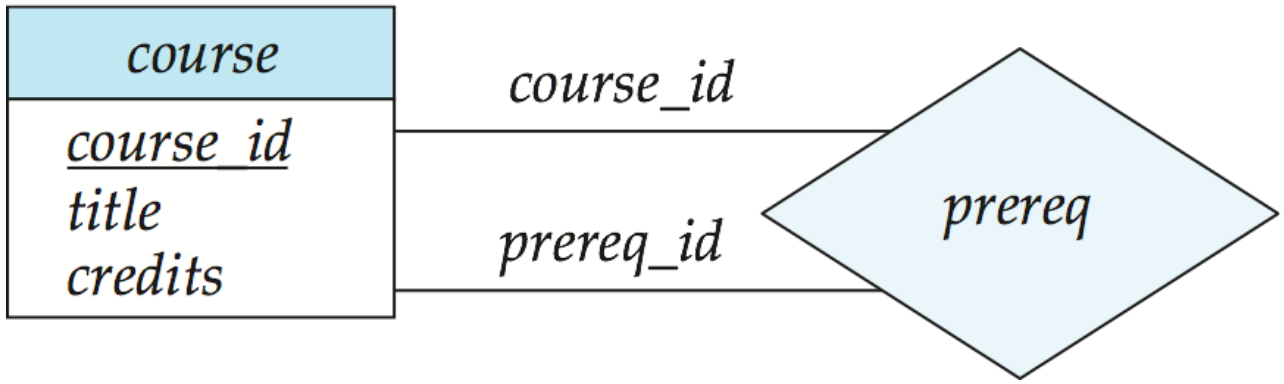
Expressing Weak Entity Sets

- A weak entity set is depicted via a double rectangle.
- The discriminator of a weak entity set is underlined 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*)



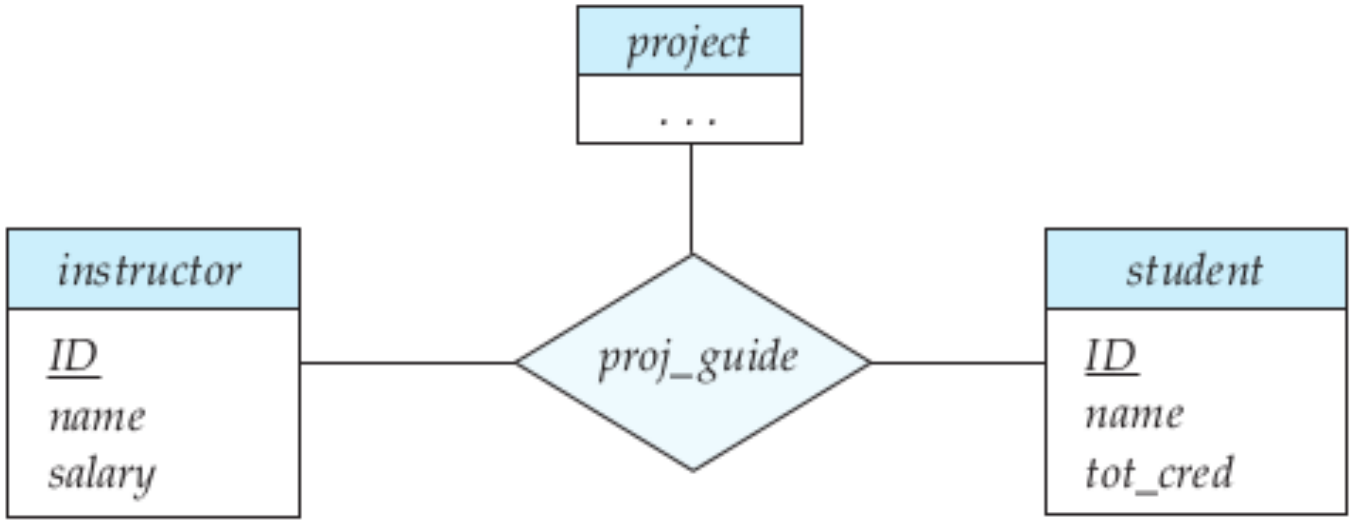
Roles

- 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**.



Non-binary Relationship Sets

- three entity sets *instructor*, *student*, and *project*, related through the relationship set *proj_guide*.



NOTE: arrows out of a nonbinary relationship set can be interpreted in different ways.

Cardinality Constraints on Ternary Relationship

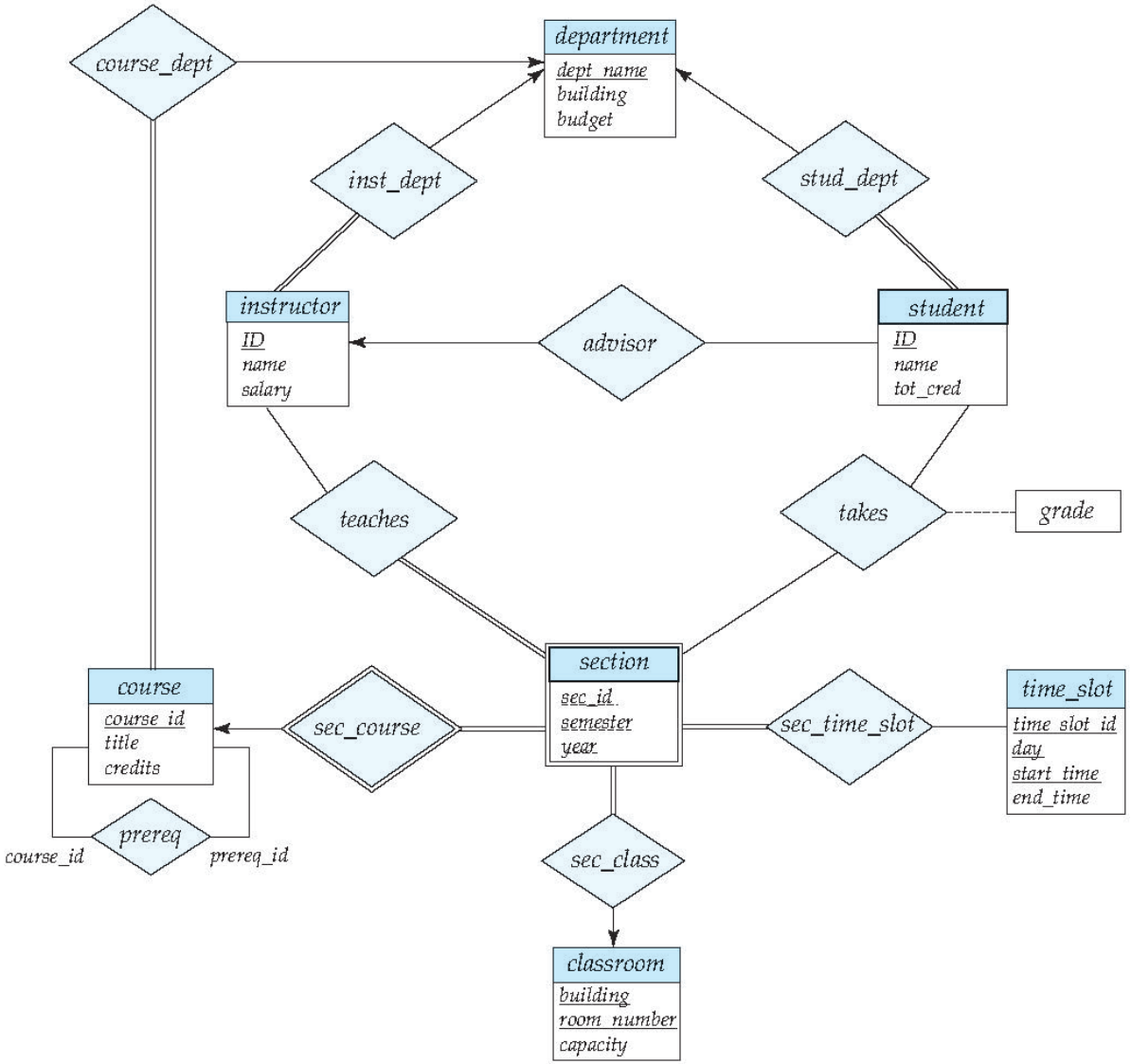
- We allow at most one arrow out of a ternary (or greater degree) relationship to indicate a cardinality constraint
- For example, an arrow from *proj_guide* to *instructor* indicates each student has at most one guide for a project
- If there is more than one arrow, there are two ways of defining the meaning.
  - For example, a ternary relationship *R* between *A*, *B* and *C* with arrows to *B* and *C* could mean

    1. Each *A* entity is associated with a unique entity from *B* and *C* or

    2. Each pair of entities from (*A, B*) is associated with a unique *C* entity, and each pair (*A, C*) is associated with a unique *B*

  - Each alternative has been used in different formalisms
  - To avoid confusion we outlaw more than one arrow

E-R Diagram for a University Enterprise



Outline

- Design Process
- E-R Data Modeling
- E-R Constraints
- Removing Redundancy
- E-R Diagram
- **Reduction to Relation Schemas**
- Redundancy of Schemas
- Extended E-R Features
- Design Issues
- Alternative Notions of Data Modeling

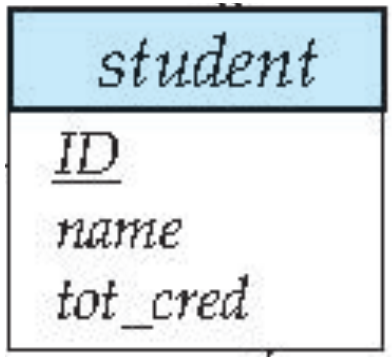
Reduction to Relation Schemas

- 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 relation schemas.
- For each entity set and relationship set there is a unique relation schema that is assigned the name of the corresponding entity set or relationship set.
- Each schema has a number of columns (generally corresponding to attributes), which have unique names.

Representing Entity Sets

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

  student(<u>*ID*</u>, *name*, *tot_cred*)



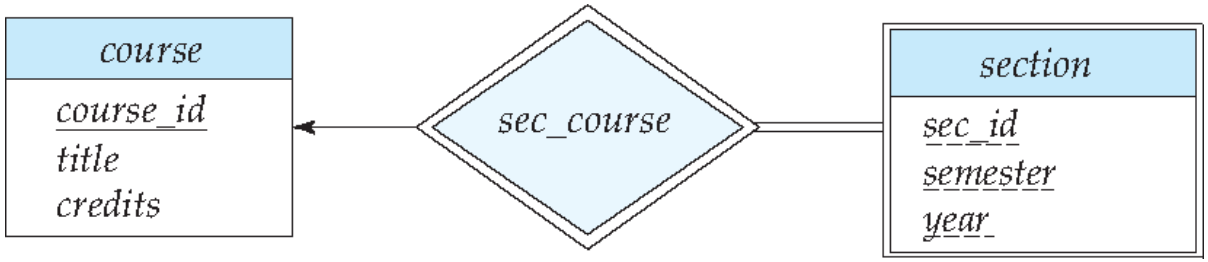
Representing Entity Sets

classroom (*building*, *room_number*, *capacity*)
department (*dept_name*, *building*, *budget*)
course (*course_id*, *title*, *credits*)
instructor (*ID*, *name*, *salary*)
student (*ID*, *name*, *tot_cred*)

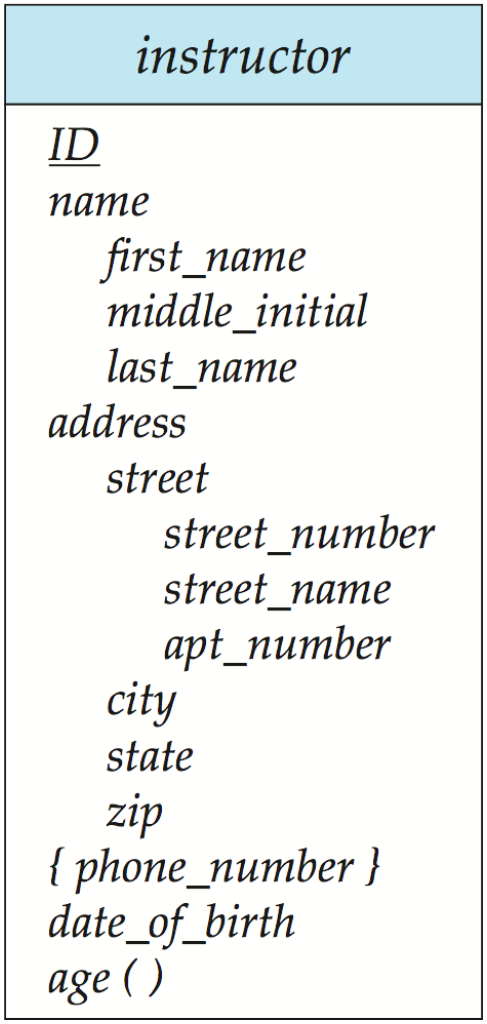
Representing Entity Sets

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

  section (course_id, sec_id, sem, year)



Representation of Entity Sets with Composite Attributes



- Composite attributes are flattened out by creating a separate attribute for each component attribute
  - Example: given entity set *instructor* with composite attribute *name* with component attributes *first_name* and *last_name* the schema corresponding to the entity set has two attributes *name_first_name* and *name_last_name*
    - Prefix omitted if there is no ambiguity (*name_first_name* could be *first_name)*
- Ignoring multivalued attributes, extended instructor schema is
  - *instructor(ID,*
    first_name, middle_initial, last_name,
    street_number, street_name,
    apt_number, city, state, zip_code,
    date_of_birth)

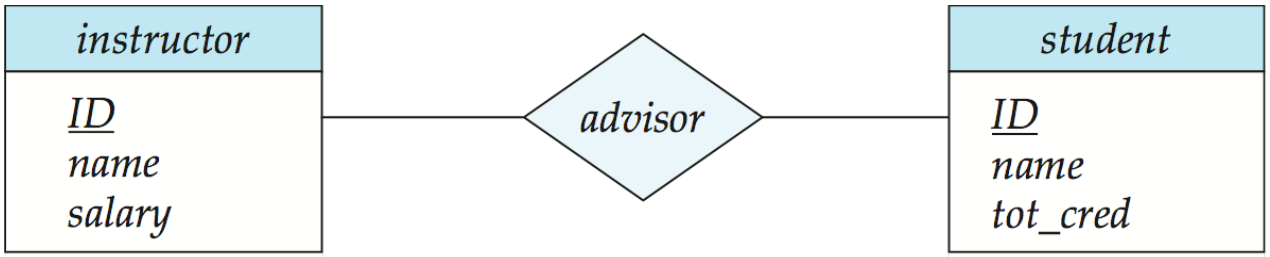
Representation of Entity Sets with Multivalued Attributes

- A multivalued attribute *M* of an entity *E* is represented by a separate schema *EM*
- Schema *EM* has attributes corresponding to the primary key of *E* and an attribute corresponding to multivalued attribute *M*
- Example: Multivalued attribute *phone_number* of *instructor* is represented by a schema:
  inst_phone= (*ID*, *phone_number*)
- Each value of the multivalued attribute maps to a separate tuple of the relation on schema *EM*
  - For example, an *instructor* entity with primary key 22222 and phone numbers 456-7890 and 123-4567 maps to two tuples:
    (22222, 456-7890) and (22222, 123-4567)

Representing Relationship Sets

- A binary 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*)



Representing Relationship Sets

- For a binary one-to-one relationship set, the primary key of either entity set can be chosen as the primary key. The choice can be made arbitrarily.
- For a binary many-to-one or one-to-many relationship set, the primary key of the entity set on the “many” side of the relationship set serves as the primary key.
- For an n-ary relationship set without any arrows on its edges, the union of the primary key-attributes from the participating entity sets becomes the primary key.
- For an n-ary relationship set with an arrow on one of its edges, the primary keys of the entity sets not on the “arrow” side of the relationship set serve as the primary key for the schema. Recall that we allowed only one arrow out of a relationship set.

Representing Relationship Sets

teaches (<u>ID</u>, <u>course_id</u>, <u>sec_id</u>, <u>semester</u>, <u>year</u>)
takes (<u>ID</u>, <u>course_id</u>, <u>sec_id</u>, <u>semester</u>, <u>year</u>, *grade*)
prereq (<u>course_id</u>, <u>prereq_id</u>)
advisor (<u>s_ID</u>, *i_ID*)
sec_course (<u>course_id</u>, <u>sec_id</u>, <u>semester</u>, <u>year</u>)
sec_time_slot (<u>course_id</u>, <u>sec_id</u>, <u>semester</u>, <u>year</u>, *time_slot_id*)
sec_class (<u>course_id</u>, <u>sec_id</u>, <u>semester</u>, <u>year</u>, *building*, *room_number*)
inst_dept (<u>ID</u>, *dept_name*)
stud_dept (<u>ID</u>, *dept_name*)
course_dept (<u>course_id</u>, *dept_name*)

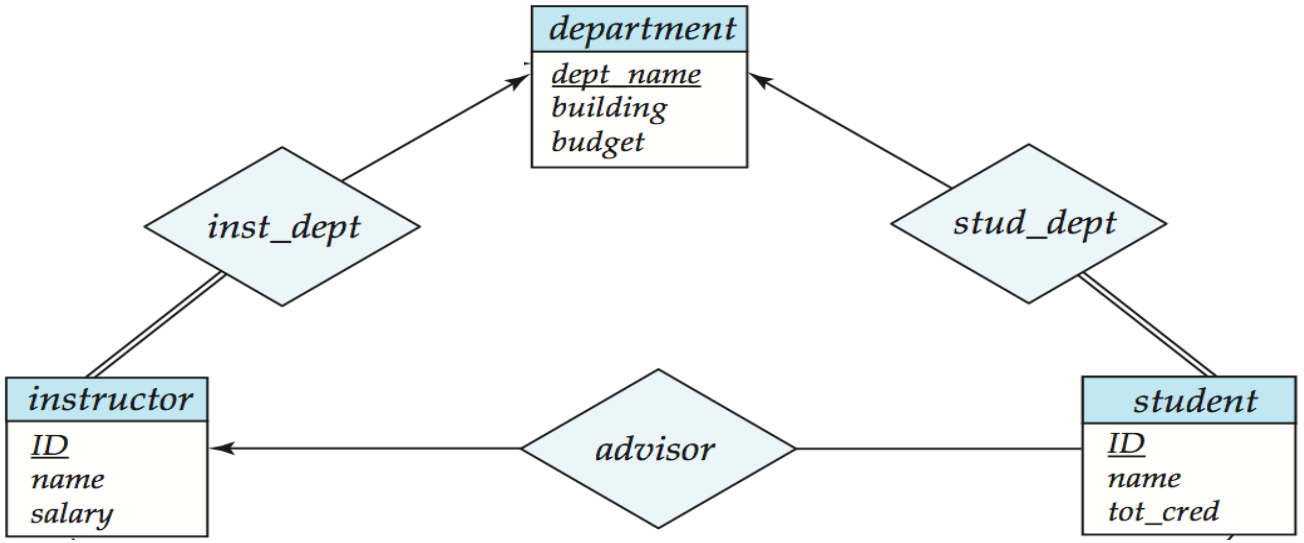
Outline

- Design Process
- E-R Data Modeling
- E-R Constraints
- Removing Redundancy
- E-R Diagram
- Reduction to Relation Schemas
- **Redundancy of Schemas**
- Extended E-R Features
- Design Issues
- Alternative Notions of Data Modeling

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* or *stud_dept*, add an attribute *dept_name* to the schema arising from entity set *instructor* or *student*



Redundancy of Schemas

- 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

- 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

Redundancy of Schemas

- The redundant relations in the E-R diagram:
  - *inst_dept*

    => instructor: {ID, name, dept_name, salary}
  - *stud_dept.*

    => student: { ID, name, dept_name, tot_cred}
  - *course_dept.*

    => course: {course_id, title, dept_name, credits}
  - *sec_class*

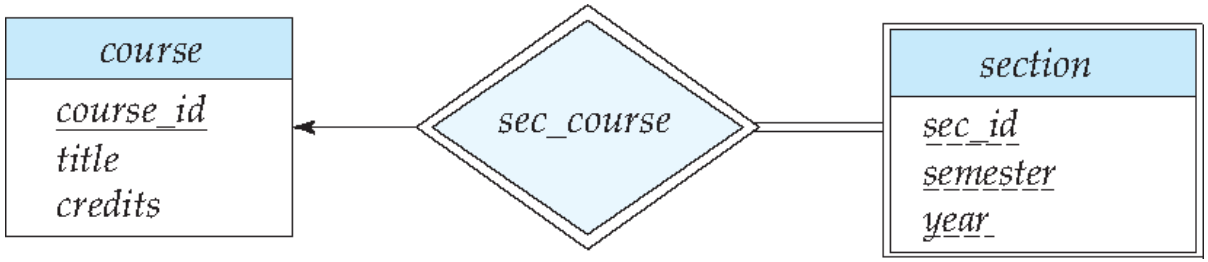
    => section: {course_id, sec_id, semester, year, building, room_number}.
  - *sec_time_slot*

    => section: {course_id, sec_id, semester, year, building, room_number, time_slot_id}

Redundancy of Schemas

- 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



Outline

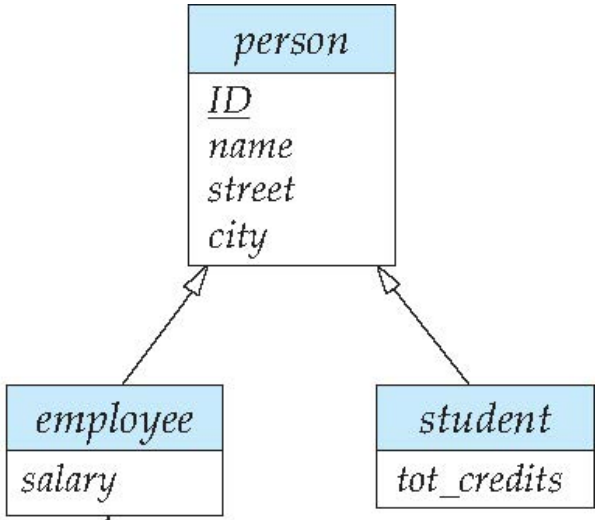
- Design Process
- E-R Data Modeling
- E-R Constraints
- Removing Redundancy
- E-R Diagram
- Reduction to Relation Schemas
- Redundancy of Schemas
- **Extended E-R Features**
- Design Issues
- Alternative Notions of Data Modeling

Specialization

- An entity set may include subgroupings of entities that are distinct from other entities in the set. (e.g., some attributes that are not shared by all the entities in the entity 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.
- The process of designating subgroupings within an entity set is called specialization.
- Depicted by a *triangle* component labeled ISA (e.g., *instructor* "is a" *person*).
- **Attribute inheritance** – a lower-level entity set inherits all the attributes and relationship participation of the higher-level entity set to which it is linked.
- Higher- and lower-level entity sets also may be designated by the terms **superclass** and **subclass**,

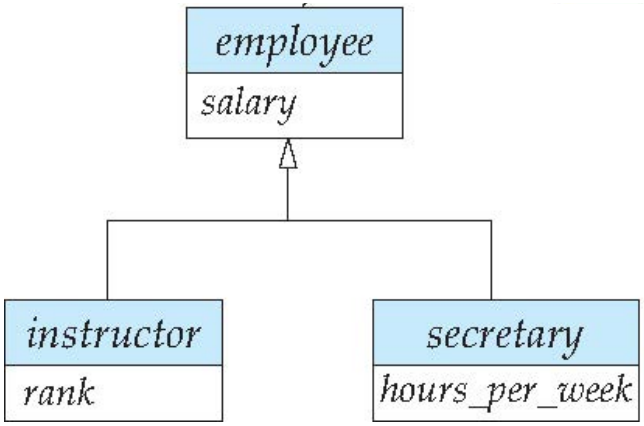
Specialization Example

- the entity set person may be further classified as:
  - employee
  - *student*



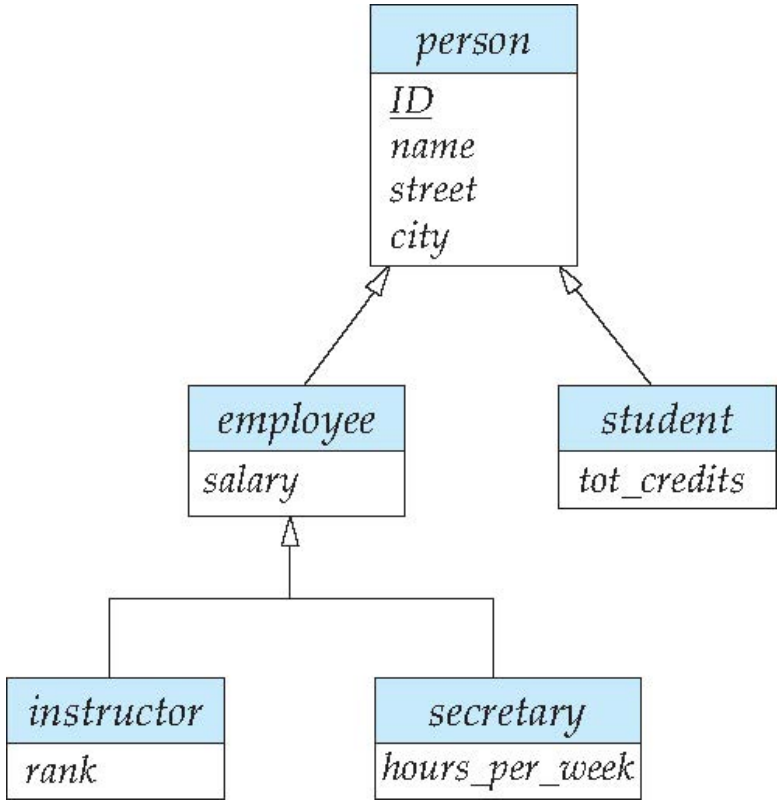
Specialization Example

- university employees may be further classified as:
  - instructor
  - secretary



Specialization Example

- **Overlapping**
  - An entity may belong to multiple specialized entity sets
  - Example: *employee* and *student*
- **Disjoint**
  - Entities must belong to at most one specialized entity set
  - *instructor* and *secretary*



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

Generalization

- **A bottom-up design process** – combine a number of entity sets that share the same features into a higher-level entity set.
- Specialization and generalization are simple inversions of each other; they are represented in an E-R diagram in the same way.
- The terms specialization and generalization are used interchangeably.
- Differences in the two approaches may be characterized by their starting point and overall goal.
- Whether a given portion of an E-R model was arrived at by specialization or generalization, the outcome is basically the same

Attribute Inheritance

- A crucial property of the higher- and lower-level entities created by specialization and generalization is **attribute inheritance**.
- The attributes of the higher-level entity sets are inherited by the lower-level entity sets.
- Attribute inheritance applies through all tiers of lower-level entity sets; thus, *instructor* and *secretary*, which are subclasses of employee, inherit all the attributes from *person*, in addition to inheriting attributes from *employee*.

Relationship Inheritance

- A lower-level entity set (or subclass) also inherits participation in the relationship sets in which its higher-level entity (or superclass) participates.

- Like attribute inheritance, participation inheritance applies through all tiers of lower-level entity sets.
  - Example: if the *person* entity set participates in a relationship *person_dept* with *department*. Then, the *student*, *employee*, *instructor* and *secretary* entity sets, which are subclasses of the *person* entity set, also implicitly participate in the *person_dept* relationship with *department*. The above entity sets can participate in any relationships in which the *person* entity set participates.

Design Constraints on a 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

Design Constraints on a Specialization/Generalization

- 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).

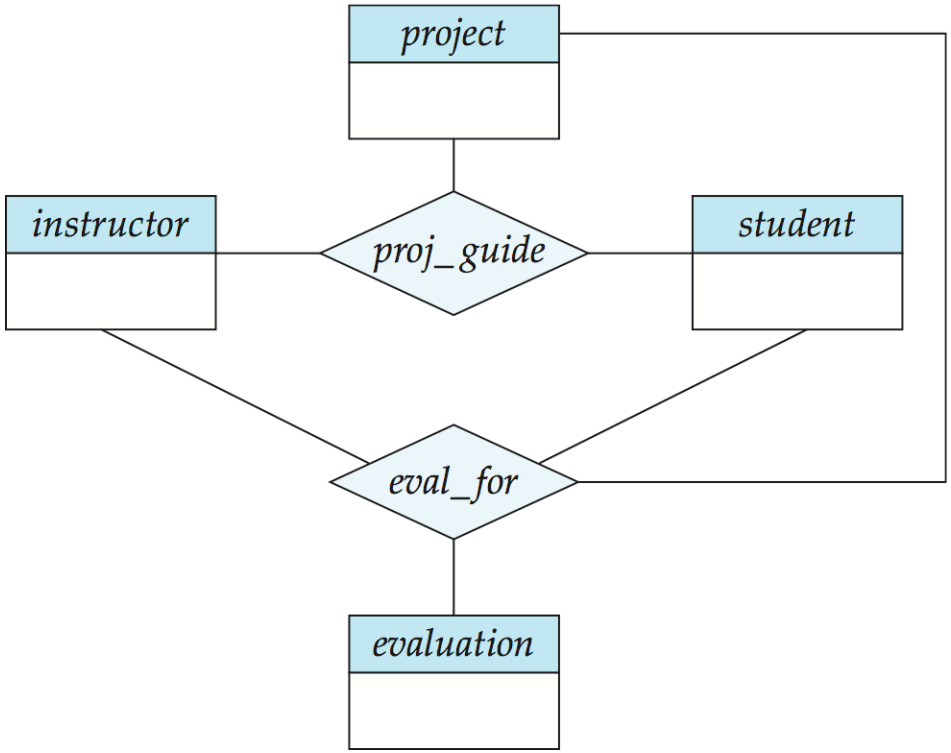
- The completeness constraint for a generalized higher-level entity set is usually total. Because the higher-level entity set arrived at through generalization is generally composed of only those entities in the lower-level entity sets.
  - Example: the *student* generalization is total; i.e., all student entities must be either graduate or undergraduate.

Design Constraints on a Specialization/Generalization

- Certain insertion and deletion requirements follow from the constraints that apply to a given generalization or specialization.
- When a total completeness constraint is in place, an entity inserted into a higher-level entity set must also be inserted into at least one of the lower-level entity sets.
- An entity that is deleted from a higher-level entity set also is deleted from all the associated lower-level entity sets to which it belongs.

Aggregation

- Consider the ternary relationship *proj_guide*, which we saw earlier
- Suppose we want to record monthly evaluation reports of a student by a guide on a project

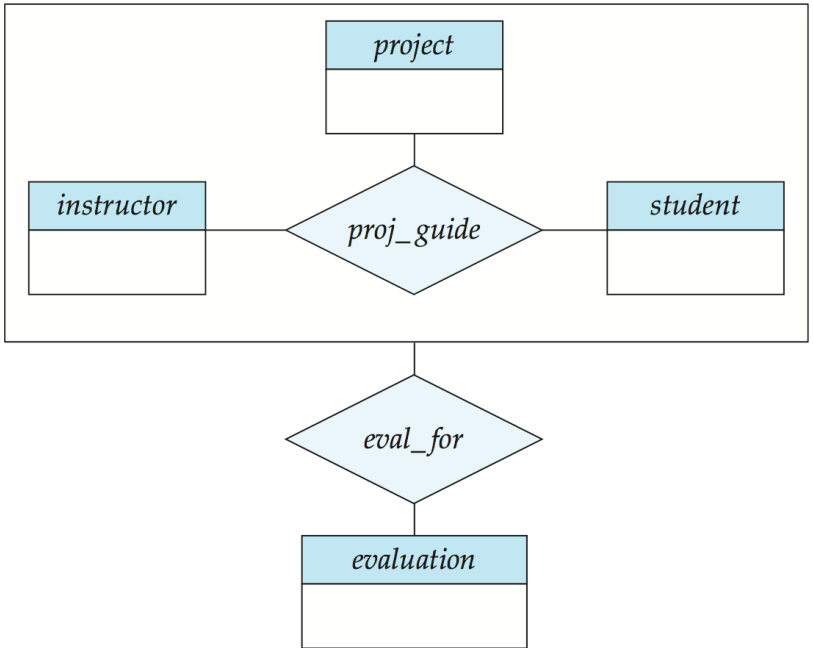


Aggregation

- Relationship sets *eval_for* and *proj_guide* represent overlapping information
  - Every *eval_for* relationship corresponds to a *proj_guide* relationship
  - However, some *proj_guide* relationships may not correspond to any *eval_for* relationships
    - So we can't discard the *proj_guide* relationship
- Eliminate this redundancy via *aggregation*
  - Treat relationship as an abstract entity
  - Allows relationships between relationships
  - Abstraction of relationship into new entity

Aggregation

- Eliminate this redundancy via *aggregation* without introducing redundancy, the following diagram represents:
  - A student is guided by a particular instructor on a particular project
  - A student, instructor, project combination may have an associated evaluation



Representing Aggregation via Schemas

- To represent aggregation, create a schema containing
  - Primary key of the aggregated relationship

  (No separate relation is required to represent the aggregation; the relation created from the defining relationship is used instead.)
  - The primary key of the associated entity set
  - Any descriptive attributes

- In our example:
  - The schema *evaluation* is:

    evaluation (evaluation_id, report)
  - The schema *eval_for* is:

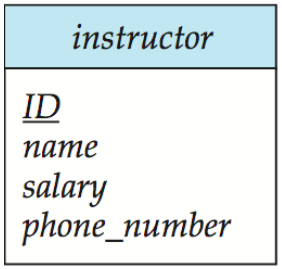
    eval_for (*s_ID, project_id, i_ID, evaluation_id*)

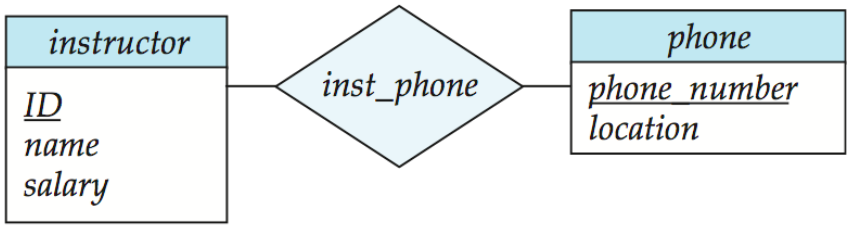
Outline

- Design Process
- E-R Data Modeling
- E-R Constraints
- Removing Redundancy
- E-R Diagram
- Reduction to Relation Schemas
- Redundancy of Schemas
- Extended E-R Features
- **Design Issues**
- Alternative Notions of Data Modeling

Entities vs. Attributes

- Use of entity sets vs. attributes

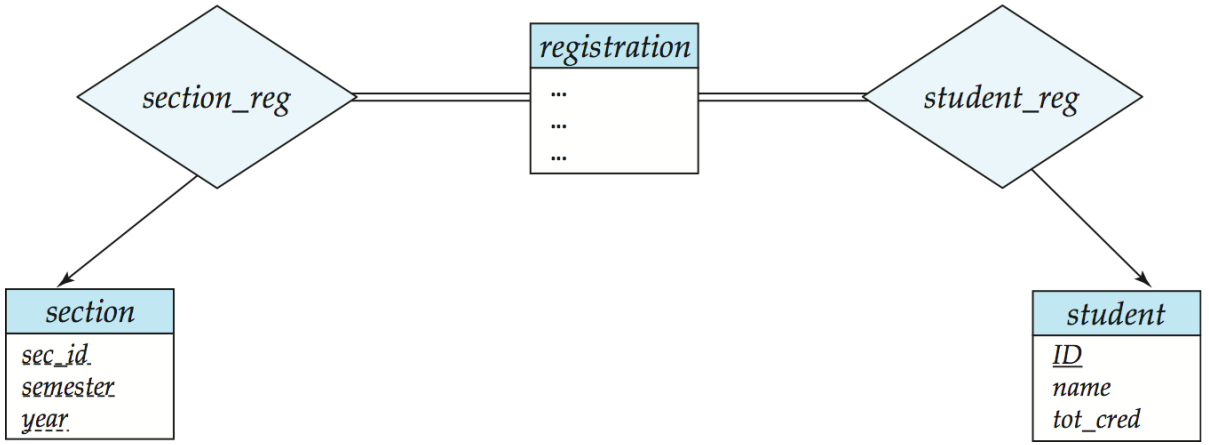




- Use of phone as an entity allows extra information about phone numbers (plus multiple phone numbers)

Entities vs. Relationship sets

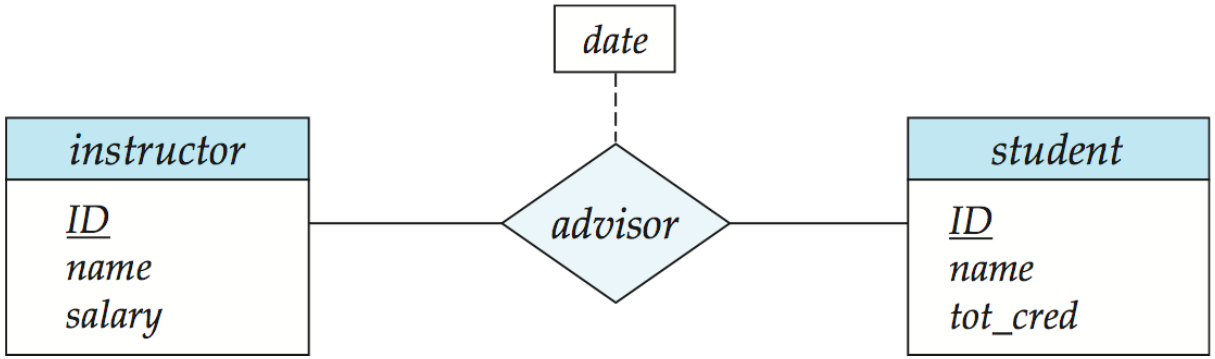
- Use of entity sets vs. relationship sets

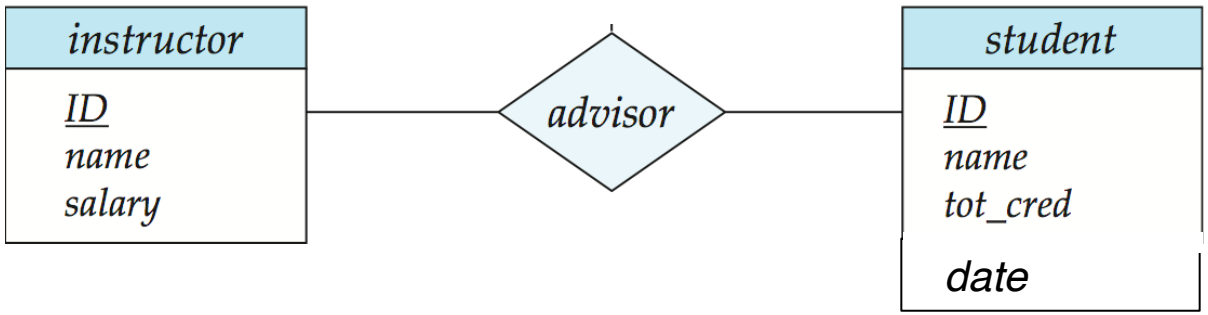


- Possible guideline is to designate a relationship set to describe an action that occurs between entities

Placement of Relationship Attributes

- **Placement of relationship attributes**
  - Example: attribute date as attribute of advisor or as attribute of student?





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
  - But there are some relationships that are naturally non-binary
    - Example: *proj_guide*

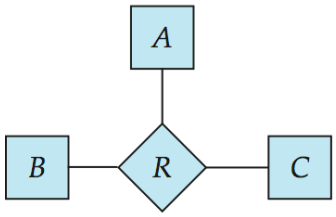
Converting Non-Binary Relationships to Binary Form

- 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:

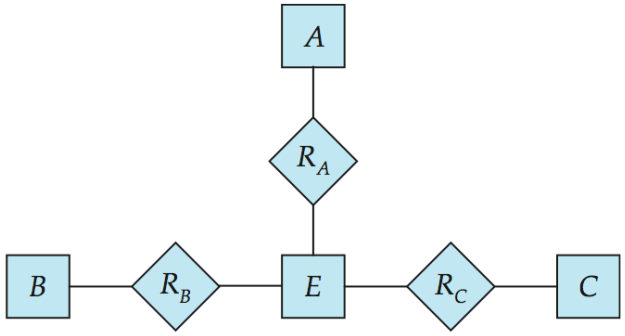
    1. R_A, relating *E* and *A*
    2. R_B, relating *E* and *B*
    3. R_C, relating *E* and *C*

  - Create an identifying attribute for *E and* 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

- In case no artificial entity is used for this representation, there may be instances in the translated schema that cannot correspond to any instance of *R*
  - *Example: we can record that instructor K works on projects A and B with students X and Y; however, we cannot record that K works on project A with student X and works on project B with student Y, but does not work on project A with Y or on project B with X.*

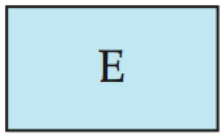
E-R Design Decisions

- The use of an attribute or entity set to represent an object.
- Whether a real-world concept is best expressed by an entity set or a relationship set.
- The use of a ternary relationship versus a pair of binary relationships.
- The use of a strong or weak entity set.
- The use of specialization/generalization – contributes to modularity in the design.
- The use of aggregation – can treat the aggregate entity set as a single unit without concern for the details of its internal structure.

Outline

- Design Process
- E-R Data Modeling
- E-R Constraints
- Removing Redundancy
- E-R Diagram
- Reduction to Relation Schemas
- Redundancy of Schemas
- Extended E-R Features
- Design Issues
- **Alternative Notions of Data Modeling**

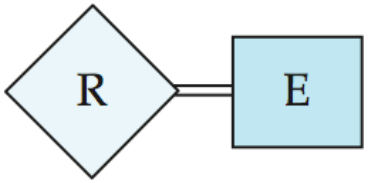
Summary of Symbols Used in E-R Notation



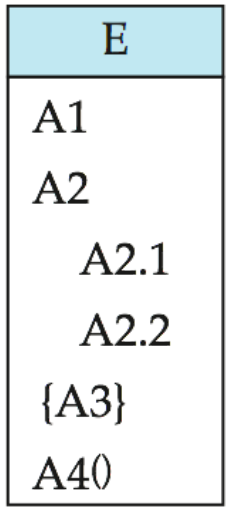
entity set

relationship set

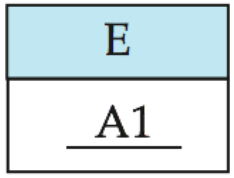
identifying relationship set for weak entity set



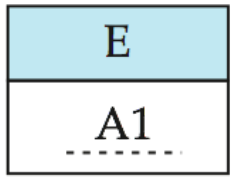
total participation of entity set in relationship



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

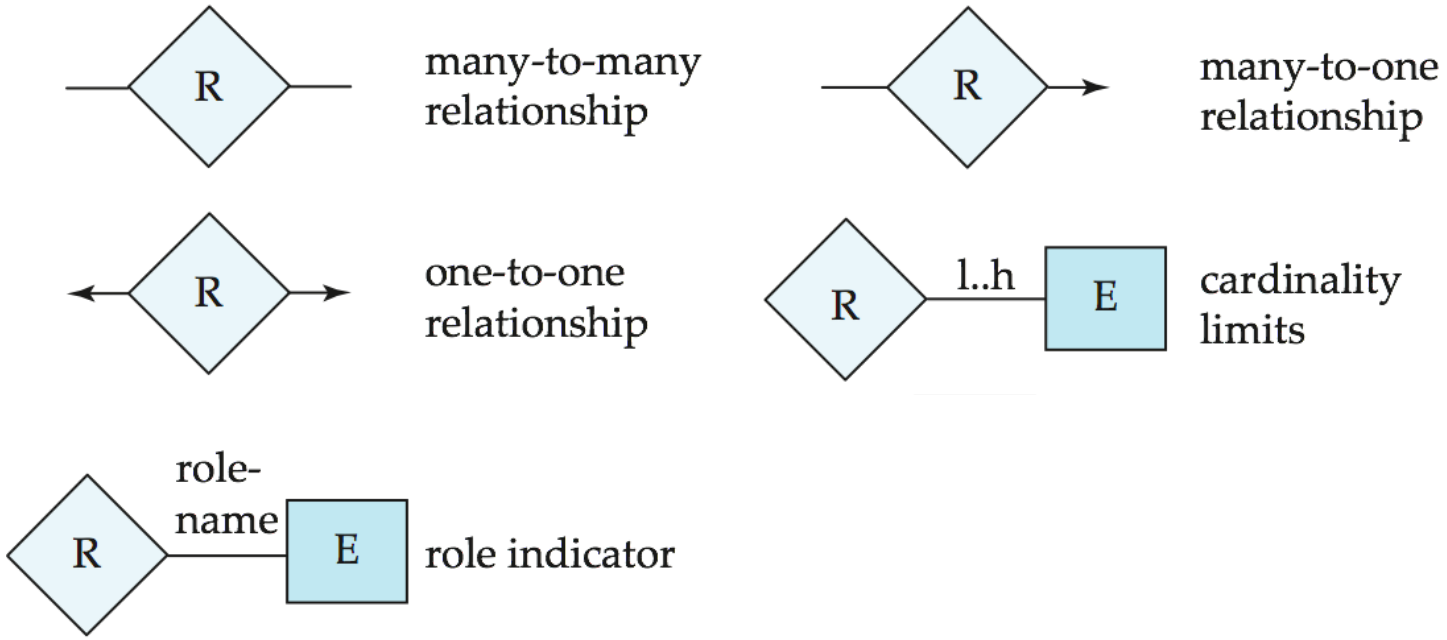


primary key

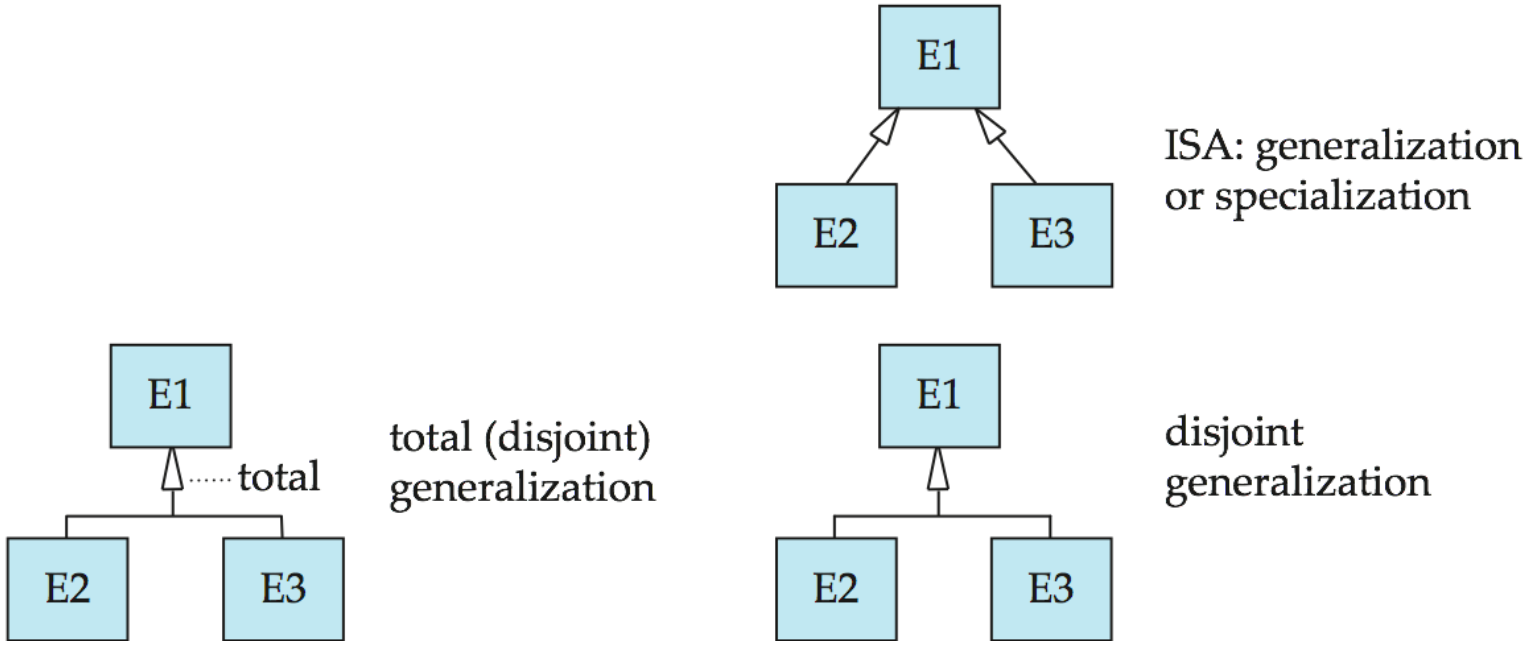


discriminating attribute of weak entity set

Summary of Symbols Used in E-R Notation

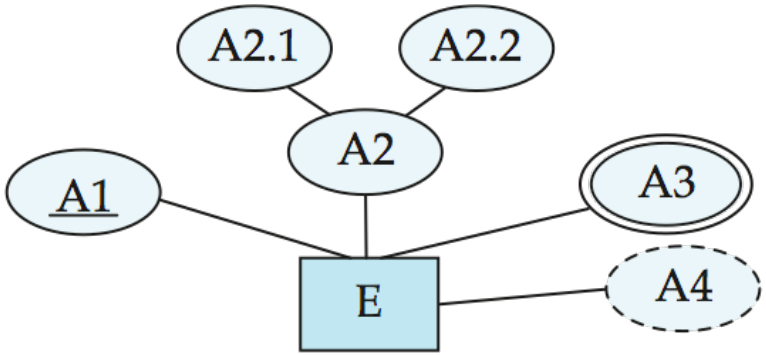


Summary of Symbols Used in E-R Notation

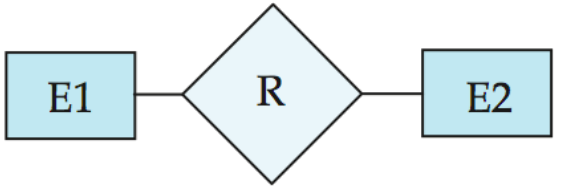


Alternative ER Notations

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

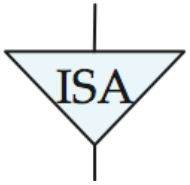


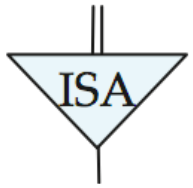
- Relationship attributes can be similarly represented, by connecting the ovals to the diamond representing the relationship.



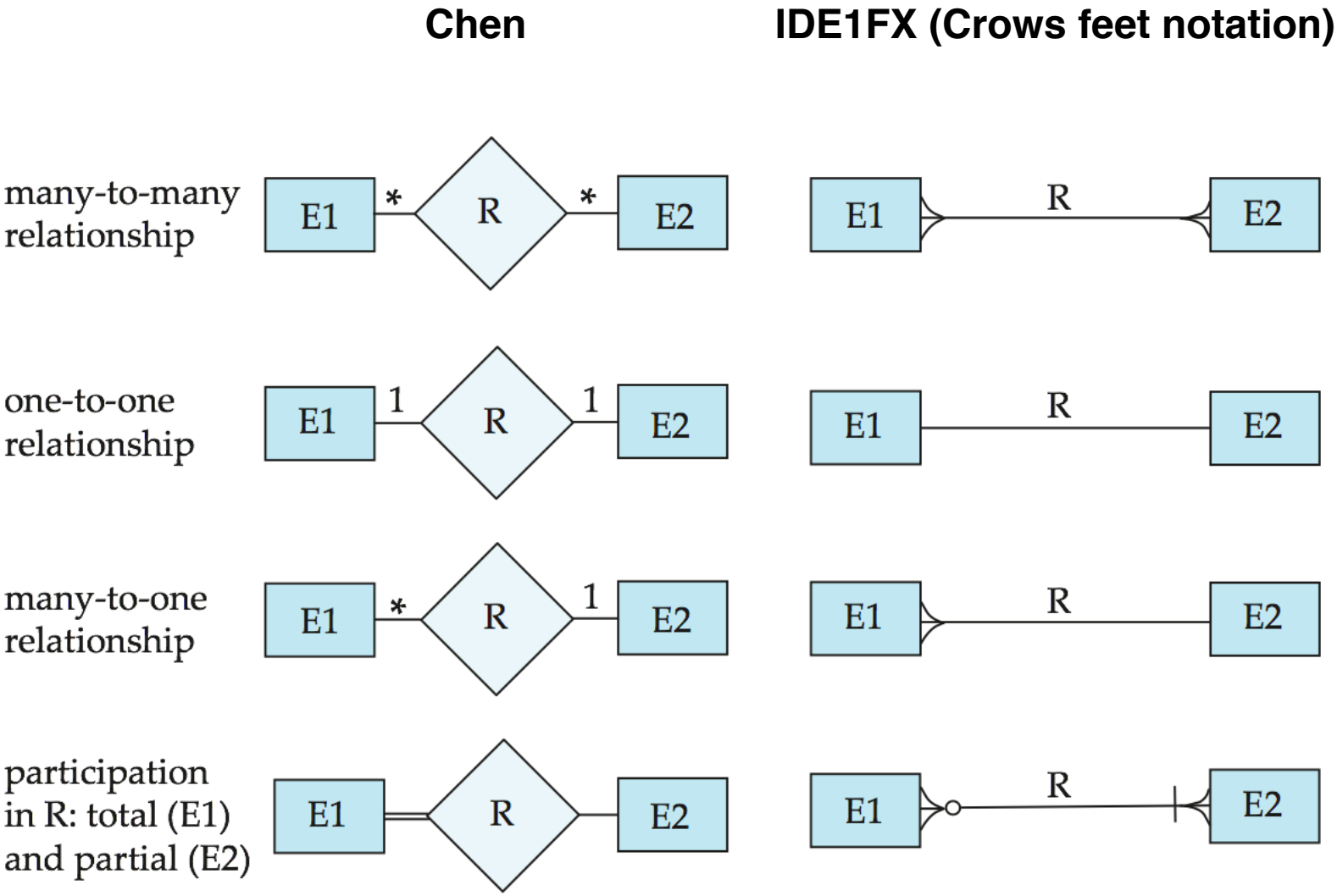
Alternative ER Notations

weak entity set 

generalization 

total generalization 

Alternative ER Notations



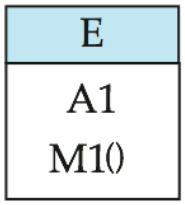
UML

- **UML**: Unified Modeling Language

- UML has many components to graphically model different aspects of an entire software system
  - Class diagram.
    - is similar to an E-R diagram, but with differences
  - Use case diagram
  - Activity diagram
  - Implementation diagram

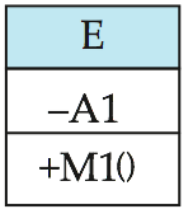
ER vs. UML Class Diagrams

ER Diagram Notation



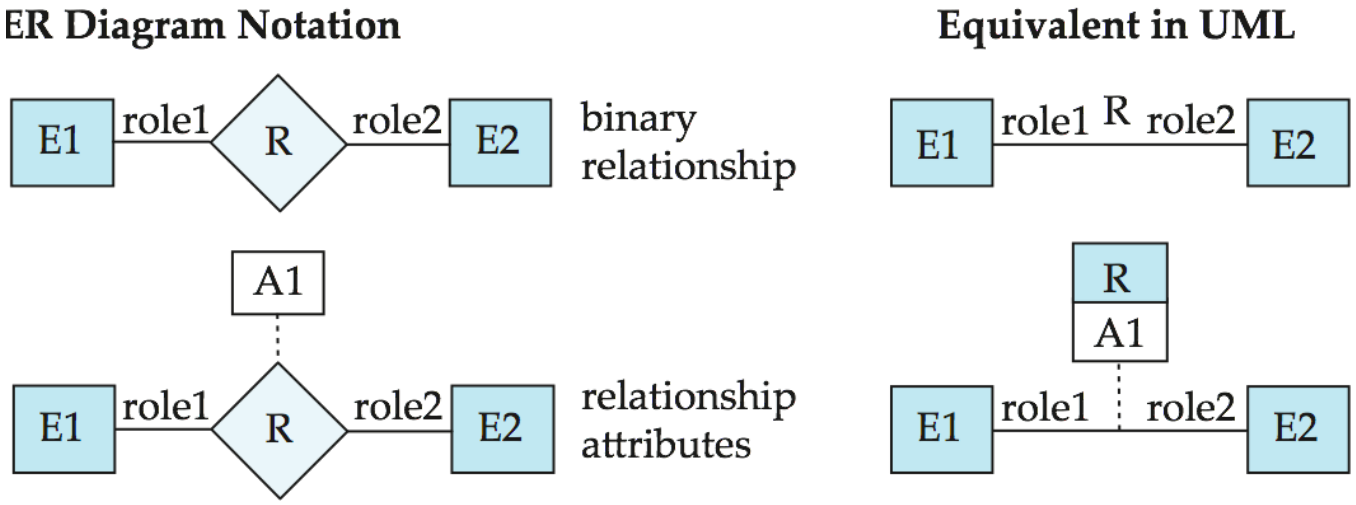
entity with attributes (simple, composite, multivalued, derived)

Equivalent in UML



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

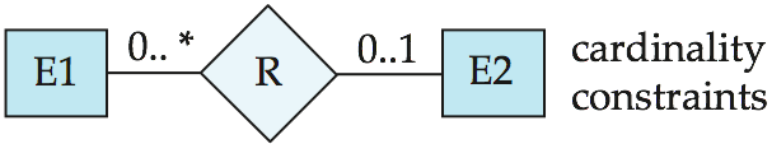
ER vs. UML Class Diagrams



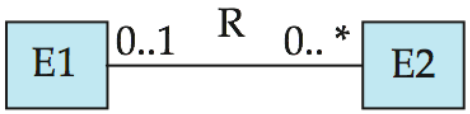
- Binary relationship sets are represented in UML by just drawing a line connecting the entity sets. The relationship set name is written adjacent to the line.
- The role played by an entity set in a relationship set may also be specified by writing the role name on the line, adjacent to the entity set.
- The relationship set name may alternatively be written in a box, along with attributes of the relationship set, and the box is connected, using a dotted line, to the line depicting the relationship set.

ER vs. UML Class Diagrams

ER Diagram Notation



Equivalent in UML

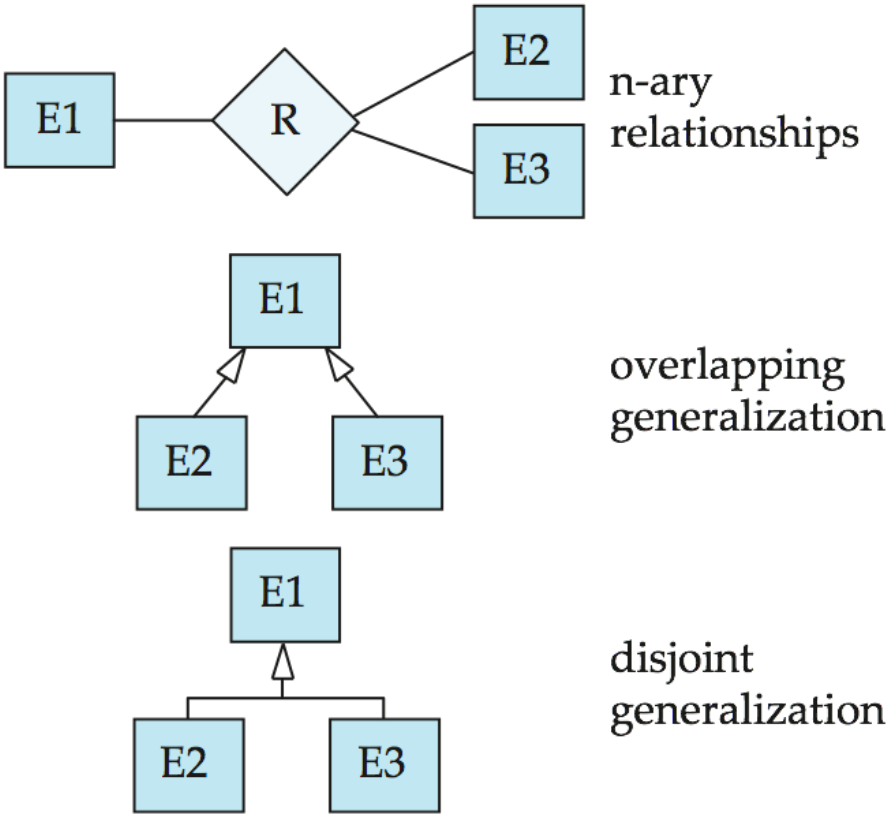


*Note reversal of position in cardinality constraint depiction

ER vs. UML Class Diagrams

ER Diagram Notation

Equivalent in UML



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

Questions?

Based on the slides of the course book