# Principles of Database Systems (CS307)

## Lecture 8: Database Design using E-R Model


**Zhong-Qiu Wang**

Department of Computer Science and Engineering
Southern University of Science and Technology


- Most contents are from slides made by Stéphane Faroult and the authors of Database System Concepts (7th Edition).
- Their original slides have been modified to adapt to the schedule of CS307 at SUSTech.
- The slides are largely based on the slides provided by Dr. Yuxin Ma

# Outline

## Part 1: Relational Language

**Basic SQL**
- Create Tables and Insert Data
- Basic Queries

**Intermediate SQL**
- Advanced Join
- Subquery
- Window Function
- *Transactions*

**Advanced SQL**
- Function
- Procedure
- Trigger
- View

## Part 2: DB Design and Application

**Relational DB Design**
- The Entity-Relationship Model
- E-R Diagram Design

**Relational Algebra**

**Normalization**
- Functional Dependency
- Normal Forms
- FD Theory
- Decomposition w/ FD

**Application Design**
- Use Databases in Programs
- Complex Data Types (JSON, XML, etc.)

## Part 3: Internal Mechanism Of Relational DB Systems

**Storage Management**
- Physical and Logical Storage Structure
- Indexing

**Query Management**
- Query Processing
- Query Optimization

**Transaction Management**
- Transaction
- Concurrency

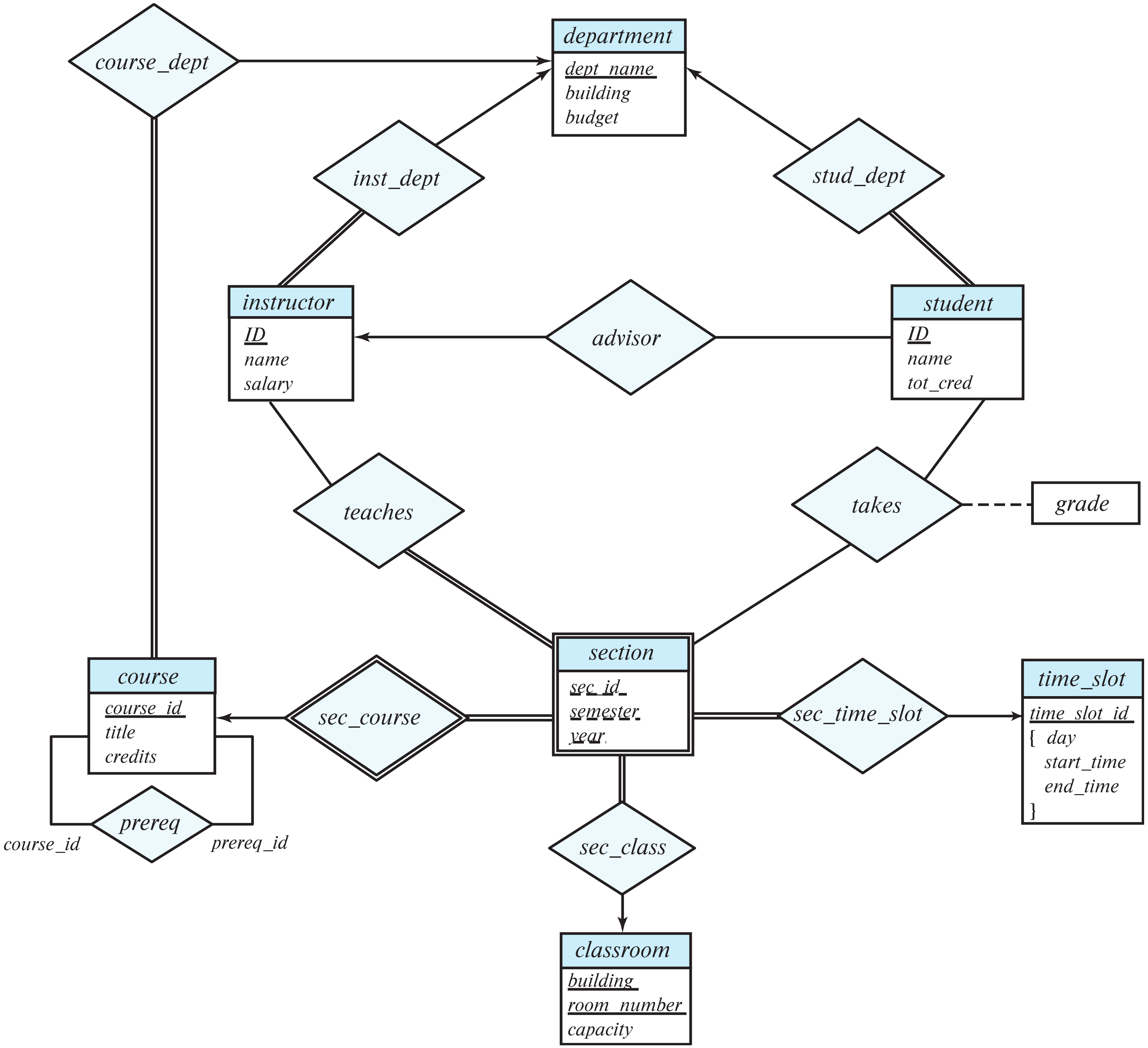
## Part 4: Miscellaneous Topics

**Big Data**
- Big Data Storage
- Query and Process Big Data

**Advanced Topics**
- Beyond Relational DB
- Data Analytics

# Entity-Relationship Model (E-R Model)
# Entity-Relationship Diagram (E-R Diagram)

# The New Running Example



# Design Phases

- Initial phase: characterize fully the data needs of the prospective database users
  - Interact extensively with domain experts
  - The outcome is a specification of user requirements

# Design Phases

- Second phase: choosing a data model
  - E.g., relational model, entity-relationship model, semi-structured data model, and object-oriented data model
  - Applying the concepts of the chosen data model
  - Translating these requirements into a conceptual schema of the database
    - Detailed overview of the enterprise
  - A fully developed conceptual schema indicates the functional requirements of the enterprise
    - Describes operations (e.g., update, retrieval, delete) that will be performed on the data
- Entity-relationship model is typically used
  - Outcome is an E-R diagram that provides a graphic representation of the schema
    - Entities that are represented in the database
    - Attributes of the entities
    - Relationships among entities
    - Constraints on the entities and relationship

# 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, and how should the attributes be distributed among the various relation schemas?
  - Physical Design – Deciding on the physical layout of the database
    - E.g., the form of file organization and choice of index structures
  - Physical schema is easy to change after application is built
    - But logical schema is not, because changes may affect a number of queries and updates scattered across application code

# 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
    - E.g., store course identifier and title of a course for each course offering
      - Only store course identifier is sufficient
    - Redundant representation of information may lead to data inconsistency among the various copies of information
      - E.g., update is not performed on all the copies
  - **Incompleteness**: a bad design may make certain aspects of the enterprise difficult or impossible to model
    - E.g., only have entity for course offering, but without entity for courses
      - Impossible to model new courses that are not offered yet

# Design Alternatives

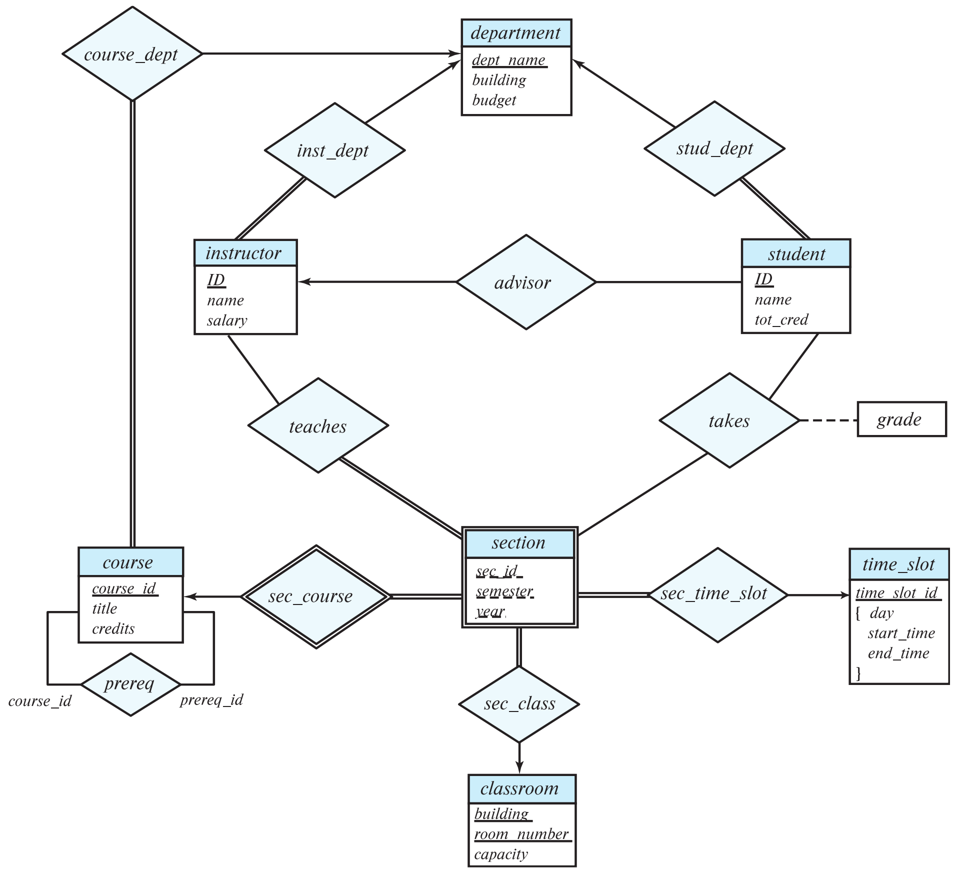
- Avoiding bad designs is not enough
  - There may be many good designs from which we must choose

- For example, a customer who buys a product
  - The sale activity is a relationship between the customer and the product?
  - The sale activity is a relationship among the customer, the product, and the sale itself?
    - i.e., the sale can be considered as an entity

- Database design can be difficult
  - When #entities and #relationships are large

# Design Approaches

- Entity-Relationship Model (covered in this chapter)
  - Specifies an enterprise schema representing overall logical structure of a database
  - Models an enterprise as [a collection of entities and relationships]
  - Represented diagrammatically by an entity-relationship diagram (E-R diagram)
    - Express overall logical structure of a database graphically




- Normalization Theory (coming in the next few weeks)
  - Formalize what designs are bad, and test for them

# Entity and Entity Sets

- An entity is an object that exists and is distinguishable from other objects
  - Concrete entity: specific person, company, plant, book
  - Abstract entity: flight reservation, course, course offering

- 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
  - Entity sets may not be disjoint (e.g., person vs. instructor and student)

# Entity and Entity Sets

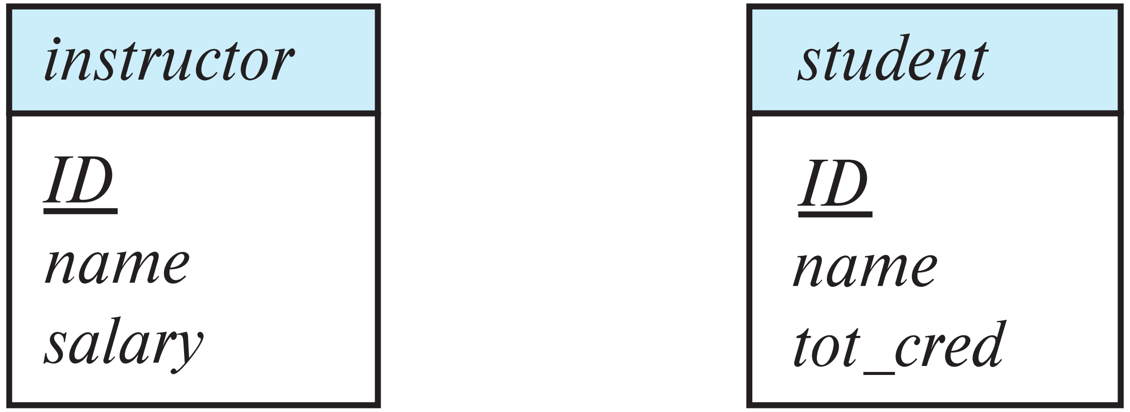
- An entity is represented by a set of attributes; i.e., descriptive properties possessed by all members of an entity set
- Example:

```
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
  - Government-issued ID number as the primary key
    - May have privacy and security issues
  - Enterprise-issued ID number

# 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

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

- A relationship set is a set of relationships of the same type (e.g., advising)
  - A mathematical relation among $n \geq 2$ (possibly non-distinct) entities, each taken from entity sets

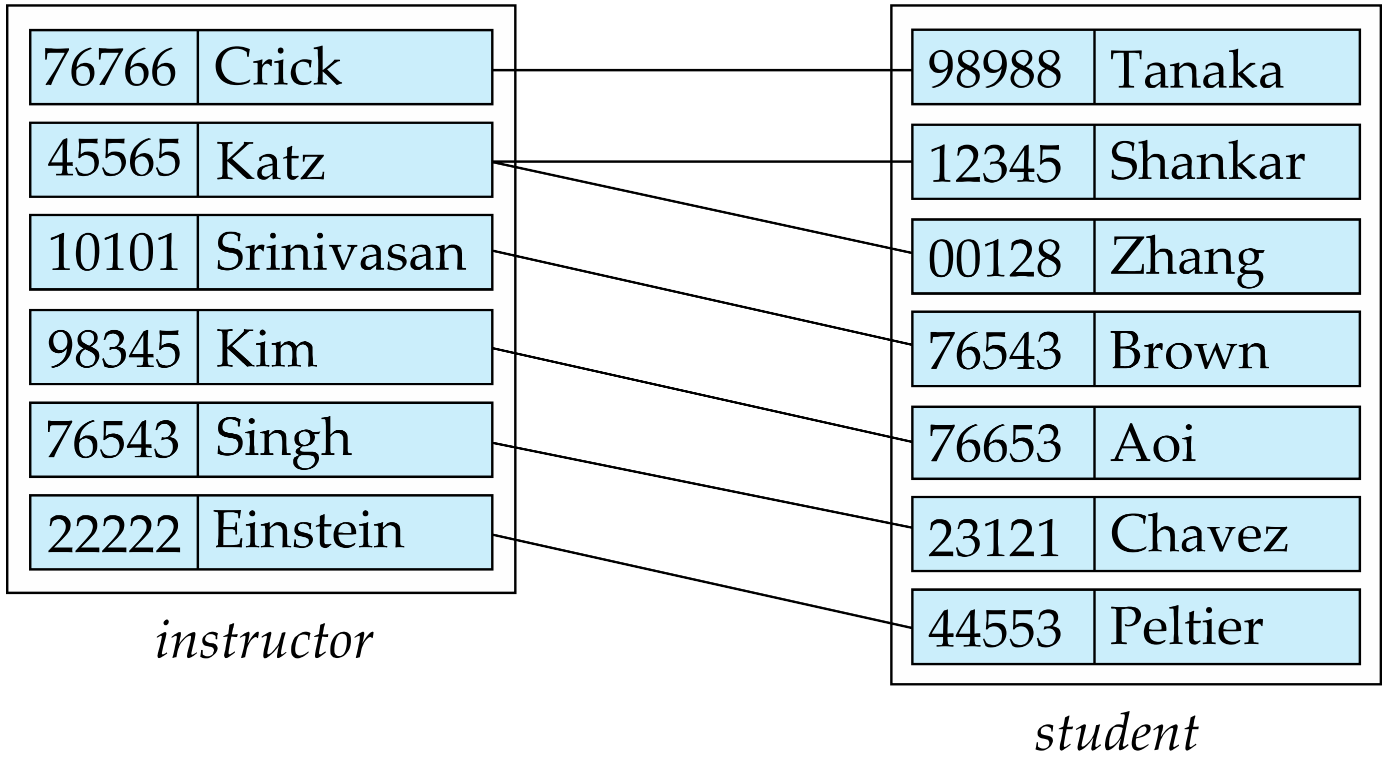
$$\{(e_1, e_2, \ldots e_n) \mid e_1 \in E_1, e_2 \in E_2, \ldots, e_n \in E_n\}$$

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

  - Example: $(44553,22222) \in$ advisor
  - The association between entity sets is referred to as participation
    - Entity sets $E_1, E_2, \ldots, E_n$ participate in relationship set R

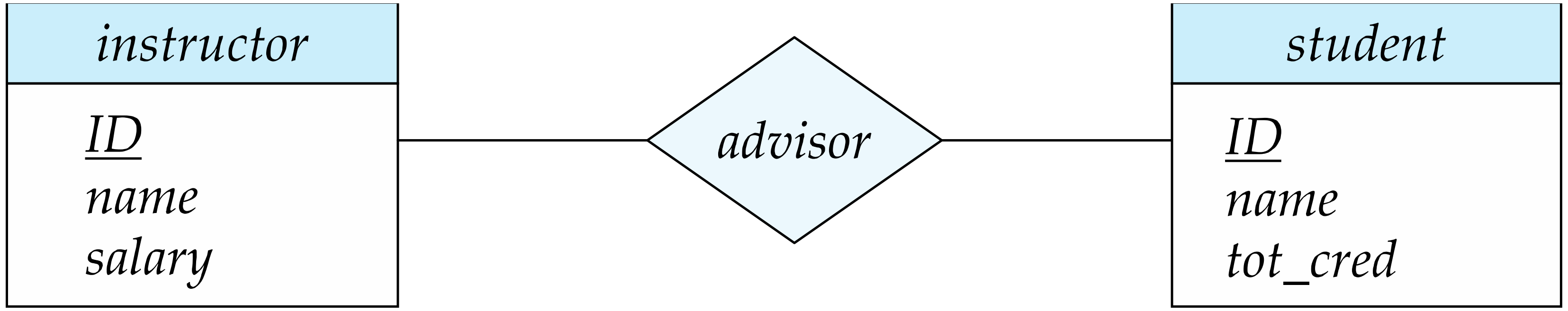
# Relationship Sets

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



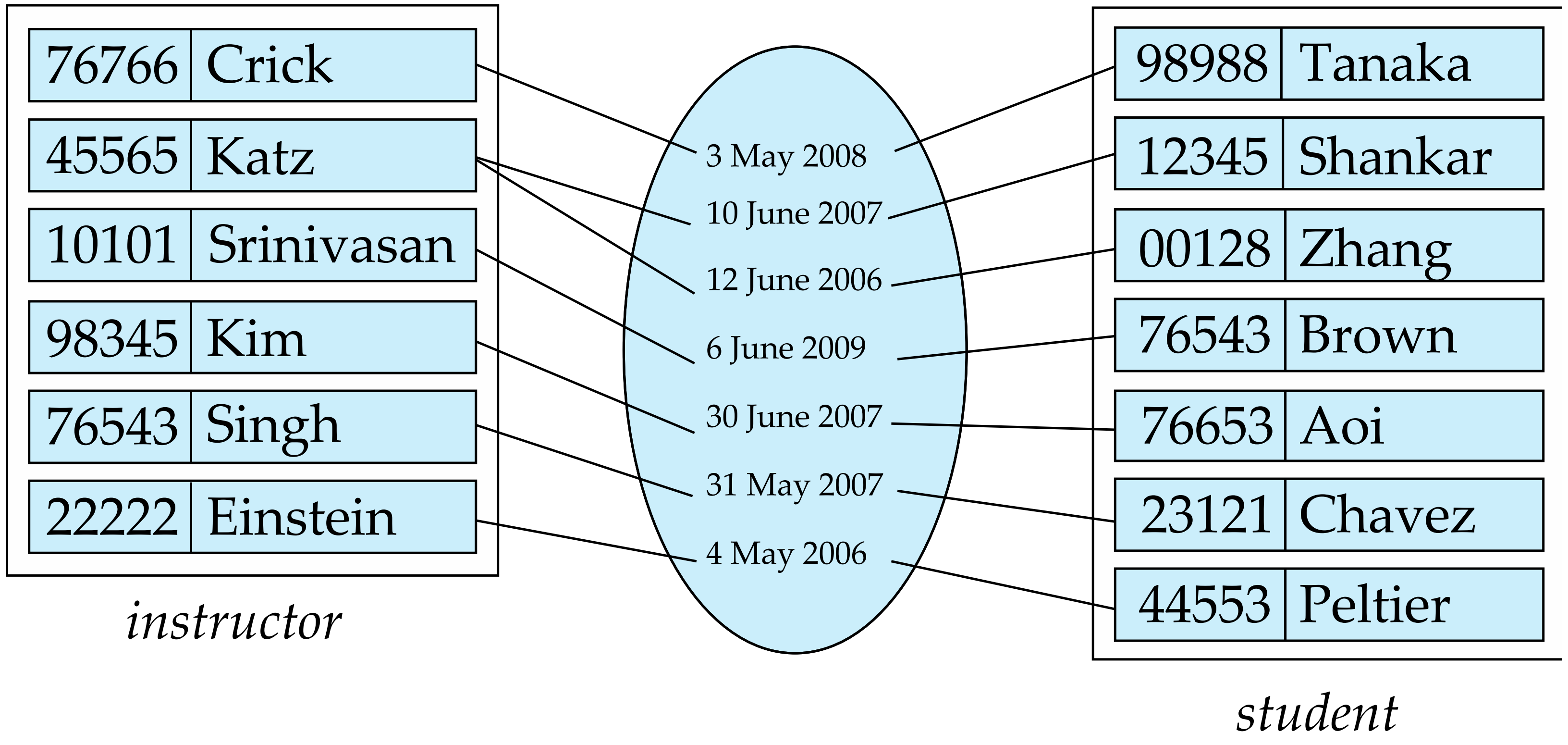
# Representing Relationship Sets via E-R Diagrams

- Use diamonds to represent relationship sets



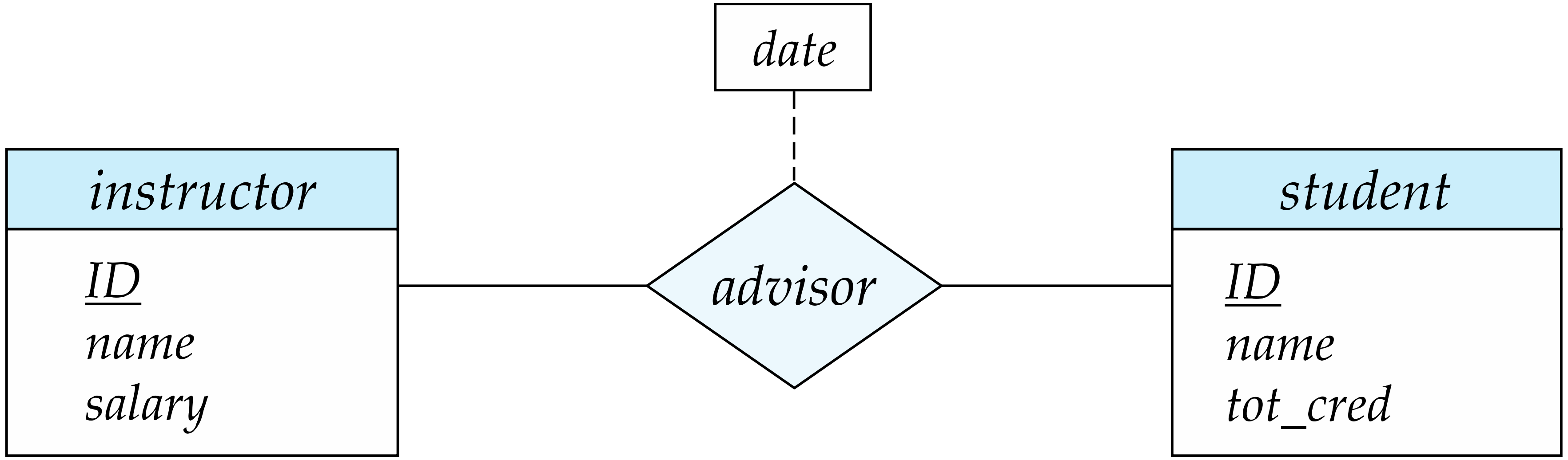
# Relationship Sets (Cont.)

- A descriptive attribute can be associated with a relationship set.
  - E.g., 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



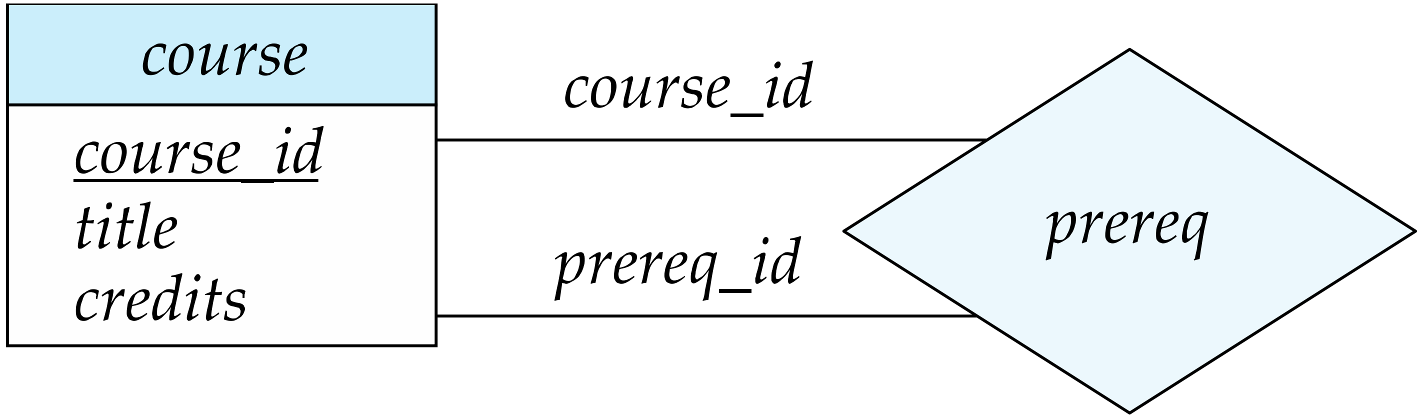
# Relationship Sets with Attributes

- Represented by using an undivided rectangle
- Linked to the diamond representing the relationship set via a dashed line



# Roles

- Entity sets of a relationship need not be distinct (i.e., can be the same entity set)
  - We can create self-pointing relationships for an entity set
  - Each occurrence of an entity set plays a "role" in the relationship
  - Example: A relationship set to represent the prerequisites of a course
    - E.g., Data Structure depends on Introduction to Programming
    - The labels `course_id` and `prereq_id` are called roles




- If the entity sets participating in a relationship are distinct,
  - Their roles are implicit and usually are not specified

# Degree of a Relationship Set

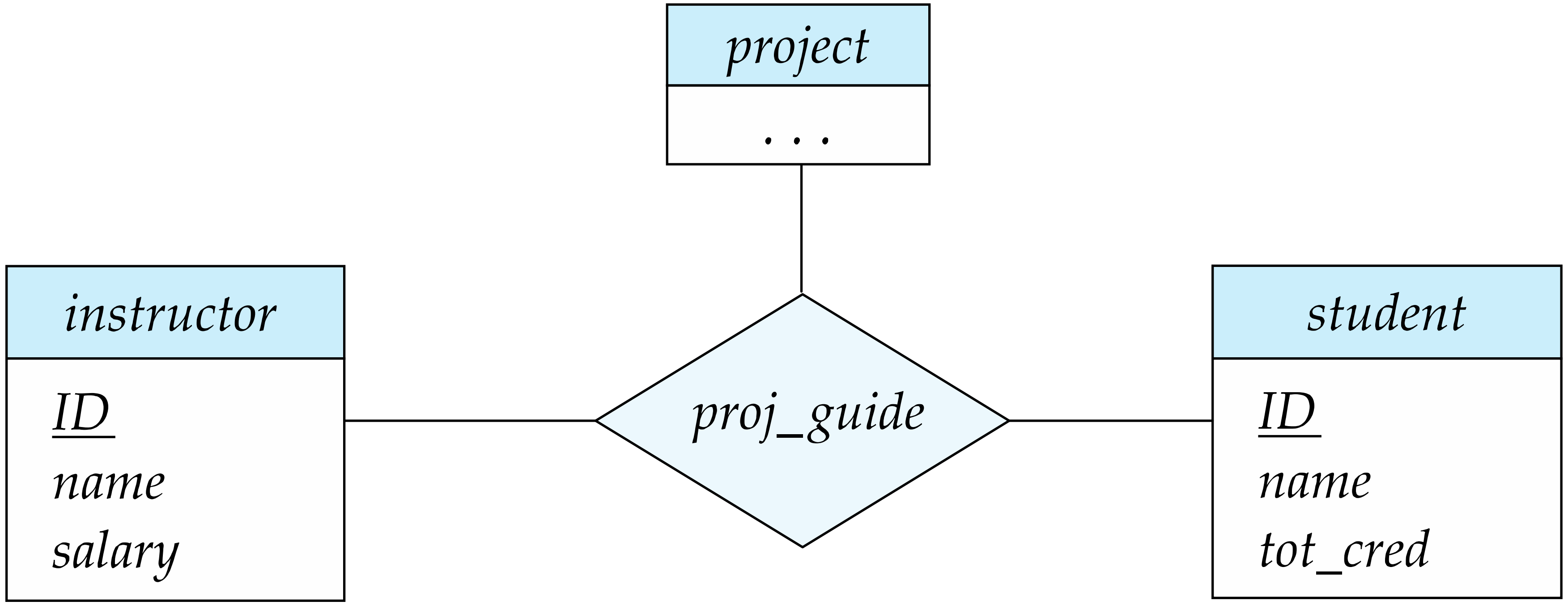
- Defined as the number of entity sets participating in a relationship set

- 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
  - E.g., students work on research projects under the guidance of an instructor
    - relationship `proj_guide` is a ternary relationship among instructor, student, and project
      - A particular student is guided by a particular instructor on a particular project

# Non-binary Relationship Sets

- Although 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

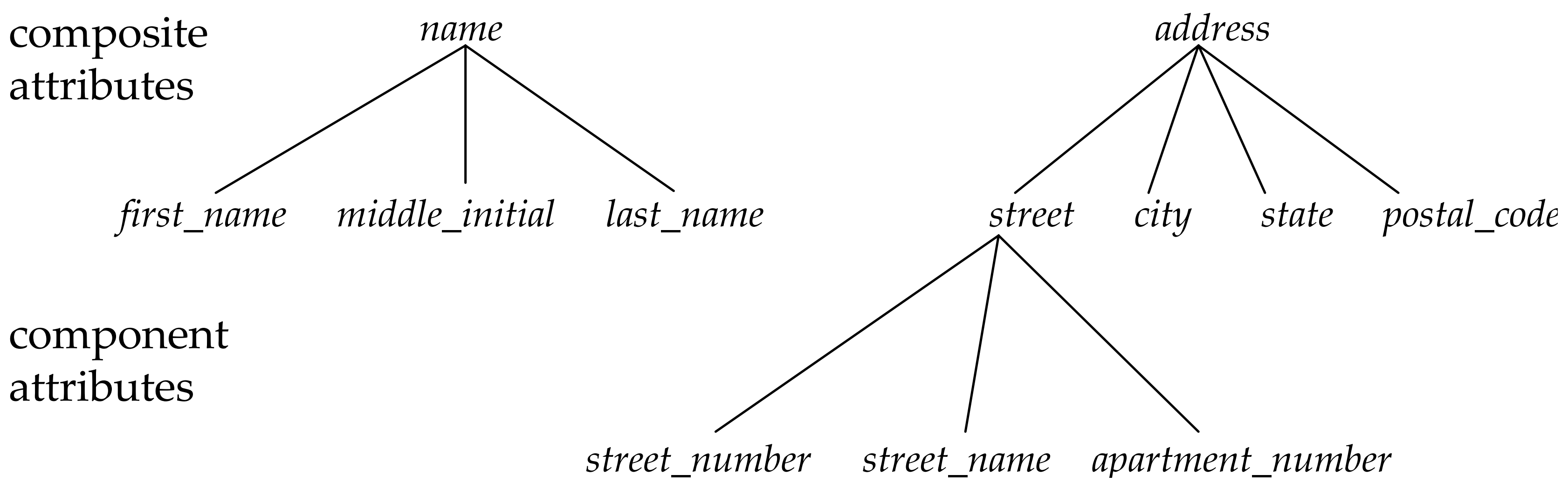


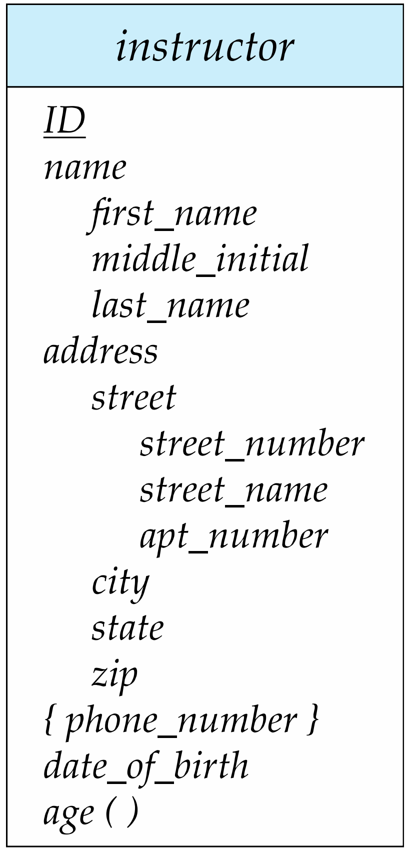
# Complex Attributes

- Attribute types
  - Simple (i.e., not divided into subparts) and composite (i.e., divided into subparts) attributes

# Composite Attributes

- Composite attributes allow us to divided attributes into subparts
  - Sometimes we may only use part of the attributes
  - In this case, composite attribute is a good design choice
- Allow us to group together related attributes, making the modeling cleaner
- A composite attribute may appear as a hierarchy



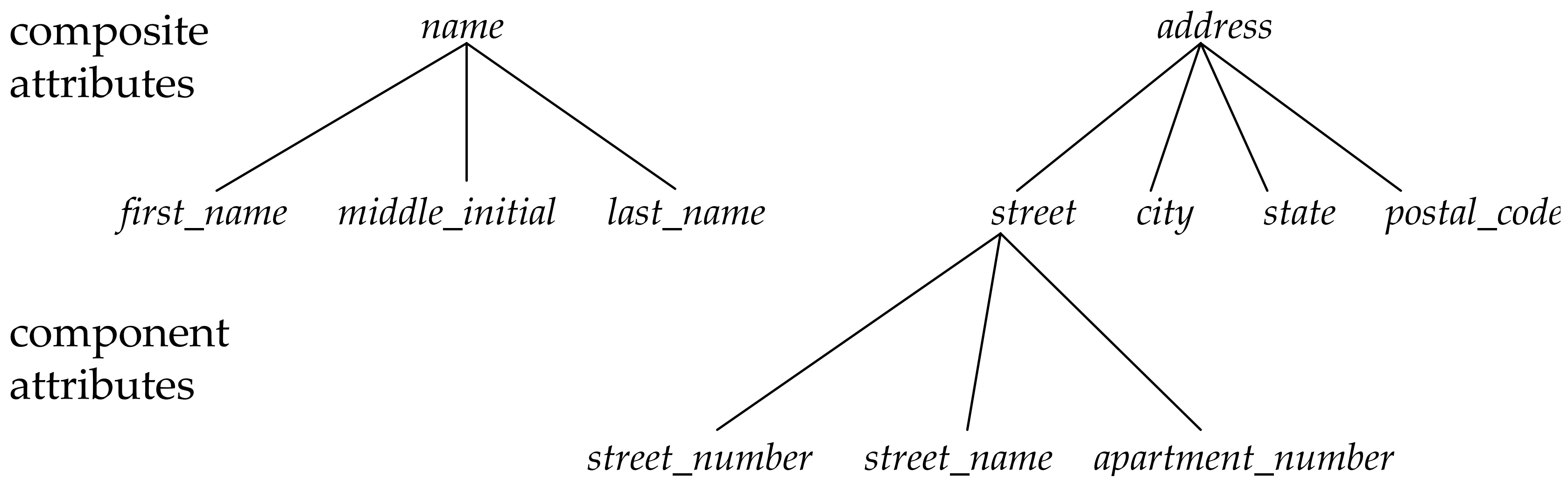

E-R notation

# Complex Attributes

- Attribute types
  - Simple (i.e., not divided into subparts) and composite (i.e., divided into subparts) attributes
  - Single-valued and multivalued attributes
    - Single-valued attribute: e.g., for a student, only one *student_id*
    - Multivalued attribute
      - *phone_numbers*: a person can have 0, 1, or multiple phone numbers at the same time
      - *grades*: each student may have multiples grades in a course
      - *department_name*: an instructor may be hired by 0, 1, or multiple departments
  - Derived attributes
    - Can be computed from other attributes
    - Example: *age* computed based on *date_of_birth*, #students advised by an instructor
- **Domain**: set of permitted values for each attribute
  - *course_id* might be the set of all text strings of a certain length
  - *semester* might be strings from the set {Fall, Winter, Spring, Summer}

# E-R Notations of Composite Attributes

- E-R notations for
  - Composite attributes
  - Multivalued attributes
  - Derived attributes

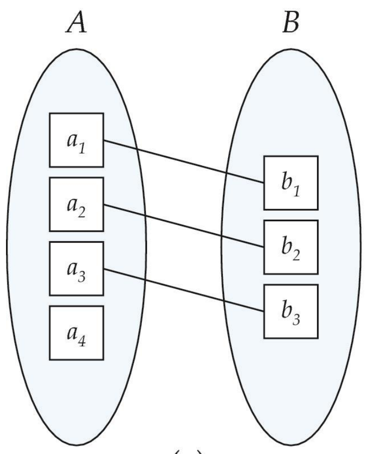



*instructor*

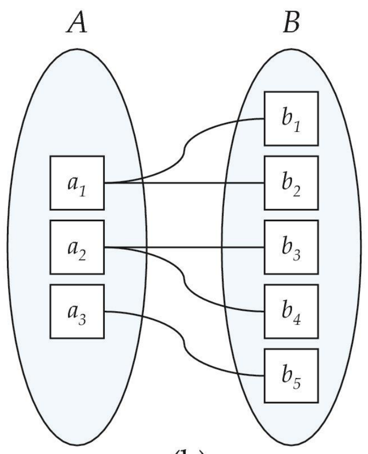
*<u>ID</u>*
*name*
    *first_name*
    *middle_initial*
    *last_name*
*address*
    *street*
        *street_number*
        *street_name*
        *apt_number*
    *city*
    *state*
    *zip*
*{ phone_number }*
*date_of_birth*
*age ( )*

# Mapping Cardinality Constraints

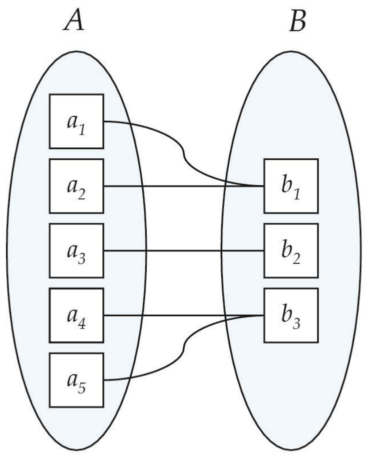
- Mapping Cardinality（映射基数）
  - Express the number of entities to which another entity can be associated via a relationship set
    - Most useful in describing binary relationship sets
    - Can also help describe non-binary relationship sets
- For a binary relationship set, the mapping cardinality must be one of the following
  - One to one, one to many, many to one, many to many



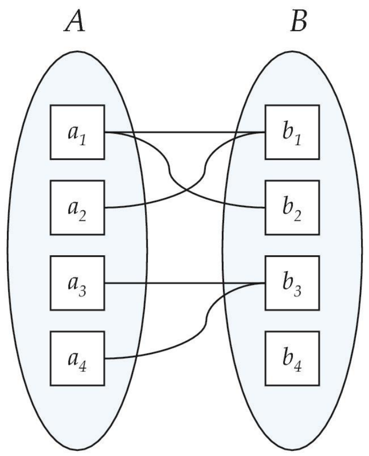

(a)




(b)



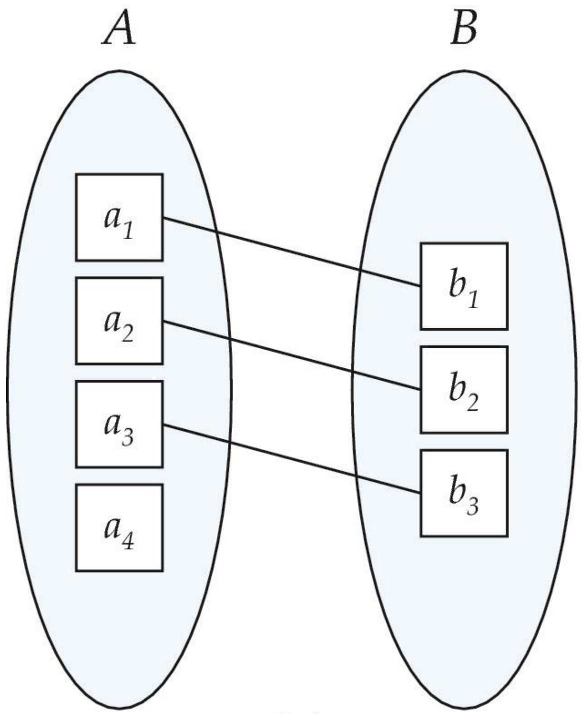

(a)




(b)

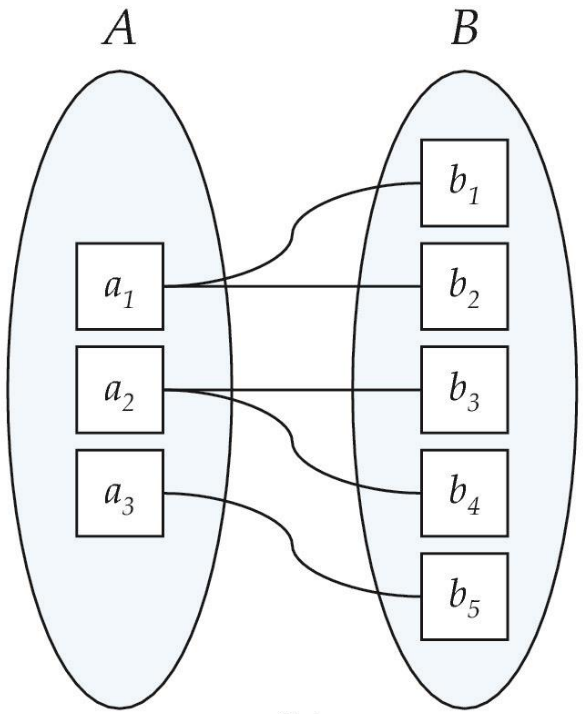
# Mapping Cardinalities

- Every entity in A is associated with at most one entity in B
- Same for entity set B




(a)

One to one




(b)

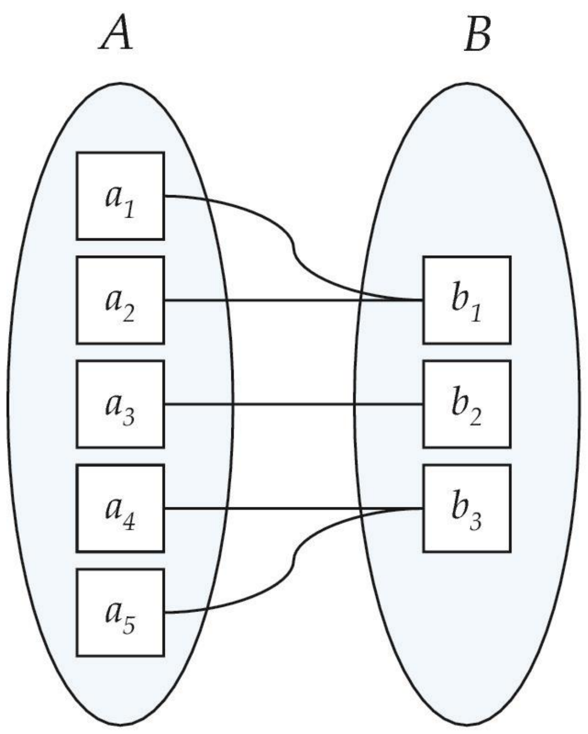
One to many

- Every entity in A is associated with 0, 1, or more entities in B
- Every entity in B is associated with at most one entity in A

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

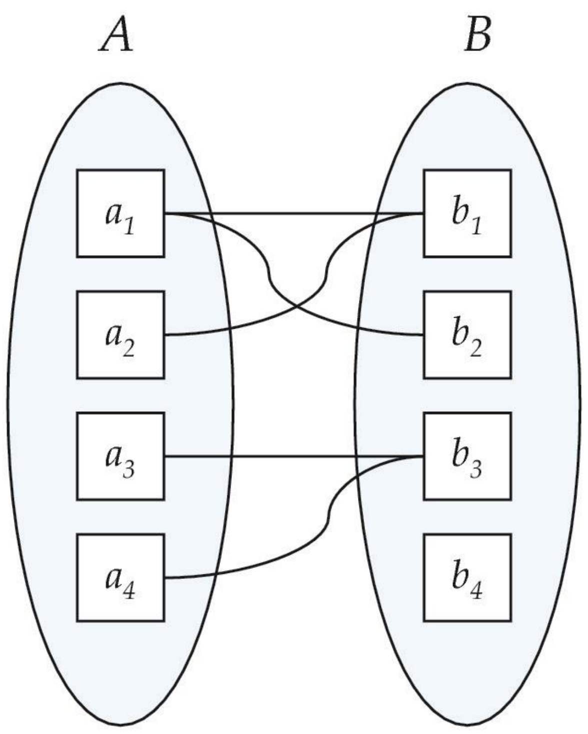
# Mapping Cardinalities

- Every entity in A is associated with at most one entity in B
- Every entity in B is associated with 0, 1 or more entities in A




(a)

Many to one




(b)

Many to many

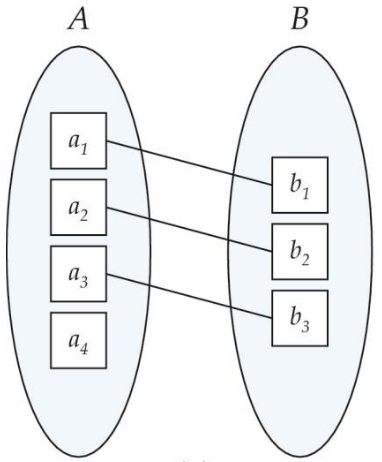
- Every entity in A is associated with 0, 1, or more entities in B
- Same for B

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

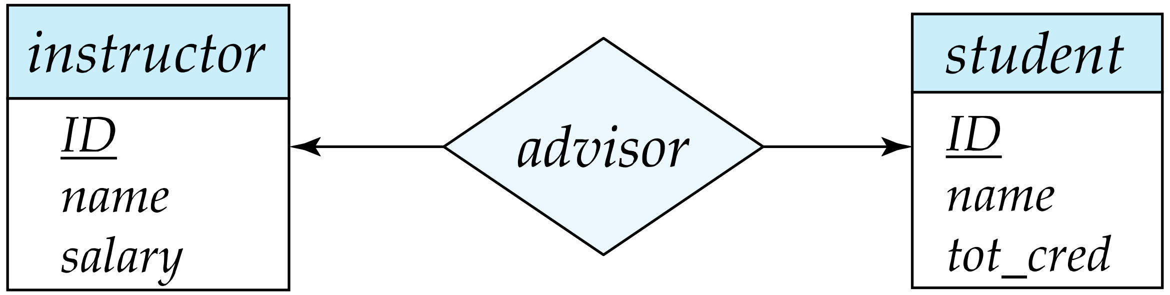
# Representing Cardinality Constraints in ER Diagram

- We express cardinality constraints by:
  - drawing either a directed line (→), signifying "one"
  - or an undirected line (—), signifying "many"
- … between the relationship set and the entity set

- <u>One-to-one relationship</u> between an instructor and a student :
  - A student is associated with at most one instructor via the relationship advisor

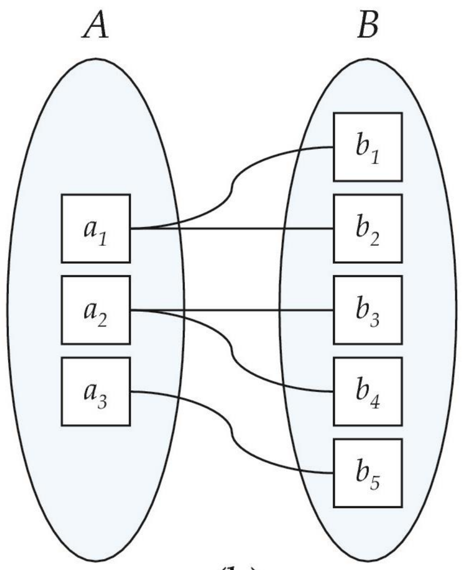


(a)

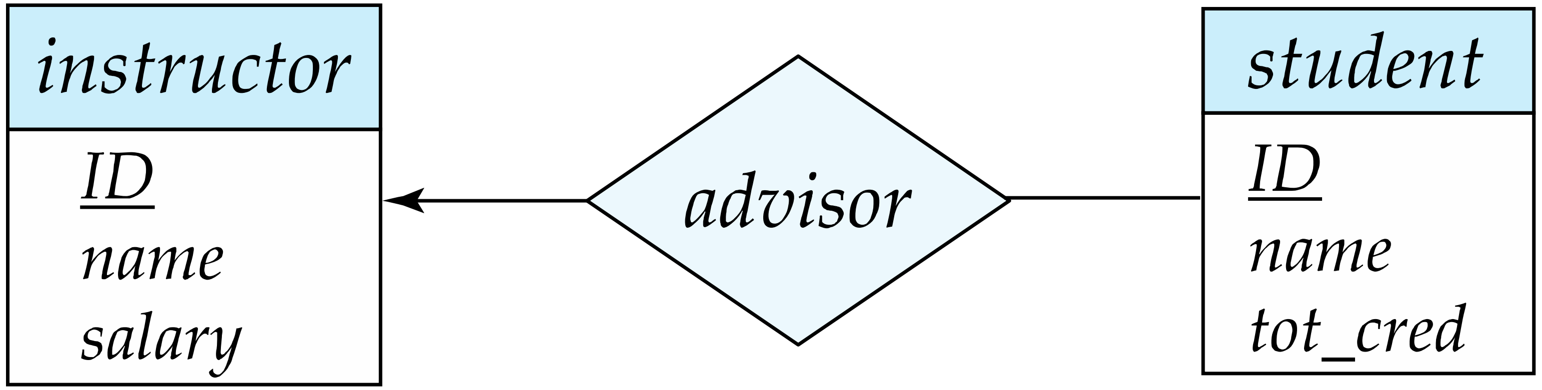


# Representing Cardinality Constraints in ER Diagram

- 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

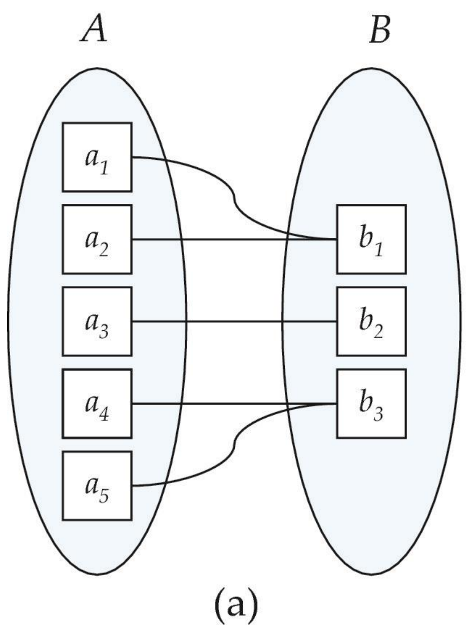


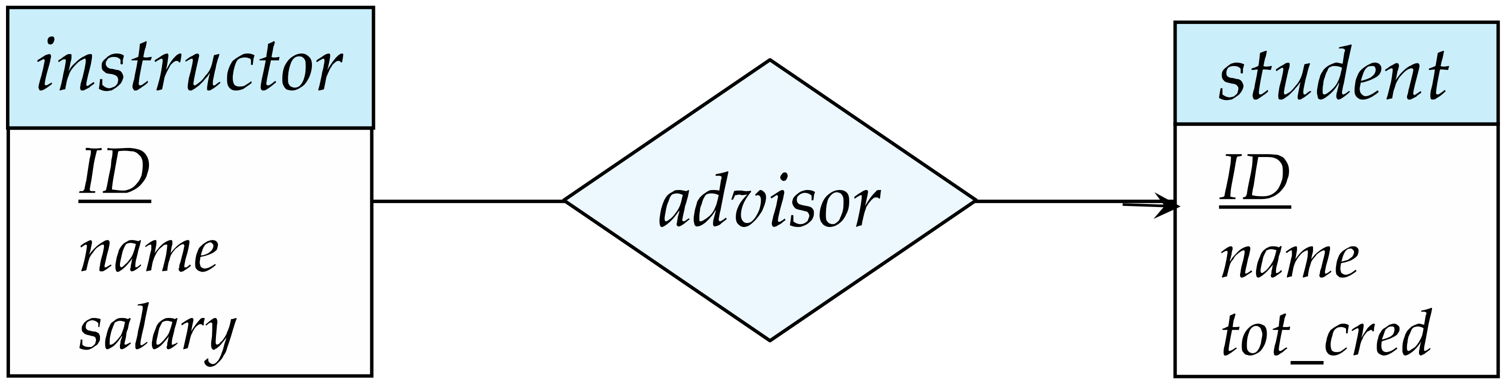
(b)



# Representing Cardinality Constraints in ER Diagram

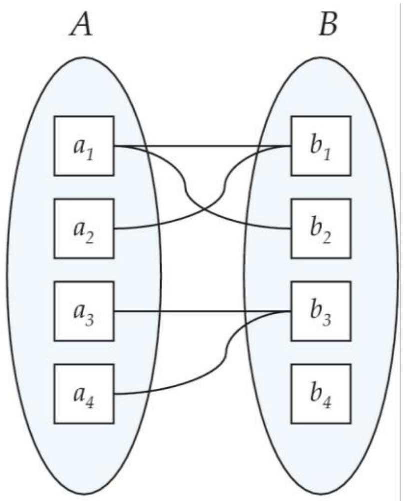
- In a many-to-one relationship between an instructor and a student,
  - An instructor is associated with at most one student via advisor
  - A student is associated with several (including 0) instructors via advisor



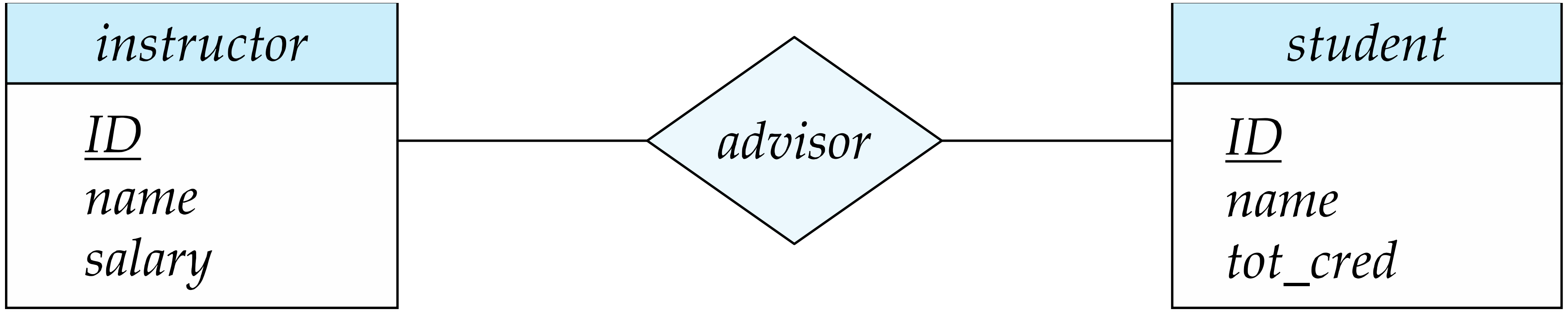



# Representing Cardinality Constraints in ER Diagram

- 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

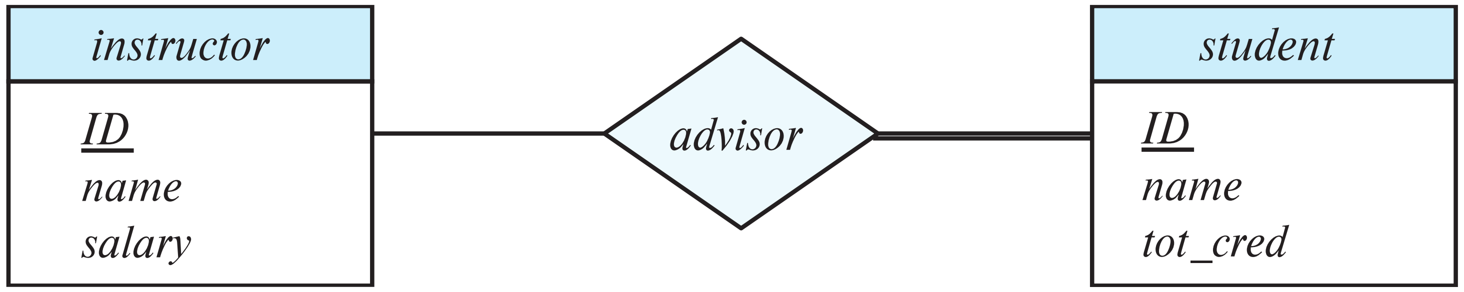


(b)



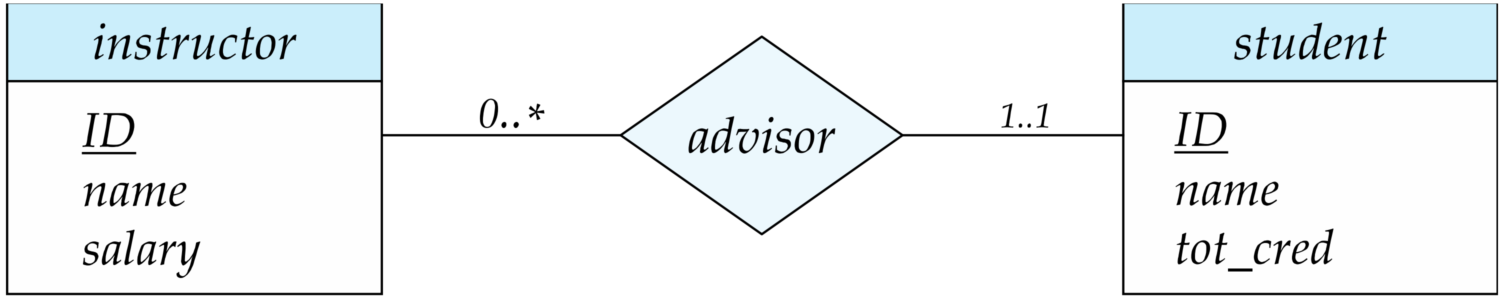
# Total and Partial Participation

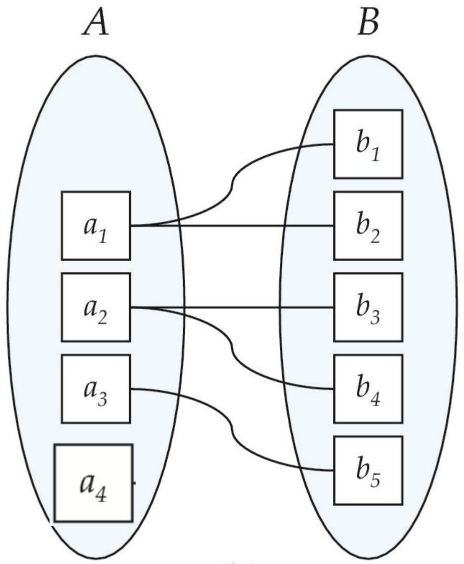
- Total participation (indicated by *undirected double line, i.e., =*)
  - Every entity in the entity set participates in at least one relationship in the relationship set
    - E.g., Participation of student in advisor relation is total
      - i.e., every student must have an associated instructor
- Partial participation (undirected line, i.e., —)
  - Some entities may not participate in any relationship in the relationship set
    - E.g., 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
  - An entity can participate in at minimum l and at maximum h relationships
  - 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



(b)

one to many,
not many to one

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

# Primary Key

- Superkey, candidate key, and primary key also apply to entity and relationship sets (like in relation schemas)
- Primary key provides a way to specify how entities and relations are distinguished
  - Every entity and relationship must be able to be uniquely identified

# Primary Key for Entity Sets

- By definition, individual entities are distinct
  - From database perspective, their differences must be expressed in terms of their attributes

- The values of the attributes 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

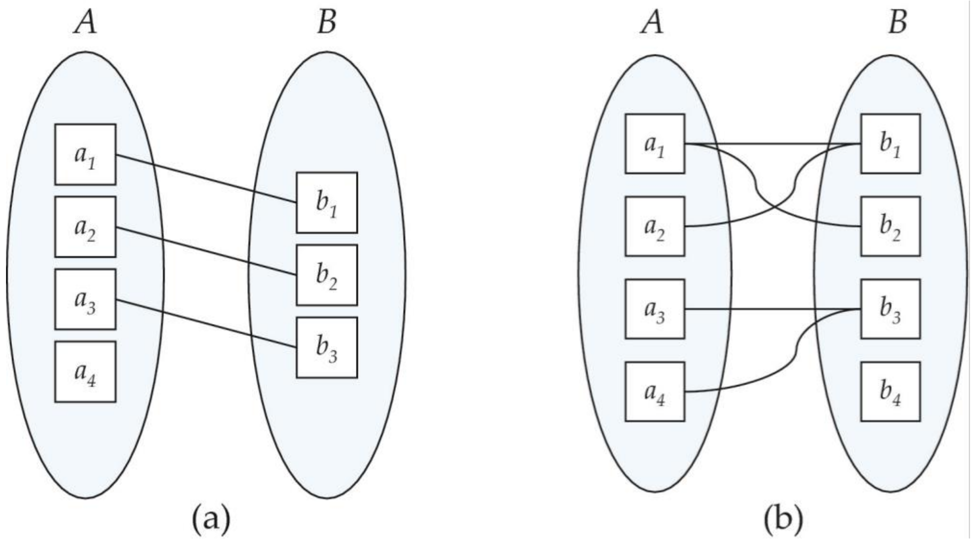
- A key for an entity is a set of attributes that <u>suffice to distinguish entities from each other</u>

# 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 E1, E2, ..., En
  - The union of the primary keys of entity sets E1, E2, ...En form a superkey for R
    - The superkey plus the attributes a1, a2, .., am associated with it describes a relationship instance
- 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 union of the primary keys is a minimal superkey and is chosen 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.

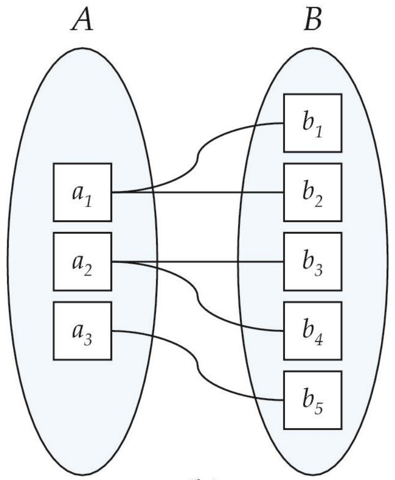



* *K* is a **superkey** of *R* if values for *K* are sufficient to identify a unique tuple of each possible relation *r(R)*

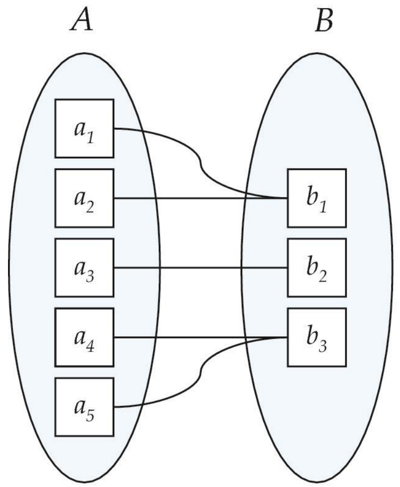
- Example: {*ID*} and {ID,name} are both superkeys of *instructor*.

# Choice of Primary key for Binary Relationship

- One-to-Many relationships
  - The primary key of the "Many" side is a minimal superkey and is used as the primary key
    - E.g., each student can have at most one advisor → then the primary key of advisor is simply the primary key of student.
- Many-to-one relationships
  - The primary key of the "Many" side is a minimal superkey and is used as the primary key



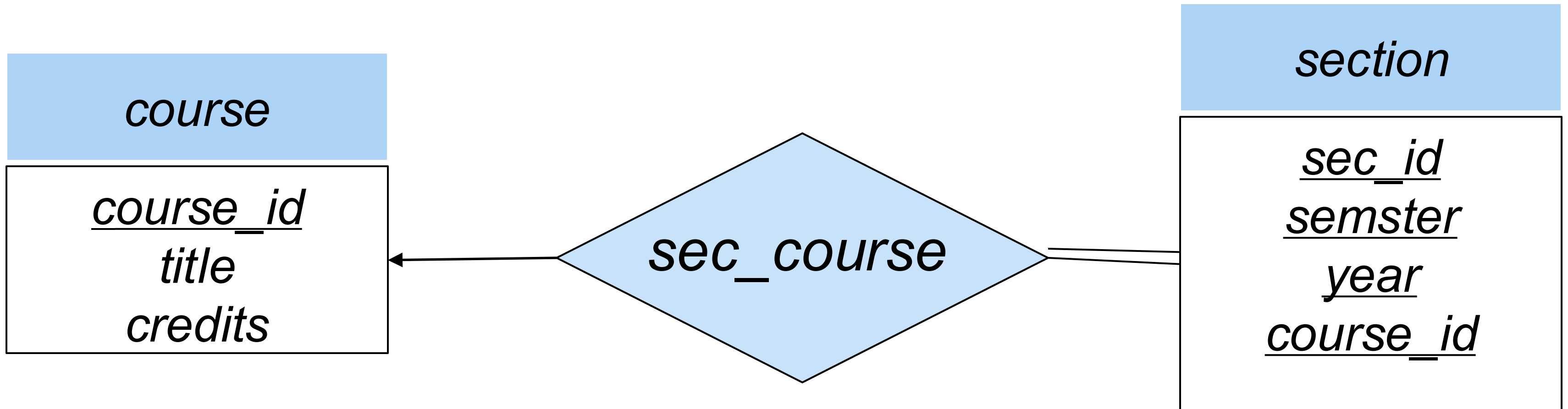
(b)



(a)

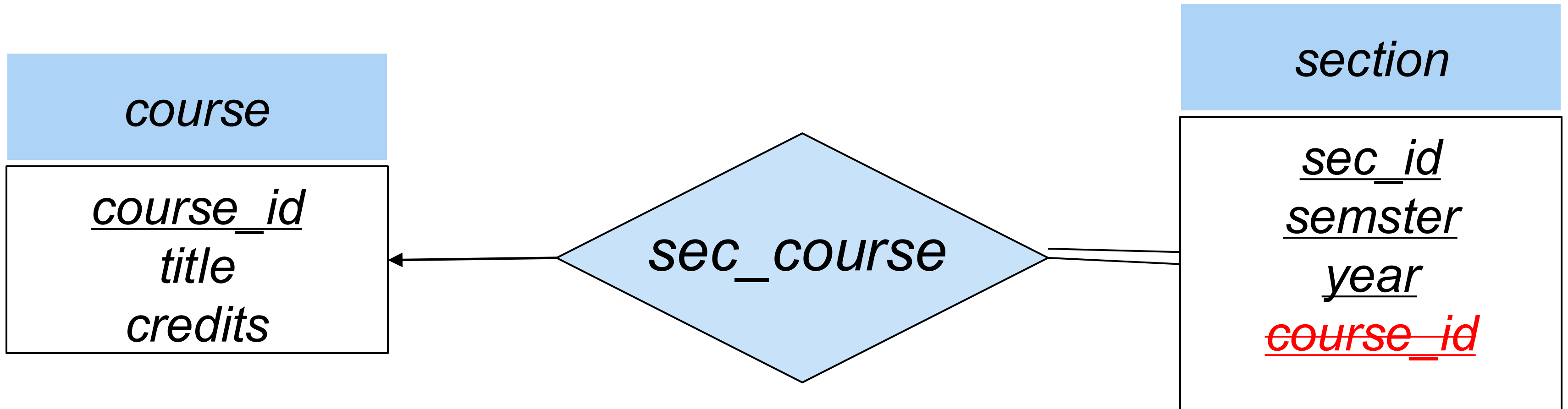
# Weak Entity Sets (弱实体集)

- Consider a *section* entity, which is uniquely identified by *course_id*, *semester*, *year*, and *sec_id*
  - Clearly, *section* entities are related to *course* entities
  - Suppose we create a relationship set *sec_course* between *section* and *course*
    - 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 （弱实体集 ）

- 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, *section* then does not have enough attributes to identify a particular section entity uniquely
  - Although each section entity is distinct, sections for different courses may share the same sec_id, year, and semester



# Weak Entity Sets （弱实体集 ）

- To deal with this problem, we treat the relationship *sec_course* as a special relationship that provides extra information, in this case, *course_id*, to identify section entities uniquely
- A weak entity set is one whose existence is dependent on another entity, called its identifying entity
  - Weak entity set: entity set that does not have sufficient attributes to form a primary key
  - 弱实体的 存在 依赖于另一个实体
- 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
  - 弱实体的 标识 依赖于另一个实体

# Weak Entity Sets （弱实体集）

- An entity set that is not a weak entity set is termed a strong entity set （强实体集）

- 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

# 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
  - The identifying relationship is many-to-one from the weak entity set to the identifying entity set
  - The participation of the weak entity set in the relationship is total
  - The identifying relationship set should not have any descriptive attributes, since any such attributes can instead be associated with the weak entity set

# Expressing Weak Entity Sets

- The relational schema, eventually created for *section*, does have the attribute *course_id*,
  - Even though we have dropped the attribute *course_id* from *section* in E-R diagram

- In the relational schema:
  - Primary key for *section*: (*course_id*, *sec_id*, *semester*, *year*)
    - Primary key of identifying (strong) entity set, plus discriminators of weak entity set
  - We have constraints on the *section* schema, with the attribute *course_id* referencing the primary key of the *course* schema

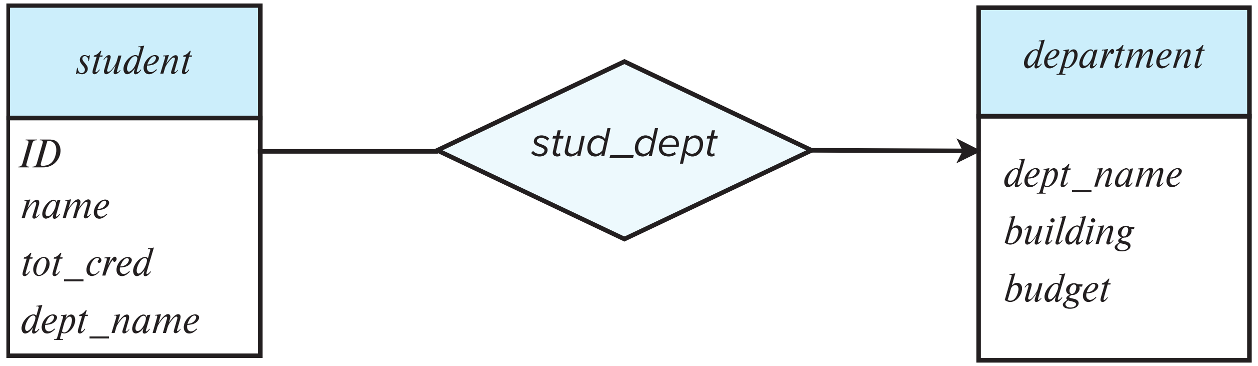
# Removing Redundant Attributes in Entity Sets

# Redundant Attributes

- Database usually starts with
  - Identifying entity sets that should be included
    - E.g., in university organization, we have instructors and students
  - Choosing appropriate attributes
    - Represent the various values we want to capture in the database
    - Depending on the designer, who has a good understanding of the enterprise
    - E.g., for instructor, we can record
      - ID, name, dept name, and salary
      - phone number, office number, home page, and others
- Once entities and attributes are chosen, relationship sets among the entities are formed
  - The relationship sets may lead some attributes to be redundant, and need to be removed

# Redundant Attributes

- Suppose we have entity sets:
  - student = {ID, name, tot_cred, dept_name}
  - department = {dept_name, building, budget}
- We model the fact that each student has an associated department using a relationship set stud_dept
- In the E-R diagram, the attribute dept_name in student replicates the one in the relationship and is redundant
  - and needs to be removed

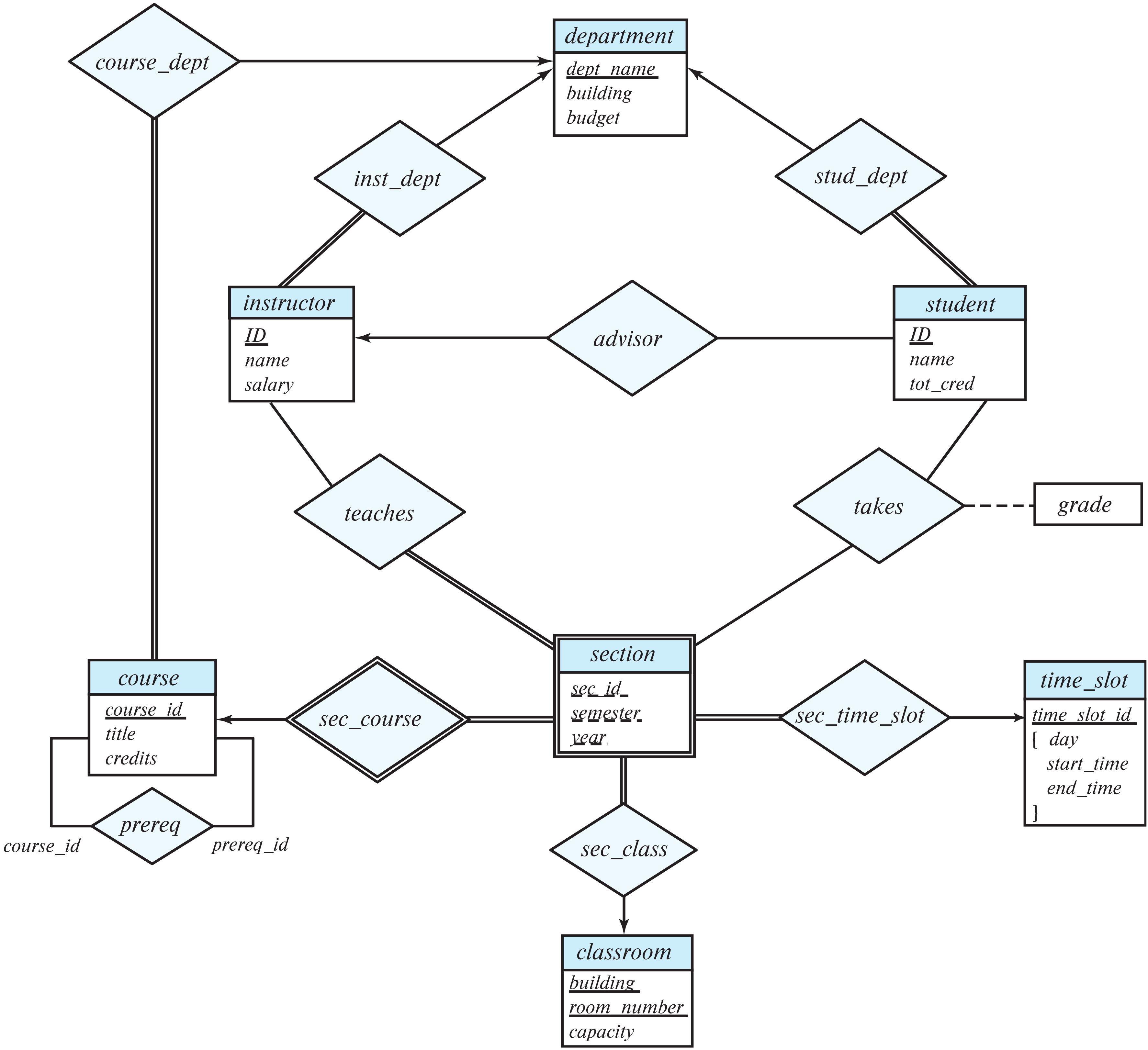



(a) Incorrect use of attribute

  - BUT: when converting back to tables from E-R diagrams, in some cases the attribute gets reintroduced
    - Depending on the mapping cardinality (e.g., when each student has at most one department)
- A good entity-relationship design does not contain redundant attributes

# Quick Example

- Each instructor must have exactly one associated department
- Each instructor can have at most one associated department
- Every course must be in some department
- Every student must be majoring in some department
- A course (and a student) can be related to only one department
- Relationship set *takes* has a descriptive attribute grade
- Each student has at most one advisor
- *section* is a weak entity set, with attributes sec_id, semester, and year forming the discriminator
- *sec_course* is the identifying relationship set relating weak entity set *section* to the strong entity set *course*



# Reduction to Relation Schemas

# Reduction to Relation Schemas

- Both E-R model and relational database model are abstract, logical representations of real-world enterprises
  - We can convert an E-R design into a relational design
- 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 schemas
  - For each entity set and relationship set, there is a unique schema that is assigned the name of the corresponding entity set or relationship set
  - Each schema has a number of attributes, with unique names

# Representing Entity Sets

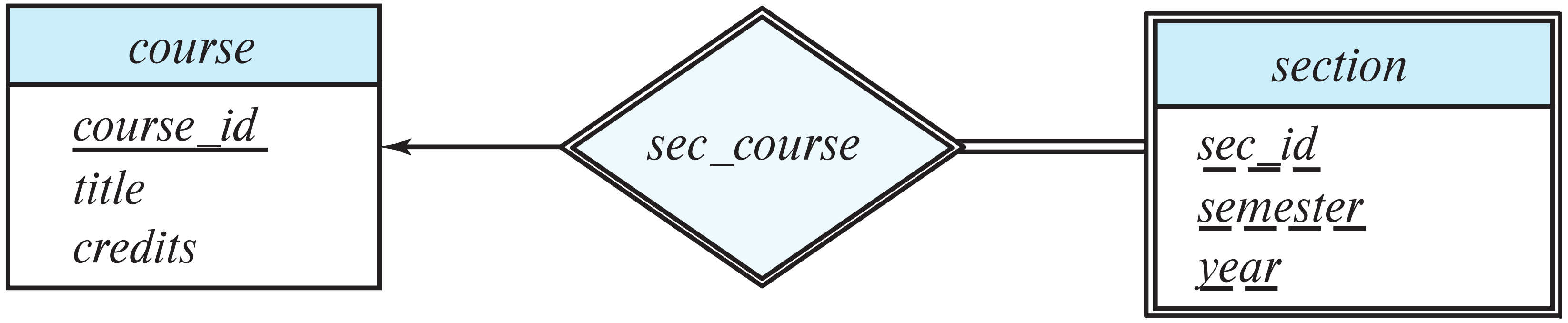
- A strong entity set reduces to a schema with the same attributes
  - Each tuple in a relation on this schema corresponds to one entity of the entity set

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

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

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

- Example



# Representation of Entity Sets with Composite Attributes

- Composite attributes are flattened out by creating a separate attribute for each component attribute
  - E.g., 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 (e.g., 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)

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

# 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
    - E.g., multivalued attribute phone_number of instructor is represented by a schema
    - Foreign-key constraint on *inst_phone*: ID references the instructor relation

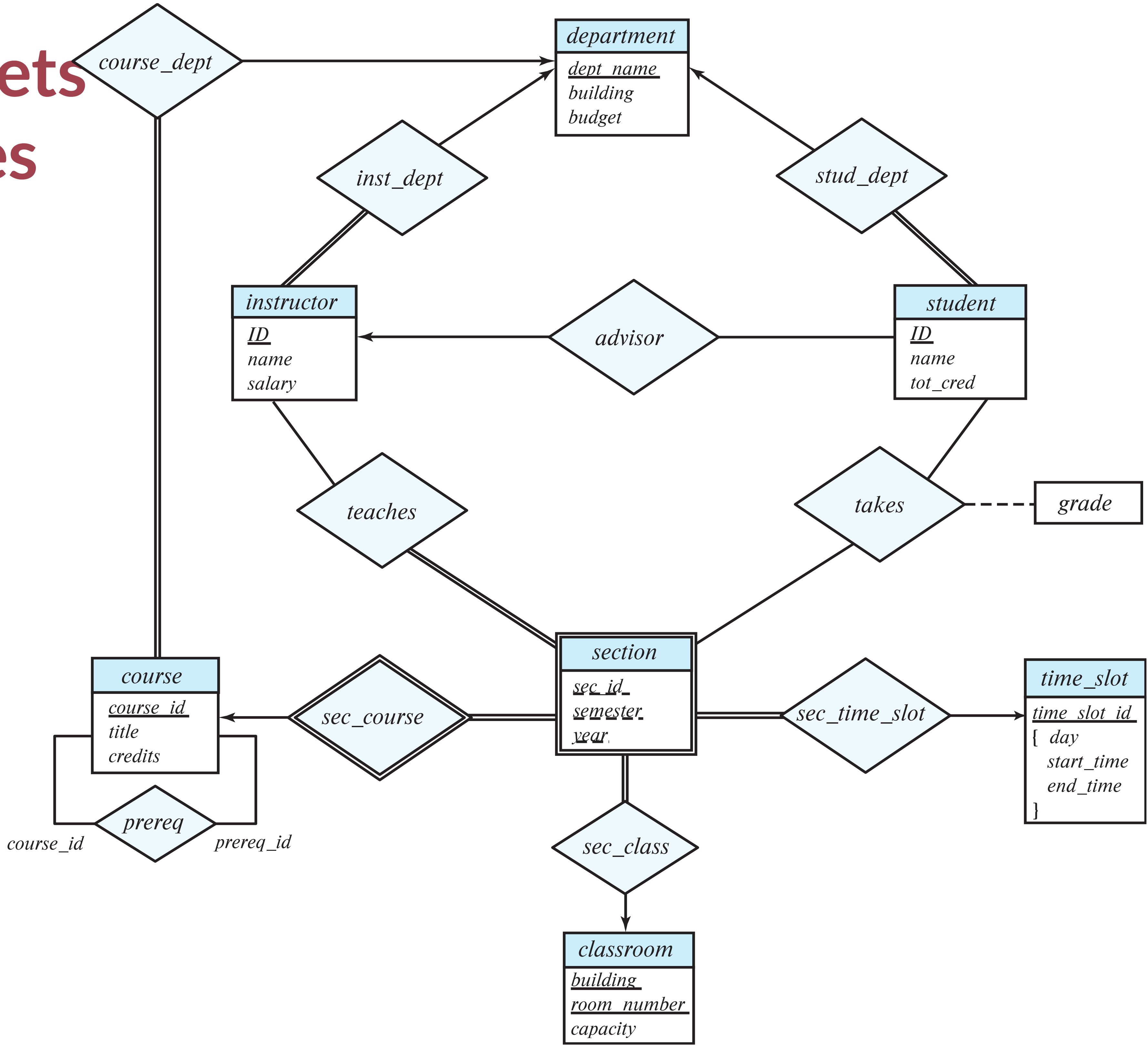
*inst_phone*= ( <u>*ID*</u>, <u>*phone_number*</u>)

  - Each value of the multivalued attribute maps to a separate tuple of the relation schema EM
    - E.g., 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)*

# Representation of Entity Sets with Multivalued Attributes

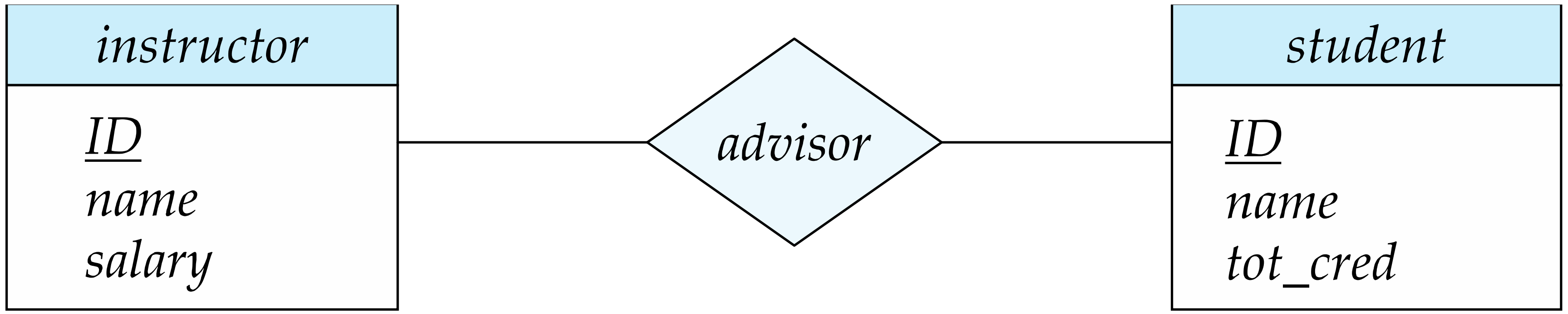
- time_slot = (time_slot_id, day, start_time, end_time)



# Representing Relationship Sets

- A many-to-many relationship set is represented as a schema with
  - Attributes consisted of the primary keys of the two participating entity sets
  - Descriptive attributes of the relationship set
- Example: schema for relationship set advisor
  - With two foreign keys created

$$advisor = (\underline{s\_id}, \underline{i\_id})$$

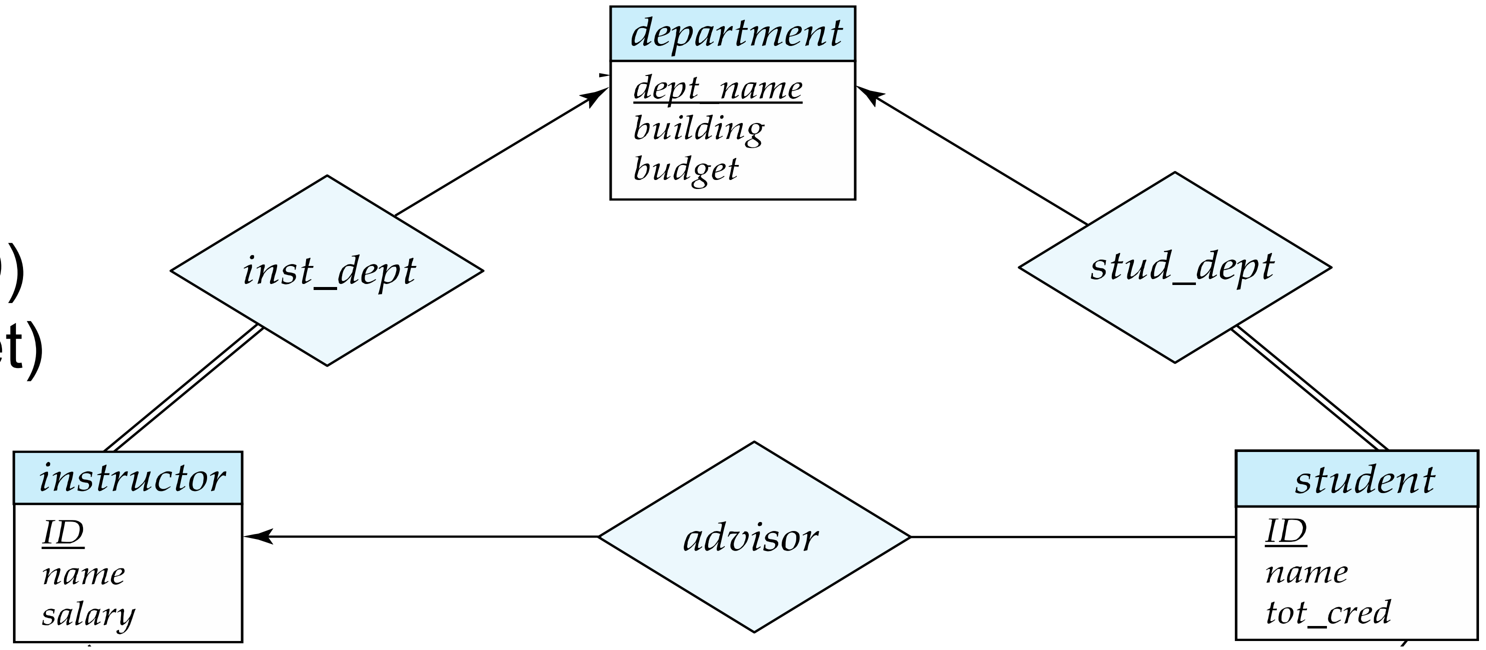


# Redundancy of Schemas

- Many-to-one and one-to-many relationship sets that are total on the many-side can be represented by adding an extra attribute to the "many" side, containing the primary key of the "one" side
  - Example: Instead of creating a schema for relationship set *inst_dept*, add an attribute *dept_name* to the schema arising from entity set *instructor*

Relation schemas: inst_dept, advisor and stud_dept are redundant

- instructor=(ID, name, salary, dept_name)
- student=(ID, name, tot_cred, instructor.ID)
- department=(dept_name, building, budget)



# 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 nulls

# Redundancy of Schemas

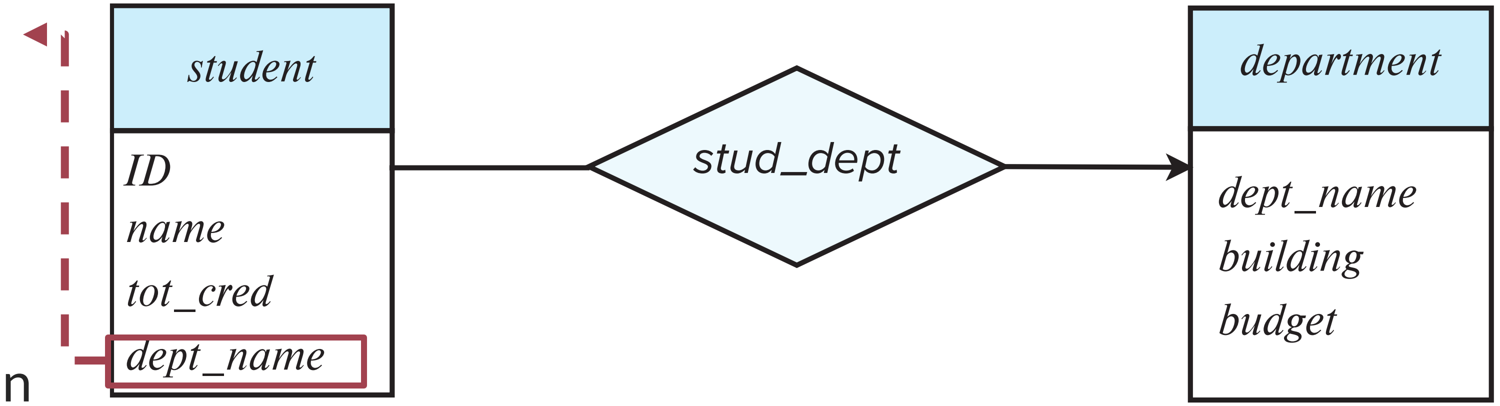
- The schema corresponding to a relationship set linking a weak entity set to its identifying strong entity set is **redundant**
  - Example: The *section* schema already contains the attributes that would appear in the *sec_course* schema

- Primary key for *section:* (*course_id*, *sec_id*, *semester*, *year*)
  - Primary key of identifying (strong) entity set, plus discriminators of weak entity set
  - We have constraints on the *section* schema, with the attribute *course_id* referencing the primary key of the *course* schema

# Design Issues

# Common Mistakes in E-R Diagrams

- Examples of erroneous E-R diagrams
  - (a) Use unnecessary attribute in entity set
    - … which is the primary key of another entity
    - Problem: data redundancy
      - The relationships are already presented in the relationship set *stud_dept*

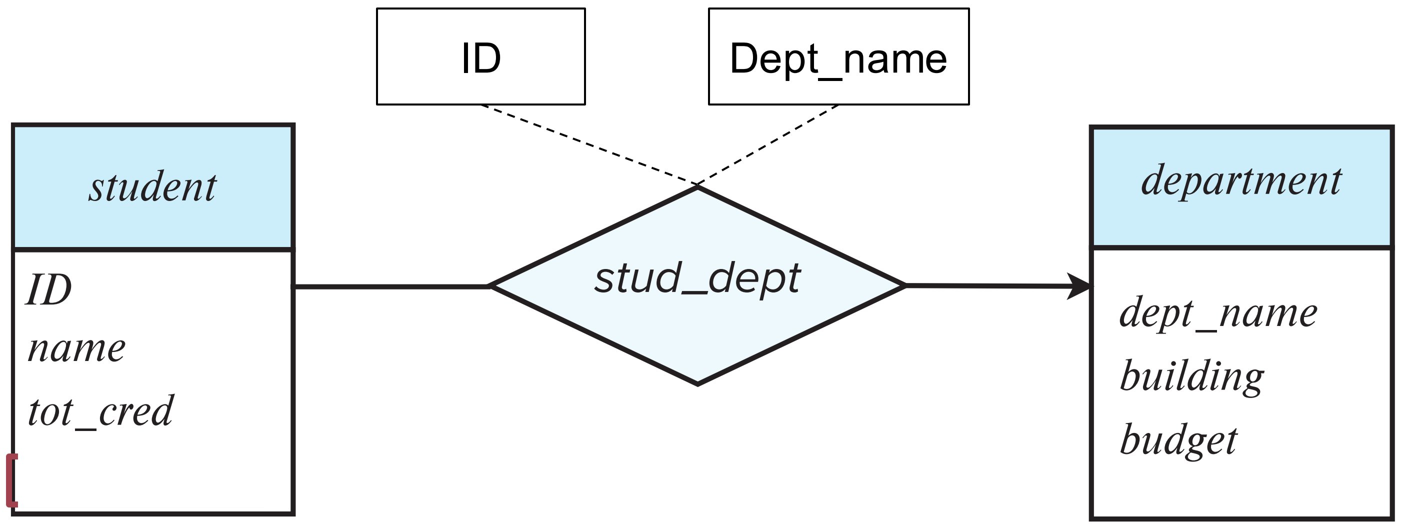



(a) Incorrect use of attribute

We should use *stud_dept*, as it makes the relationship between student and department explicit, rather than implicit via an attribute

# Common Mistakes in E-R Diagrams

- Examples of erroneous E-R diagrams
  - (b) Unnecessary attribute in relationship set
    - Designate the primary-key attributes of the related entity sets as attributes of the relationship set
    - Problem: data redundancy
      - primary-key attributes are already implicit in the relationship set*

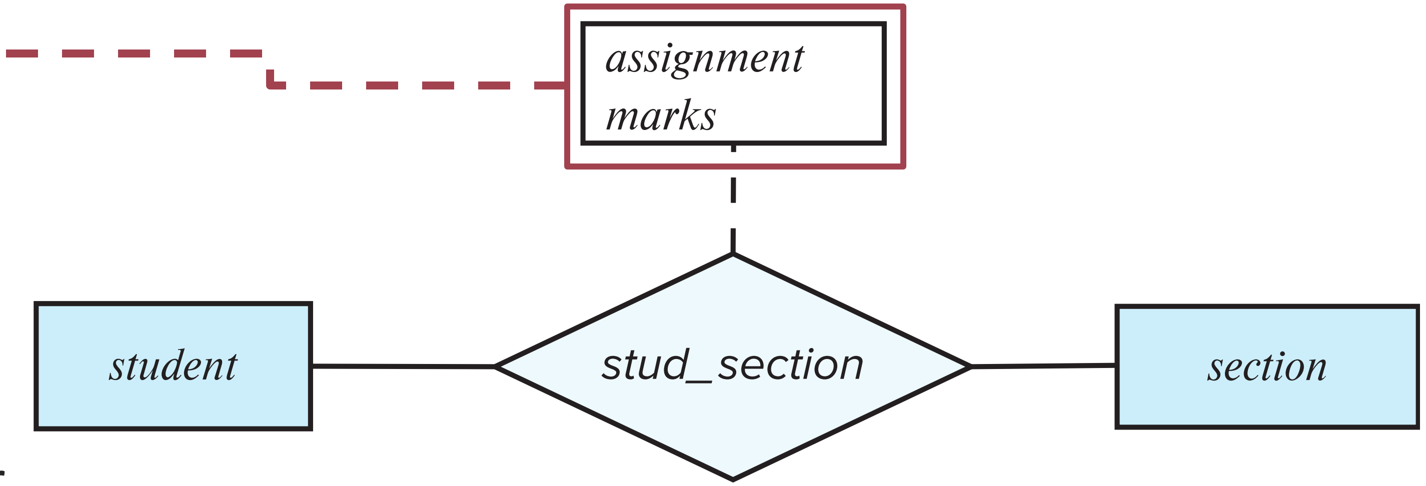



(a) Incorrect use of attribute

* When we create a relation schema from the E-R schema, the attributes may appear in a schema created from the stud_dept relationship set; however, they should not appear in the stud_dept relationship set in the E-R diagram

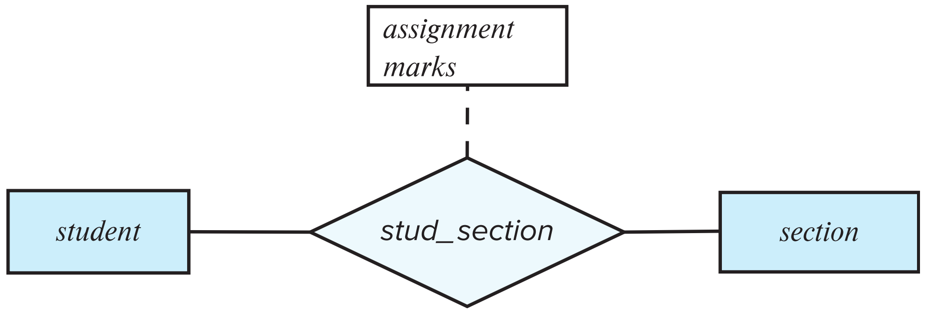
# Common Mistakes in E-R Diagrams

- Examples of erroneous E-R diagrams
  - (c) Erroneous relationship attributes
    - Use a relationship with a single-valued attribute in a situation that requires a multivalued attribute
    - Problem
      - can only represent a single assignment for a given student-section pair
      - cannot represent multiple assignments released in the same section for a given student-section pair




(b) Erroneous use of relationship attributes

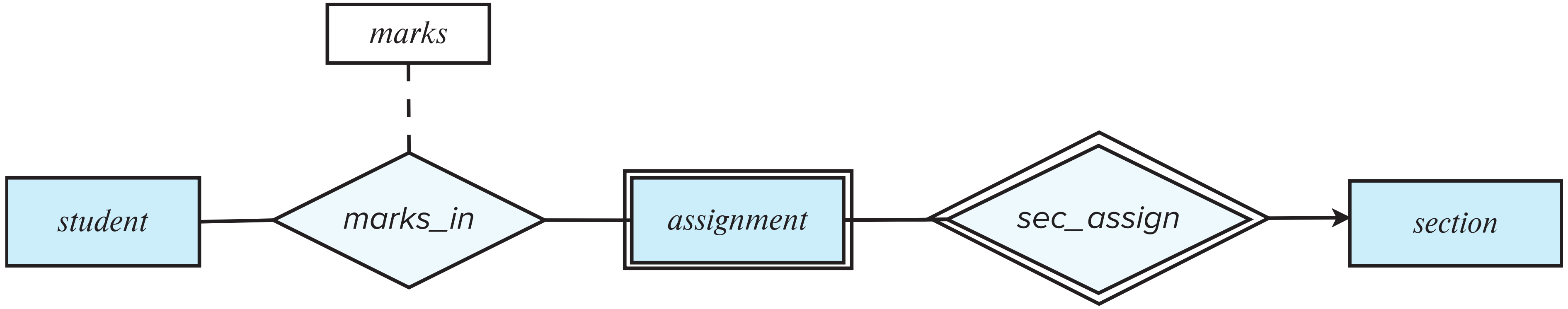
# Common Mistakes in E-R Diagrams




(b) Erroneous use of relationship attributes

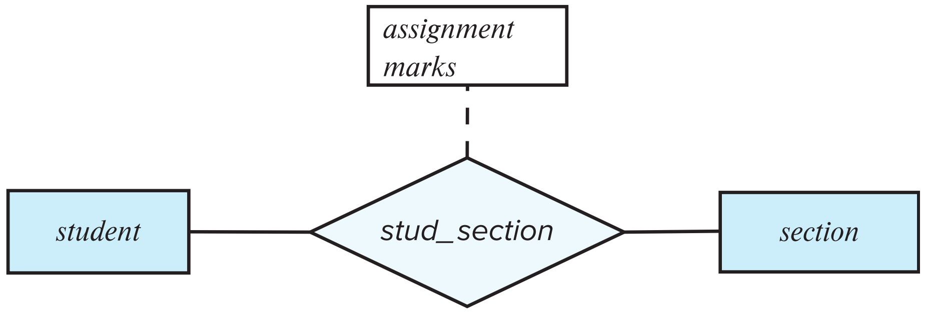
- Examples of erroneous E-R diagrams
  - (c) Erroneous relationship attributes
    - Use a relationship with a single-valued attribute in a situation that requires a multivalued attribute
    - Problem: cannot represent multiple assignments released in the same section for a given student-section pair

- Solutions:
  - **1) Weak entity set**
    - model *assignment* as a weak entity identified by *section*
    - add a relationship *marks_in* between *assignment* and *student*
    - *marks_in* has an attribute *marks*



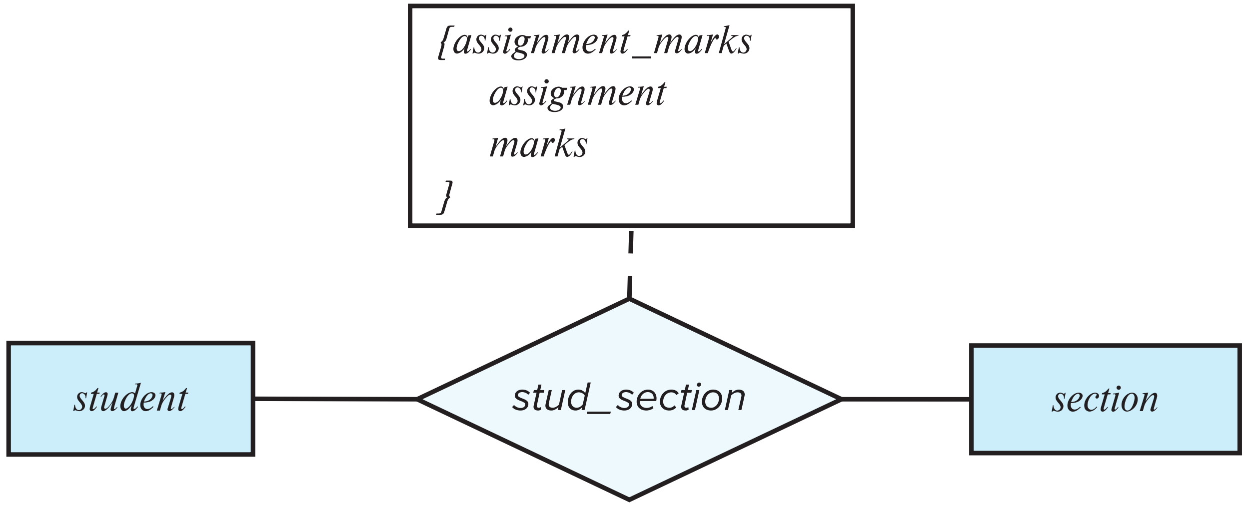

(c) Correct alternative to erroneous E-R diagram (b)

# Common Mistakes in E-R Diagrams




(b) Erroneous use of relationship attributes

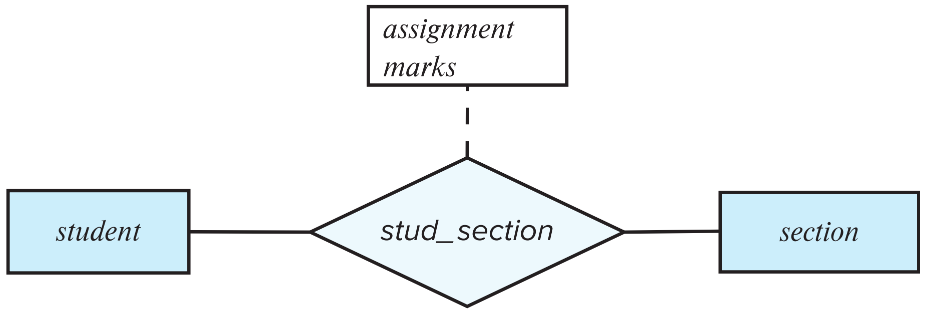
- Examples of erroneous E-R diagrams
  - (c) Erroneous relationship attributes
    - Problem: cannot represent multiple assignments released in the same section




(d) Correct alternative to erroneous E-R diagram (b)

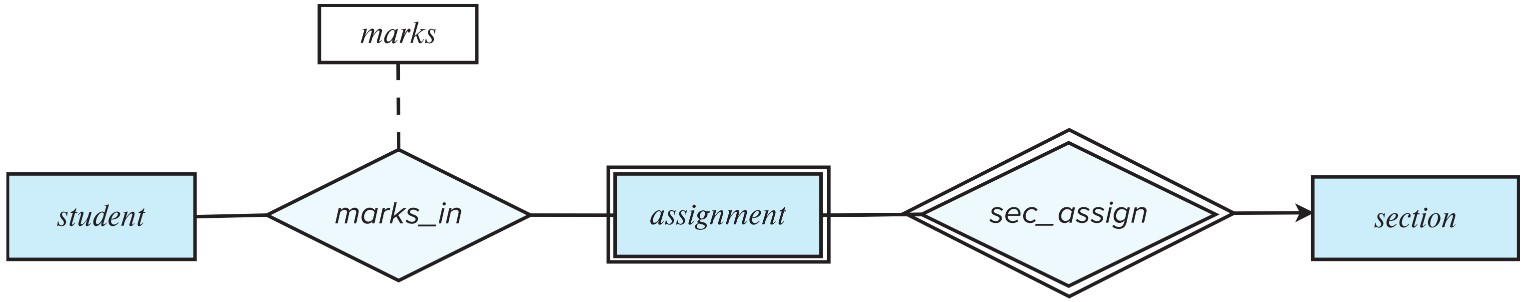
- Solutions:
  - 1) Weak entity set
  - **2) Composite attributes**
    - use a multivalued composite attribute *{assignment marks}*, where *assignment_marks* has component attributes *assignment* and *marks*

# Common Mistakes in E-R Diagrams

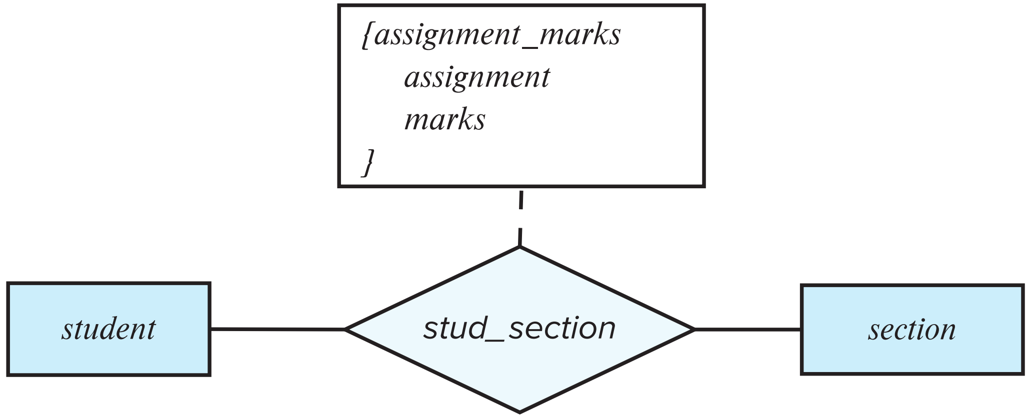



(b) Erroneous use of relationship attributes

- Examples of erroneous E-R diagrams
  - (b) Erroneous relationship attributes
    - Problem: cannot represent multiple assignments released in the same section
- Solutions:
  - 1) Weak entity set
  - 2) Composite attributes




(c) Correct alternative to erroneous E-R diagram (b)

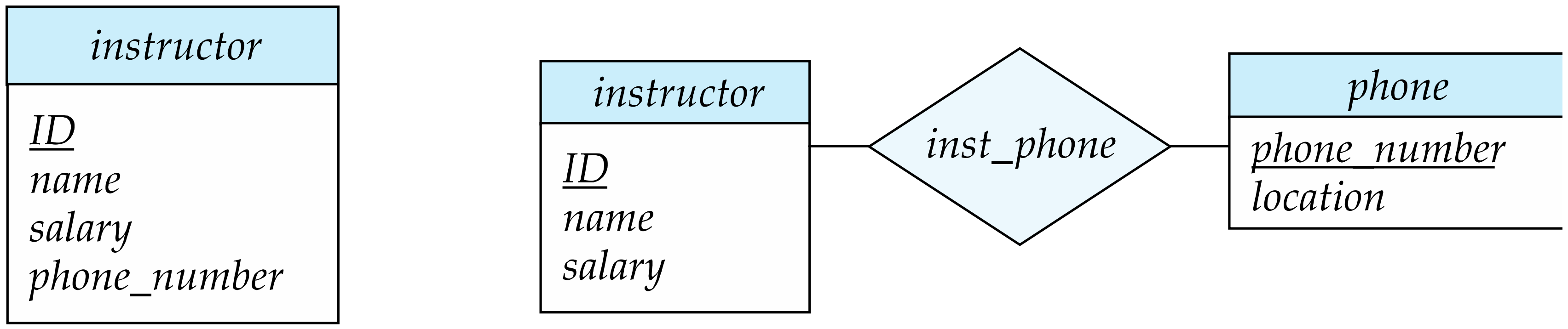



(d) Correct alternative to erroneous E-R diagram (b)

Using weak entity set is better, since it allows recording other information about the assignment, such as maximum marks or deadlines

# Use of Entity Sets vs. Attributes

- Use entity sets or attributes?



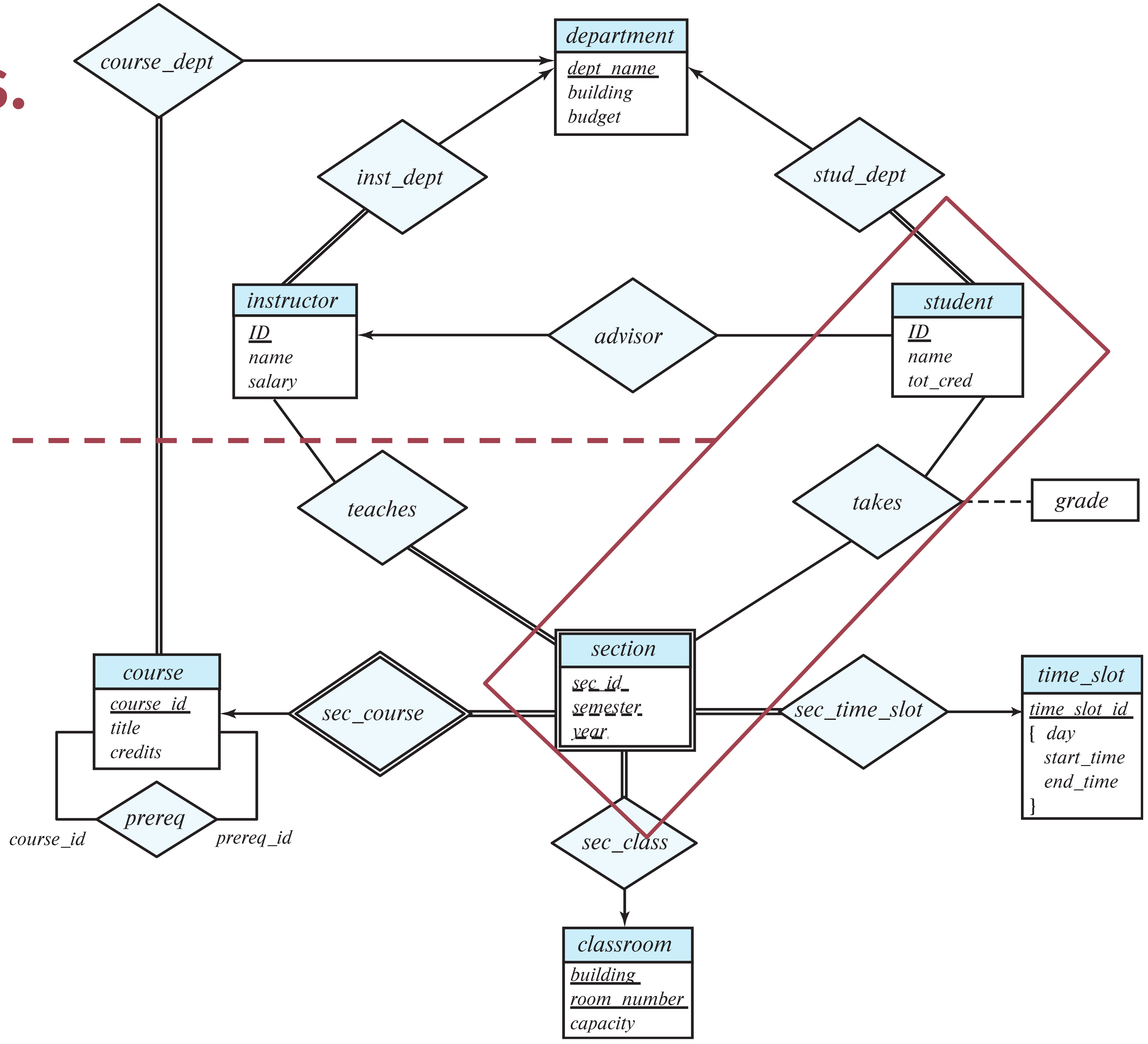

- Use of *phone* as an entity allows extra information about phone numbers
  - E.g. location (Home phone, mobile (cell) phone, office phone), phone type / brand
  - Allow each instructor to have multiple (including zero) phone numbers
- Treating *phone_number* as an attribute implies that instructors have precisely one phone number each
  - But this can be avoided by configuring *phone_number* as a multi-valued attribute
- Treating phone as an entity is more general than treating it as an attribute
  - We can keep extra information about a phone

# Use of Entity Sets vs. Relationship Sets

- Use entity sets or relationship sets? sometimes it is difficult to answer
  - We could use an entity set to model a relationship set
  - We could use a relationship set to model an entity set
  - **A possible guideline**: Use a relationship set to describe an action that occurs between entities

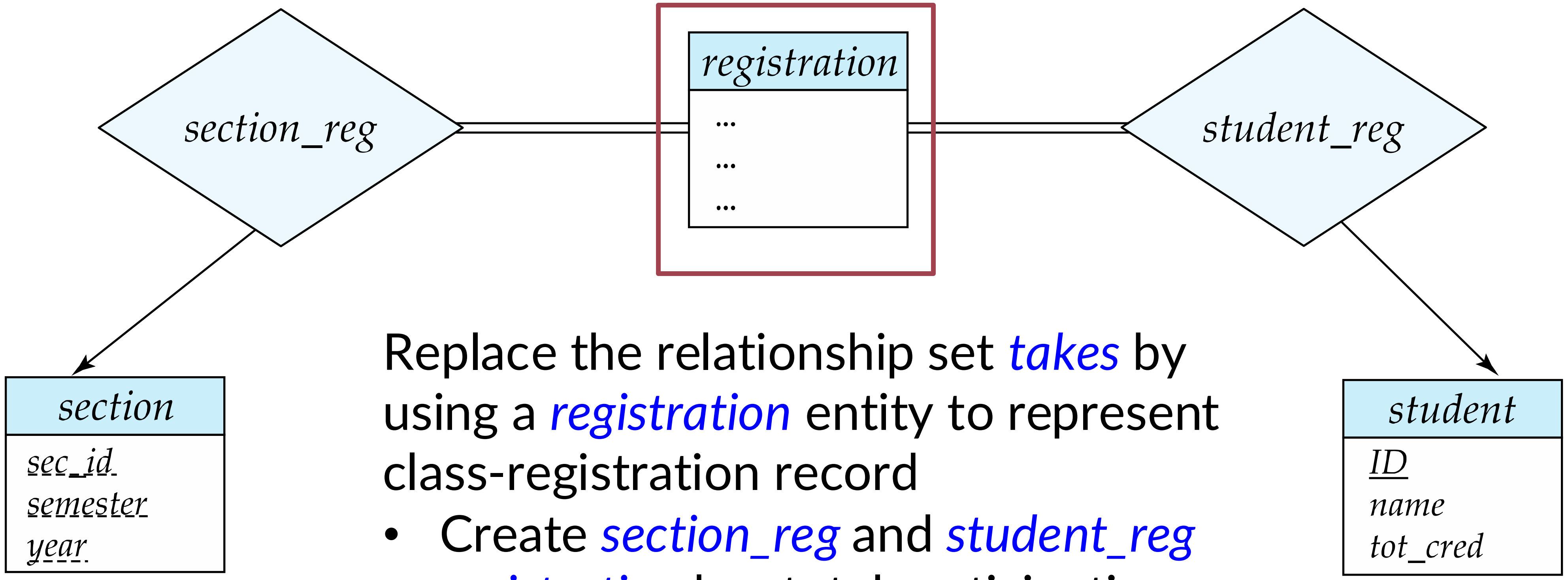
# Use of Entity Sets vs. Relationship Sets

- Example: *takes*



# Use of Entity Sets vs. Relationship Sets

- Use entity sets or relationship sets? sometimes it is difficult to answer
  - **A possible guideline**: Use a relationship set to describe an action that occurs between entities



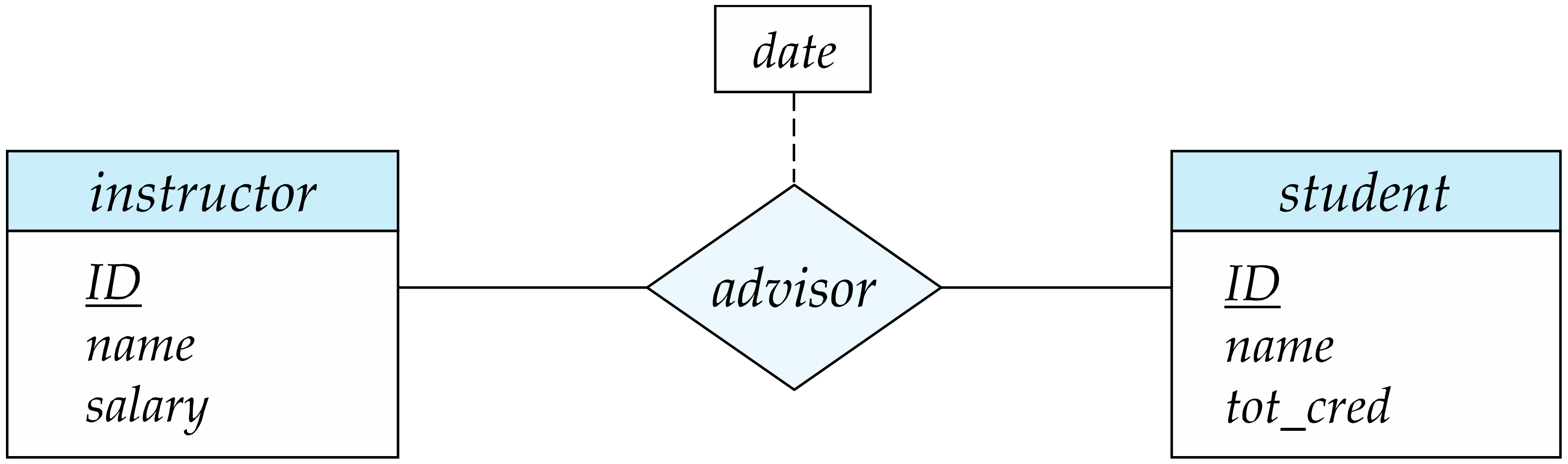

Replace the relationship set *takes* by using a *registration* entity to represent class-registration record

- Create *section_reg* and *student_reg*
- *registration* has total participation

  - Using *take* could be more compact and probably preferable
  - But by using a *registration* entity, we can associate other info with a course-registration record

# Use of Entity Sets vs. Relationship Sets

- Use entity sets or relationship sets? sometimes it is difficult to answer
  - **A possible guideline**: Use a relationship set to describe an action that occurs between entities
    - This guideline can be used for designing relationship attributes
      - For example, attribute *date* as attribute of advisor or as attribute of student

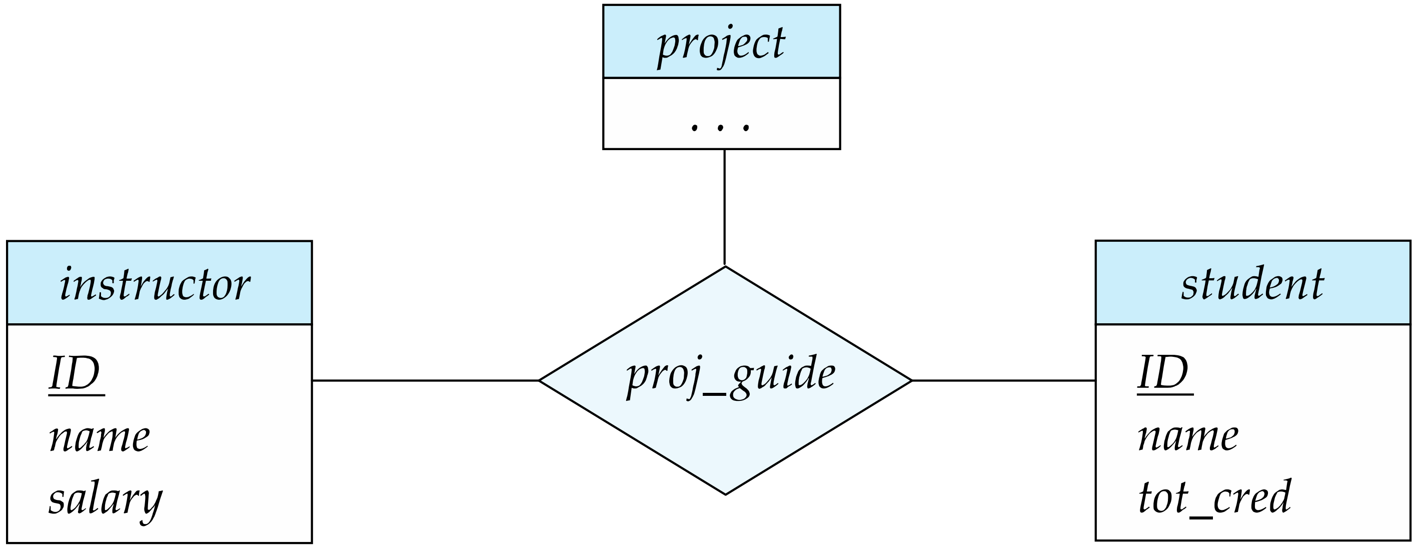


# Binary vs. Non-Binary Relationships

- Relationships in databases are often binary
- Some relationships that appear to be non-binary may be better represented using binary relationships
  - E.g., a ternary relationship *parents*, relating a child to his/her father and mother, is best replaced by two binary relationships, *father* and *mother*
    - Using two binary relationships allows partial information (e.g., only mother being known)
  - If *parent* is used, a null value would be required if only mother is known

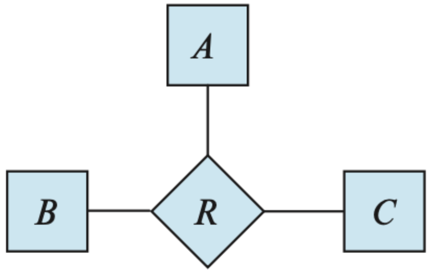
# Binary vs. Non-Binary Relationships

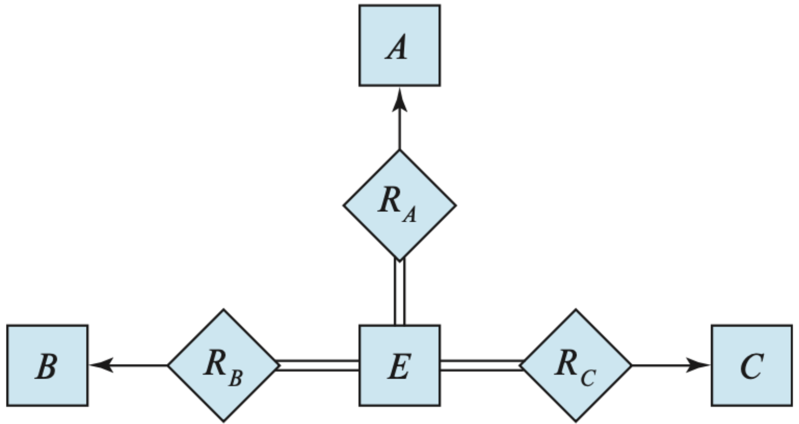
- Relationships in databases are often binary
- Some relationships that appear to be non-binary may be better represented using binary relationships
  - E.g., a ternary relationship *parents*, relating a child to his/her father and mother, is best replaced by two binary relationships, *father* and *mother*
    - Using two binary relationships allows partial information (e.g., only mother being known)
  - But there are some relationships that are naturally non-binary
    - Example: *proj_guide*



# Binary vs. Non-Binary Relationships

- It is possible to replace any non-binary (n-ary, for n > 2) relationship set by a number of distinct binary relationship sets
  - Replace relationship set R with entity set E, and relationship sets $R_A$, $R_B$ and $R_C$
  - Participation of E in $R_A$, $R_B$ and $R_C$ is total
  - Any descriptive attributes in R are assigned to E; a special identifying attribute is created for E
    - This could make the design complicated and cost more storage
- n-ary relationship set shows more clearly several entities participate in a single relationship
- Some constraints on ternary relationship cannot be translated into constraints on binary relationships
  - E.g., R, which is many-to-one from A, B to C, cannot be easily satisfied by $R_A$, $R_B$ and $R_C$
- Some relationships that are naturally non-binary
  - Better not using binary relationships to model



# 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, an n-ary relationship set shows more clearly that several entities participate in a single relationship.

# E-R Design Decisions

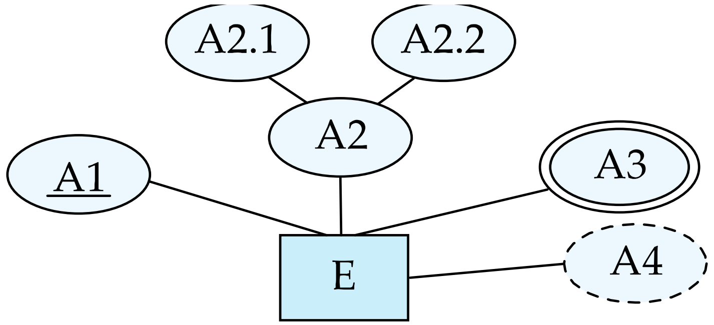
- The use of an attribute or entity set to represent an object
- Whether a real-world concept is best expressed by an entity set or a relationship set
- The use of a ternary relationship vs. a pair of binary relationships
- The use of a strong or weak entity set

- ** Extra:*
  - ** 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*

# Self Study: Alternative ER Notations

- No universal standard for E-R diagram notation, and different books and E-R diagram software use different notations.
- Chapter 7.10, Database System Concepts (7th Edition)

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

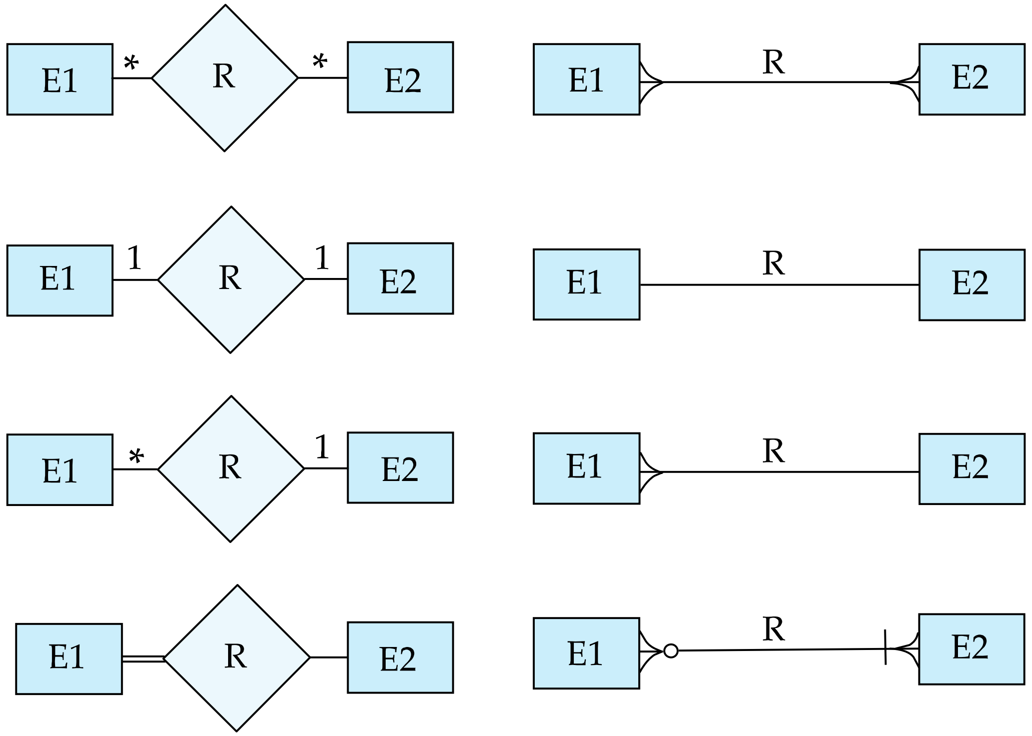



many-to-many relationship

one-to-one relationship

many-to-one relationship

participation in R: total (E1) and partial (E2)



# Summary

- Database design mainly involves the design of the database schema
  - To represent overall logical structure of the database
- E-R model provides a convenient graphical representation to view data, relationships, and constraints
- Entity, entity set, relationship, relationship sets
- Superkey, candidate key, and primary key apply to entity and relationship sets (like in relation schemas)
  - Primary key of a relationship set is composed of attributes from one or more of the related entity sets
- Mapping cardinality: number of entities another entity can be associated to via a relationship set
- Weak entity set: entity set that does not have sufficient attributes to form a primary key
- Strong entity set: entity set with a primary key
- Concepts and objects may, in certain cases, be represented by entities, relationships, or attributes
- A database design based on an E-R diagram can be represented by a collection of relation schemas
- Common mistakes to avoid in E-R design

# Normalization: A First Look

# 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
    - E.g., store course identifier and title of a course for each course offering
      - Only store course identifier is sufficient
    - Redundant representation of information may lead to data inconsistency among the various copies of information
      - E.g., update is not performed on all the copies
  - **Incompleteness**: a bad design may make certain aspects of the enterprise difficult or impossible to model
    - E.g., only have entity for course offering, but without entity for courses
      - Impossible to model new courses that are not offered yet

# Design Alternatives

- Avoiding bad designs is not enough
  - There may be many good designs from which we must choose
- For example, a customer who buys a product
  - The sale activity is a relationship between the customer and the product?
  - The sale activity is a relationship among the customer, the product, and the sale itself?
    - i.e., the sale can be considered as an entity
- Database design can be difficult
  - When #entities and #relationships are large
- Do we have any guidelines on how to get a good design?
  - Normal Forms (**范式**)!

# Normalization （规范化）

- In practice, we usually just satisfy 1NF, 2NF and 3NF

| | UNF (1970) | 1NF (1970) | 2NF (1971) | 3NF (1971) | EKNF (1982) | BCNF (1974) | 4NF (1977) | ETNF (2012) | 5NF (1979) | DKNF (1981) | 6NF (2003) |
|---|---|---|---|---|---|---|---|---|---|---|---|
| Primary key (no duplicate tuples)[4] | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ |
| Atomic columns (cells cannot have tables as values)[5] | ✗ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ |
| Every non-trivial functional dependency either does not begin with a proper subset of a candidate key or ends with a prime attribute (no partial functional dependencies of non-prime attributes on candidate keys)[5] | ✗ | ✗ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ |
| Every non-trivial functional dependency either begins with a superkey or ends with a prime attribute (no transitive functional dependencies of non-prime attributes on candidate keys)[5] | ✗ | ✗ | ✗ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ |
| Every non-trivial functional dependency either begins with a superkey or ends with an elementary prime attribute | ✗ | ✗ | ✗ | ✗ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | N/A |
| Every non-trivial functional dependency begins with a superkey | ✗ | ✗ | ✗ | ✗ | ✗ | ✓ | ✓ | ✓ | ✓ | ✓ | N/A |
| Every non-trivial multivalued dependency begins with a superkey | ✗ | ✗ | ✗ | ✗ | ✗ | ✗ | ✓ | ✓ | ✓ | ✓ | N/A |
| Every join dependency has a superkey component[8] | ✗ | ✗ | ✗ | ✗ | ✗ | ✗ | ✗ | ✓ | ✓ | ✓ | N/A |
| Every join dependency has only superkey components | ✗ | ✗ | ✗ | ✗ | ✗ | ✗ | ✗ | ✗ | ✓ | ✓ | N/A |
| Every constraint is a consequence of domain constraints and key constraints | ✗ | ✗ | ✗ | ✗ | ✗ | ✗ | ✗ | ✗ | ✗ | ✓ | ✗ |
| Every join dependency is trivial | ✗ | ✗ | ✗ | ✗ | ✗ | ✗ | ✗ | ✗ | ✗ | ✗ | ✓ |

https://en.wikipedia.org/wiki/Database_normalization

# First Normal Form (1NF，第一范式)

- A relational schema *R* is in 1NF if the domains of all attributes of R are atomic
  - Domain is atomic if its elements are considered to be indivisible units
  - Examples of non-atomic domains:
    - Set of names, composite attributes
    - Identification numbers like CS307 that can be broken up into parts
      - However, in practice, we can also consider it atomic
  - Non-atomic values complicate storage and encourage redundant (repeated) storage of data

# First Normal Form (1NF)

- Example: Non-atomic attribute

| station_id | name | location |
|---|---|---|
| 1 | Luohu(罗湖) | 114.11833 , 22.53111 |
| 2 | Guomao(国贸) | 114.11889 , 22.54 |
| 3 | Laojie(老街) | 114.11639 , 22.54444 |
| 4 | Grand Theater(大剧院) | 114.10333 , 22.54472 |
| 5 | Science Museum(科学馆) | 114.08972 , 22.54333 |
| 6 | Huaqiang Rd(华强路) | 114.07889 , 22.54306 |
| 7 | Gangxia(岗厦) | 114.06306 , 22.53778 |
| 8 | Convention and Exhibition Center Station(会展中心) | 114.05472 , 22.5375 |
| 9 | Shopping Park(购物公园) | 114.05472 , 22.53444 |
| 10 | Xiangmihu(香蜜湖) | 114.034 , 22.5417 |

# First Normal Form (1NF)

- Fix it by splitting the names into two columns

| station_id | english_name | chinese_name | longitude | latitude |
|---|---|---|---|---|
| 1 | Luohu | 罗湖 | 114.11833 | 22.53111 |
| 2 | Guomao | 国贸 | 114.11889 | 22.54 |
| 3 | Laojie | 老街 | 114.11639 | 22.54444 |
| 4 | Grand Theater | 大剧院 | 114.10333 | 22.54472 |
| 5 | Science Museum | 科学馆 | 114.08972 | 22.54333 |
| 6 | Huaqiang Rd | 华强路 | 114.07889 | 22.54306 |
| 7 | Gangxia | 岗厦 | 114.06306 | 22.53778 |
| 8 | Convention and Exhibition Cent… | 会展中心 | 114.05472 | 22.5375 |
| 9 | Shopping Park | 购物公园 | 114.05472 | 22.53444 |
| 10 | Xiangmihu | 香蜜湖 | 114.034 | 22.5417 |

# First Normal Form (1NF)

- Another example: Starring
  - Problems: 1) Redundant names; 2) difficulties in updating/deleting a specific person; 3) extra cost in splitting names; 4) difficulties in making statistics

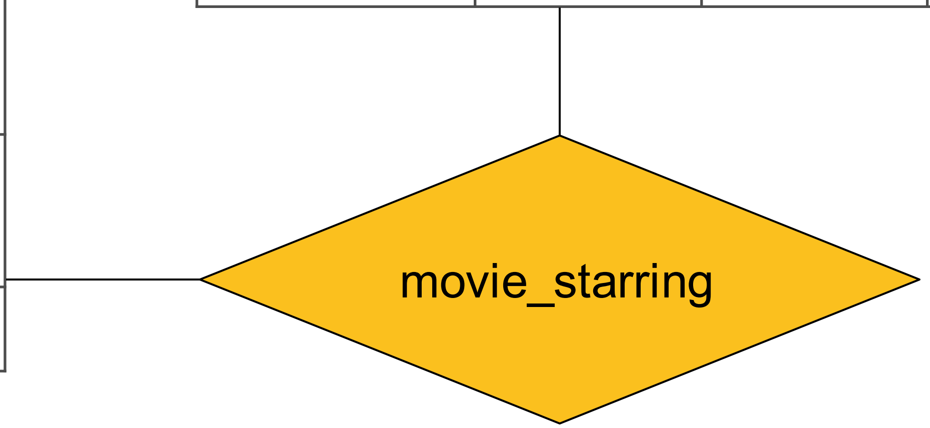
| Movie ID | Movie Title | Country | Year | Director | Starring |
|---|---|---|---|---|---|
| 0 | Citizen Kane | US | 1941 | welles, o. | Orson Welles, Joseph Cotten |
| 1 | La règle du jeu | FR | 1939 | Renoir, J. | Roland Toutain, Nora Grégor, Marcel Dalio, Jean Renoir |
| 2 | North By Northwest | US | 1959 | HITCHCOCK, A. | Cary Grant, Eva Marie Saint, James Mason |
| 3 | Singin' in the Rain | US | 1952 | Donen/Kelly | Gene Kelly, Debbie Reynolds, Donald O'Connor |
| 4 | Rear Window | US | 1954 | Alfred Hitchcock | James Stewart, Grace Kelly |

# First Normal Form (1NF)

- Fix it by treating the column as a multi-valued attribute
  - *movie_starring* table has two foreign keys, *movid_id* and *star_id*

| Movie ID | Movie Title | Country | Year | Director |
|---|---|---|---|---|
| 0 | Citizen Kane | US | 1941 | welles, o. |
| 1 | La règle du jeu | FR | 1939 | Renoir, J. |
| 2 | North By Northwest | US | 1959 | HITCHCOCK, A. |
| 3 | Singin' in the Rain | US | 1952 | Donen/Kelly |
| 4 | Rear Window | US | 1954 | Alfred Hitchcock |

| Star ID | Firstname | Lastname | Born | Died |
|---|---|---|---|---|
| 1 | | | | |
| 2 | | | | |
| 3 | | | | |



# Second Normal Form (2NF，第二范式)

- A relation satisfying 2NF must:
  - be in 1NF
  - not have any non-prime attribute that is dependent on any proper subset of any candidate key of the relation
    - A non-prime attribute of a relation is an attribute that is not a part of any candidate key of the relation
  - 不 包含只依赖于 主键中部分属性 的非主属性
    - “非主属性”是指 不属于 任何候选键的属性

# Second Normal Form (2NF)

- Example: consider this table with the composite primary key (*station_id*, *line_id*)

| station_id | english_name | chinese_name | district | line_id | line_color | operator |
|---|---|---|---|---|---|---|
| 1 | Luohu | 罗湖 | Luohu | 1 | Green | Shenzhen Metro Corporation |
| 2 | Guomao | 国贸 | Luohu | 1 | Green | Shenzhen Metro Corporation |
| 3 | Laojie | 老街 | Luohu | 1 | Green | Shenzhen Metro Corporation |
| 4 | Grand Theater | 大剧院 | Luohu | 1 | Green | Shenzhen Metro Corporation |
| 4 | Grand Theater | 大剧院 | Luohu | 11 | Purple | Shenzhen Metro Corporation |
| 4 | Grand Theater | 大剧院 | Luohu | 2 | Orange | Shenzhen Metro Corporation |
| 3 | Laojie | 老街 | Luohu | 3 | DeepSkyBlue | Shenzhen Metro No.3 Line |

  - The columns *line_color* and *operator* are not related to *station_id*
    - They are only related to *line_id*, which is only part of (a subset of) the primary key
  - Similarly, *english_name*, *chinese_name*, and *district* are not related to *line_id*
    - They are only related to *station_id*, which is only part of (a subset of) the primary key
  - 非主属性 *line_color*, *operator*, *english_name*, *chinese_name*, *district* 只依赖于主键中的部份属性

# Second Normal Form (2NF)

- Example: Consider this table with the composite primary key *(station_id, line_id)*

| station_id | english_name | chinese_name | district | line_id | line_color | operator |
|---|---|---|---|---|---|---|
| 1 | Luohu | 罗湖 | Luohu | 1 | Green | Shenzhen Metro Corporation |
| 2 | Guomao | 国贸 | Luohu | 1 | Green | Shenzhen Metro Corporation |
| 3 | Laojie | 老街 | Luohu | 1 | Green | Shenzhen Metro Corporation |
| 4 | Grand Theater | 大剧院 | Luohu | 1 | Green | Shenzhen Metro Corporation |
| 4 | Grand Theater | 大剧院 | Luohu | 11 | Purple | Shenzhen Metro Corporation |
| 4 | Grand Theater | 大剧院 | Luohu | 2 | Orange | Shenzhen Metro Corporation |
| 3 | Laojie | 老街 | Luohu | 3 | DeepSkyBlue | Shenzhen Metro No.3 Line |

- Problem when not meeting 2NF: Insertion and deletion anomaly
    - We CANNOT insert a new station with no lines assigned yet (unless using NULLs)
    - If we delete a line, all stations associated with this line will be deleted as well

# Second Normal Form (2NF)

- Fix it by
  - Splitting the two unrelated parts into two different tables of entities
  - And create a relationship set (if it is the many-to-many relationship between the two entities)
- By the way…
  - A relation with a single-attribute primary key is <u>automatically in 2NF</u> once it meets 1NF

stations

| station_id | english_name | chinese_name | district |
|---|---|---|---|
| 1 | Luohu | 罗湖 | Luohu |
| 2 | Guomao | 国贸 | Luohu |
| 3 | Laojie | 老街 | Luohu |
| 4 | Grand Theater | 大剧院 | Luohu |

Foreign key

primary key(line_id, station_id)

line_detail

| line_id | station_id | num | dist |
|---|---|---|---|
| 1 | 1 | 1 | 0 |
| 1 | 2 | 2 | 1 |
| 1 | 3 | 3 | 1 |
| 1 | 4 | 4 | 1 |
| 11 | 4 | 21 | <null> |
| 2 | 4 | 26 | 2 |
| 3 | 3 | 10 | 2 |

Foreign key

lines

| line_id | line_color | operator |
|---|---|---|
| 1 | Green | Shenzhen Metro Corporation |
| 2 | Orange | Shenzhen Metro Corporation |
| 3 | DeepSkyBlue | Shenzhen Metro No.3 Line |
| 11 | Purple | Shenzhen Metro Corporation |

# Third Normal Form (3NF，第三范式)

- A relation satisfying 3NF must:
  - be in 2NF
  - all the attributes in a table are determined only by the candidate keys of that relation, not by any non-prime attributes
  - 所有属性 只依赖于候选键，不依赖于任意非主属性

# Third Normal Form (3NF)

- Example: Consider this table which describes the bus lines and their stops
  - Primary key (*bus_line*)

| bus_line | station_id | chinese_name | english_name | district |
|---|---|---|---|---|
| B796 | 21 | 鲤鱼门 | Liyumen | Nanshan |
| M343 | 21 | 鲤鱼门 | Liyumen | Nanshan |
| M349 | 21 | 鲤鱼门 | Liyumen | Nanshan |
| M250 | 26 | 坪洲 | Pingzhou | Bao'an |
| 374 | 61 | 安托山 | Antuo Hill | Futian |
| B733 | 61 | 安托山 | Antuo Hill | Futian |
| B828 | 120 | 临海 | Linhai | Nanshan |

- *station_id* depends on the primary key (*bus_line*)
- However, the columns *chinese_name*, *english_name*, and *district* depend on *station_id*, which is not the primary key.
  - They only have "indirect/transitive" dependence （非直接/传递依赖 ） on the primary key
- Problem: Data redundancy

# Third Normal Form (3NF)

- Example: Consider this table which describes the bus lines and their stops
  - Primary key (*bus_line*)

| bus_line | station_id | chinese_name | english_name | district |
|---|---|---|---|---|
| B796 | 21 | 鲤鱼门 | Liyumen | Nanshan |
| M343 | 21 | 鲤鱼门 | Liyumen | Nanshan |
| M349 | 21 | 鲤鱼门 | Liyumen | Nanshan |
| M250 | 26 | 坪洲 | Pingzhou | Bao'an |
| 374 | 61 | 安托山 | Antuo Hill | Futian |
| B733 | 61 | 安托山 | Antuo Hill | Futian |
| B828 | 120 | 临海 | Linhai | Nanshan |

- Problem when not meeting 3NF:
  - Data redundancy
    - » the attributes for a station have been stored multiple times
  - Insertion and deletion anomaly
    - » inserting a new bus line with no station becomes impossible without NULLs
    - » deleting a station/bus line may also delete corresponding bus lines/stations

# Third Normal Form (3NF)

- Fix it by:
  - <u>Create a new table</u> with *station_id* as the primary key
    - i.e., the column which *chinese_name*, *english_name*, and *district* depend on
  - <u>Move all columns</u> which depend on the new primary key into the new table
    - … and, only leave the primary key of the new table (*station_id*) in the original table
  - (*In practice, if necessary) Add a foreign-key constraint
    - Not related to relational database modeling, only in implementations

stations

| station_id | chinese_name | english_name | district |
|---|---|---|---|
| 21 | 鲤鱼门 | Liyumen | Nanshan |
| 26 | 坪洲 | Pingzhou | Bao'an |
| 61 | 安托山 | Antuo Hill | Futian |
| 120 | 临海 | Linhai | Nanshan |
| 121 | 宝华 | Baohua | Bao'an |

Foreign key

bus_lines

| station_id | bus_line |
|---|---|
| 21 | B796 |
| 21 | M343 |
| 21 | M349 |
| 26 | M250 |
| 61 | 374 |
| 61 | B733 |
| 120 | B828 |
| 121 | B828 |
| 121 | M235 |

# Normalization

- In practice, we usually just satisfy 1NF, 2NF and 3NF

| | UNF (1970) | 1NF (1970) | 2NF (1971) | 3NF (1971) | EKNF (1982) | BCNF (1974) | 4NF (1977) | ETNF (2012) | 5NF (1979) | DKNF (1981) | 6NF (2003) |
|---|---|---|---|---|---|---|---|---|---|---|---|
| Primary key (no duplicate tuples)[4] | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ |
| Atomic columns (cells cannot have tables as values)[5] | ✗ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ |
| Every non-trivial functional dependency either does not begin with a proper subset of a candidate key or ends with a prime attribute (no partial functional dependencies of non-prime attributes on candidate keys)[5] | ✗ | ✗ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ |
| Every non-trivial functional dependency either begins with a superkey or ends with a prime attribute (no transitive functional dependencies of non-prime attributes on candidate keys)[5] | ✗ | ✗ | ✗ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ |
| Every non-trivial functional dependency either begins with a superkey or ends with an elementary prime attribute | ✗ | ✗ | ✗ | ✗ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | N/A |
| Every non-trivial functional dependency begins with a superkey | ✗ | ✗ | ✗ | ✗ | ✗ | ✓ | ✓ | ✓ | ✓ | ✓ | N/A |
| Every non-trivial multivalued dependency begins with a superkey | ✗ | ✗ | ✗ | ✗ | ✗ | ✗ | ✓ | ✓ | ✓ | ✓ | N/A |
| Every join dependency has a superkey component[8] | ✗ | ✗ | ✗ | ✗ | ✗ | ✗ | ✗ | ✓ | ✓ | ✓ | N/A |
| Every join dependency has only superkey components | ✗ | ✗ | ✗ | ✗ | ✗ | ✗ | ✗ | ✗ | ✓ | ✓ | N/A |
| Every constraint is a consequence of domain constraints and key constraints | ✗ | ✗ | ✗ | ✗ | ✗ | ✗ | ✗ | ✗ | ✗ | ✓ | ✗ |
| Every join dependency is trivial | ✗ | ✗ | ✗ | ✗ | ✗ | ✗ | ✗ | ✗ | ✗ | ✗ | ✓ |

https://en.wikipedia.org/wiki/Database_normalization

# Normalization

**Every non key attribute must provide a fact about the key, the whole key, and nothing but the key.**

2NF

3NF

**William Kent (1936 – 2005)**

William Kent. "A Simple Guide to Five Normal Forms in Relational Database Theory", Communications of the ACM 26 (2), Feb. 1983, pp. 120–125.