Chapter 7: Entity-Relationship Model

Database System Concepts, 6th Ed.

Chapter 7: Entity-Relationship Model

- Design Process
- Modeling
- Constraints
- E-R Diagram
- Design Issues
- Weak Entity Sets
- Extended E-R Features
- Design of the Bank Database
- Reduction to Relation Schemas
- Database Design
- UML

Design Phases

- The initial phase of database design is to characterize fully the data needs of the prospective database users
- Next, the designer chooses a data model and, by applying the concepts of the chosen data model, translates these requirements into a conceptual schema of the database
- A fully developed conceptual schema also indicates the functional requirements of the enterprise
- In a “specification of functional requirements”, users describe the kinds of operations (or transactions) that will be performed on the data

Design Phases (Cont.)

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

- Logical Design – Deciding on the database schema Database design requires that we find a "good" collection of relation schemas
  - Business decision – What attributes should we record in the database?
  - Computer Science decision – What relation schemas should we have and how should the attributes be distributed among the various relation schemas?
- Physical Design – Deciding on the physical layout of the database

Design Approaches

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

Chapter 7: Entity-Relationship Model

- Database Design for a University
- Introduction to Relational Model
- Entity-Relationship Model
  - Conceptual representation of the data
  - “Reality” meets “bits and bytes”
  - Must make sense, and be usable by other people

Entity-Relationship Model

Entity-relationship Model

Typically used for conceptual database design

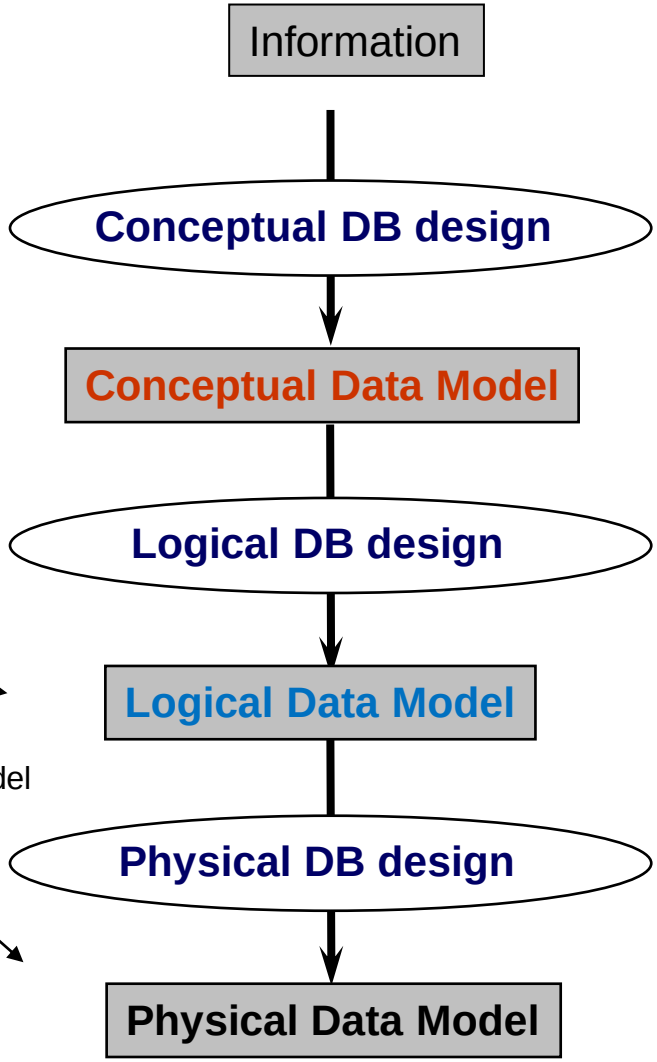
Three Levels of Modeling

Relational Model

Typically used for logical database design

Mapping the conceptual schema defined using the entity-relationship model into a relation schema

System specific database Schema includes file organization and choice of index structures



Database Design for a University

- A real university database would be much more complex than this design
- However, use this simplified model to help you understand conceptual ideas without getting lost in details of a complex design
- The initial specification of user requirements may be based on interviews with the database users, and on the designer's own analysis of the organization
- The description that arises from this design phase serves as the basis for specifying the conceptual structure of the database

Database Design for a University

- The major characteristics of the university
  - The **University** is organized into **departments**
  - **Each department** is identified by a unique name (*dept name), is located in a particular building, and has a budget*
  - **Each department** has **a list of courses** it offers
  - **Each course** has associated with it a *course id, title, dept name, and credits, and may also have associated prerequisites*
  - **Instructors** are identified by their unique *ID*
  - ***Each instructor** has name, associated* department (*dept name), and salary*
  - **Students** are identified by their unique *ID*
  - ***Each student** has a name, an associated* major department (*dept name), and tot cred (total credit hours the student* earned thus far)

Database Design for a University

- The university maintains **a list of classrooms**, specifying the name of the *building, room number, and room capacity*
- The university maintains **a list of all classes (sections)** taught
- **Each section** is identified by a *course id, sec id, year, and semester, and has associated with it* a *semester, year, building, room number, and time slot id (the time slot when the* class meets)
- The **department** has **a list of teaching assignments** specifying, for each instructor, the sections the instructor is teaching
- The **university has a list of all student course registrations, specifying, for each student, the courses and the associated sections that the student has taken** (registered for)

Relational Model

Example of a Relation

Attributes (or columns)

ID	*name*	*dept_name*	*salary*
10101	Srinivasan	Comp. Sci.	65000
12121	Wu	Finance	90000
15151	Mozart	Music	40000
22222	Einstein	Physics	95000
32343	El Said	History	60000
33456	Gold	Physics	87000
45565	Katz	Comp. Sci.	75000
58583	Califieri	History	62000
76543	Singh	Finance	80000
76766	Crick	Biology	72000
83821	Brandt	Comp. Sci.	92000
98345	Kim	Elec. Eng.	80000

Tuples (or rows)

Shows a *relationship among a set of values*

Table/Relation

A collection of such relationships

Attribute Values

Relational Model

- Database
  - A database consists of multiple relations
  - Information about an enterprise is broken up into parts

    instructor
    student
    advisor

  - Design:

    univ (*instructor -ID, name, dept_name, salary, student_Id*, ..)
- Relation Schema
  - Introduced by Dr. E.F. (“Ted”) Codd in 1970… based on set theory
  - Collection of relation schemas prepares a whole database
  - Describes
    - A blueprint of a database
      - Outlines the way data is organized into tables
    - Structure and constraints of data representing in a particular domain
  - Each tuple is divided into fields called **Domain**

Relation Schema and Instance

- $A_1, A_2, \ldots, A_n$ are *attributes*
- $R = (A_1, A_2, \ldots, A_n)$ is a *relation schema*

  Example:

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

- Formally, given sets $D_1, D_2, \ldots. D_n$ a **relation *r*** is a subset of

  $D_1 \times D_2 \times \ldots \times D_n$

  Thus, a relation is a set of *n*-tuples $(a_1, a_2, \ldots, a_n)$ where each $a_i \in D_i$

- **Relation Instance**
  - The current values of a relation are specified by a table

Relational Model-Attribute Types

- **Domain of the attribute**
  - The set of allowed values for each attribute
  - Attribute values are (normally) required to be **atomic**; that is, indivisible
  - e.g. 1) For the instructor phone field is used, If the instructor
    - Has one phone, single value in a field
      - Atomic
    - Have series of phones, multiple values in a field separated by delimiter (comma/space)
      - Nonatomic (as sub part is useful)
  - e.g. 2) Phone is having country + state code + number
    - Atomic / NonAtomic
    - Application dependent
      - If all together as a one number, then Atomic
      - If subparts to treat separately, then NonAtomic

Relational Model-Attribute Types

- **Not** allowed
  - Composite (address-street, city, state…)
  - Multi-valued (nonatomic) attributes are
- **Domain of the attribute**
  - The special value ***null*** is a member of every domain
    - Can be utilized if the value is unknown or does not exist
      - e.g. If the instructor does not have a phone

Relational Model

- Keys
  - Super key
  - Candidate key
  - Primary key
- Constraints
  - Foreign key
- Schema diagram
- Query language

Entity-Relationship (ER) Model

ER Model

- How to design a database schema
- ER-data model
  - Identify entities
  - How those entities are related
  - Associated set of constraints
  - How to transform an E-R design into a set of basic relation schemas and how to capture constraints
- Database design is good or bad and the process of creating good designs using a broader set of constraints
  - Not part of this ppt

ER-Model

- Developed to facilitate database design by allowing specification of an enterprise schema that represents the overall logical structure of a database
- Very useful in mapping the meanings and interactions of real-world enterprises onto a conceptual schema
  - Because of this usefulness, many database-design tools draw on concepts from the ER model
- The ER data model employs three basic concepts:
  - Entity sets,
  - Relationship sets,
  - Attributes

ER-Model

- Diagrammatic representation of a high-level data model
  - Instead of representing all data in tables, it distinguishes between basic objects, called *entities, and relationships among these objects*
- *Often used as a* first step in database-schema design
- A ***database*** can be modeled as:
  - **A collection of entities** and
  - **Relationship among entities**

ER-Model Entity

- An **entity is a "thing" or "object" in the real world that is distinguishable from** all other objects
  - Example: specific person, company, event, plant
- Entity has a set of properties, and the values for some set of properties may **uniquely identify an entity**
- For example, person in a university is an entity
  - Have a *person id property whose* value uniquely identifies that person
  - Value 677-89-9011 for *person id* would uniquely identify one particular person in the university
- Another example, courses can be thought of as entities, and *course id uniquely identifies a course entity in* the university

ER-Model Entity

- **Types of Entity**
  - **Concrete Entity** (Such as a person or a book)
  - **Abstract Entity** (Course, Course offering or Flight reservation)
- In University example, entities are:
  - Instructors, Students: Concrete Entities
  - Departments, Courses and Course offerings: Abstract Entities

ER-Model Entity Set

- **A set of entities of the same type that share the same properties,** or attributes
  - Entity set *instructor:* The set of all people who are instructors at a given university
  - *Entity set student:* The set of all students in the university
  - Entity set song: The collection of all songs in a database
  - Other Example: set of all persons, companies, trees, holidays

ER-Model Attributes

- An entity is represented by a set of **attributes**
- **Attributes are descriptive** properties possessed by each member of an entity set
- Each entity may **have its own value for each attribute**
  - Possible attributes of the *instructor entity set are*
    - *ID, name, dept name, and salary*
  - *In real life, there would be further attributes, such as*
    - *street* number, apartment number, state, postal code, country
    - but we omit them to keep our examples simple
  - Possible attributes of the *course entity set are*
    - *course id, title, dept name, and credits*
- A subset of the attributes form a **primary key** of the entity set; i.e., uniquely identifying each member of the set

Entity Sets *instructor* and *student*

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

instructor

student-ID	student_name
98988	Tanaka
12345	Shankar
00128	Zhang
76543	Brown
76653	Aoi
23121	Chavez
44553	Peltier

student

Relationship Sets

- To manipulate the data better way
- To enforce constraint(s)

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

  Example:

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

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

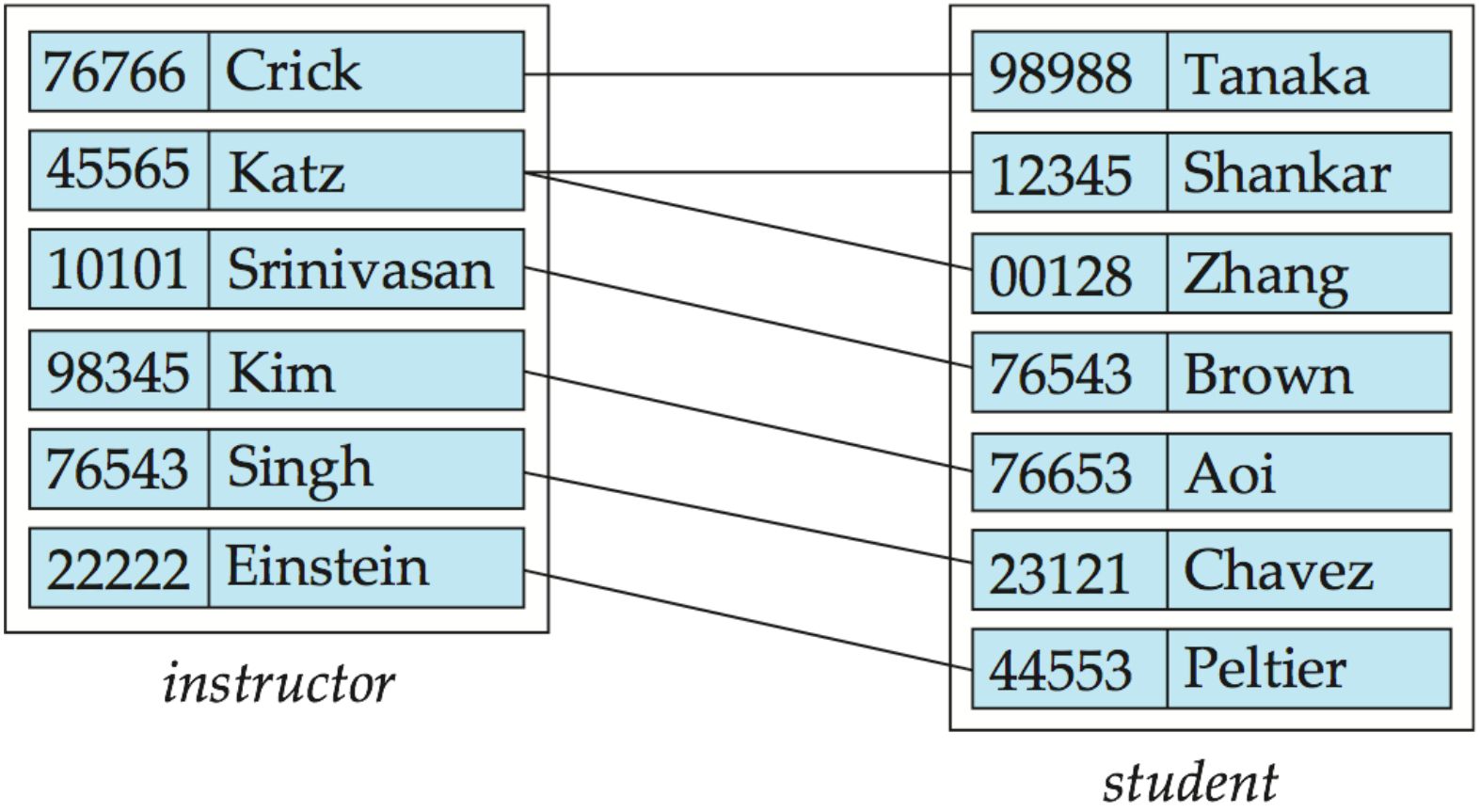
$$\{(e_1, e_2, \ldots e_n) \mid e_1 \in E_1, e_2 \in E_2, \ldots, e_n \in E_n\}$$

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

  - Example:

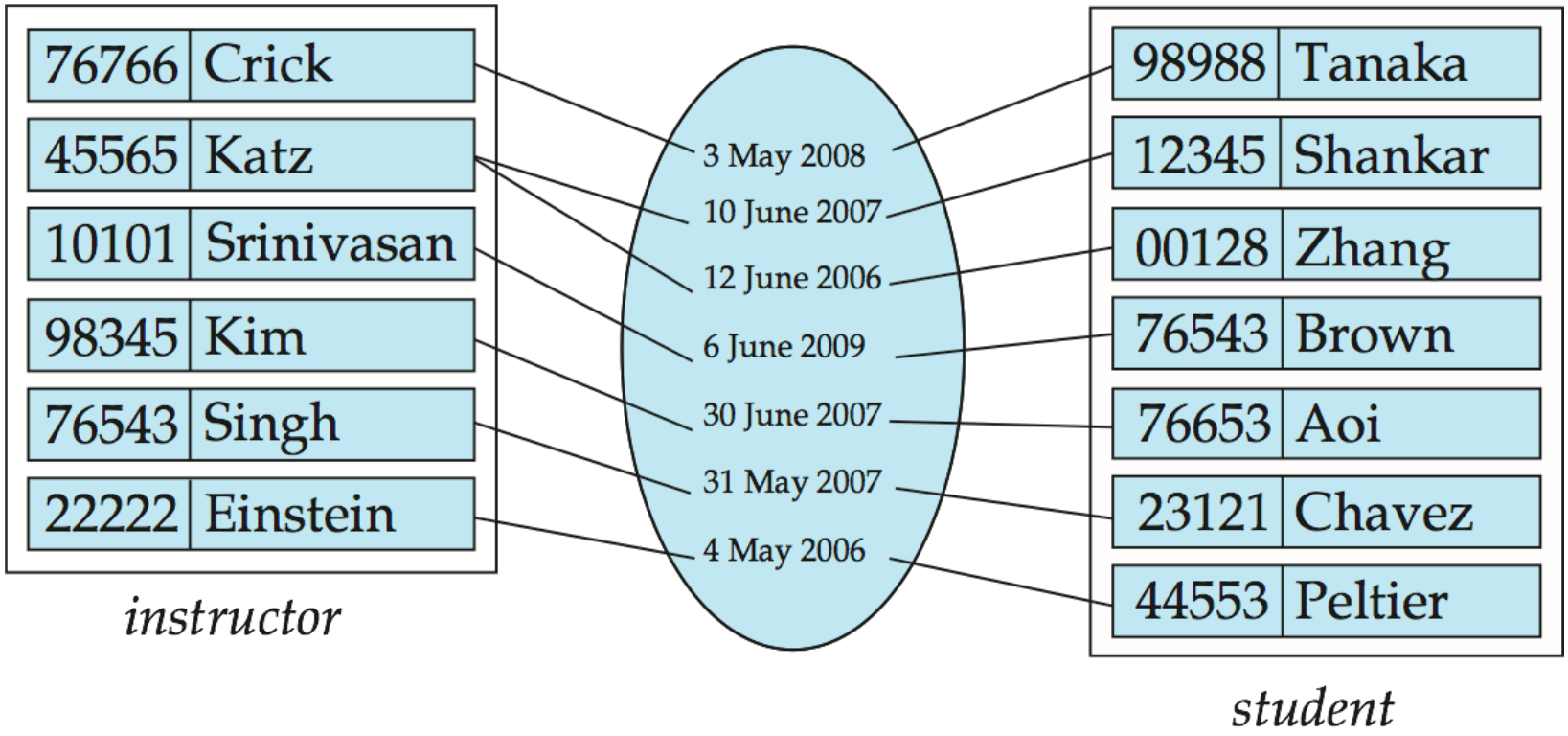
$$(44553,22222) \in advisor$$

Relationship Set *advisor*



Relationship Sets (Cont.)

- An **attribute** can also be property of 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



Degree of a Relationship Set

- **Binary relationship**
  - Involve two entity sets (or degree two)
    - Most relationship sets in a database system are binary
    - Example: Student registers a course
- Relationships between more than two entity sets are **rare** (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*

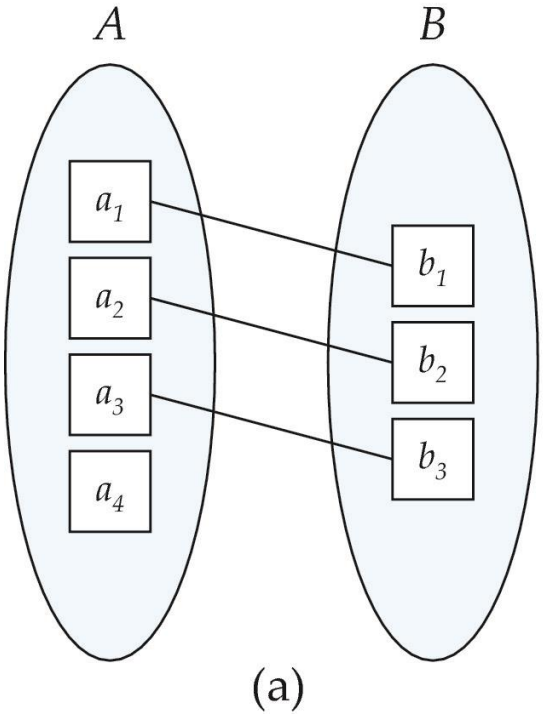
Mapping Cardinality Constraints

- **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

An entity in *A is associated with*
at most one entity in B, and
An entity in *B is associated with*
at most one entity in A



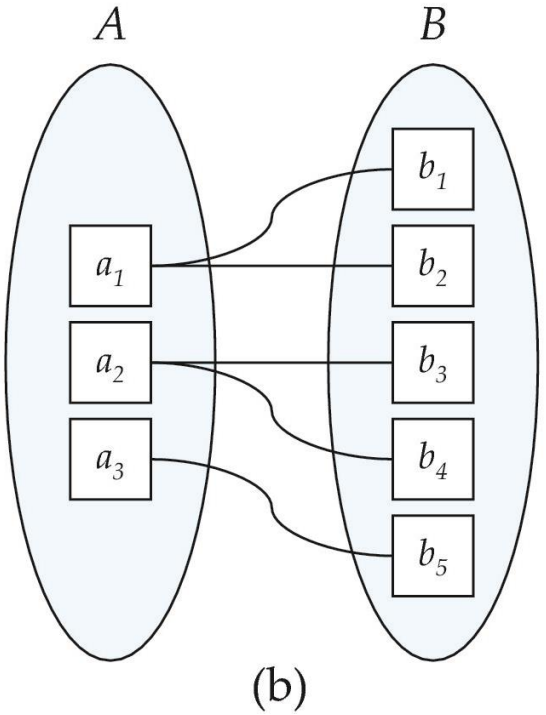
Note: Some elements in *A* and *B* may not be mapped to any elements in other set

Example
An instructor is associated with at most one student via *advisor*
and a student is associated with at most one instructor via *advisor*

Mapping Cardinalities

One to many

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

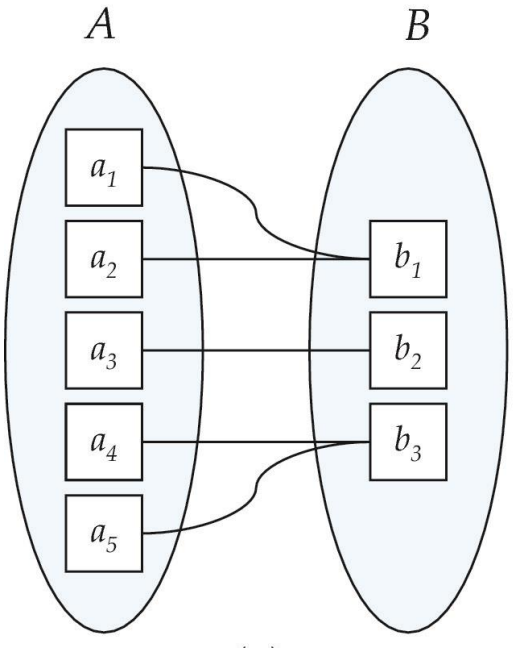


Example
An instructor can advise several students and a student can be advised by only one instructor

The relationship set from *instructor to student is* one-to-many

Mapping Cardinalities

Many to one



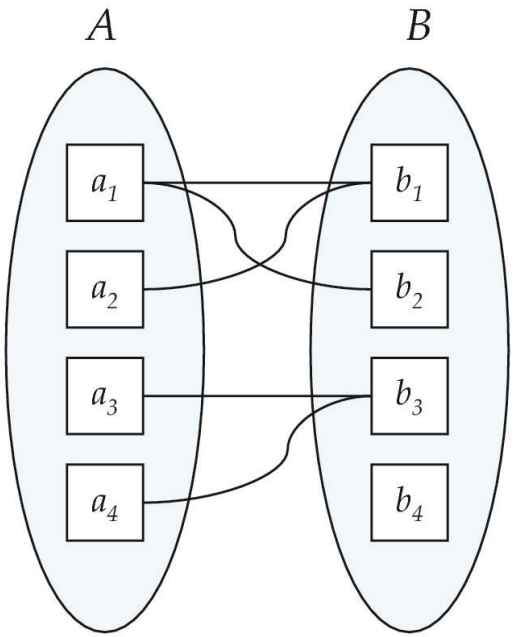
An entity in *A is associated with at most one entity in B*
And
An* entity in *B, however, can be associated with any number (zero or more) of* entities in *A

Example
An instructor may advise at most one student,
But a student may have many advisors

The relationship set from *instructor to student is* many-to-one

Mapping Cardinalities

Many to many



An entity in *A is associated with any number (zero or more)* of entities *in B, And*
an entity in B is associated with any number (zero or* more) of entities in *A

Example
A student can be advised by several instructors (as in the case of students advised jointly)

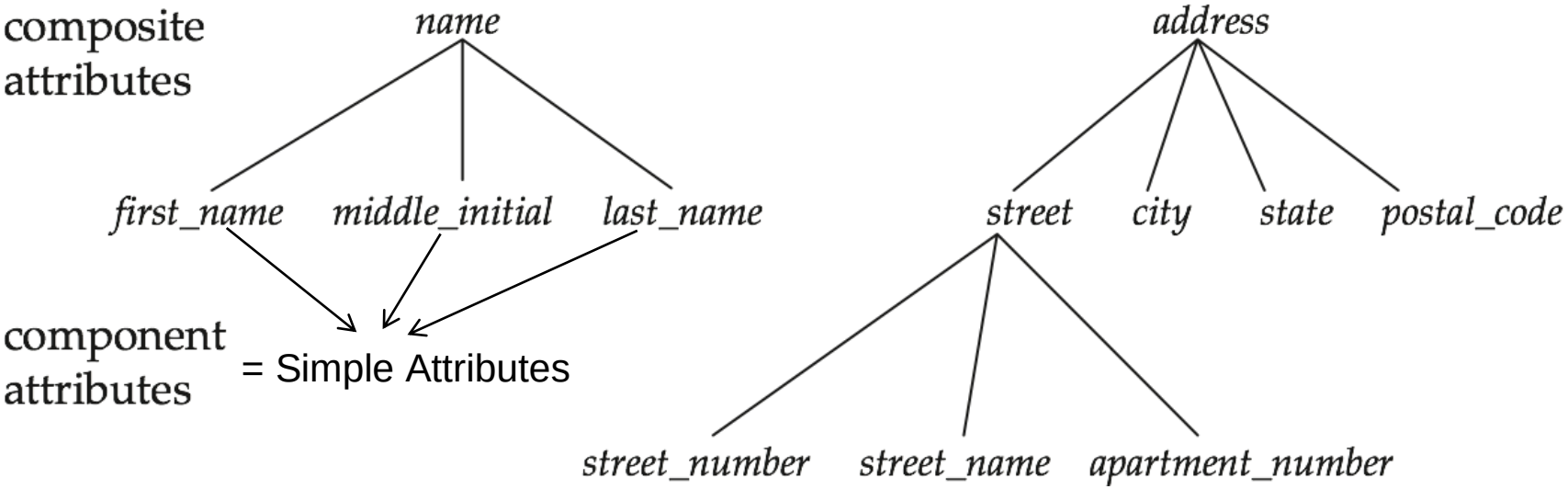
Attributes

- An entity is represented by a set of attributes, that is descriptive properties possessed by all members of an entity set
  - Example:

    instructor = (*ID, name, street, city, salary*)
    course= (*course_id, title, credits*)
- **Domain** – the set of permitted values for each attribute
- Attribute types:
  - **Simple** and **composite** attributes
  - **Single-valued** and **multivalued** attributes
  - **Derived** attributes

Attributes

- Attribute types:
  - **Simple -** Example: First_Name, Salary
  - **Composite**



Attribute Types

- **Simple -** Example: First Name, Salary
- **Composite -** Example: Name, Address
- **Single-valued -** Example: phone_number
- **Multivalued** attributes - Example: *phone_numbers*
- **Derived** attributes
  - Can be computed from other attributes
  - The value of a derived attribute is **not stored but is computed when required**
  - Example
    - Age, derived from given date_of_birth
    - *Students advised (Represents how* many students an instructor advise) derived by counting the number of *student entities associated with that instructor*

Keys

- Used to specify how entities within a given entity set are distinguished
- *In other words, no two entities in an entity* set are allowed to have exactly the same value for all attributes
- Key are applicable to relation schemas also **to identify relationships uniquely**, and thus distinguish relationships from each other

Types:

- **Super key**
- **Candidate key**
- **Primary key**
- **Foreign key**

Keys

- **Super key**
  - A set of one or more attributes taken collectively whose values uniquely determine each entity in the entity set
  - Example
    - ID and name of instructor
    - ID and title of *course*
    - *Course_ID and title of course*

Keys

- **Candidate key**
  - Requires super keys for which no proper subset is a super key
  - A minimal super key of an entity set is called **candidate key**
  - Example
    - *ID* is candidate key of *instructor*, *Course_ID* is candidate key of *course*
  - Several distinct sets of attributes can also serve as a candidate key
  - Example
    1. {*name, dept name*} *can also distinguish instructor*
       - *Then, both {ID} and {name, dept name}* are candidate keys
    2. **Check:** *{ID, name}*
       - *Can also distinguish instructor*
       - *But, this combination can not form a candidate key*
       - ***As* the attribute *ID alone is a candidate key***

Keys

- **Primary key**
  - Although several candidate keys may exist, one of the candidate keys is selected to be the **Primary Key** by the database designer to identify tuples within a relation
  - Examples
    - Instructor ID
    - In the United States, the social-security number attribute of a person would be a candidate key
    - In India, VoterID, PAN ID

Keys

- **Primary key**
  - Must be chosen with care, avoid repeated value attribute
    - Example: name of a person
      - Because there may be many people with the same name
  - Select attribute whose values are never, or very rarely, changed
    - Example: address field of a person
      - Because, it is likely to change

  NOTE: Primary key attributes in entity, relation should be underlined in the model and listed before the other attributes in schema

Keys

- **Foreign key**
  - An entity, say *e1, may include among its attributes the primary key of another* entity, say *e2*
  - *This attribute is called a* ***foreign key from e1, referencing e2***
    - Entity *e1 is called "**referencing entity" of the foreign key dependency***
    - Entity e*2 is called "**referenced entity" of the foreign key***
  - Example
    - Attribute *dept name in instructor is a foreign key from instructor, referencing department,* since *dept name is the primary key of department*
      - *In any database instance,* given any tuple, say *ta, from the instructor entity, there must be some tuple, say tb, in the department entity such that the value of the dept name attribute of ta is* the same as the value of the primary key, *dept name, of tb*

Keys for Relationship Sets

- The combination of primary keys of the participating entity sets forms a super key of a relationship set
- *Means* ***a pair of entity sets can have at most one relationship in a particular relationship set***
- Example
  - (*s_id, i_id*) is the super key of *advisor*
- If the relationship set *R has no attributes associated with it, then the primary key of the relationship set R is a set of* attributes

  primary-key(E1) $\cup$ ***primary-key(E2)*** $\cup \cdots \cup$ ***primary-key(En)***

Keys for Relationship Sets

- If the relationship set *R has attributes a1, a2, . . . , am associated with it, then the primary key of the relationship set R is a* the set of attributes

  primary-key(E1) ∪ primary-key(E2) ∪ · · · ∪ primary-key(En) ∪ {a1, a2, . . . , am}

  - Example
    - For entity sets *instructor and student and the relationship set advisor with attribute date*

- In both of the above cases, the set of attributes *primary-key(E1) ∪ primary-key(E2) ∪ · · · ∪ primary-key(En)* forms a superkey for the relationship set

- The structure of the primary key for the relationship set **ALSO depends on the mapping cardinality of the relationship set**

Keys for Relationship Sets

- Primary key of *advisor relationship*
  - If, **one-to-one relationships**
    - Either candidate key can be used as the primary key
  - If **many-to-one relationship**
    - From ***student to instructor*** is
      - *That is, each student* can have at most one advisor
      - *Simply the* primary key of *student*
    - *From **instructor to student***
      - *That is, an instructor can advise only one student*
      - *Simply the primary key of instructor*
  - If **many-to-many relationship**
    - *Union of the primary keys of instructor and student*

Redundant Attributes

- Suppose we have entity sets
  - *instructor*, with attributes including *dept_name*
  - *department*

  and a relationship
  - ***inst_dept*** relating *instructor* and *department*
- Attribute *dept_name* in entity *instructor* is redundant since there is an explicit relationship *inst_dept* which relates instructors to departments
  - The attribute replicates information present in the relationship, and should be removed from *instructor*
  - BUT: when converting back to tables, in some cases the attribute gets reintroduced, we will see later

University Schema- Entity Set, Attributes, Primary Key

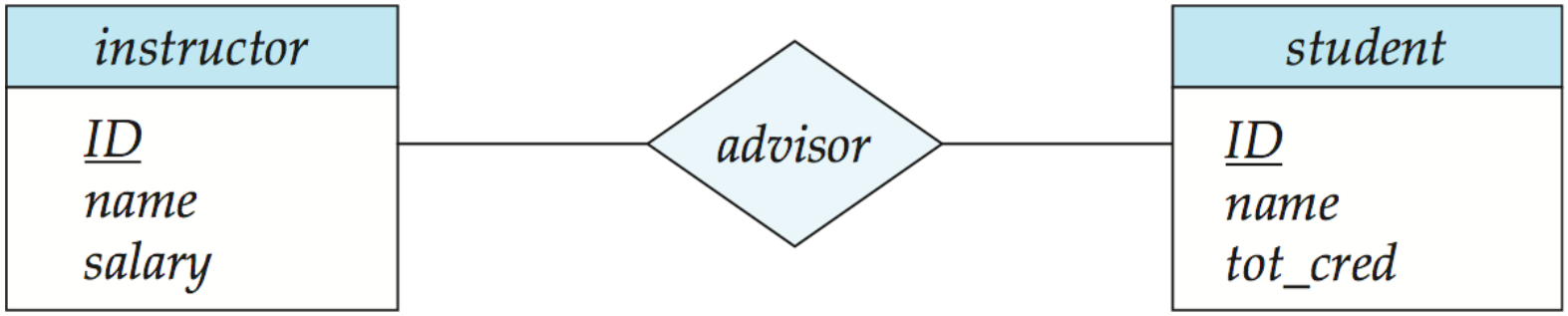
1. **Student**
   - **Student (*ID, name, tot cred*)**
2. **Instructor**
   - **Instructor (*ID, name, salary*)**
3. **Department**
   - **Department(*dept name, building, budget*)**
4. **Classroom**
   - **Classroom(*building, room number, capacity*)**
5. **Course**
   - **Course(*course id, title, credits*)**
6. **Section**
   - **Section(*course id, sec id, semester, year*)**
7. **Time slot**
   - **Timeslot(*time slot id, {(day, start time, end time) }*)**

A good entity-relationship design does not contain redundant attributes

University Schema- Relationship Sets

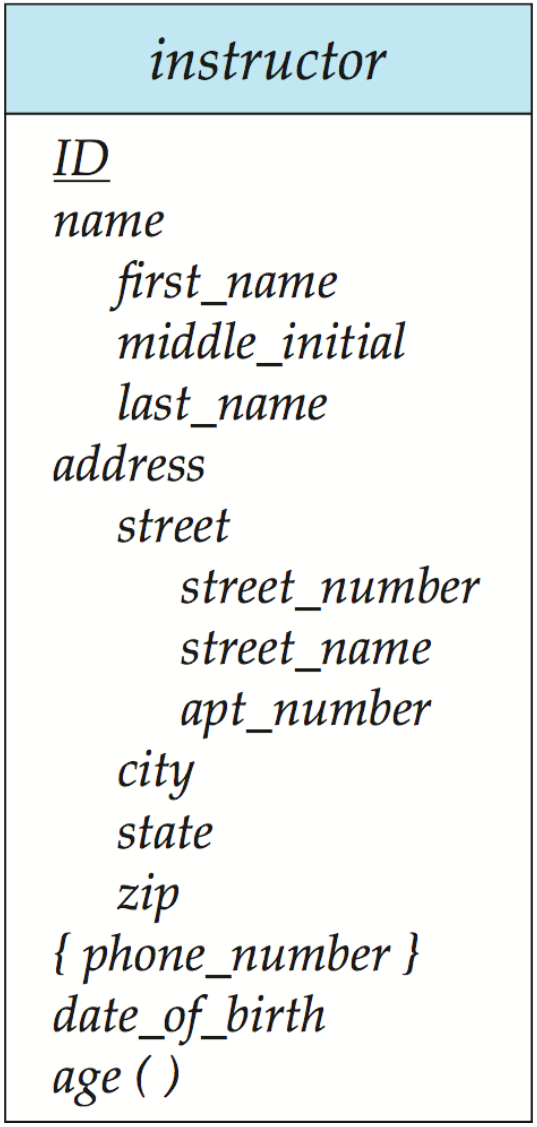
1. inst dept: relating instructors with departments
2. stud dept: relating students with departments
3. teaches: relating instructors with sections
4. takes: relating students with sections, with a descriptive attribute *grade*
5. course dept: relating courses with departments
6. sec course: relating sections with courses
7. sec class: relating sections with classrooms
8. sec time slot: relating sections with time slots
9. advisor: relating students with instructors
10. prereq: relating courses with prerequisite courses

E-R Diagrams

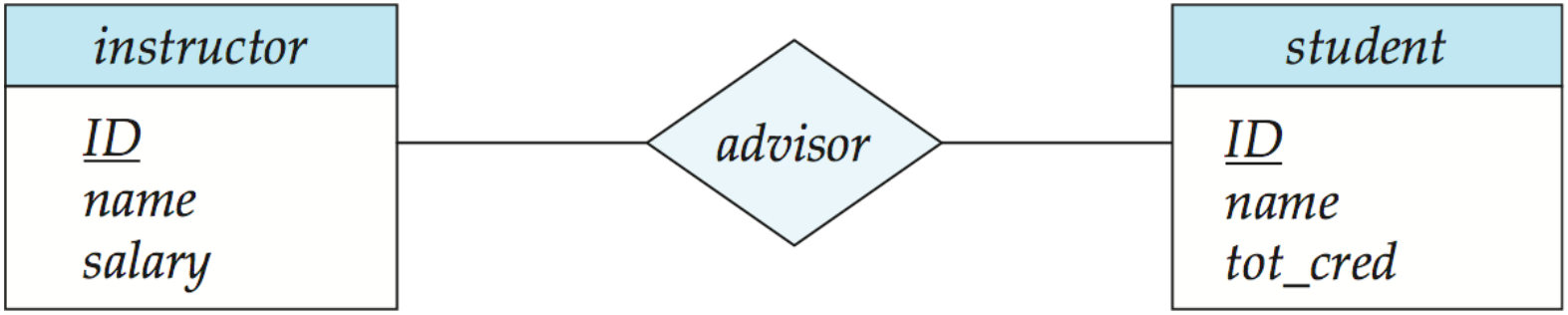


- Rectangles represent entity sets
- Diamonds represent relationship sets
- Attributes listed inside entity rectangle
- Underline indicates primary key attributes

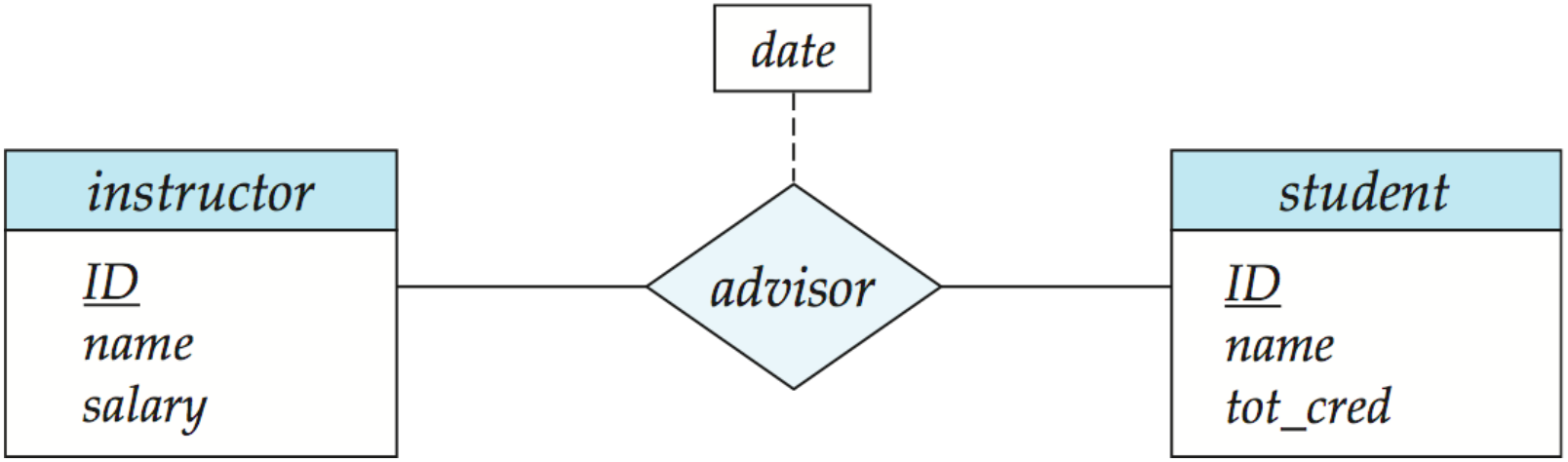
Entity With Composite, Multivalued and Derived Attributes



Relationship Sets

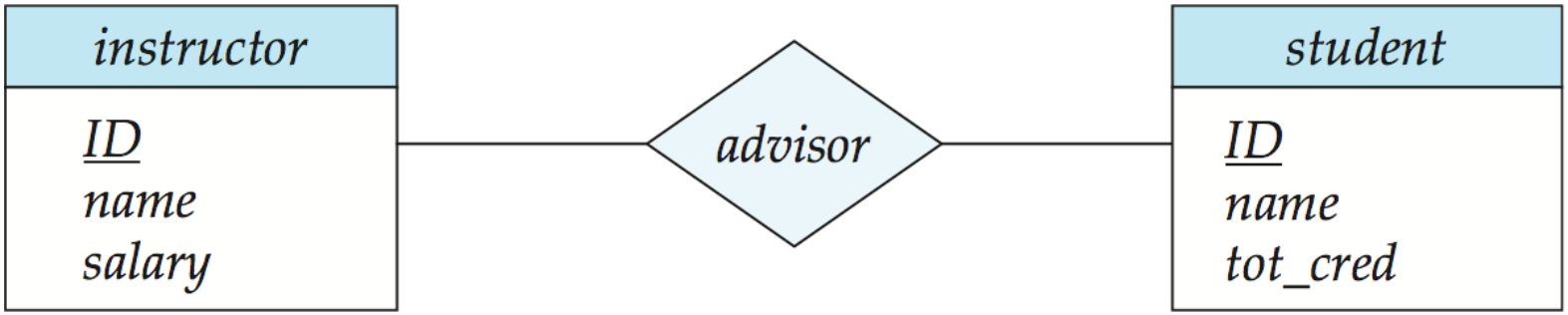


Relationship Sets with Attributes



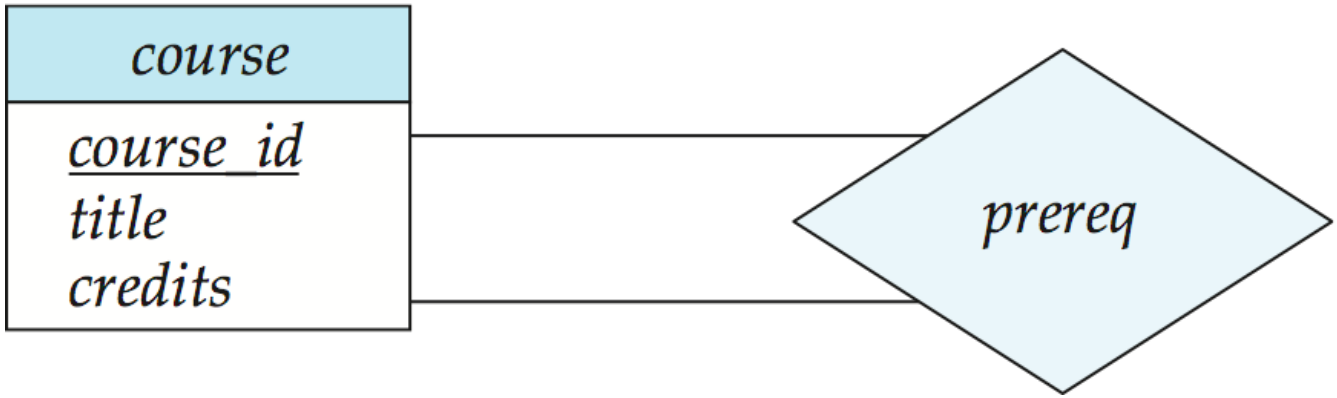
Roles

- The function that an entity plays in a relationship is called that entity's **role**
- Almost, entity sets participating in a relationship set are generally **distinct**
  - Roles are implicit and are not usually specified



Roles

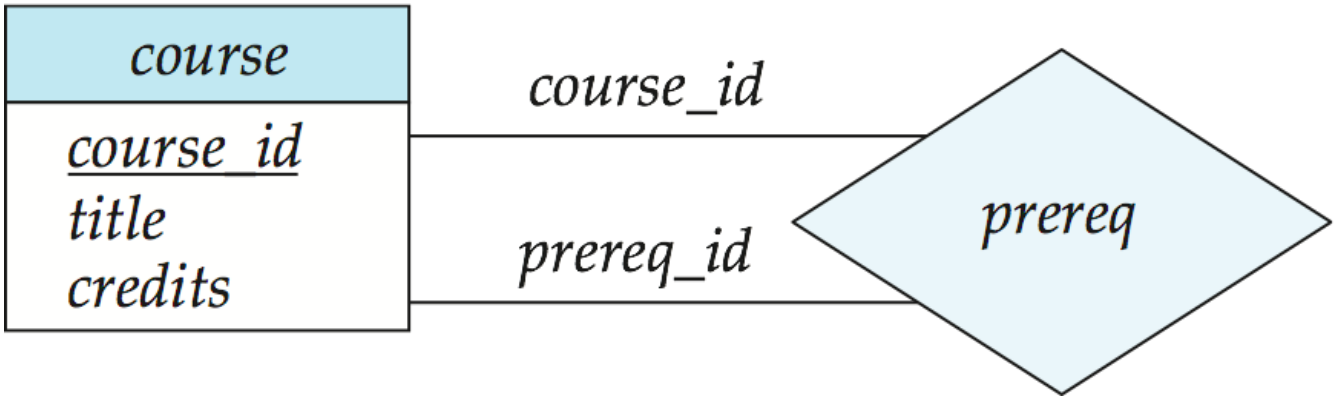
- But, if same entity set is used for the relationship i.e. the same entity set participates in a relationship set more than once, in different roles
- Example, Course has Prerequisite Course



- Roles are useful when the meaning of a relationship needs clarification
- In this type of relationship set, sometimes called a **recursive relationship set, explicit role names are necessary to** specify how an entity participates in a relationship instance

Roles

- For example, consider the entity set *course that records information about all the courses offered in the* university
  - To depict the situation where one course (C2) is a prerequisite for another course (C1) we have relationship set *prereq that is* ***modeled by ordered*** **pairs of** ***course entities***
  - *The first course of a pair takes the role of courseC1, whereas* the second takes the role of prerequisite course C2
  - All relationships of *prereq are* ***characterized by (C1, C2) pairs****; and* ***(C2, C1) pairs are excluded***



- The labels "*course_id*" and "*prereq_id*" are called **roles**

Cardinality Constraints

- Express the number of entities to which another entity can be associated via a relationship set
- Express cardinality constraints by drawing either
  - **A directed line (→), signifying "one," or**
  - **An undirected line (—), signifying "many,"**

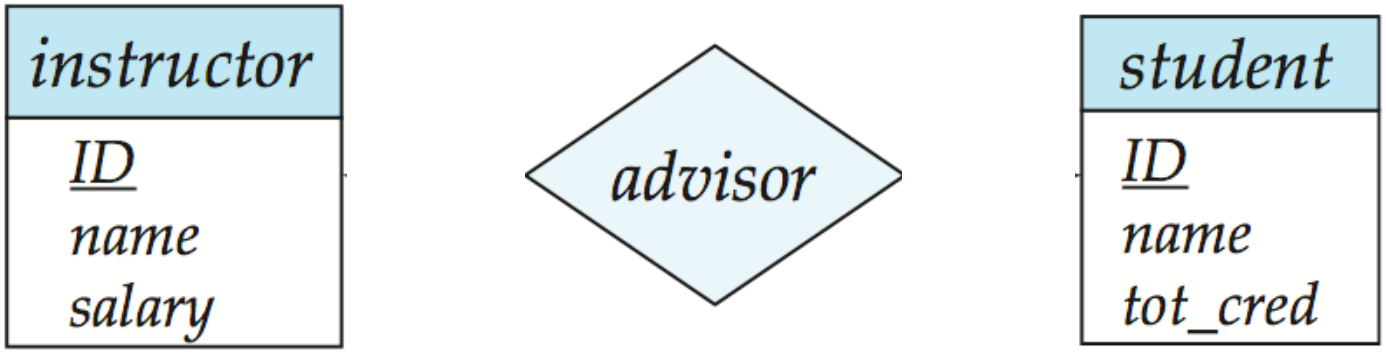
  between the relationship set and the entity set

One-to-One Relationship

- One-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 at most one instructor via *advisor*

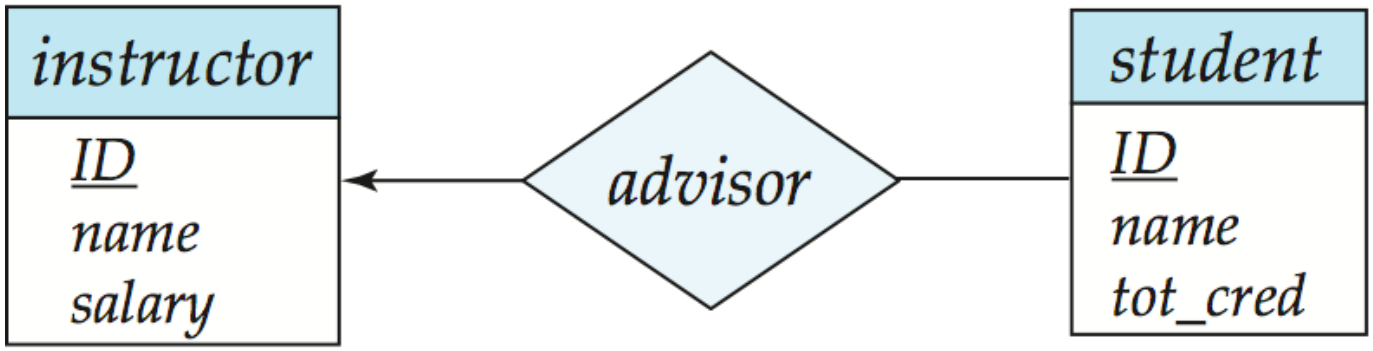


One-to-Many Relationship

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

And

  - A student is associated with at most one instructor via advisor,

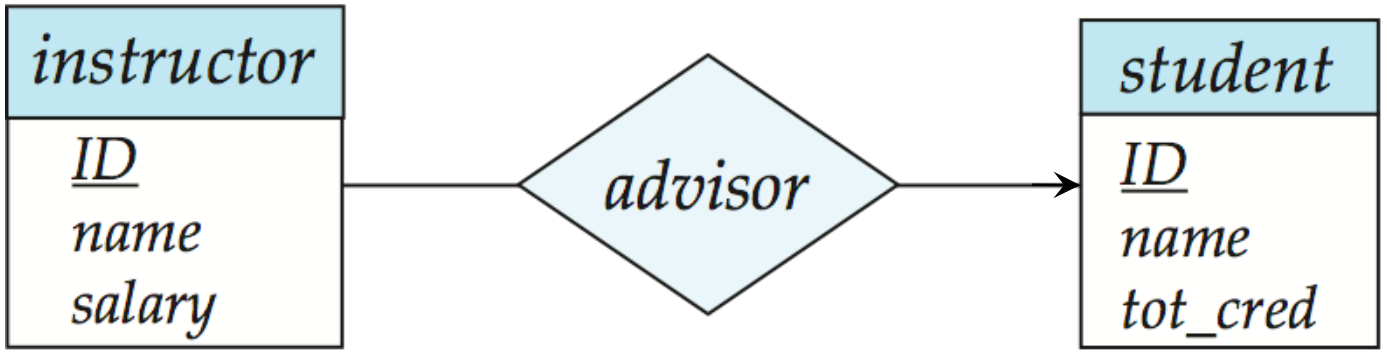


Many-to-One Relationships

- 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*

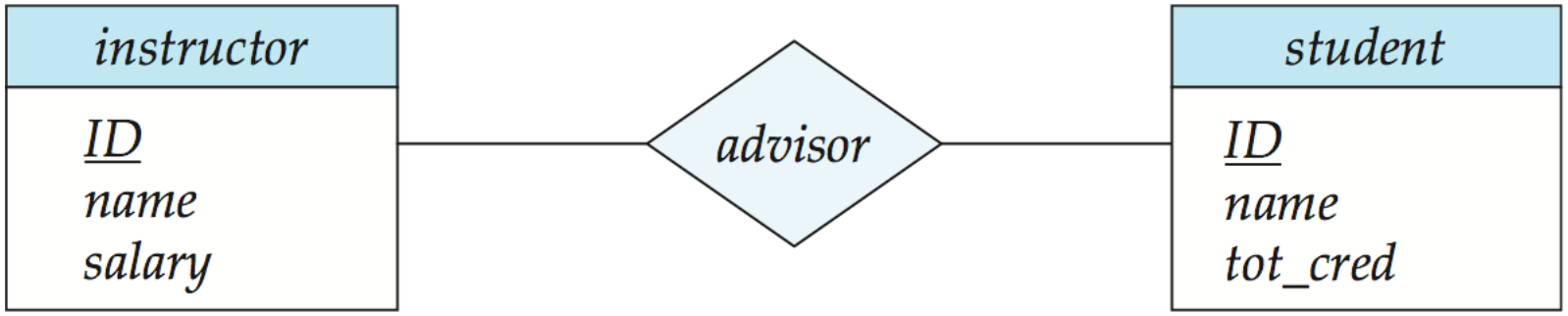


Many-to-Many Relationship

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

And

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



Exercise

1. Identify the Entity Attributes and Primary Key for
    1. Car
    2. Player
2. Discover Primary key for the following:
    1. Match(TeamA, TeamB, Date, TeamAScore, TeamBScore, RefID)
3. Describe the following Relationship:

4. Draw the following relationship:
    1. A vehicle has many wheels, each wheel has one vehicle

Identify the Entity Attributes and Primary Key

1. Identify the Entity Attributes and Primary Key for
    1. Car
    2. Player

- Entity Car
    - Colour
    - Number of doors
    - Convertible
    - Registration number

- Car(<u>Registration number</u>, Colour, Number of doors, Convertible)

Identify the Primary Key for Entity Attributes:

- Entity:Player
  - Name: string, Position: string, Number: integer, injured: boolean, Team: string

Attribute	Unique	Reason
Name	No	might have several players with the same name
Position	No	might have two goalies
Number	No	might be storing the details of multiple teams, in which case each number has several players from different teams
Injured	No	several people might be injured at the same time
Team	No	several players can play for the same team

- I
- Player(<u>PlayerID</u>, Name, Position, Number, Injured, Team)

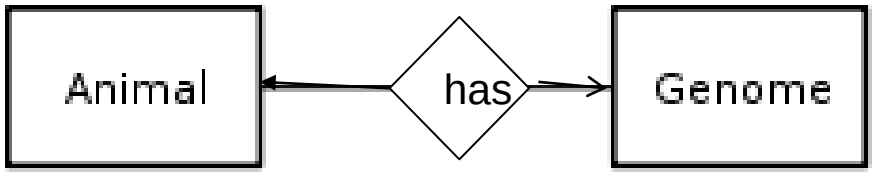
Discover Primary keys

2. Discover Primary key for the following:

- Match(TeamA, TeamB, Date, TeamAScore, TeamBScore, RefID)

Match(<u>TeamA</u>, <u>TeamB</u>, <u>Date</u>, TeamAScore, TeamBScore, RefID)

Describe the following Relationships:



Answer :
An animal has one genome
A genome describes only one animal

Draw the following relationships:

- A vehicle has many wheels, each wheel has one vehicle

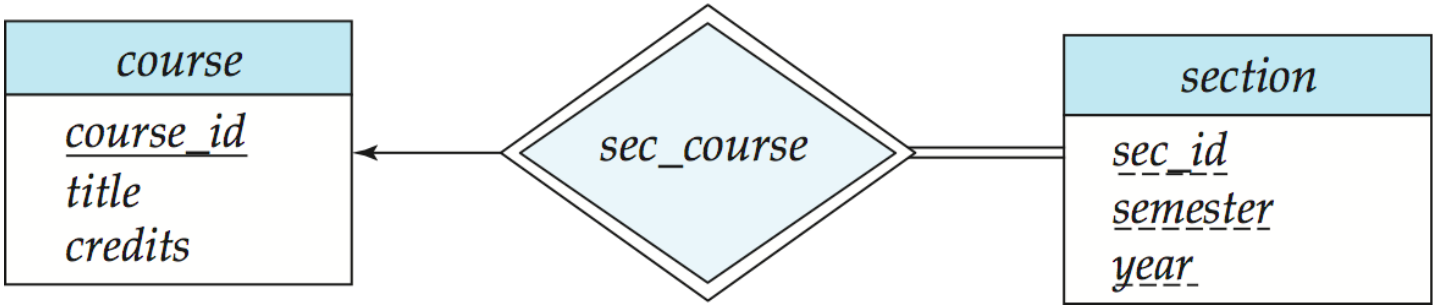
vehicle ← wheel

Weak Entity Sets

- An entity set that does not have a primary key
- The existence of a weak entity set depends on the existence of a **identifying entity set**
  - It must relate to the identifying entity set via a total, one-to-many relationship set from the identifying to the weak entity set
  - **Identifying relationship** depicted using a double diamond
- The **discriminator** (*or partial key)* of a weak entity set is the set of attributes that distinguishes among all the entities of a weak entity set
- The primary key of a weak entity set is formed by the primary key of the strong entity set on which the weak entity set is existence dependent, plus the weak entity set's discriminator

Weak Entity Sets (Cont.)

- Underline the discriminator of a weak entity set with a dashed line
- Put the identifying relationship of a weak entity in a double diamond
- Primary key for *section* – (*course_id, sec_id, semester, year*)

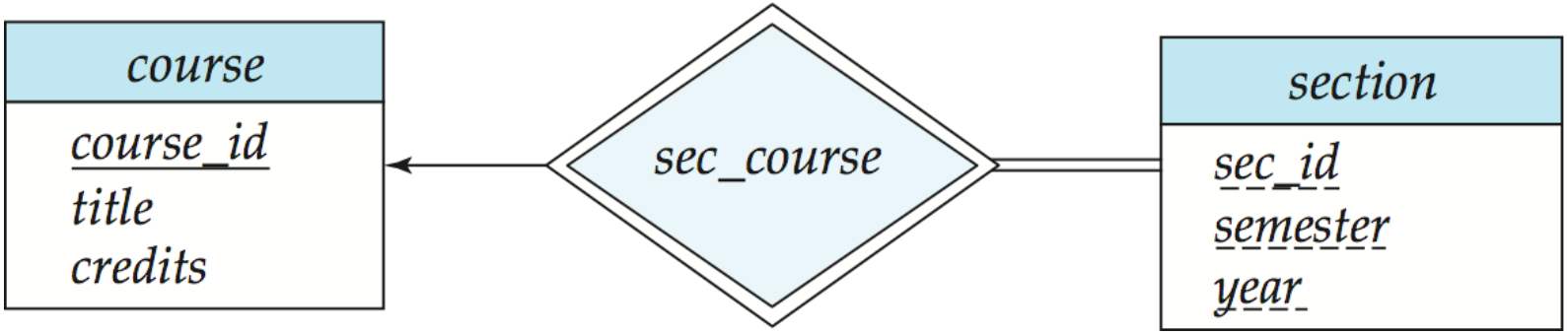


Weak Entity Sets (Cont.)

- Note: the primary key of the strong entity set is not explicitly stored with the weak entity set, since it is implicit in the identifying relationship
- If *course_id* were explicitly stored, *section* could be made a strong entity, but then the relationship between *section* and *course* would be duplicated by an implicit relationship defined by the attribute *course_id* common to *course* and *section*

Participation of an Entity Set in a Relationship Set

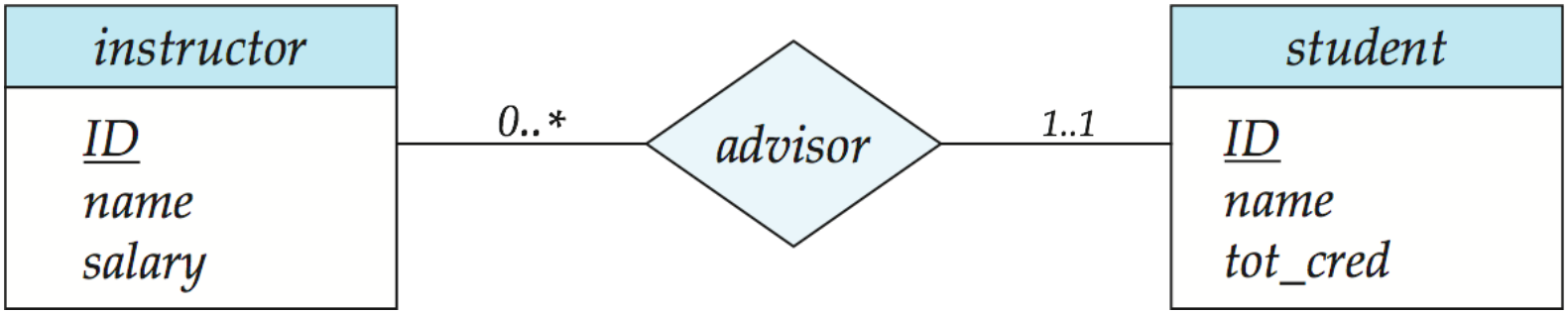
- **Total participation (indicated by double line):** every entity in the entity set participates in at least one relationship in the relationship set
  - E.g., Participation of *section* in *sec_course* is total
    - Every *section* must have an associated course



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

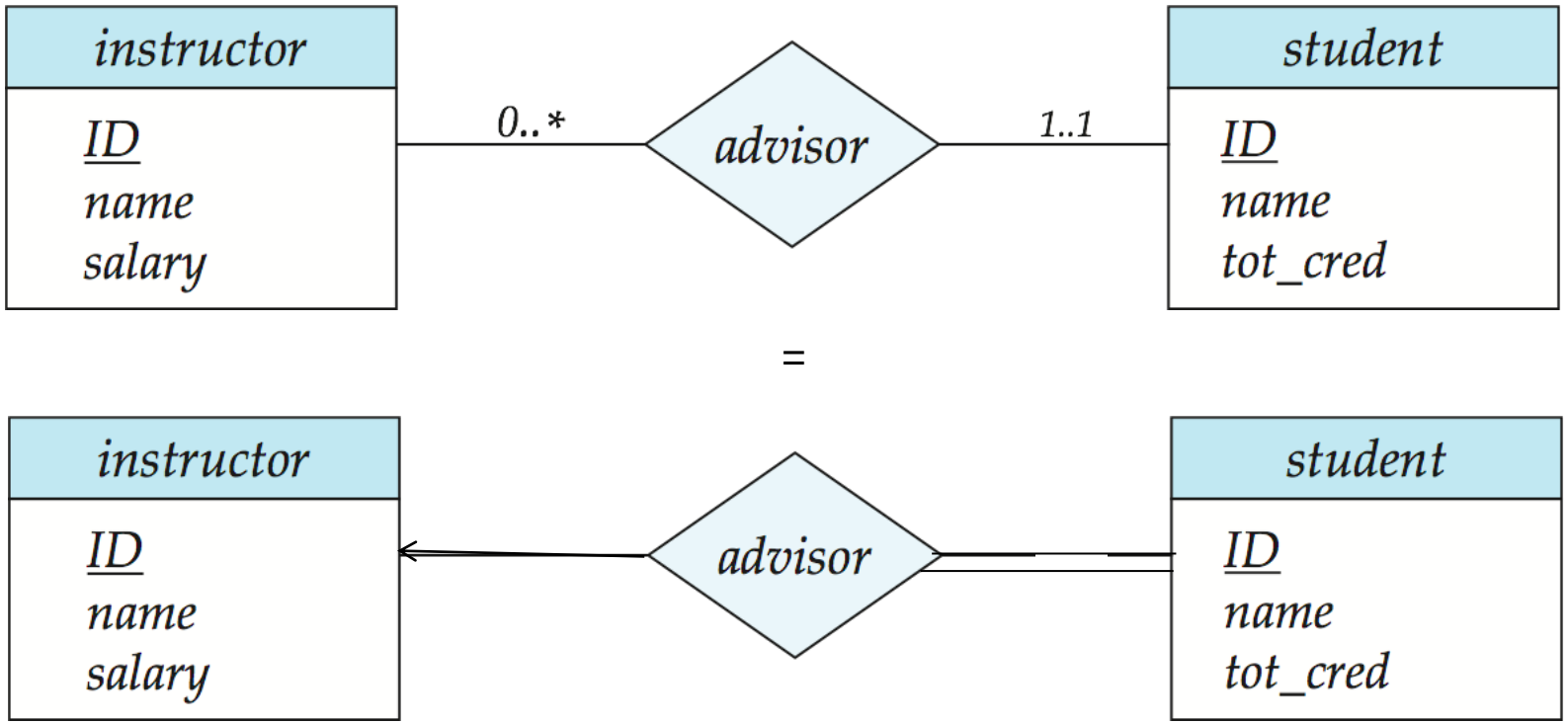
Alternative Notation for Cardinality Limits

- Cardinality limits can also **express participation 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*



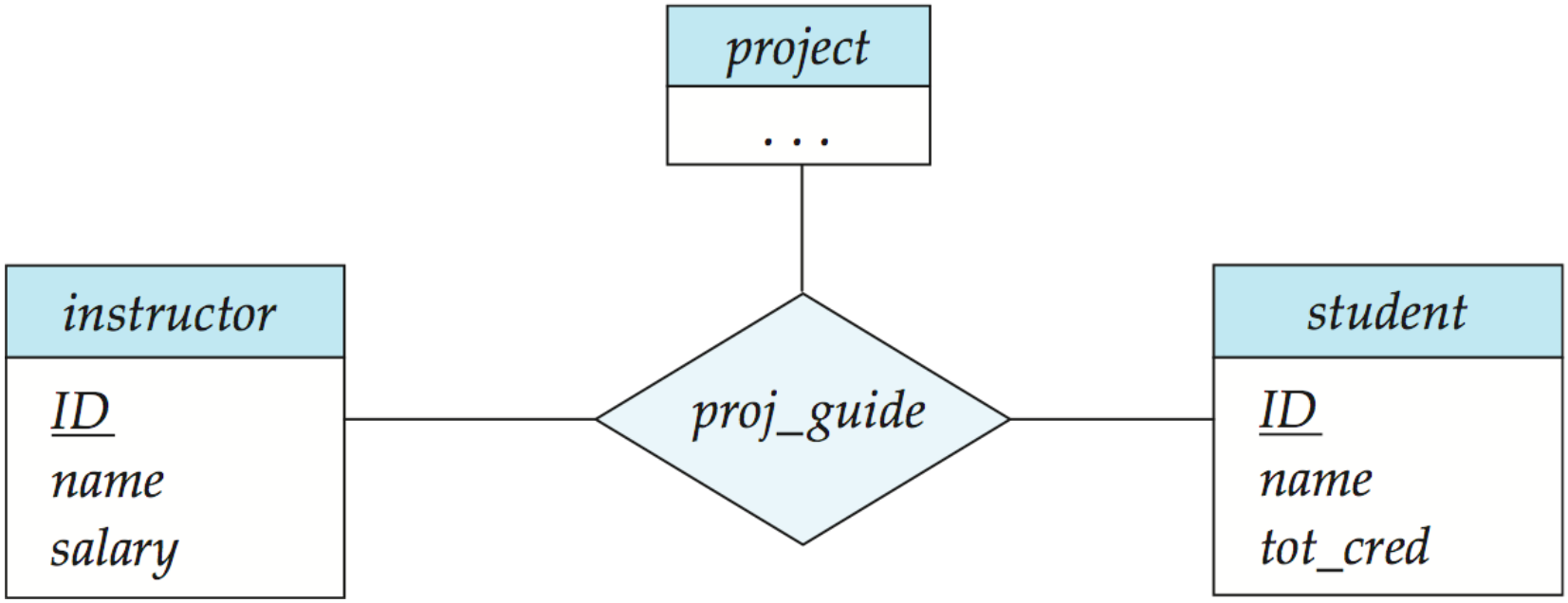
- Between *advisor and student has a* cardinality constraint of 1*..1*
  - That is, **each student must have exactly one advisor**
- The limit 0*..∗* on the line between *advisor and instructor indicates*
  - *That* ***an instructor can have*** **zero or more students**
- *T*he relationship *advisor is* ***one-to-many from instructor*** **to** ***student,*** *and* ***further the participation of student in advisor is total, implying*** **that a student must have an advisor**

Alternative Notation for Cardinality Limits



One to Many Relationship

E-R Diagram with a Ternary Relationship



Cardinality Constraints on Ternary Relationship

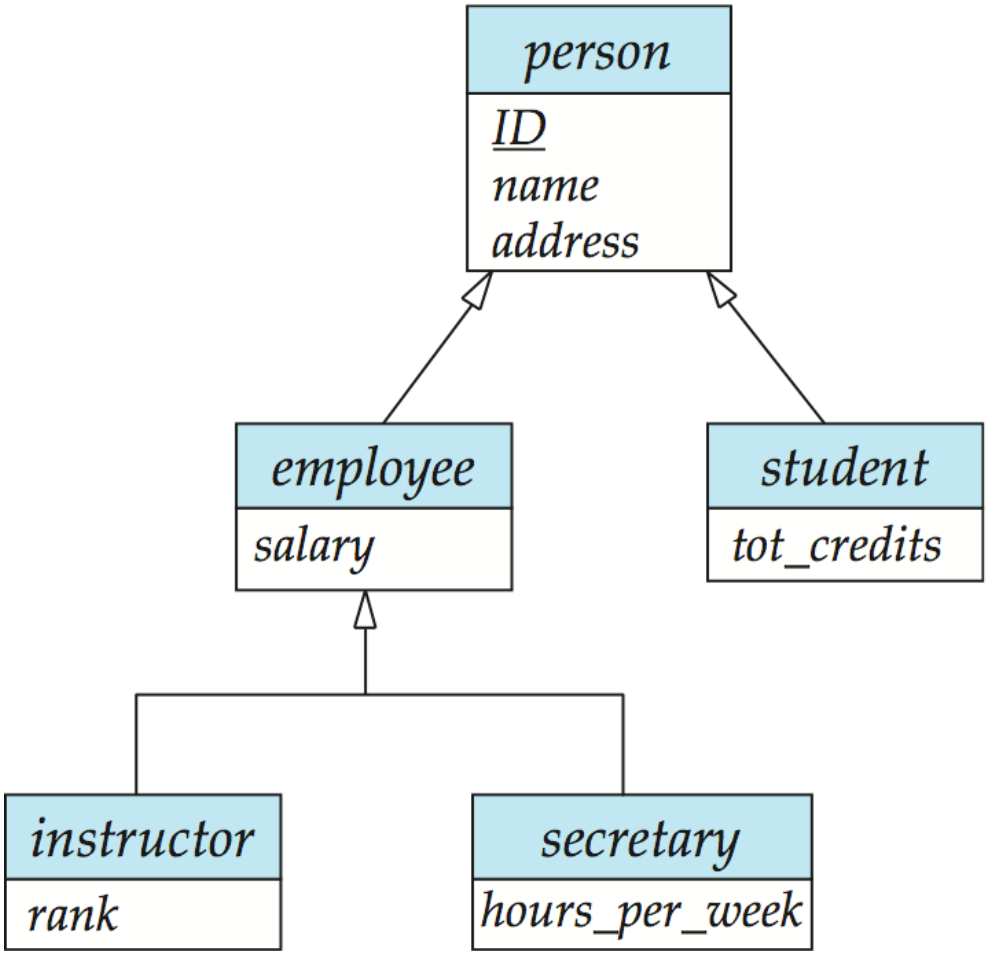
- Allow at most one arrow out of a ternary (or greater degree) relationship to indicate a cardinality constraint
- E.g., an arrow from *proj_guide* to *instructor* indicates each student has at most one guide for a project

Extended ER Features

Extended E-R Features: Specialization

- Top-down design process;
- To desig subgroupings within an entity set that are distinctive from other entities in the set
- These subgroupings become lower-level entity sets that have attributes or participate in relationships that do not apply to the higher-level entity set
- Depicted by a hollow *triangle* arrow component labeled ISA (E.g., *instructor* “is a” *person*)
- **Attribute inheritance** – a lower-level entity set inherits all the attributes and relationship participation of the higher-level entity set to which it is linked

Specialization Example



Extended ER Features: Generalization

- **A bottom-up design process** – combine a number of entity sets that share the same features into a higher-level entity set
- Specialization and generalization are simple inversions of each other; they are represented in an E-R diagram in the same way
- The terms specialization and generalization are used interchangeably

Specialization and Generalization (Cont.)

- Can have multiple specializations of an entity set based on different features
- E.g., *permanent_employee* vs. *temporary_employee*, in addition to *instructor* vs. *secretary*
- Each particular employee would be
  - A member of one of *permanent_employee* or *temporary_employee*,
  - And also a member of one of *instructor*, *secretary*
- The ISA relationship also referred to as **superclass - subclass** relationship

Design Constraints on a Specialization/Generalization

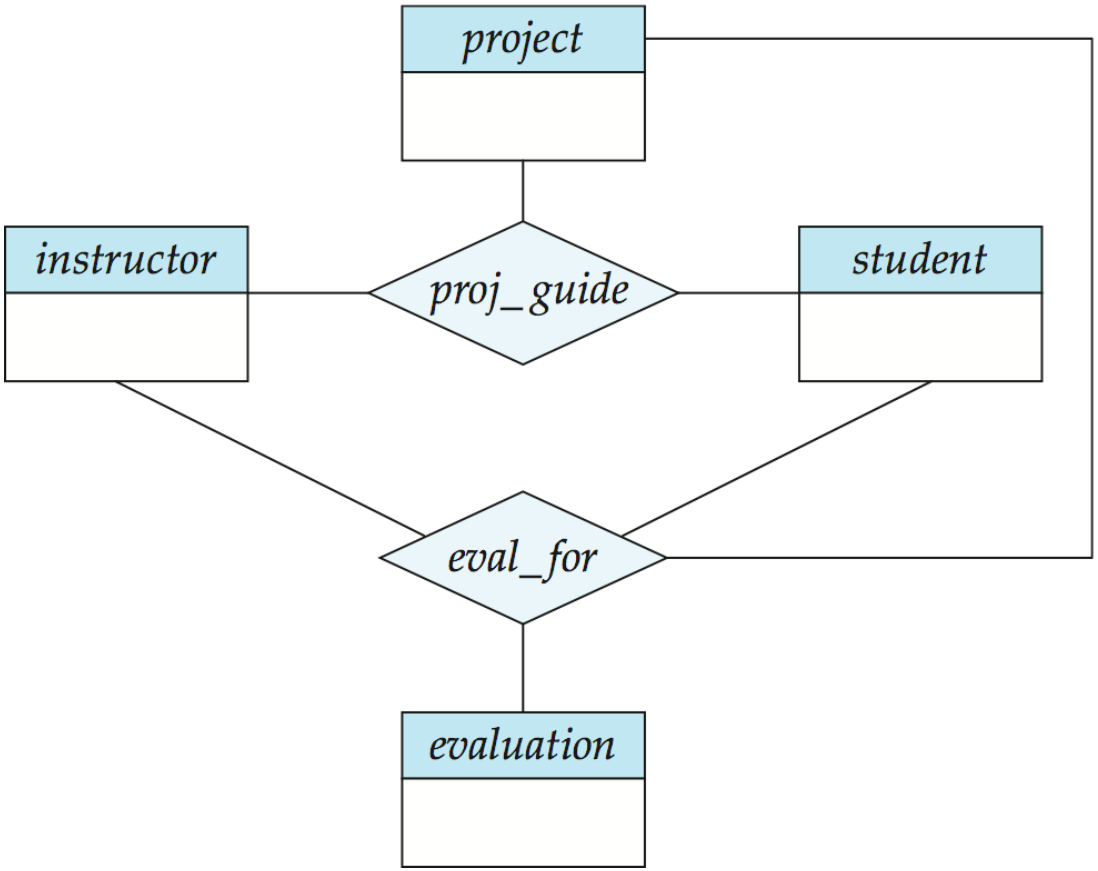
- Constraint on which entities can be members of a given lower-level entity set
  - Condition-defined
    - Example: all customers over 65 years are members of *senior-citizen* entity set; *senior-citizen* ISA *person*
  - User-defined
- Constraint on whether or not entities may belong to more than one lower-level entity set within a single generalization
  - **Disjoint**
    - An entity may belong to only one specialized lower-level entity set
    - Instructor and secretary are specialization of employee, not both
    - A single arrow is used
  - **Overlapping**
    - An entity may belong to more than one multiple lower-level entity set
    - Person is either employee or student, both is also permitted
    - Two separate arrows are used

Design Constraints on a Specialization/Generalization (Cont.)

- **Completeness constraint** -- specifies whether or not an entity in the higher-level entity set must **belong to** at least one of the lower-level entity sets within a generalization
  - **Total**: an entity must belong to one of the lower-level entity sets
    - Student generalization is total
    - All students must be either UG or PG
  - **Partial**: an entity need not belong to one of the lower-level entity sets
    - Work team entity is partial specialization
    - Employees are assigned to team after 3 months on the job
    - Some employees may not be members of any of the lower level team entity set

Aggregation

- Consider the ternary relationship *proj_guide*
- Suppose wanted to record evaluations of a student by a guide on a project

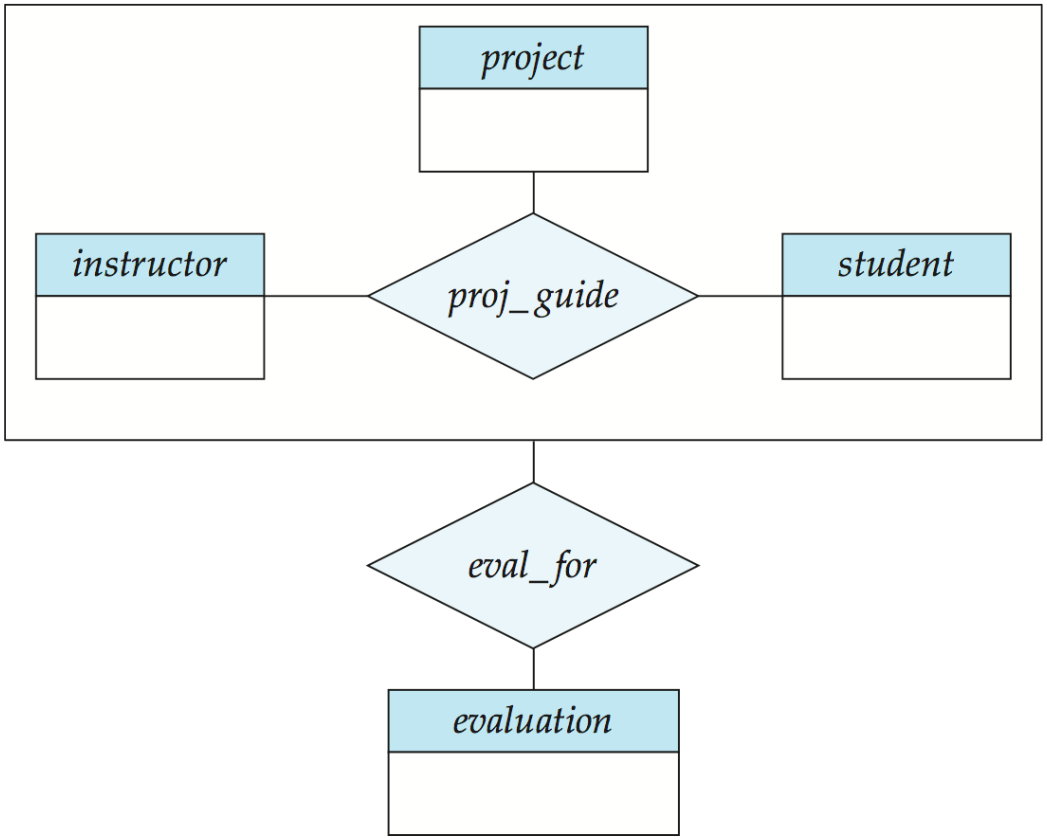


Aggregation (Cont.)

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

Aggregation (Cont.)

- Without introducing redundancy, the following diagram represents:
  - A student is guided by a particular instructor on a particular project
  - A student, instructor, project combination may have an associated evaluation



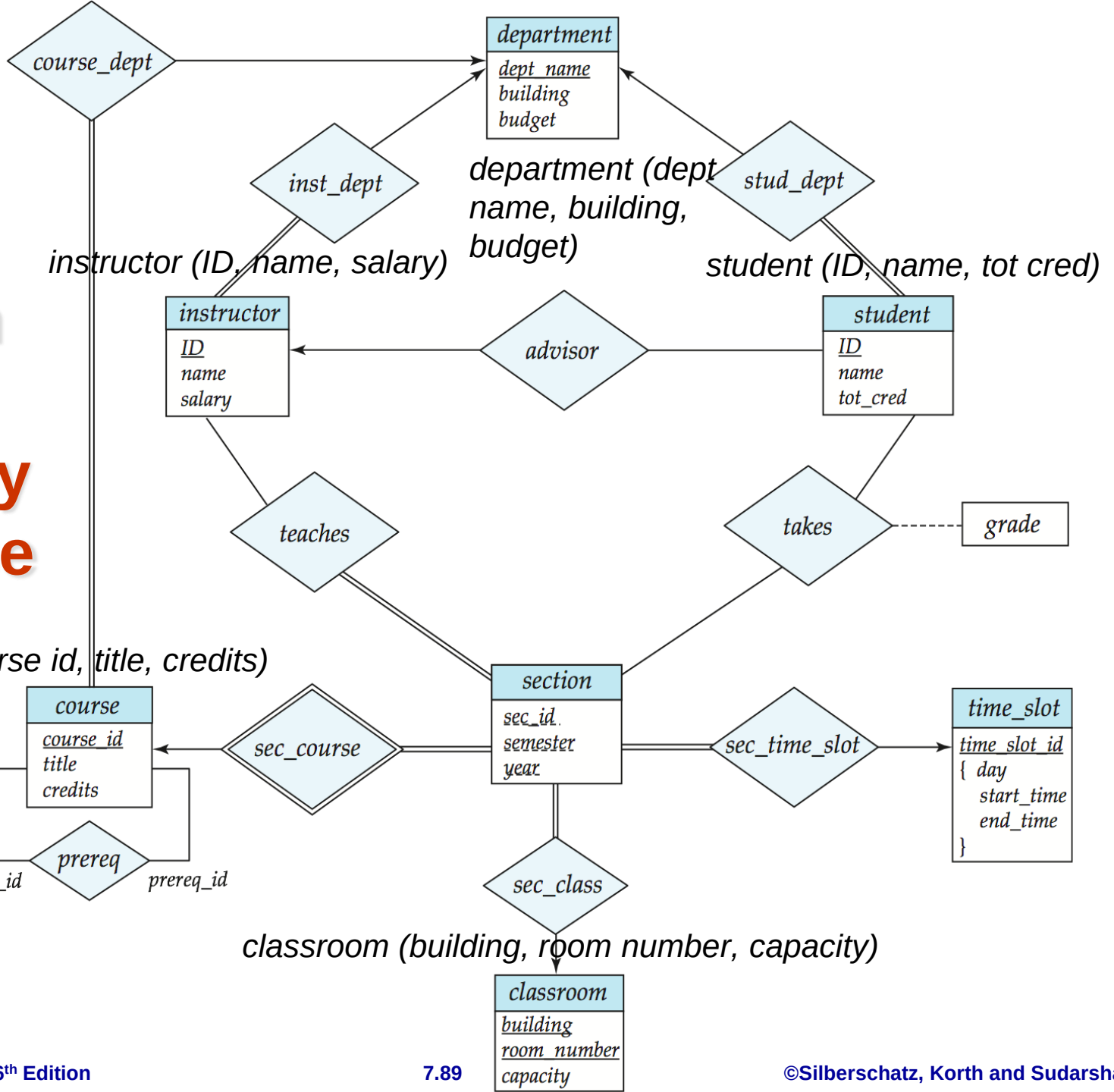
ER-Model Design

Bank Application

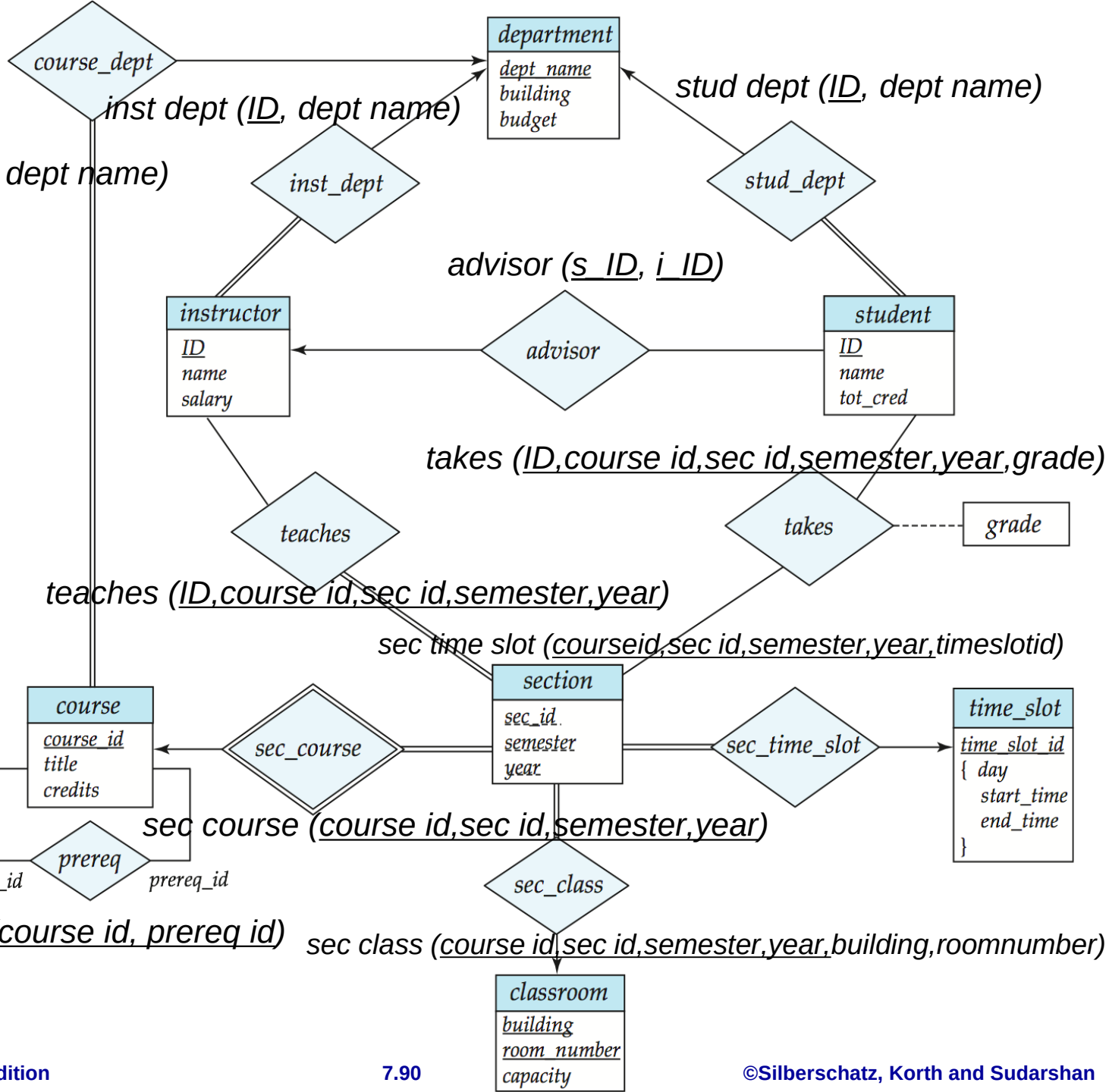
Database System Concepts, 6th Ed.

Reduction to Relational Schemas

ER Diagram for a University Enterprise



Database Schema



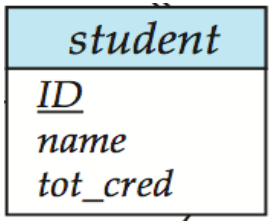
Reduction to Relation Schemas

- Entity sets and relationship sets can be expressed uniformly as *relation schemas*
  - To 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 columns (generally corresponding to attributes), which have unique names

ER-Model

Design Issue: Representing Strong Entity Set (Atomic attributes)

- A strong entity set



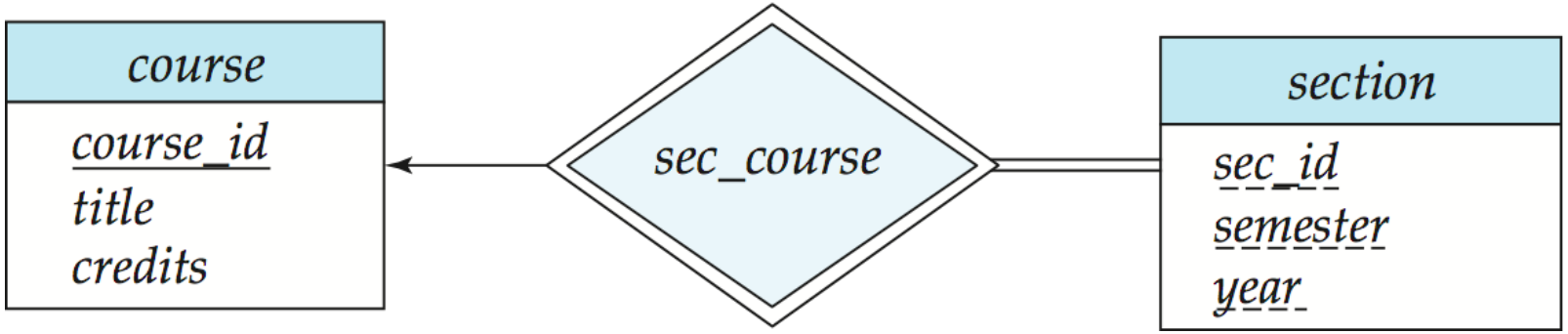
  - Reduces to a schema with the same attributes

    student(ID, name, tot_cred)

ER-Model

Design Issue: Representing Weak Entity Set (Atomic attributes)

- A weak entity set



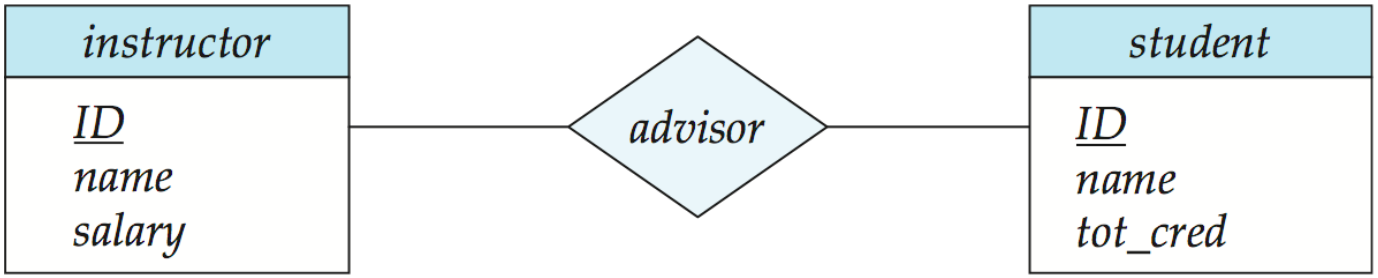
  - Becomes a table that includes a column for the primary key of the identifying strong entity set (no need of separate sec_course)
    section (course_id, sec_id, sem, year)

ER-Model

Design Issue: Representing Many to Many Relations

- A many-to-many relationship set
  - Represented as a schema with attributes for the primary keys of the two participating entity sets, and any descriptive attributes of the relationship set
- Example: schema for relationship set *advisor*

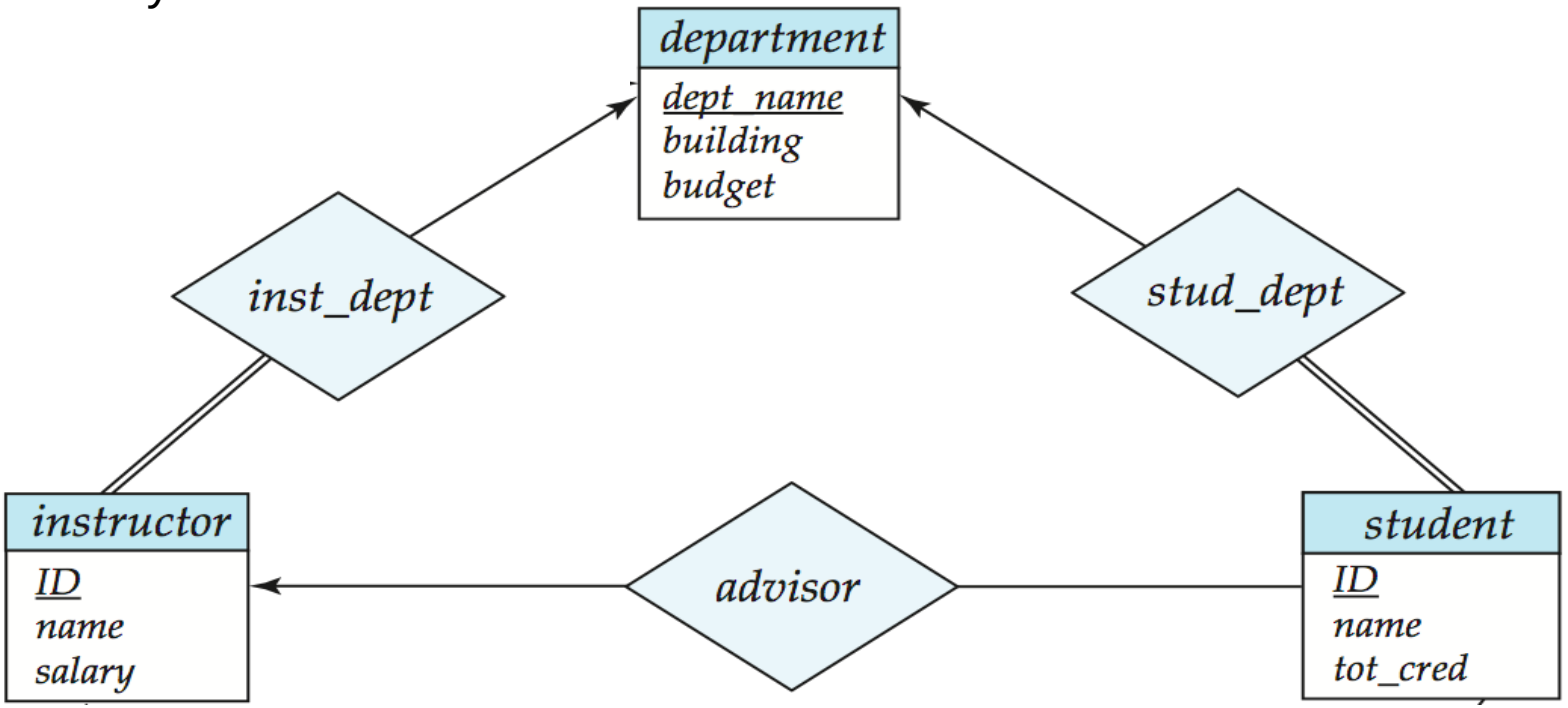
  advisor = (*s_id, i_id*)



ER-Model

Design Issue: Representing Many to 1 and 1 to Many Relations

- **Many-to-one and one-to-many relationship sets** that are **total on the many-side**
  - 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*



ER-Model

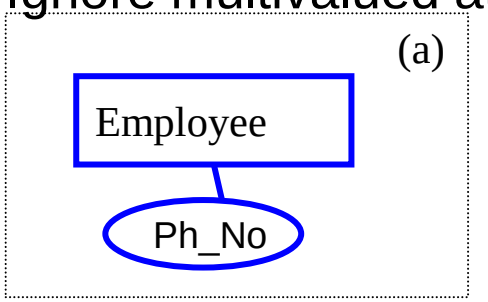
Design Issue: Representing 1 to 1 Relations

- For one-to-one relationship sets, either side can be chosen to act as the “many” side
  - That is, extra attribute can be added to either of the tables corresponding to the two entity sets

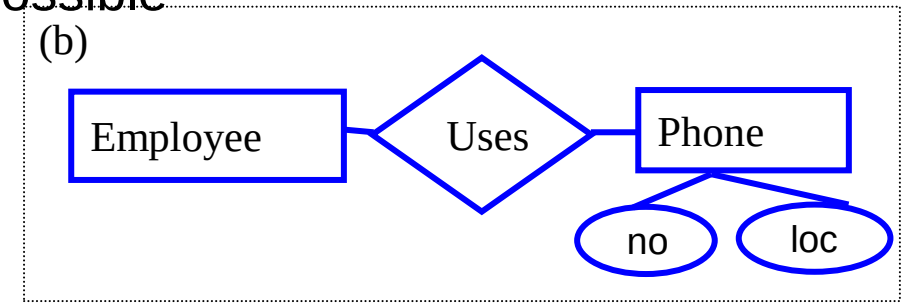
ER-Model

Design Issue: Entity Sets vs. Attributes (Multi Valued)

- An Example: Employees can have multiple phones
  - Ignore multivalued attributes, if possible



vs



Solution: Determine how phones are used

1. Can many employees share a phone?

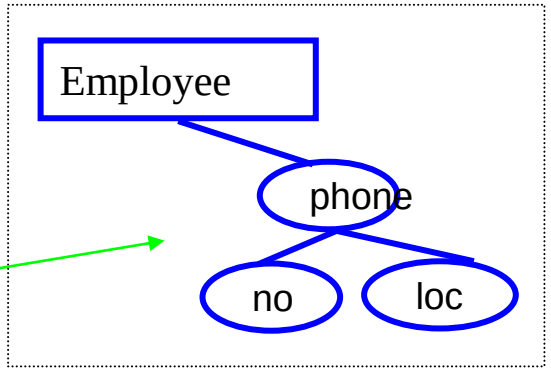
   (If yes, then (b)) → Use of phone as an entity allows extra information about phone numbers (plus multiple phone numbers)

2. Can employees have multiple phones?

   (if yes, then (b), or (a) with multivalued attributes)

3. Else

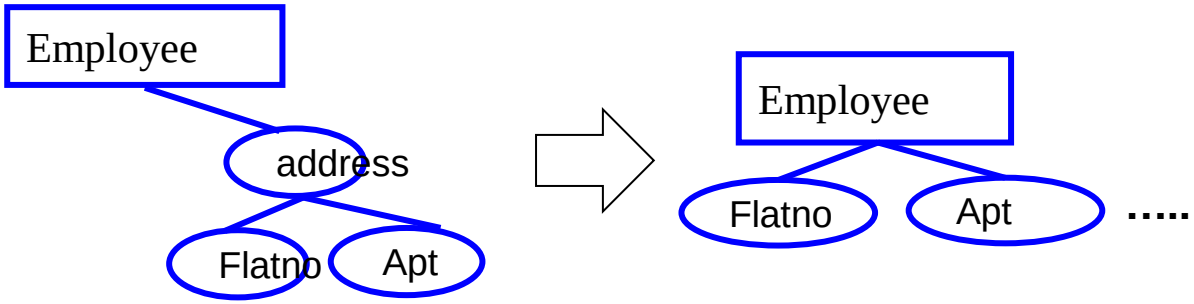
   (a), perhaps with composite attributes



ER-Model

Design Issue: Composite Attribute

- Example: Employee phone having std code + no
- Address having Flat no, Apt, street, road



Solution

Composite attributes are flattened out by creating a separate attribute for each component attribute

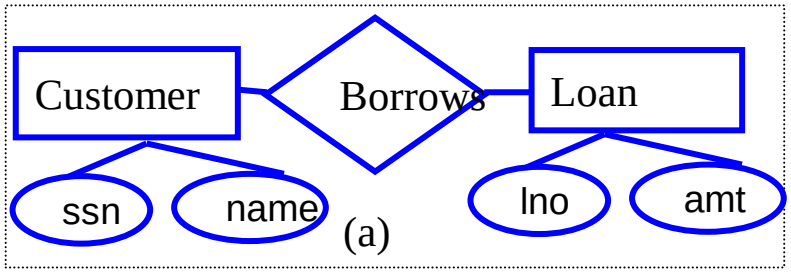
Example: The entity set has four attributes addr_FlatNo, addr_Aprt, addr_Street, and addr_road

Prefix omitted if there is no ambiguity

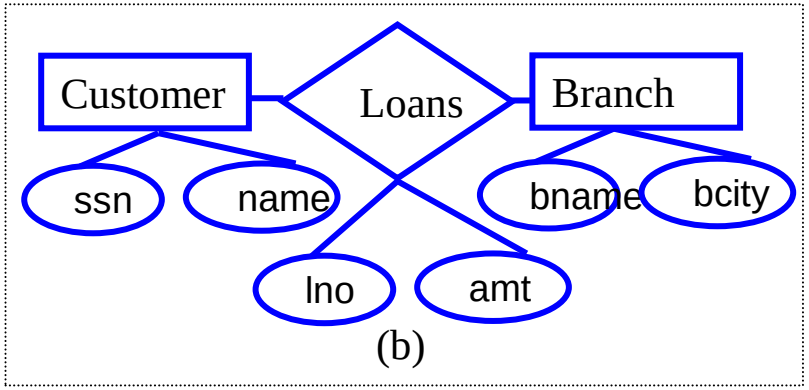
ER-Model

Design Issue: Entity Sets vs. Relationship Sets

An Example: How to model bank loans



vs



To resolve, determine how loans are issued

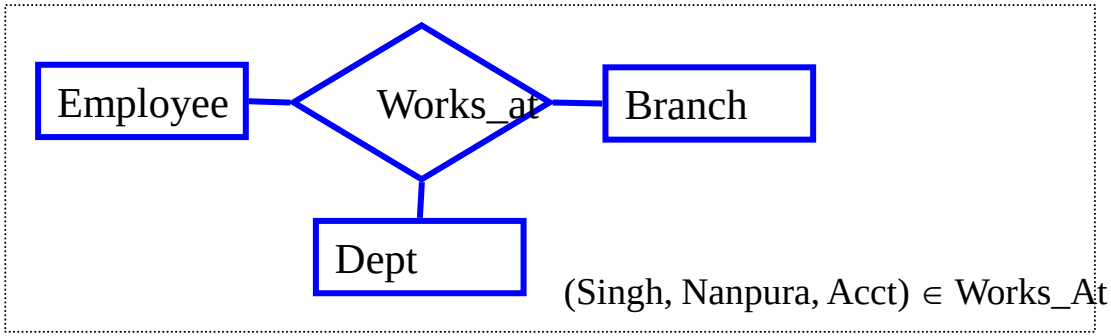
1. Can there be more than one customer per loan?
   - If yes, then (a). Otherwise, for joint acct loan info must be replicated for each customer (wasteful, potential update anomalies)
2. Is loan a noun or a verb?
   - Both, but more of a noun to a bank. (hence (a) probably more appropriate)

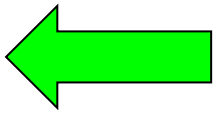
ER-Model

Design Issue: N-ary vs Binary Relationship Sets

An Example: Works_At

Ternary:



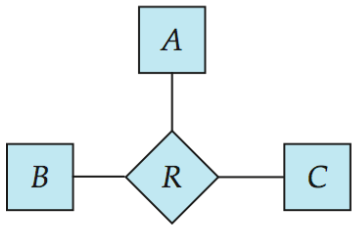


Choose n-ary when possible! (Avoids redundancy, update anomalies)

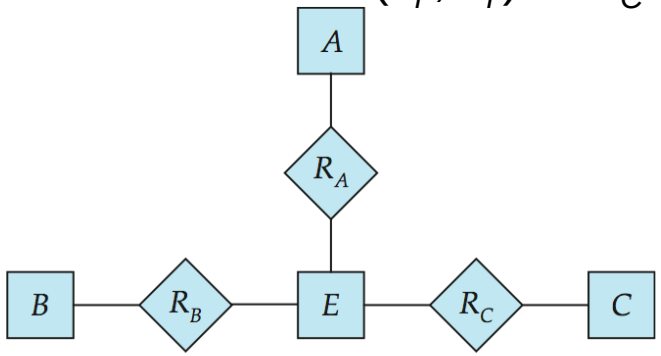
ER-Model

Design Issue: N-ary vs Binary Relationship Sets

- In general, any non-binary relationship can be represented using binary relationships by creating an artificial entity set
  - Replace *R* between entity sets A, B and C by an entity set *E*, and three relationship sets:
    1. R_A, relating *E* and *A*
    2. R_B, relating *E* and *B*
    3. R_C, relating *E* and *C*
  - Create a special identifying attribute for *E*
  - Add any attributes of *R* to *E*
  - For each relationship (a_i , b_i , c_i) in *R*, create
    1. a new entity e_i in the entity set *E*
    2. add (e_i , a_i) to R_A
    3. add (e_i , b_i) to R_B
    4. add (e_i , c_i) to R_C



(a)



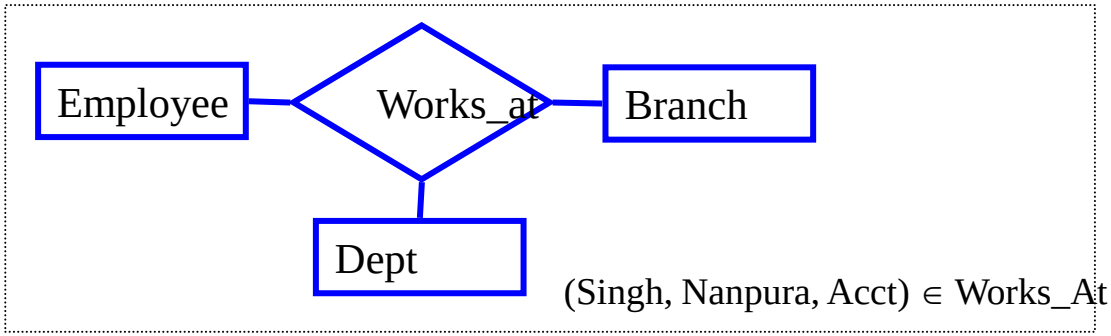
(b)

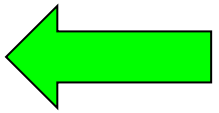
ER-Model

Design Issue: N-ary vs Binary Relationship Sets

An Example: Works_At

Ternary:

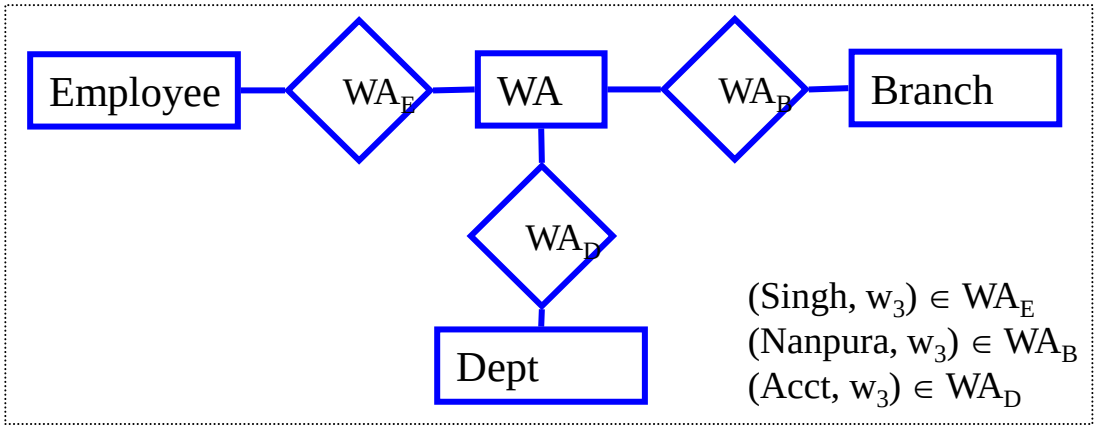




Choose n-ary when possible! (Avoids redundancy, update anomalies)

vs

Binary:



ER-Model

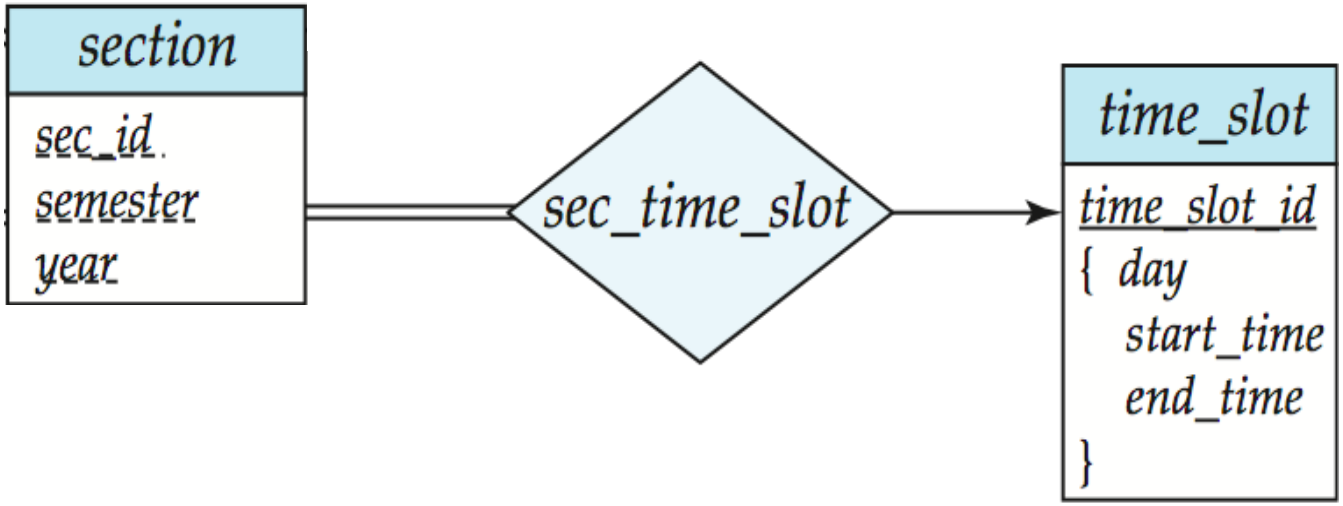
Design Issue: N-ary vs Binary Relationship Sets

- Also need to translate constraints
  - Translating all constraints may not be possible
  - There may be instances in the translated schema that cannot correspond to any instance of *R*
    - Exercise: *add constraints to the relationships* R_A, R_B *and* R_C to ensure that a newly created entity corresponds to exactly one entity in each of entity sets *A, B* and *C*
  - We can avoid creating an identifying attribute by making E a weak entity set identified by the three relationship sets

ER-Model

Design Issue: Multi valued attribute but Primary Key

- Special case: entity *time_slot* has only one attribute other than the primary-key attribute, and that attribute is multivalued
  - Optimization: Don't create the relation corresponding to the entity, just create the one corresponding to the multivalued attribute
  - *time_slot* (*time_slot_id, day, start_time, end_time*)
  - caution: *time_slot* attribute of *section (*from *sec_time_slot*) cannot be a foreign key due to this optimization



ER-Model

Design Issue: Specialization

- Method 1:
  - Form a schema for the higher-level entity
  - Form a schema for each lower-level entity set, include primary key of higher-level entity set and local attributes

schema	attributes
person	*ID, name, street, city*
student	*ID, tot_cred*
employee	*ID, salary*

  - Drawback: getting information about, an *employee* requires accessing two relations, the one corresponding to the low-level schema and the one corresponding to the high-level schema

ER-Model

Design Issue: Specialization

- Method 2:
  - Form a schema for each entity set with all local and inherited attributes

schema	attributes
person	*ID, name, street, city*
student	*ID, name, street, city, tot_cred*
employee	*ID, name, street, city, salary*

  - If specialization is total, the schema for the generalized entity set (*person*) not required to store information
    - Can be defined as a “view” relation containing union of specialization relations
    - But explicit schema may still be needed for foreign key constraints
  - Drawback: *name, street* and *city* may be stored redundantly for people who are both students and employees

ER-Model

Design Issue: Aggregation

- To represent aggregation, create a schema containing
  - primary key of the aggregated relationship,
  - the primary key of the associated entity set
  - any descriptive attributes

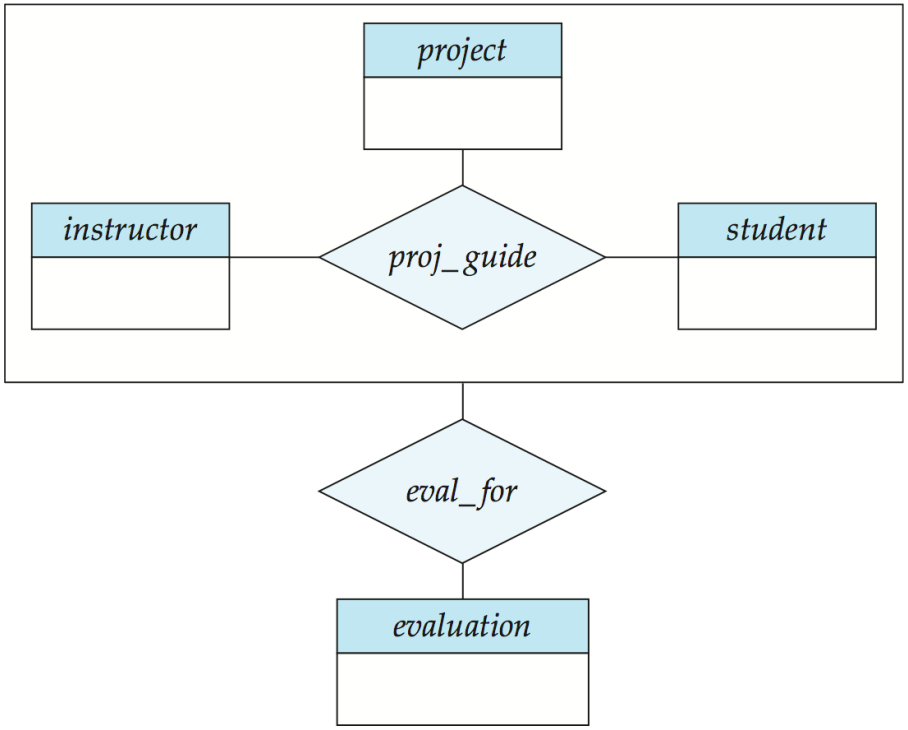
ER-Model

Design Issue: Aggregation

- For example, to represent aggregation manages between relationship works_on and entity set manager, create a schema

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

- Schema *proj_guide* is redundant provided we are willing to store null values for attribute *manager_name* in relation on schema *manages*



E-R Design Decisions

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

END of Chapter

Ch 6 Relational Algebra