CSc 484
Database Management Systems

Ken Gamradt

Spring 2024

Database Design using ER (II)

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

- **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
      - 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 repeated 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 many good designs from which we must choose

Design Approaches

- **Entity Relationship Model** (**Chapter 6**)
  - Models an enterprise as a collection of **entities** and **relationships**
    - **Entity**: a "**thing**" or "**object**" in the enterprise that is **distinguishable** from other objects
      - Described by a set of attributes
    - **Relationship**: an association among several entities
  - Represented diagrammatically by an entity-relationship diagram:
- **Normalization Theory** (**Chapter 7**)
  - Formalize what designs are bad, and test for them

Entity-Relationship Model

- 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 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

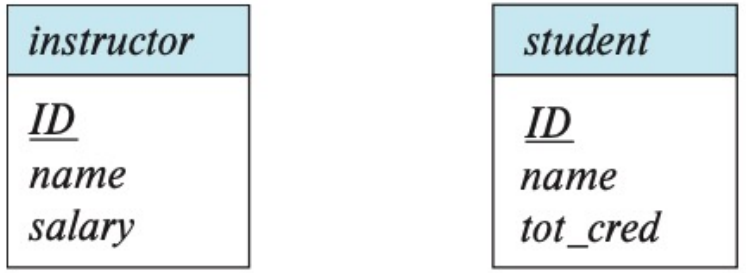
Entity Sets

- An **entity** is an object that exists and is distinguishable from other objects
  - E.g., specific person, company, event, plant, …
- An **entity set** is a set of entities of the same type that share the same properties
  - E.g., 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
  - E.g.,

    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

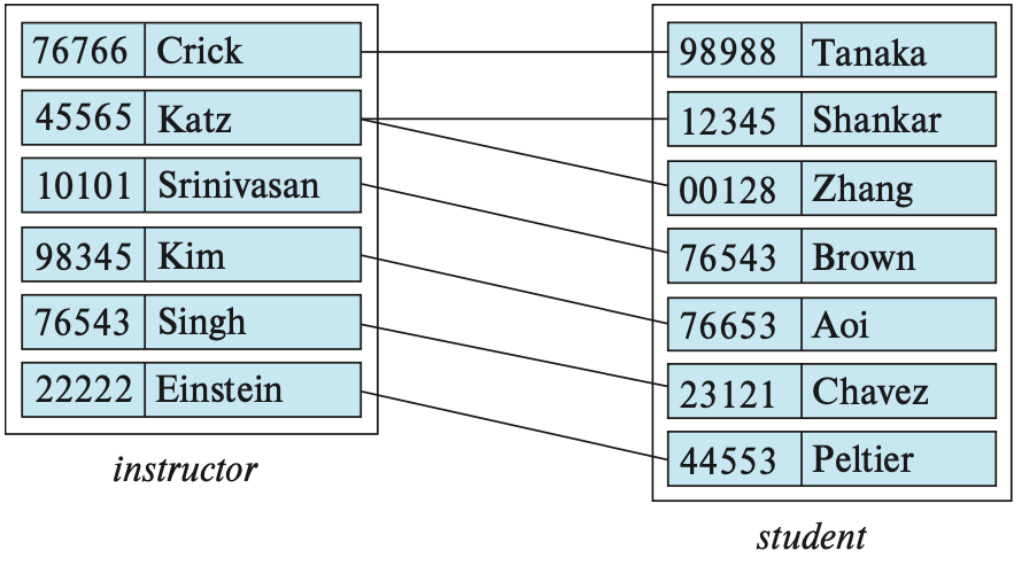
Representing Entity sets in E-R Diagram

- **Entity set**
  - Set of entities of the same type that share the same attributes
- An entity set is represented in an E-R diagram by a rectangle
  - Part 1: name of the entity set
    - Instructor
    - Student
  - Part 2: name of all **attributes**
    - ID, name, salary
    - ID, name, tot_cred
    - Primary keys are underlined
      - ID
      - ID



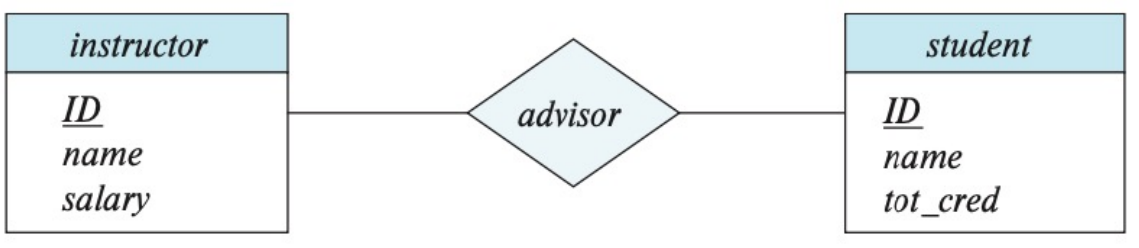
Relationship Sets

- E.g., 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



Relationship Sets

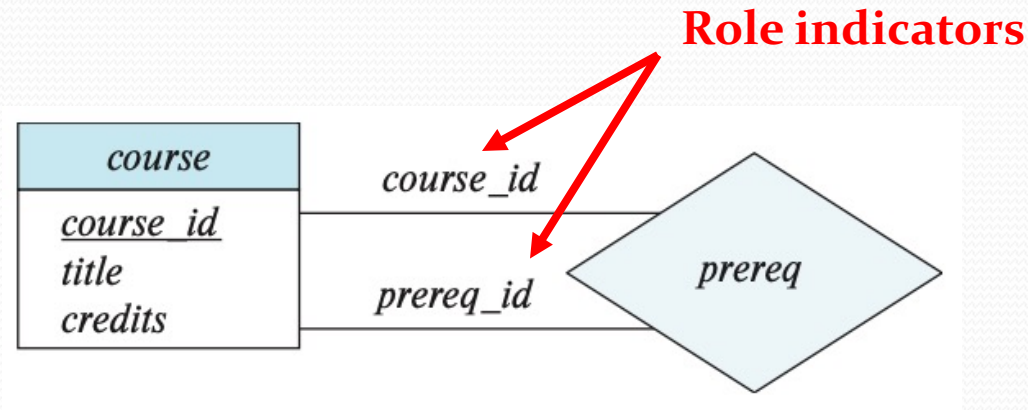
- **Relationship set**
  - Is represented in an E-R diagram by a diamond
  - Link the entity sets via lines



- Each entity plays a role in the relationship

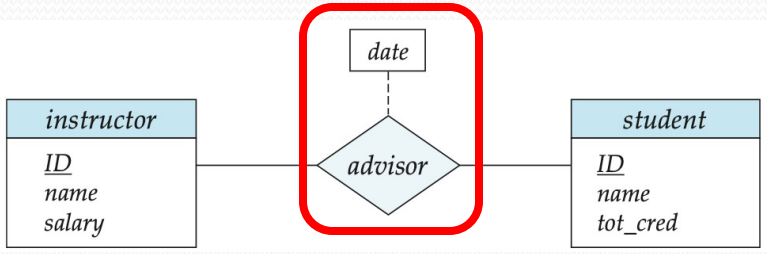
Recursive Relationship Set

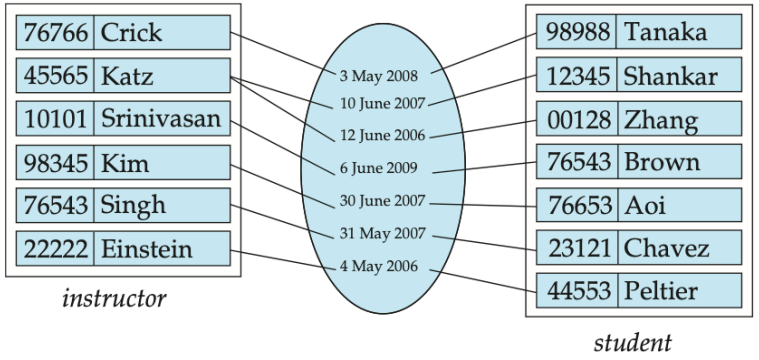
- The same entity set participates in a relationship set more than once
  - In different roles
- E.g., depict the prerequisite course for courses



Relationship Sets with Attributes

- A relationship may have attributes – **descriptive attributes**
- Attributes of a relationship set are represented using an **undivided** rectangle
- Linked by dashed line from relationship set to descriptive attribute



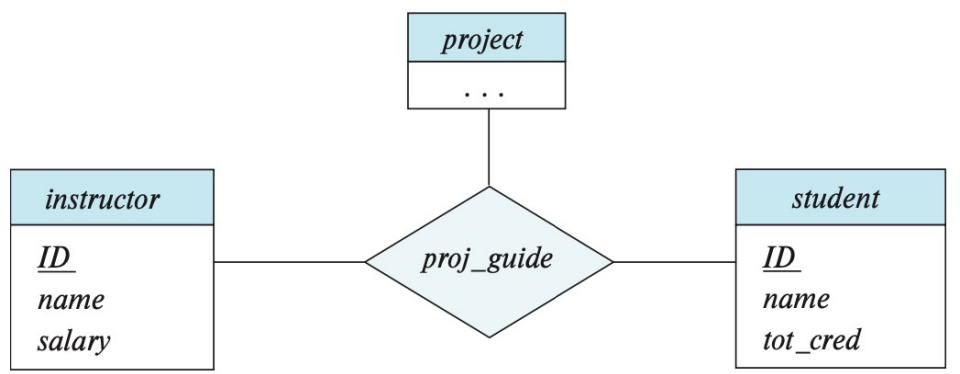


Degree of a Relationship Set

- **Binary** relationship
  - involve two entity sets
    - Degree of two
  - most relationship sets in a database system are binary
- Relationships between more than two entity sets are rare
  - E.g., 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

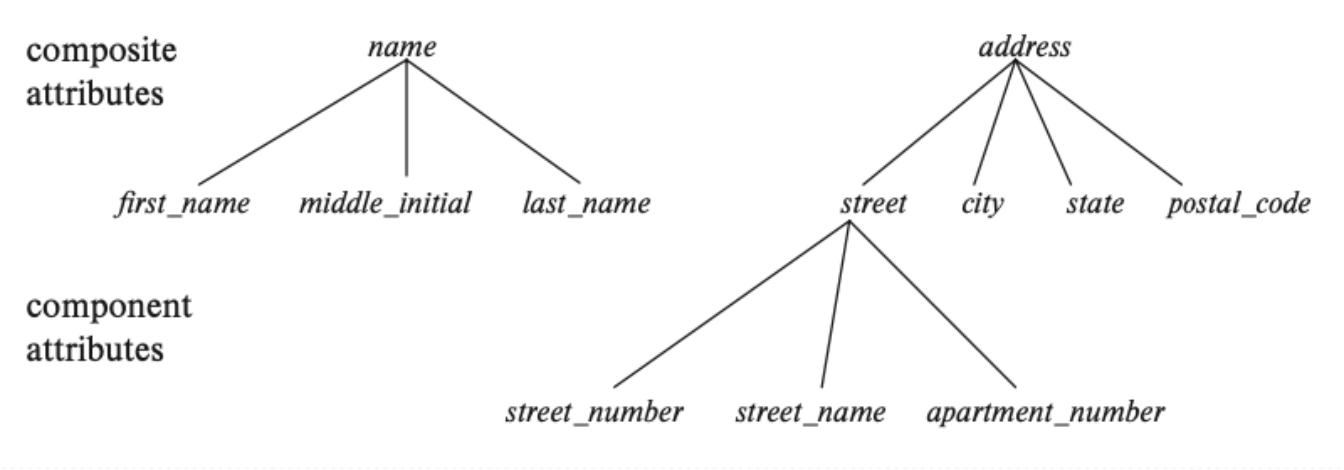


Complex Attributes

- Attribute types:
  - **Simple** and **composite** attributes
  - **Single-valued** and **multivalued** attributes
    - E.g., multivalued attribute: *phone_numbers*
  - **Derived** attributes
    - Can be computed from other attributes
    - E.g., *age* – given *date_of_birth*
- **Domain** – the set of permitted values for each attribute

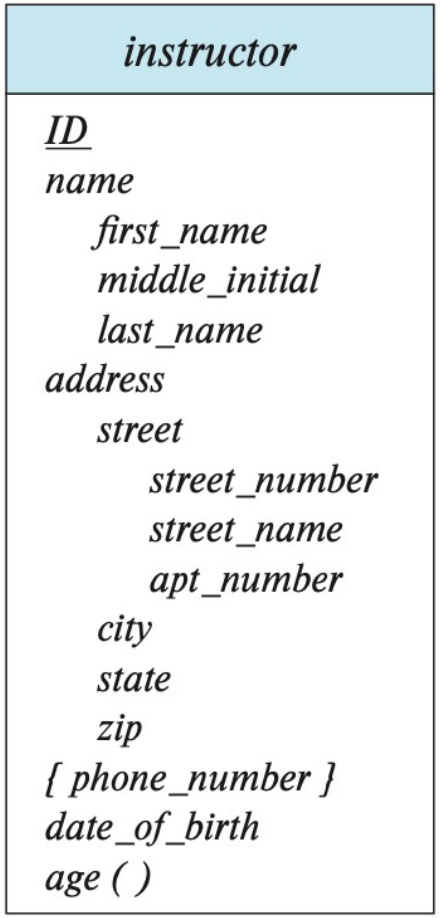
Composite Attributes

- **Simple** attribute:
  - Cannot be decomposed
    - Broken into smaller parts
- **Composite** attribute:
  - Can be decomposed into other meaningful attributes



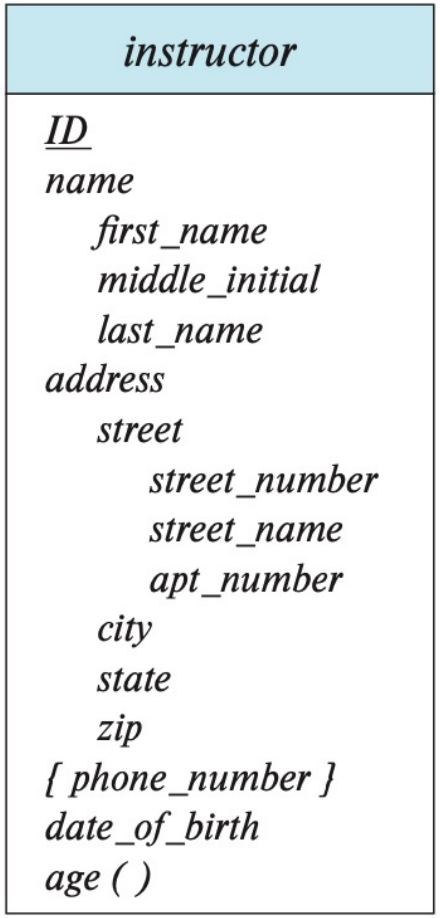
Representing Complex Attributes

- If the name and address attributes are added to the instructor entity-set
- **When to use composite attributes?**
- If the users wish to refer to
  - an entire attribute on some occasions
  - a component of other occasions
- Another benefit of using composite attributes
  - Help to group together related attributes, making the modeling cleaner



Representing Complex Attributes

- **Single-valued** attribute:
  - Has exactly one value for a particular entity
- **Multi-valued** attribute:
  - May have multiple values for a particular entity
  - E.g., an instructor may have zero or more phone numbers, and different instructors may have different numbers of phone numbers
- **Derived** attribute:
  - The value of this type of attribute can be derived from the values of other related attributes or entities



Structural Constraints

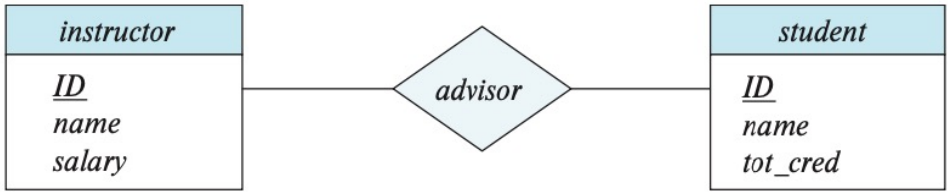
- The constraints that may be placed on entity types that participate in a relationship
  - Reflect the restrictions on the relationships as perceived in the “real world”
- Two types of constraint on relationship sets
  - **Cardinality**
  - **Participation**

Mapping Cardinalities

- **Mapping cardinalities** (cardinality ratios)
  - Express the number of entities to which another entity can be associated via a relationship set
    - Specify the minimum and/or maximum number of each entities when participating a relationship
      - Minimum cardinality: 0, 1
      - Maximum cardinality: 1, N

Entity-Relationship Model – Mapping Cardinalities

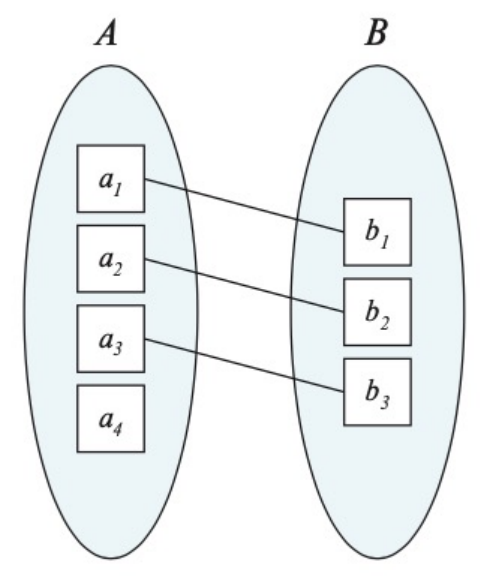
- Describe a binary relationship set
- E.g., **advisor** relationship set between **instructor** and **student** entity sets
- The mapping cardinality can be
  - **One-to-one**
  - **One-to-many**
  - **Many-to-one**
  - **Many-to-many**

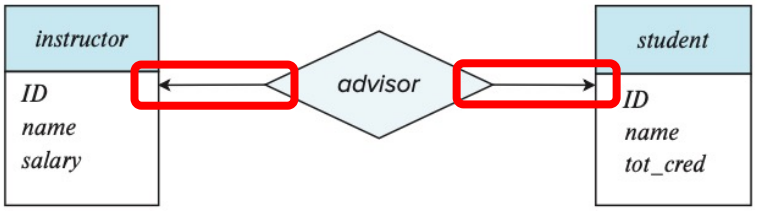


- A directed line and an undirected line are used to indicate the cardinality constraint in the E-R diagram

Mapping Cardinalities

- **One-to-one**
  - Each instructor may advise **at most one** student
  - Each student may have **at most one** advisor

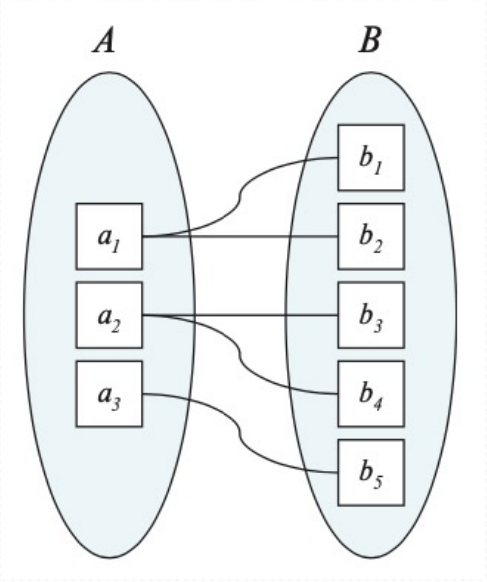


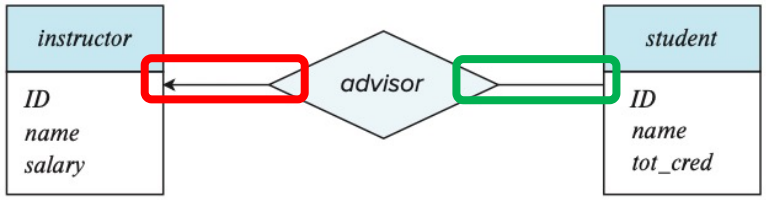


ER diagram notation:
Directed line from the relationship set to the "**one**" side of the relationship

Mapping Cardinalities

- **One-to-many**
  - Each instructor entity is associated with any number **(zero or more)** of student entities
  - Each student entity is associated with **at most one** instructor entity



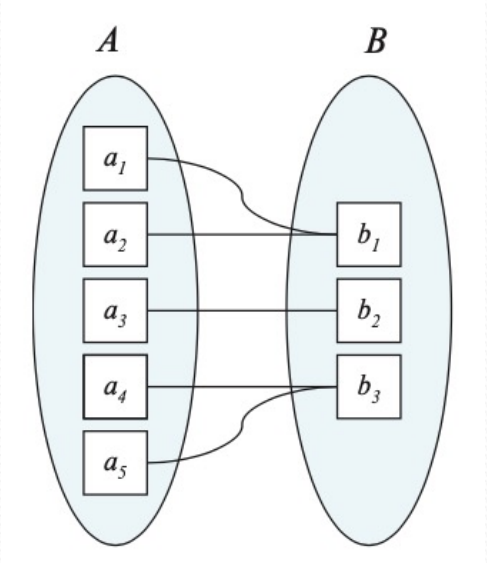


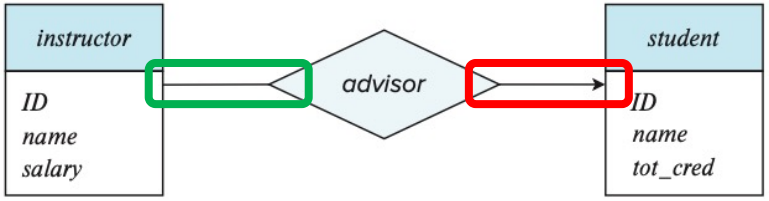
ER diagram notation:

Undirected line from the relationship set to the "**many**" side of the relationship

Mapping Cardinalities

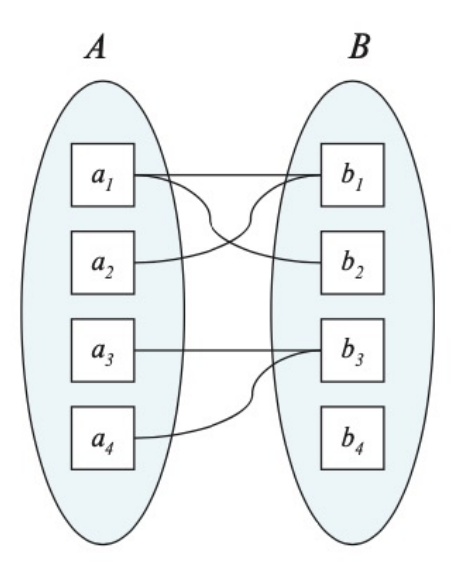
- **Many-to-one**
  - Each instructor may advise **at most one** student
  - Each student may have any number **(zero or more)** of advisors

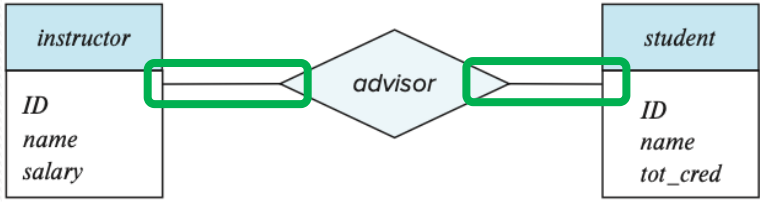




Mapping Cardinalities

- **Many-to-many**
  - Each instructor may advise any number **(zero or more)** of students
  - Each student may have any number **(zero or more)** of advisors



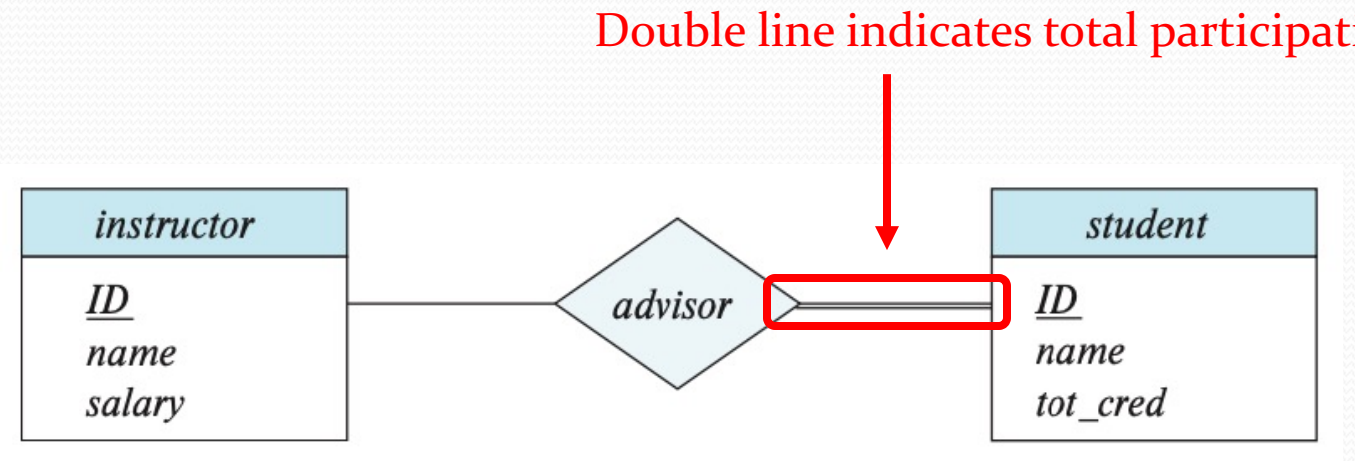


Participation

- Determine whether all or only some entities of an entity set participate in a relationship
  - **Total** participation
    - Every entity in entity set must participate in at least one relationship
  - **Partial** participation
    - Some entities do not have to participate in relationship
- **Double lines** are used in E-R diagrams to indicate **total** participation

Participation

- E.g., the university requires that each student must have an advisor



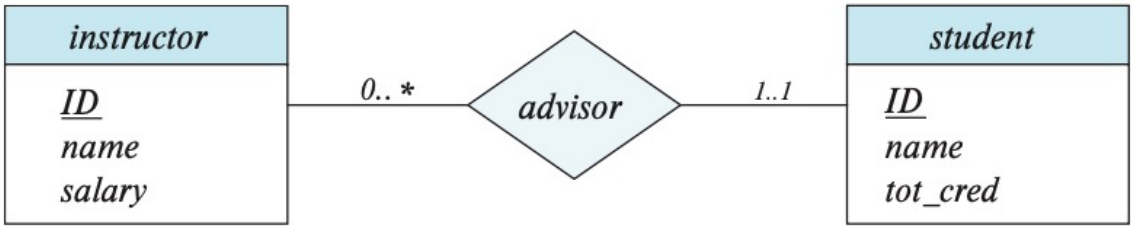
Cardinality & Participation

- Another way to indicate more complex constraints of cardinality and participation
  - Single line together with numbers (x .. y) to indicate the constraints from the requirement
    - x: meaning minimum cardinality
    - y: meaning maximum cardinality
    - x, y can be numbers (0, 1, 2, …) and * (means unlimited)
  - x = 1: total participation
  - x = 0: partial participation
  - y = 1: the entity participates in at most one relationship
  - y = *: unlimited number of entities

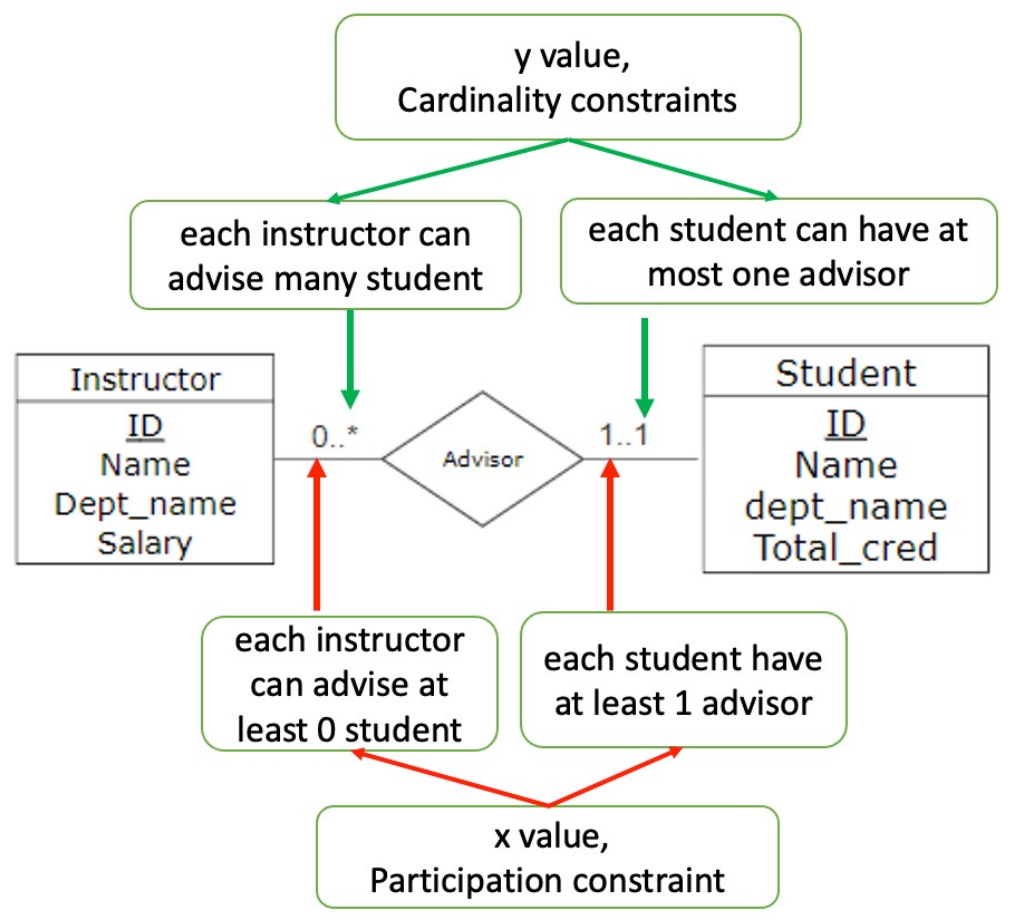
Cardinality & Participation

- The university requires
  - Each student must have at exactly one advisor
  - Each instructor can advise any number of students

- Cardinality: one-to-many from instructor to student
  - For instructors that participate
- Participation: instructor (**partial**), student (**total**)



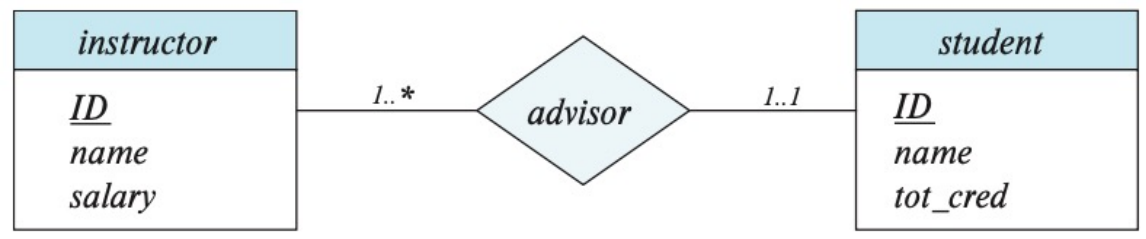
Cardinality & Participation



Cardinality & Participation

- E.g., university requires
  - Each instructor must advise at least one student
  - Each student must have exactly one advisor

- Both entity sets are total participation
  - Cardinality is one-to-many



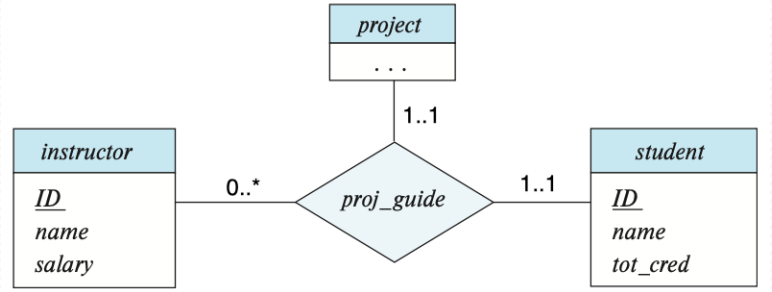
Cardinality Constraints on Ternary Relationship

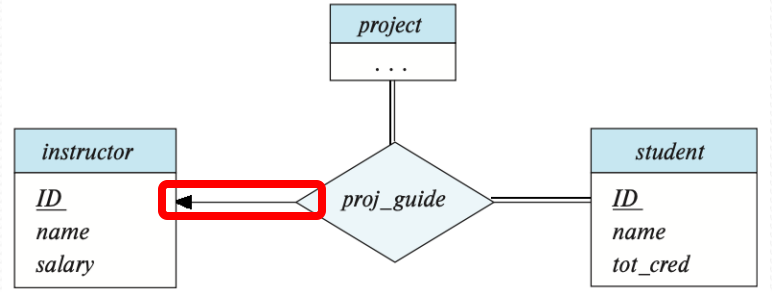
- We allow at most one arrow out of a ternary (or greater degree) relationship to indicate a cardinality constraint
- E.g., an arrow from proj_guide to instructor indicates each student has at most one guide for a project
- If there is more than one arrow, there are two ways of defining the meaning
  - E.g., 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

Cardinality Constraints on Ternary Relationship

- Constraints on non-binary relationship sets
- E.g., senior design projects
  - Instructors provide guidance for any number of students on their projects
  - Each student can be guided by exactly one instructor

Only one arrow out of a non-binary relationship set is allowed





Primary key

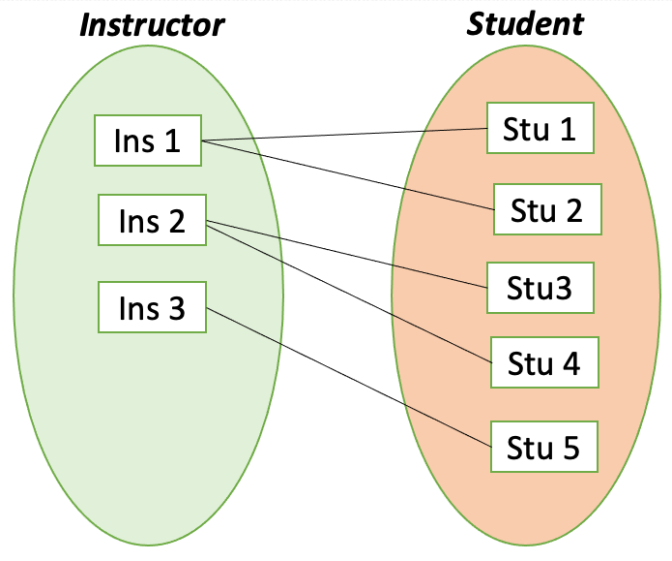
- Primary keys provide a way to specify how entities and relations are distinguished
  - **Entity sets**
  - **Relationship sets**
  - **Weak entity sets**

Primary key for Entity Sets

- Conceptually, 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 can be used to distinguish entities from each other

Primary key for Relationship sets

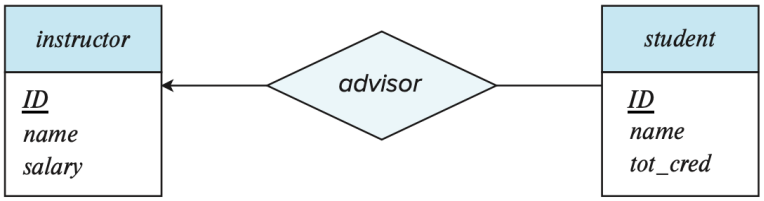
- Keys also help to identify relationships uniquely



Each relationship is unique

Entity set names used as a prefix to distinguish attributes with the same name, can be omitted if attributes with different names

(Instructor.ID, Student.ID) form a superkey for the relationship key

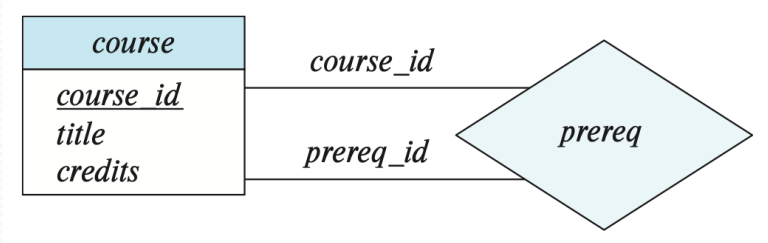


primary-key(E_1) U primary-key(E_2) U … U primary-key(E_n) can be used as a superkey to distinguish relationships

Primary key for Relationship sets

- More about the name of attributes in the superkey of a relationship set
  - In recursive relationship set, one entity set plays different roles in the relationship

- Use role names/indicators as prefixes to form a unique attribute



(course_id.ID, prereq_id.ID) forms the superkey for *prereq*

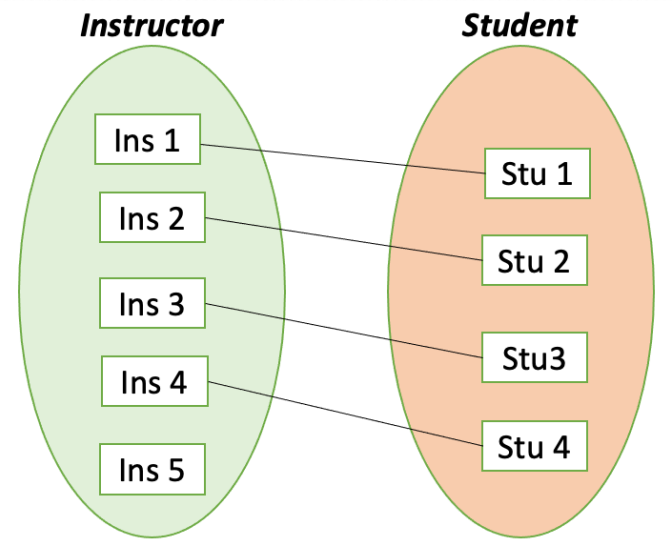
Primary key for Relationship sets

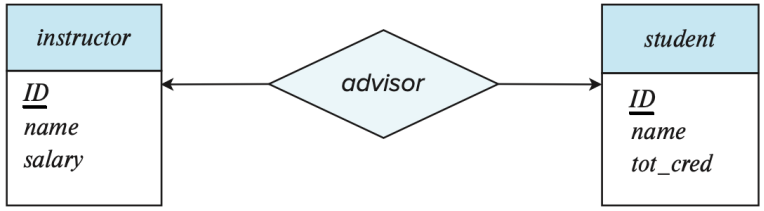
- The choice of the primary key depends on the mapping cardinality of the relationship

- Considering a binary relationship set

Primary key for Relationship sets

- **One-to-one**
  - The primary key of the relationship set can be the primary key of either one of the participating entity sets

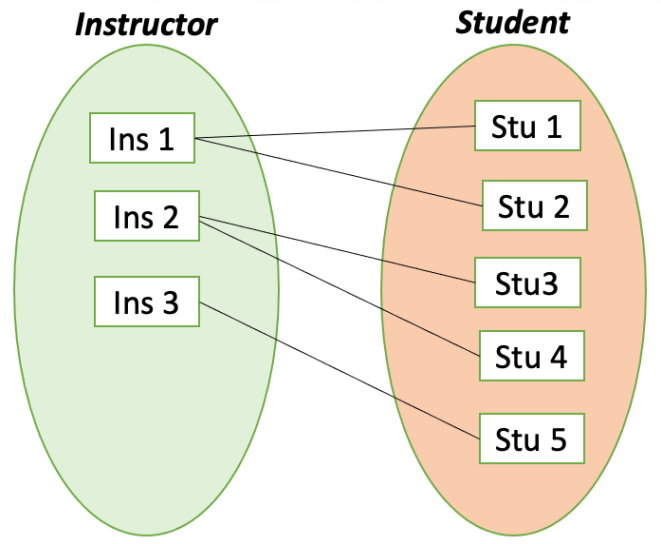


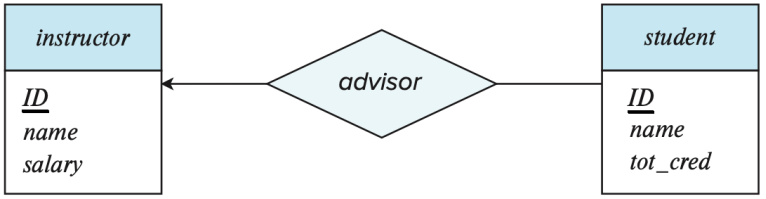


The primary key for advisor is:
Instructor.ID or ***Student.ID***

Primary key for Relationship sets

- **One-to-many**
  - The primary key for the "many" side is used as the primary key for the relationship set

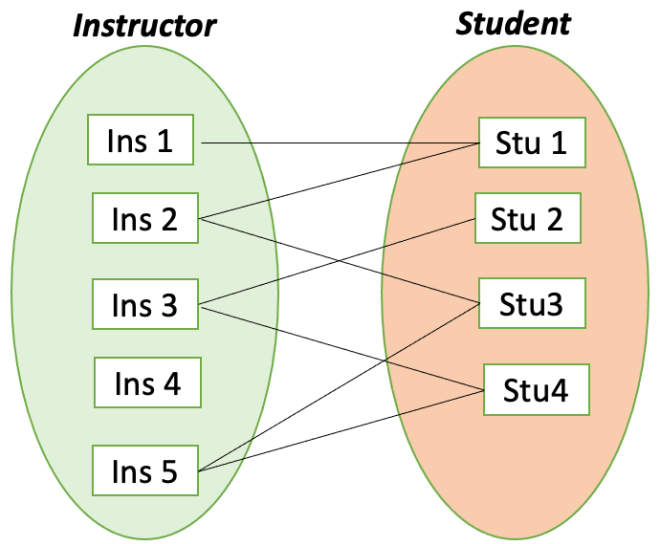


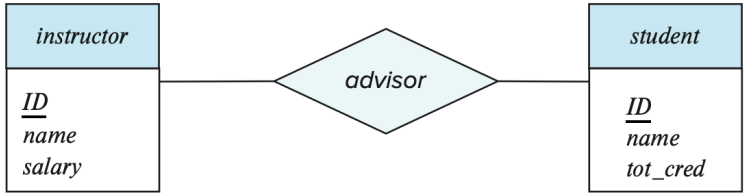


The primary key for advisor is:
Student.ID

Primary key for Relationship sets

- **Many-to-many**
  - The primary key is the union of both entity sets





The primary key for advisor is:
(Instructor.ID, Student.ID)

Acknowledgements