

## Module - III (Part-2)

### Database Design Using the E-R Model

*Lecture - 1*

## Introduction to Entity-Relationship (E-R) Model



# Design Phases

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



# Design Phases (Cont.)

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



# Design Alternatives

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



# Design Approaches

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



# Outline of the ER Model



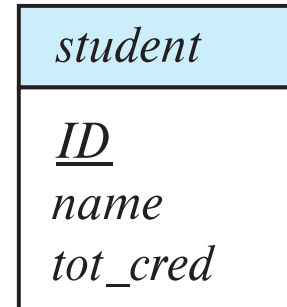
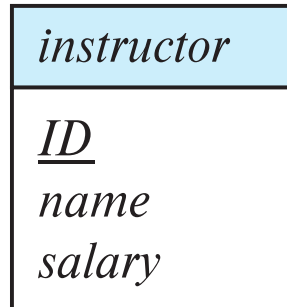
# Entity Sets

- 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, trees, holidays
- 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







# Relationship Sets

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

Example:

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

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

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

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

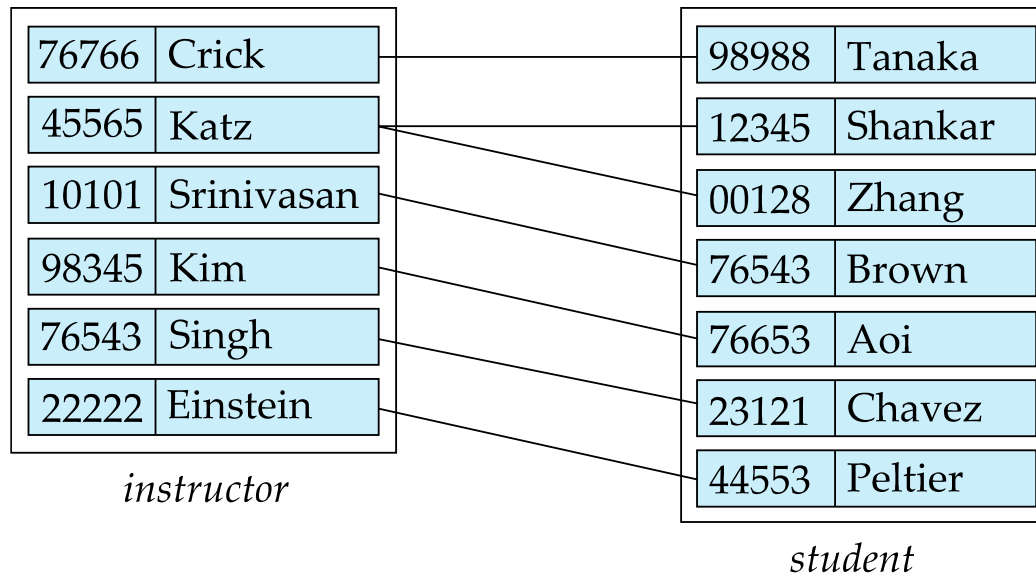
- Example:

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



# Relationship Sets (Cont.)

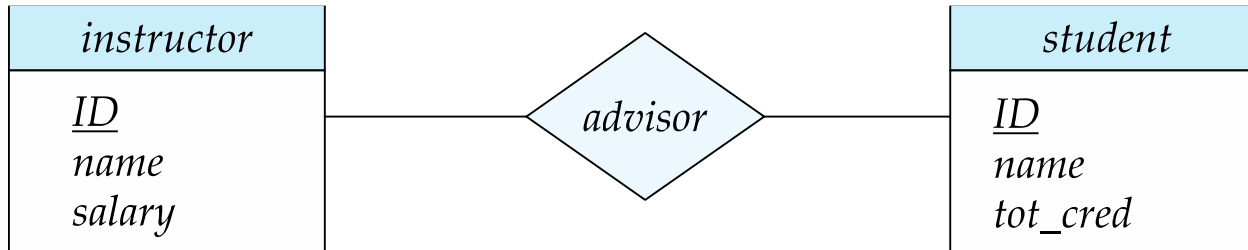
- Example: we define the relationship set *advisor* to denote the associations between students and the instructors who act as their advisors.
- Pictorially, we draw a line between related entities.





# Representing Relationship Sets via ER Diagrams

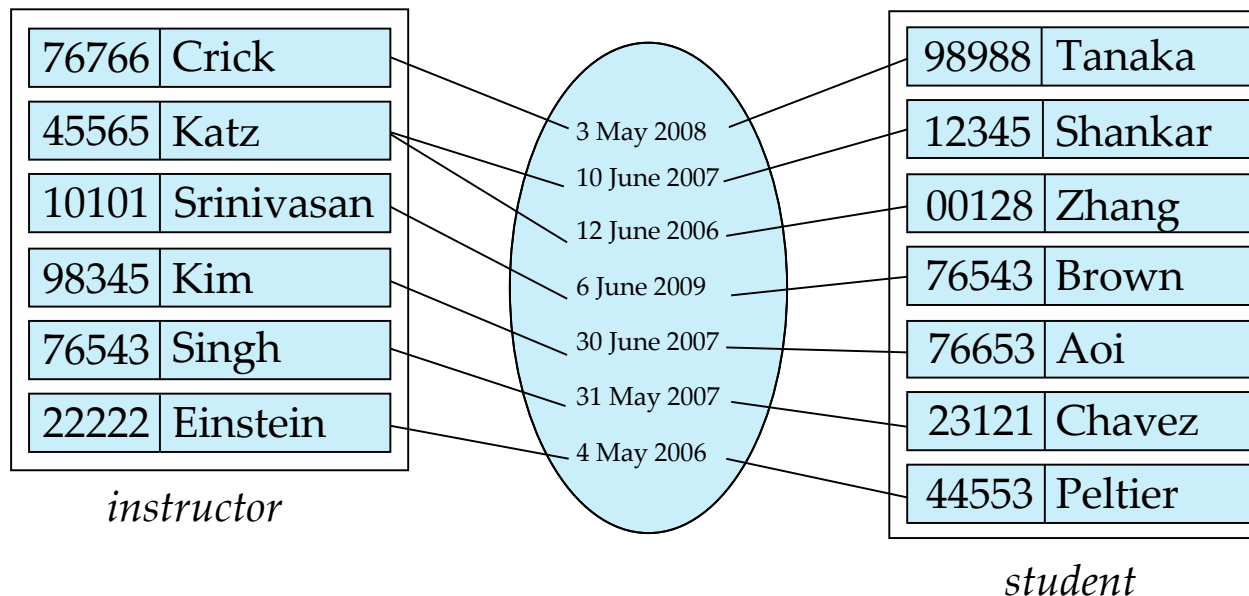
- Diamonds represent relationship sets.





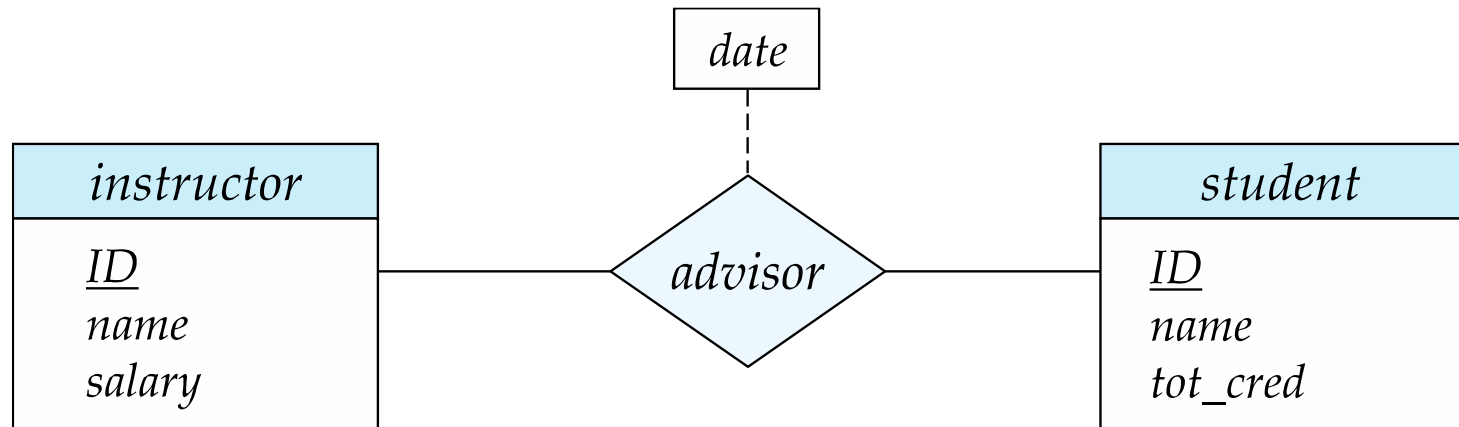
# Relationship Sets (Cont.)

- 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





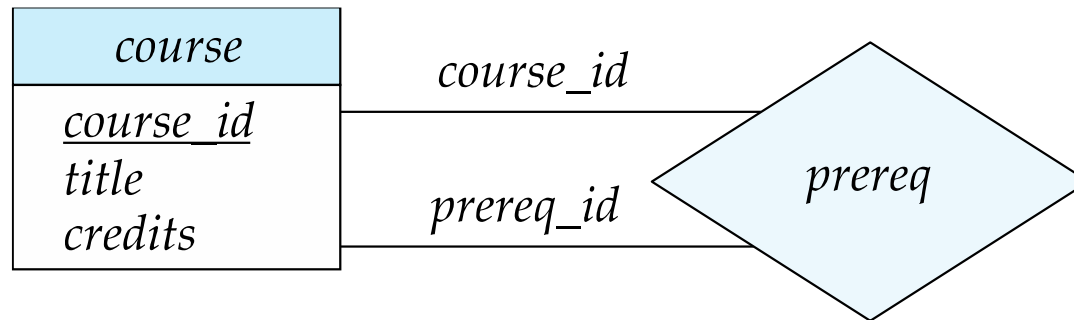
# Relationship Sets with Attributes





# 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**.





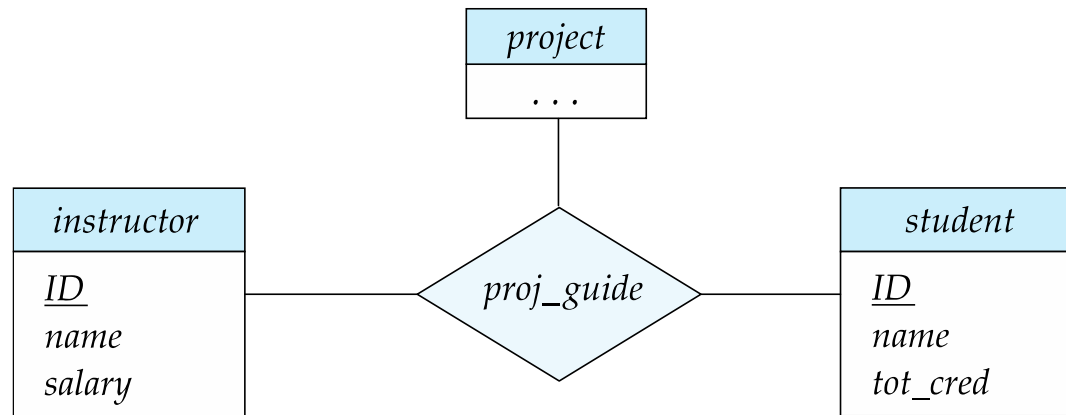
# Degree of a Relationship Set

- 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. Most relationships are binary. (More on this later.)
  - Example: *students* work on research *projects* under the guidance of an *instructor*.
  - relationship *proj\_guide* is a ternary relationship between *instructor*, *student*, and *project*



# Non-binary Relationship Sets

- Most relationship sets are binary
- There are occasions when it is more convenient to represent relationships as non-binary.
- E-R Diagram with a Ternary Relationship







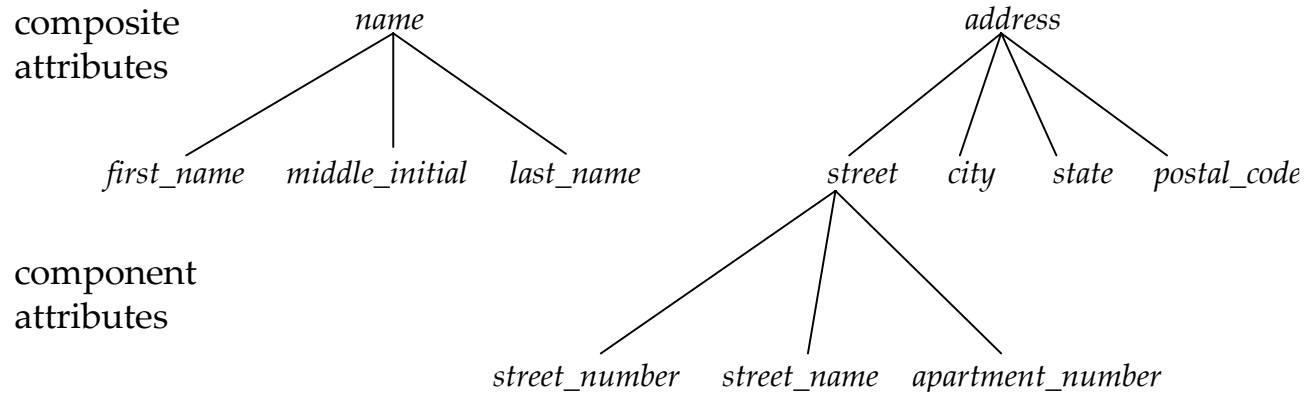
# Complex Attributes

- 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



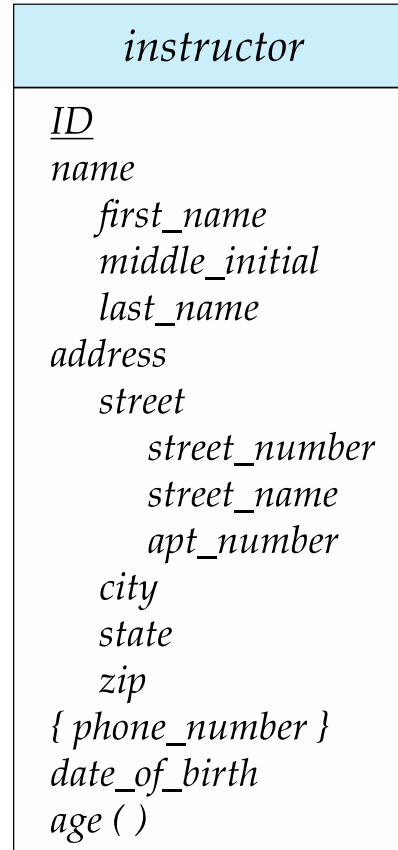
# Composite Attributes

- Composite attributes allow us to divided attributes into subparts (other attributes).





# Representing Complex Attributes in ER Diagram



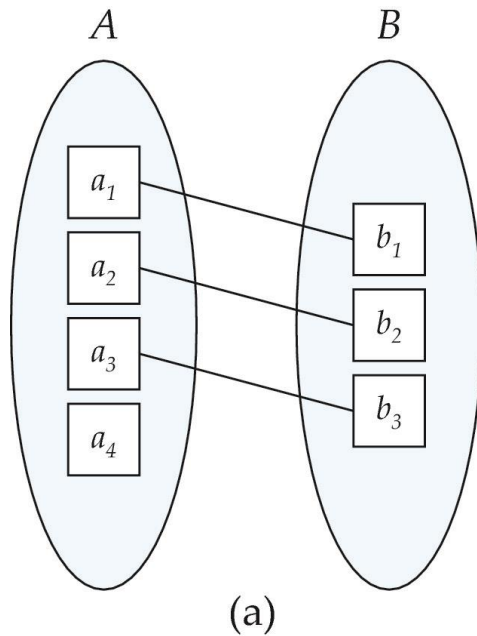


# Mapping Cardinality Constraints

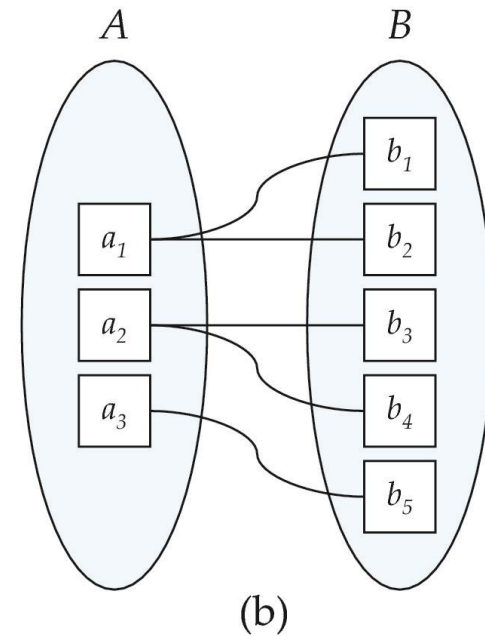
- Express the number of entities to which another entity can be associated via a relationship set.
- Most useful in describing binary relationship sets.
- For a binary relationship set the mapping cardinality must be one of the following types:
  - One to one
  - One to many
  - Many to one
  - Many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



# Mapping Cardinalities



One to one

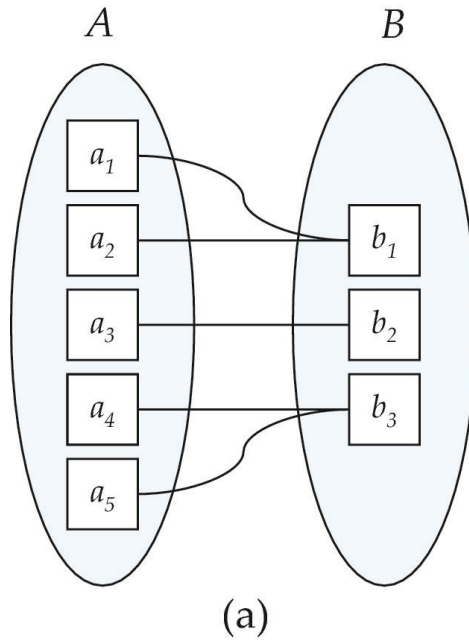


One to many

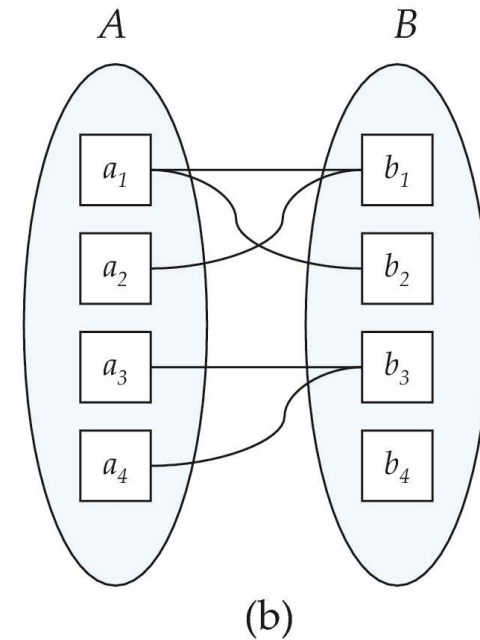
Note: Some elements in  $A$  and  $B$  may not be mapped to any elements in the other set



# Mapping Cardinalities



Many to one



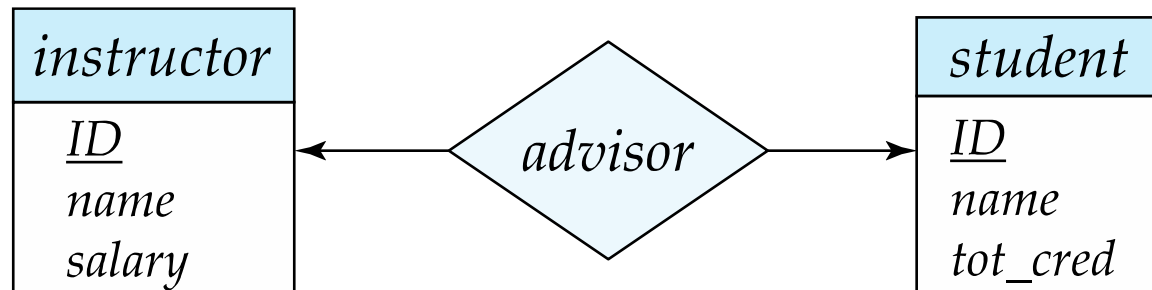
Many to many

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



# Representing Cardinality Constraints in ER Diagram

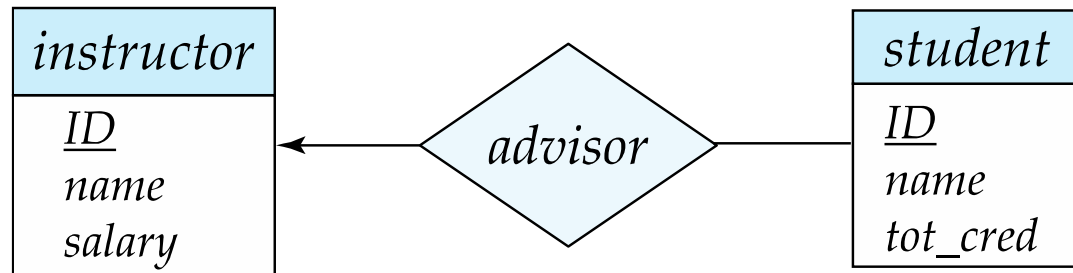
- We express cardinality constraints by drawing either a directed line ( $\rightarrow$ ), signifying “one,” or an undirected line ( $—$ ), signifying “many,” between the relationship set and the entity set.
- One-to-one relationship between an *instructor* and a *student* :
  - A student is associated with at most one *instructor* via the relationship *advisor*
  - A *student* is associated with at most one *department* via *stud\_dept*





# One-to-Many Relationship

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

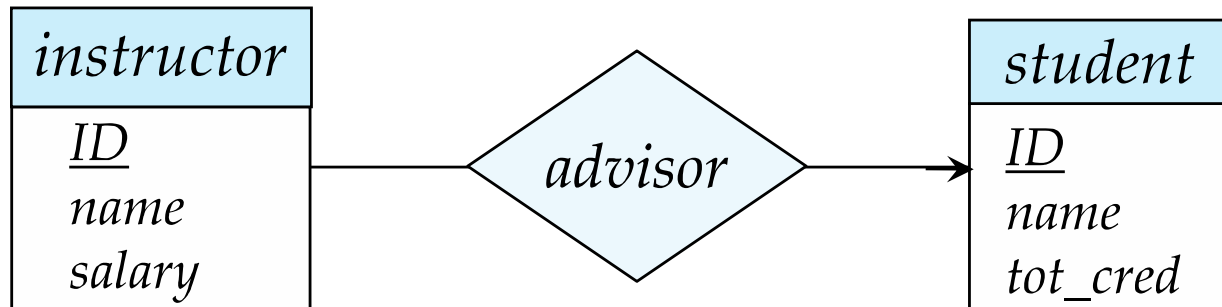






# Many-to-One Relationships

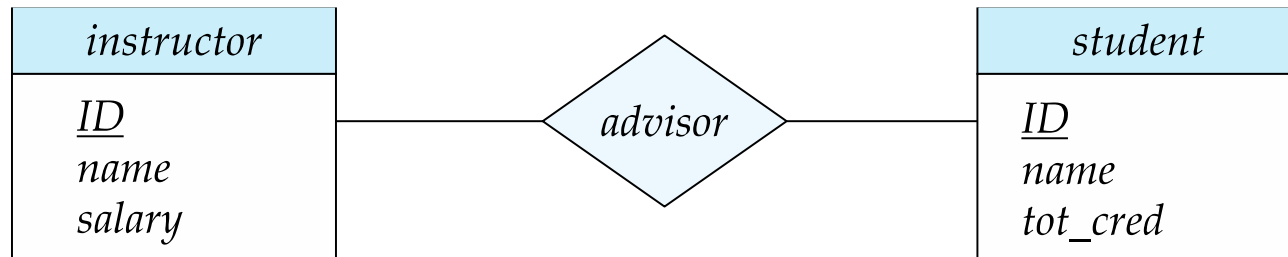
- In a many-to-one relationship between an *instructor* and a *student*,
  - an *instructor* is associated with at most one *student* via *advisor*,
  - and a *student* is associated with several (including 0) *instructors* via *advisor*





# Many-to-Many Relationship

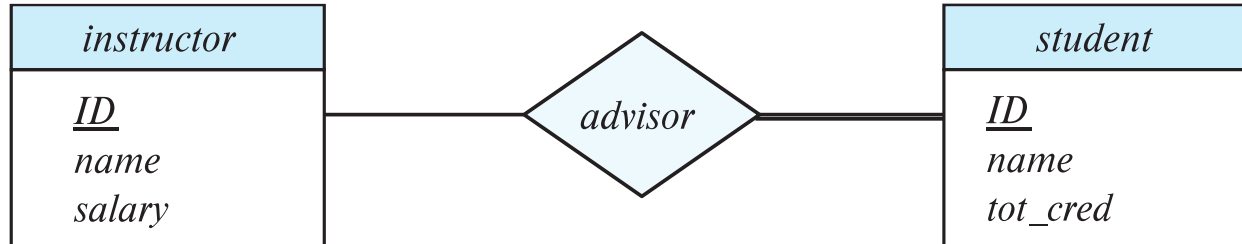
- An instructor is associated with several (possibly 0) students via *advisor*
- A student is associated with several (possibly 0) instructors via *advisor*





# Total and Partial Participation

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



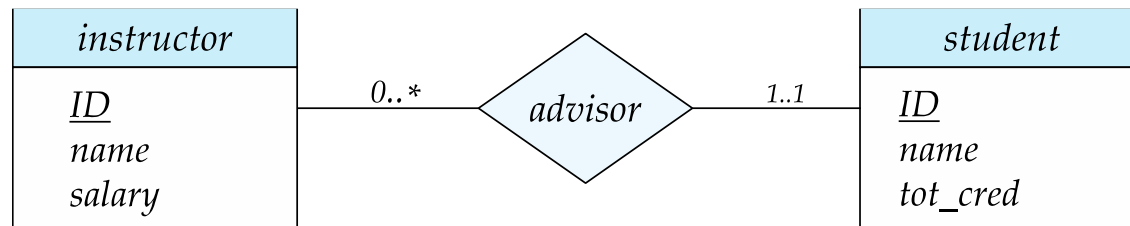
participation of *student* in *advisor* relation is total

- every *student* must have an associated instructor
- **Partial participation:** some entities may not participate in any relationship in the relationship set
  - Example: participation of *instructor* in *advisor* is partial



# Notation for Expressing More Complex Constraints

- A line may have an associated minimum and maximum cardinality, shown in the form  $l..h$ , where  $l$  is the minimum and  $h$  the maximum cardinality
  - A minimum value of 1 indicates total participation.
  - A maximum value of 1 indicates that the entity participates in at most one relationship
  - A maximum value of \* indicates no limit.
- Example



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



# 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



# 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  $E_1, E_2, \dots, E_n$
  - The primary key for  $R$  consists of the union of the primary keys of entity sets  $E_1, E_2, \dots, E_n$
  - If the relationship set  $R$  has attributes  $a_1, a_2, \dots, a_m$  associated with it, then the primary key of  $R$  also includes the attributes  $a_1, a_2, \dots, a_m$
- 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

- 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.
- 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.



## Weak Entity Sets (Cont.)

- 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.
- 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**.
- Note that the relational schema we eventually create from the entity set *section* does have the attribute *course\_id*, for reasons that will become clear later, even though we have dropped the attribute *course\_id* from the entity set *section*.



# Expressing Weak Entity Sets

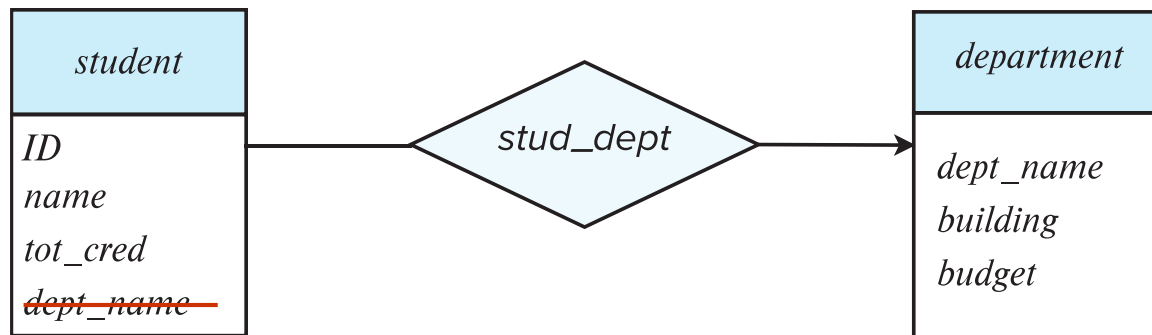
- In E-R diagrams, a weak entity set is depicted via a double rectangle.
- We underline the discriminator of a weak entity set with a dashed line.
- The relationship set connecting the weak entity set to the identifying strong entity set is depicted by a double diamond.
- Primary key for *section* –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





# Redundant Attributes

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



(a) Incorrect use of attribute



# E-R Diagram for a University Enterprise

