Database Design Using the E-R Model

CMPT 354

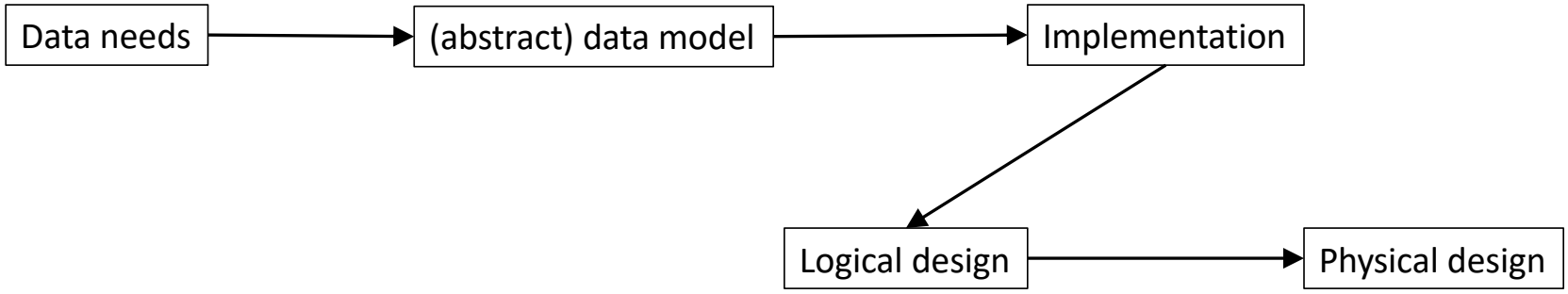
Jian Pei

jpei@cs.sfu.ca

Outline

- Overview of the Design Process
- The Entity-Relationship Model
- Complex Attributes
- Mapping Cardinalities
- Primary Key
- Removing Redundant Attributes in Entity Sets
- Reducing ER Diagrams to Relational Schemas
- Extended E-R Features
- Entity-Relationship Design Issues
- Alternative Notations for Modeling Data
- Other Aspects of Database Design

Database Design



Design Phases

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

Design Phases

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

Design Alternatives

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

Design Approaches

- Entity Relationship Model (covered in this section)
  - Model 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 (to be discussed later)
  - Formalize what designs are bad, and test for them

ER model – Database Modeling

- The ER data model was developed to facilitate database design by allowing specification of an enterprise schema that represents the overall logical structure of a database
- The ER data model employs three basic concepts
  - Entity sets
  - Relationship sets
  - Attributes
- ER diagram: the diagrammatic representation associated with the ER model, which can express the overall logical structure of a database graphically

Entity Sets

- An entity is an object that exists and is distinguishable from other objects
  - Example: specific person, company, event, plant
- An entity set is a set of entities of the same type that share the same properties
  - Example: set of all persons, companies, trees, holidays
- An entity is represented by a set of attributes – 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 uniquely identifying each member of the set

Entity Sets – Instructor and Student

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

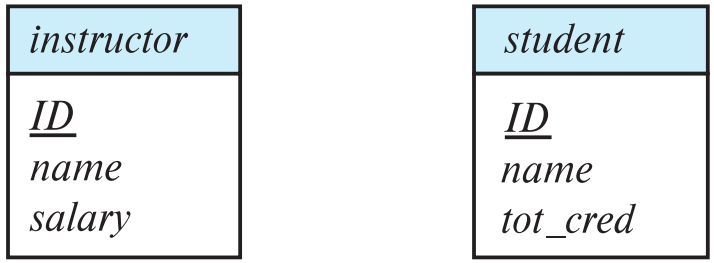
instructor

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

student

Representing Entity sets in ER Diagram

- Entity sets can be represented graphically
  - Rectangles represent entity sets
  - Attributes listed inside entity rectangle
  - Underline indicates primary key attributes



Relationship Sets

- A relationship is an association among several entities

  Example:

  44553 (Peltier) — *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

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

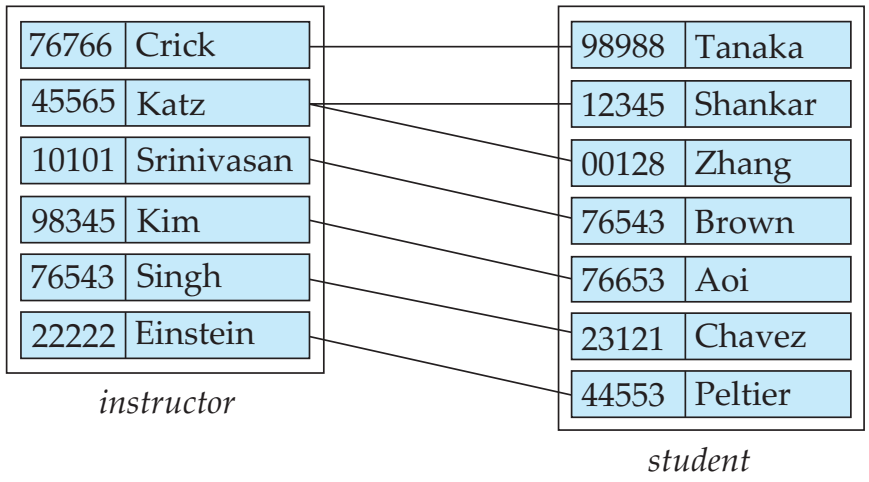
  where $(e_1, e_2, \ldots, e_n)$ is a relationship

  - Example

    $(44553, 22222) \in advisor$

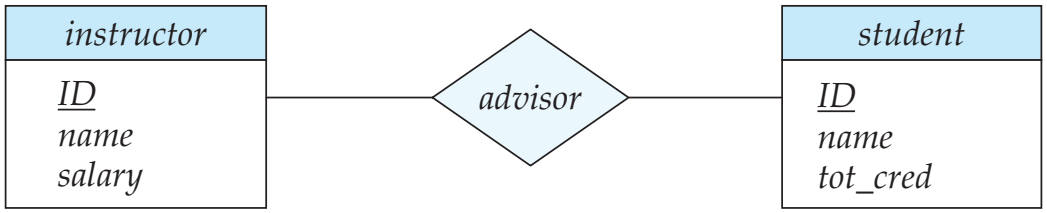
Relationship Sets

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



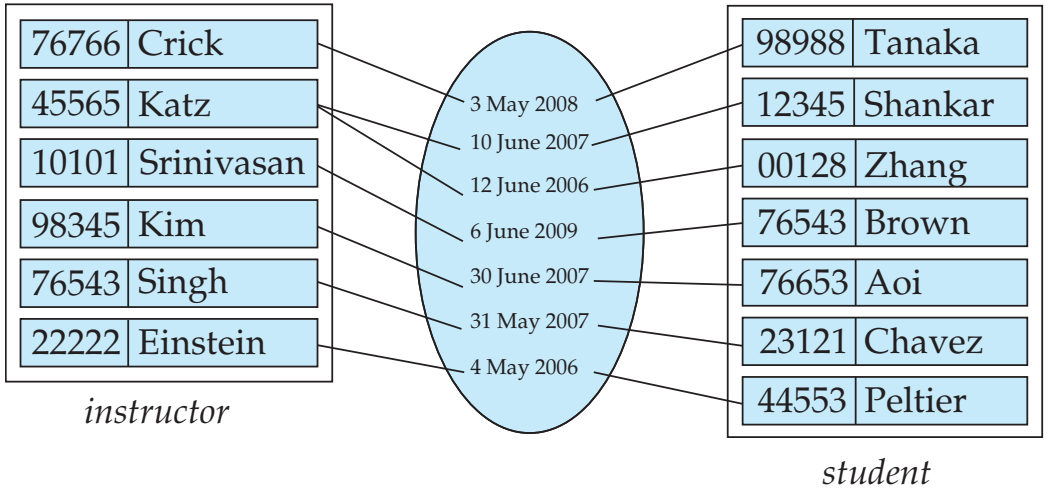
Representing Relationship Sets via ER Diagrams

- Diamonds represent relationship sets

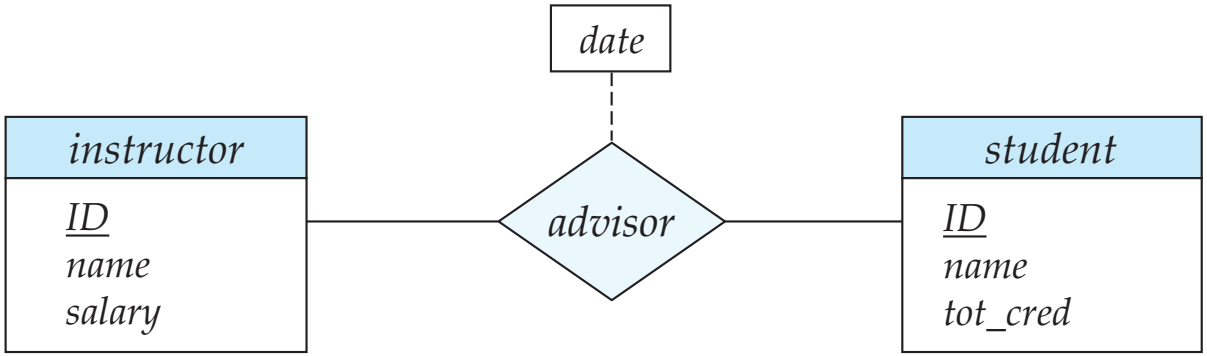


Relationship Sets

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

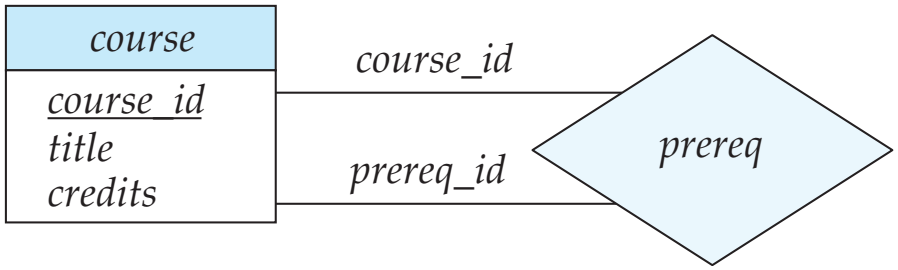


Relationship Sets with Attributes



Roles

- Entity sets of a relationship need not be distinct
  - Each occurrence of an entity set plays a “role” in the relationship
- The labels “*course_id*” and “*prereq_id*” are called **roles**

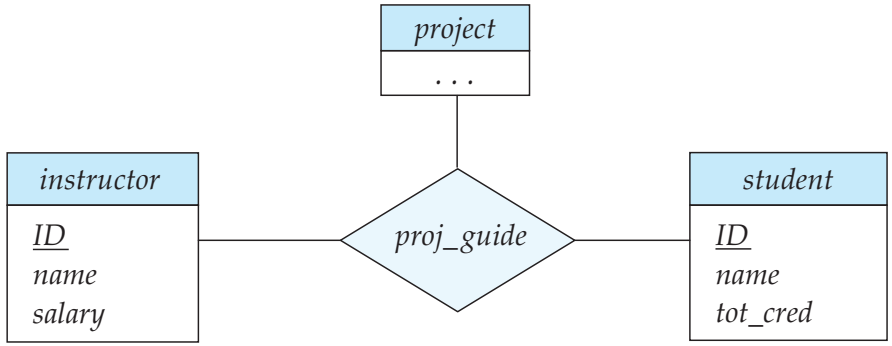


Degree of a Relationship Set

- Binary relationship
  - Involve two entity sets (or degree two)
  - Most relationship sets in a database system are binary
- Relationships between more than two entity sets are rare – most relationships are binary
  - Example: *students* work on research *projects* under the guidance of an *instructor*
  - relationship *proj_guide* is a ternary relationship between *instructor, student,* and *project*

Non-binary Relationship Sets

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



To-Do List

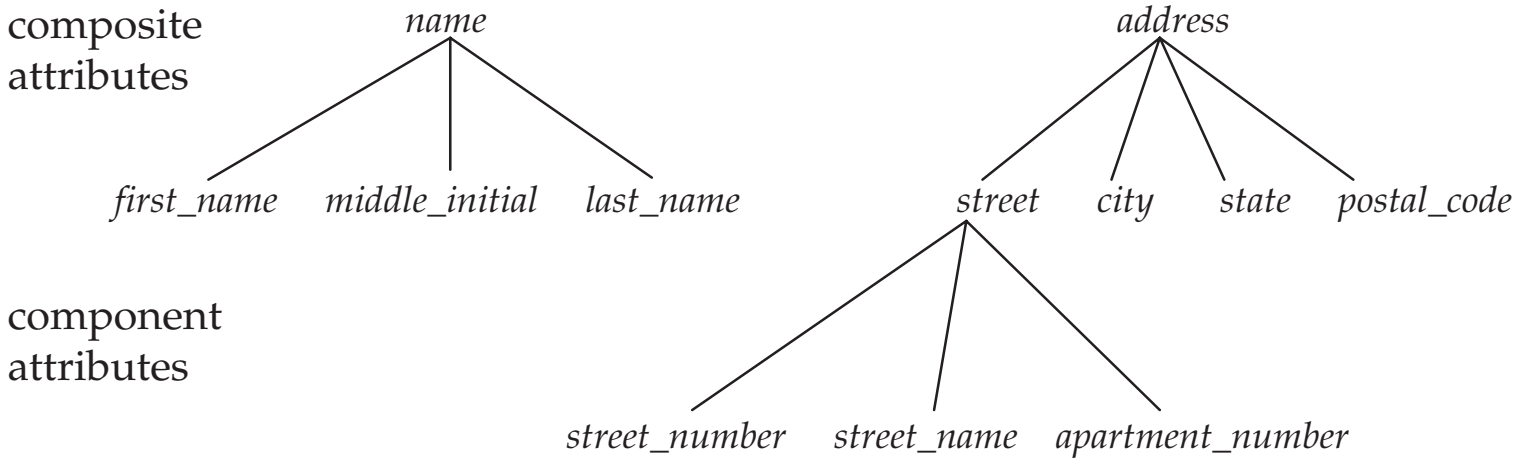
- Using ER diagram, describe entity sets *customer* and *product*, and relationship set *purchases*
- Using ER diagram, describe entity sets *employee* and *company*, and relationship set *works*
- Can you give an example where there are only entity sets but no relationship sets? Is it a good design?
- Can you give an example where there are only relationship sets but no entity sets? Is it a good design?
- Can you give an example where a relationship set has degree 1?

Complex Attributes

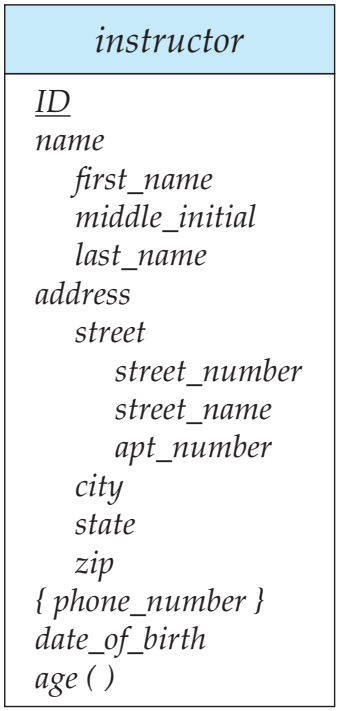
- Attribute types
  - Simple and composite attributes
  - Single-valued and multivalued attributes
    - Example: multivalued attribute: *phone_numbers*
  - Derived attributes
    - Can be computed from other attributes
    - Example: age, given date_of_birth
- Domain – the set of permitted values for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

Composite Attributes

- Composite attributes allow us to divide attributes into subparts (other attributes)



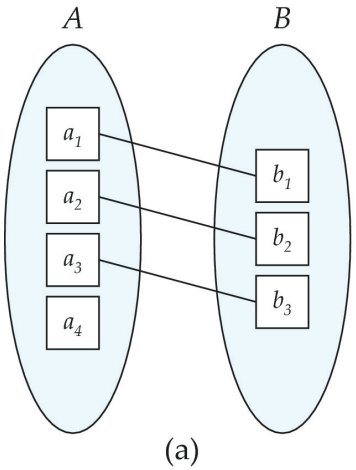
Representing Complex Attributes in ER Diagram



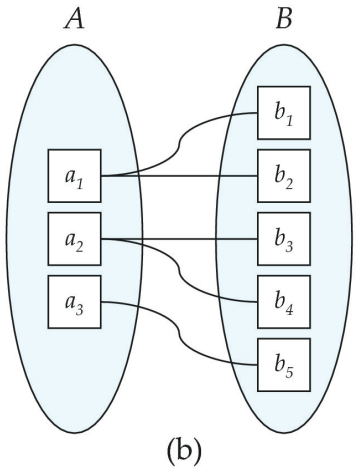
Mapping Cardinality Constraints

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

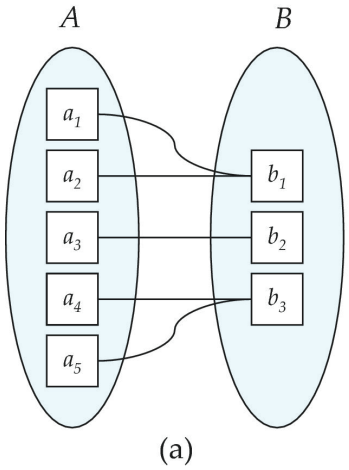
Mapping Cardinalities



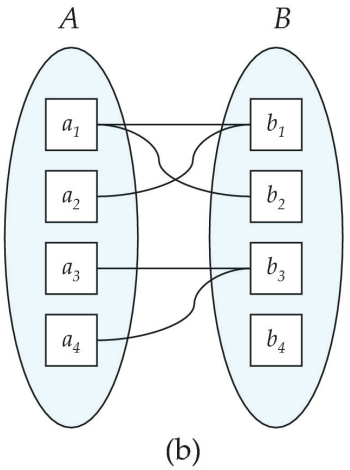
One to one



One to many



Many to one

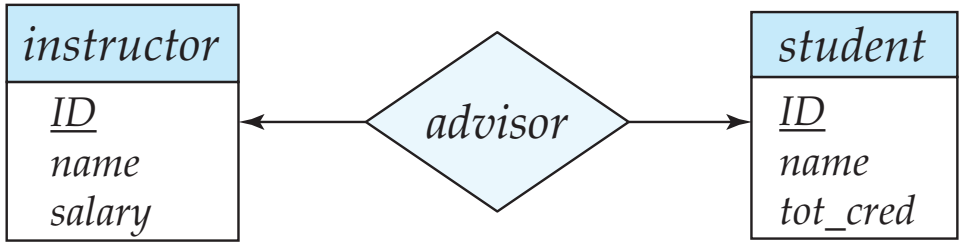


Many to many

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

Representing Cardinality Constraints in ER Diagram

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



One-to-Many Relationship

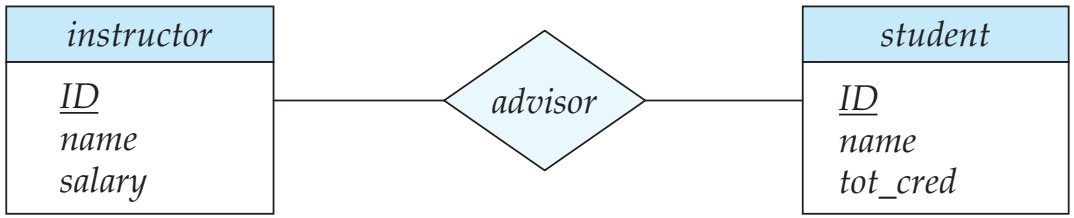
- one-to-many relationship between an *instructor* and a *student*
  - An instructor is associated with several (including 0) students via *advisor*
  - A student is associated with at most one instructor via advisor

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

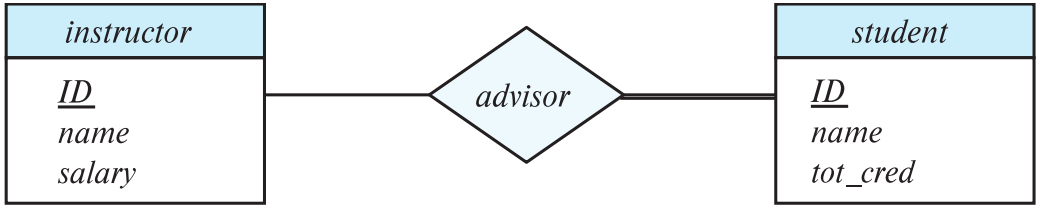
Many-to-Many Relationship

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



Total and Partial Participation

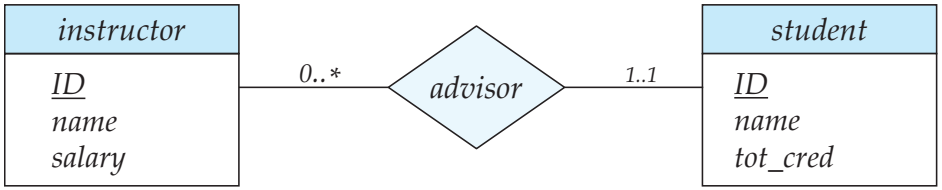
- Total participation (indicated by double line): every entity in the entity set participates in at least one relationship in the relationship set



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

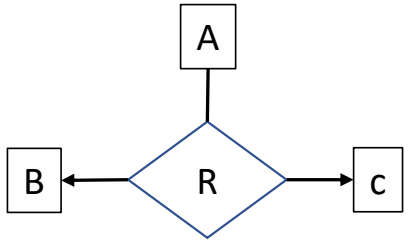
Notation for Expressing More Complex Constraints

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



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

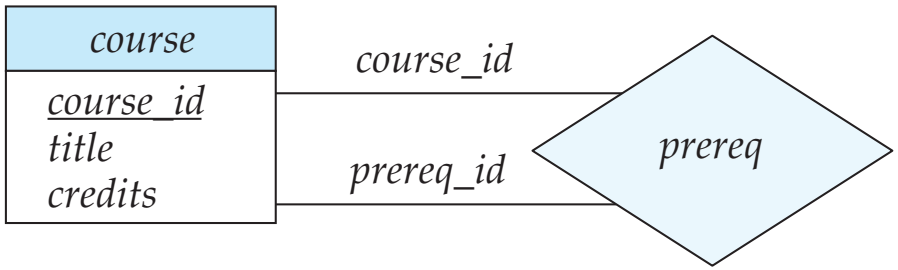
Cardinality Constraints on Ternary Relationship



- We allow at most one arrow out of a ternary (or greater degree) relationship to indicate a cardinality constraint
- For example, an arrow from *proj_guide* to *instructor* indicates each student has at most one guide for a project
- Why only one arrow? If there is more than one arrow, there are two ways of defining the meaning
  - For example, a ternary relationship *R* between *A*, *B* and *C* with arrows to *B* and *C* could mean
    1. Each *A* entity is associated with a unique entity from B and *C* or
    2. Each pair of entities from (*A, B*) is associated with a unique *C* entity, and each pair (*A, C*) is associated with a unique *B*
  - Each alternative has been used in different formalisms
  - To avoid confusion, we outlaw more than one arrow

To-Do List

- Can roles also have cardinality/partition constraints?



Primary Key

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

Primary key for Entity Sets

- By definition, individual entities are distinct
- From database perspective, the differences among individual entities must be expressed in terms of their attributes
- The values of the attribute values of an entity must uniquely identify the entity
  - No two entities in an entity set are allowed to have exactly the same value for all attributes
- A key for an entity is a set of attributes that suffice to distinguish entities from each other

To-Do List

- Can you come up with an application example where an entity set does not have a key?
- In practice, for an entity set, such as employees, you may include some more attributes, which may not be useful for the application, so that the entities are distinguishable, or include less attributes and synthesize an ID attribute. What are the pros and cons of the two approaches?
- When you design the key for an entity set in an application, such as the students in a university, if a subset of attributes distinguishes all individual entities collected in the current application, such as {firstname, lastname} distinguishes all current students, is it valid in design to use the subset of attributes as a key?

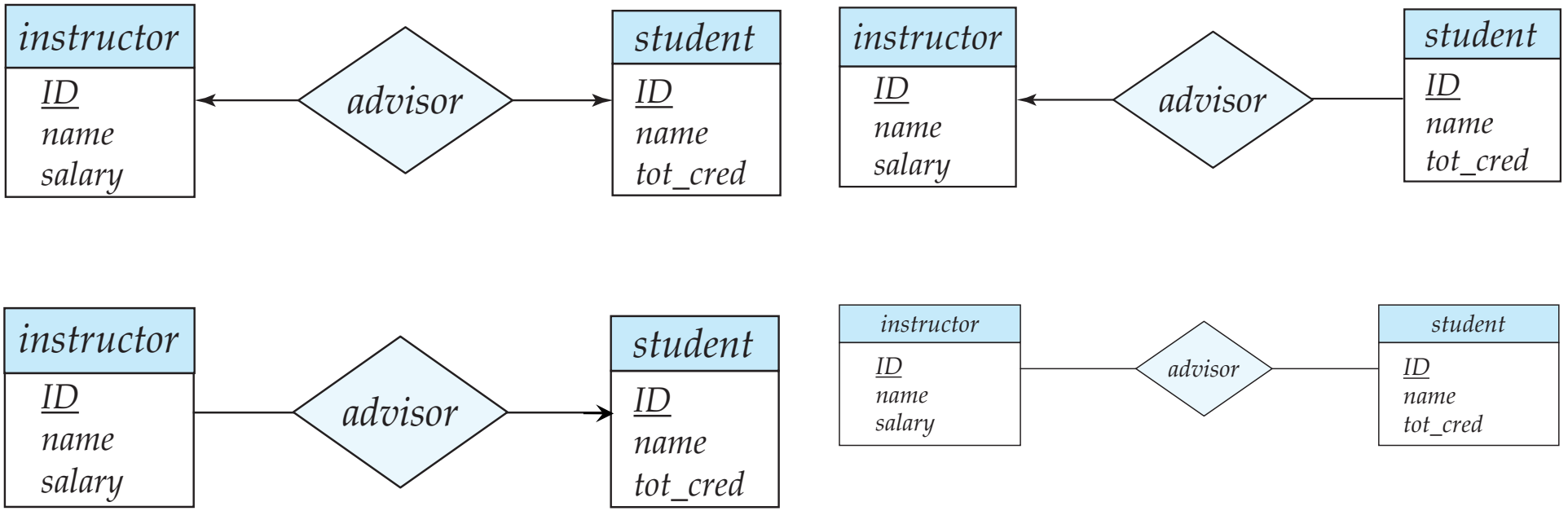
Primary Key for Relationship Sets

- To distinguish among the various relationships of a relationship set we use the individual primary keys of the entities in the relationship set
  - Let *R* be a relationship set involving entity sets E_1, E_2, …, E_n
  - The primary key for R is consists of the union of the primary keys of entity sets E_1, E_2, …, E_n
  - If the relationship set *R* has attributes a_1, a_2, …, a_m associated with it, then the primary key of *R* also includes the attributes a_1, a_2, …, a_m
- Example: relationship set "advisor"
  - The primary key consists of *instructor.ID* and s*tudent.ID*
- The choice of the primary key for a relationship set depends on the mapping cardinality of the relationship set

Choice of Primary key for Binary Relationship

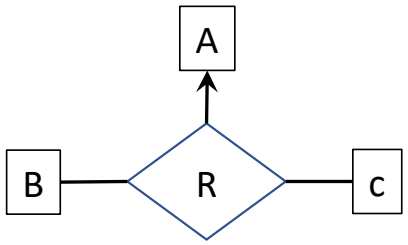
- Many-to-Many relationships: the preceding union of the primary keys is a minimal superkey and is chosen as the primary key
- One-to-Many relationships: the primary key of the “Many” side is a minimal superkey and is used as the primary key
- Many-to-one relationships: the primary key of the “Many” side is a minimal superkey and is used as the primary key
- One-to-one relationships: the primary key of either one of the participating entity sets forms a minimal superkey, and either one can be chosen as the primary key

Examples


instructor
ID
name
salary
advisor
student
ID
name
tot_cred
instructor
ID
name
salary
advisor
student
ID
name
tot_cred
instructor
ID
name
salary
advisor
student
ID
name
tot_cred
instructor
ID
name
salary
advisor
student
ID
name
tot_cred

To-Do List

- Why don't we simply use the union of the individual primary keys of the entities in the relationship set as the primary key for one-to-one, one-to-many, and many-to-one relationship sets?
  - Is the union of the individual primary keys of the entities in the relationship set a superkey?
  - Is the union a candidate key?
- How to choose the primary key for a ternary relationship set with a cardinality constraint?



Weak Entity Sets

- Consider a *section* entity, which is uniquely identified by a *course_id*, *semester, year*, and *sec_id*
  - Clearly, section entities are related to course entities
- Suppose we create a relationship set *sec_course* between entity sets *section* and *course*
  - Note that the information in *sec_course* is redundant, since *section* already has an attribute *course_id*, which identifies the course with which the section is related
- One option to deal with this redundancy is to get rid of the relationship s*ec_course*; however, by doing so the relationship between *section* and *course* becomes implicit in an attribute, which is not desirable

Weak Entity Sets

- An alternative way to deal with this redundancy is to not store the attribute *course_id* in the *section* entity and to only store the remaining attributes *section_id*, *year*, and *semester*
  - However, the entity set *section* then does not have enough attributes to identify a particular *section* entity uniquely
- To deal with this problem, we treat the relationship *sec_course* as a special relationship that provides extra information, in this case, the *course_id*, required to identify *section* entities uniquely
- A weak entity set is one whose existence is dependent on another entity, called its identifying entity
- Instead of associating a primary key with a weak entity, we use the identifying entity, along with extra attributes called discriminator to uniquely identify a weak entity

Weak Entity Sets

- An entity set that is not a weak entity set is called a strong entity set
- Every weak entity must be associated with an identifying entity – a weak entity set is existence dependent on the identifying entity set
- The identifying entity set is said to own the weak entity set that it identifies
- The relationship associating the weak entity set with the identifying entity set is called the identifying relationship
- Note that the relational schema we eventually create from the entity set *section* **does** have the attribute *course_id*, for reasons that will become clear later, even though we have dropped the attribute *course_id* from the entity set *section*

Expressing Weak Entity Sets

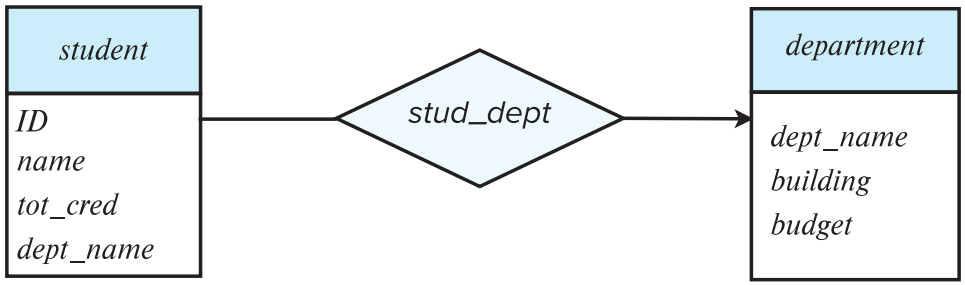
- In E-R diagrams, a weak entity set is depicted via a double rectangle
- We underline the discriminator of a weak entity set with a dashed line
- The relationship set connecting the weak entity set to the identifying strong entity set is depicted by a double diamond
- Primary key for *section* – (*course_id,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

To Do List

- Can a strong entity set own more than one weak entity set?
  - If so, can you give an example? Can those two or multiple weak entity sets be merged into one?
  - If not, why?
- Can more than one relationship set exist between two entity sets?

Redundant Attributes

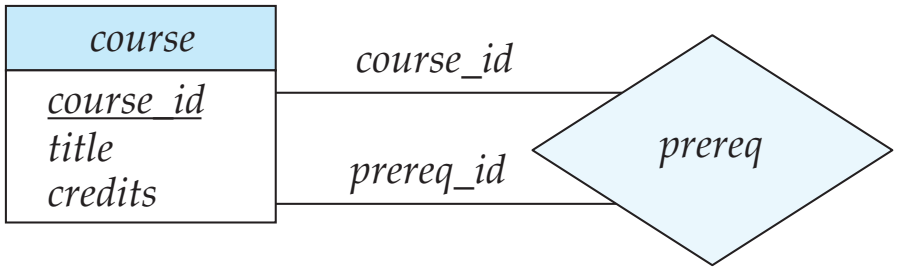
- Suppose we have entity sets:
  - *student*, with attributes: *ID*, *name, tot_cred*, *dept_name*
  - *department,* with attributes: *dept_name*, *building, budget*
- We model the fact that each student has an associated department using a relationship set *stud_dept*
- The attribute *dept_name* in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



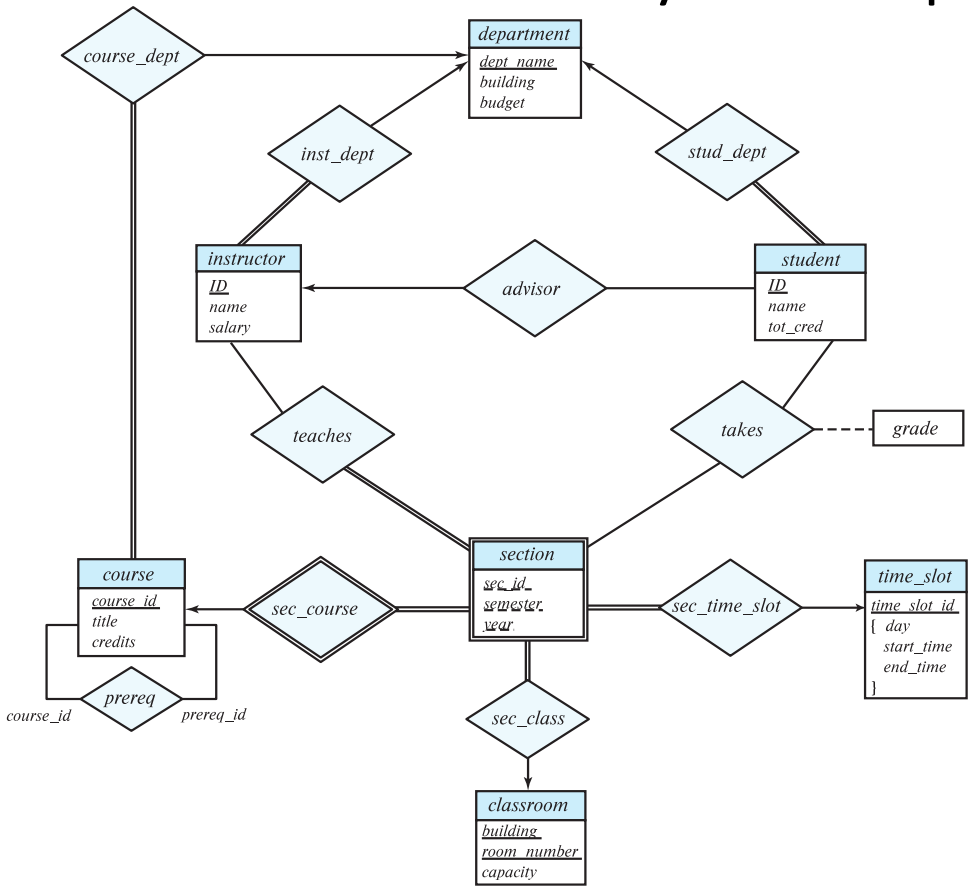
(a) Incorrect use of attribute

To-Do List

- Should every one-to-many relationship set be represented as a weak entity set?
- Can a weak entity set be many-to-many with respect to the identifying strong entity set?
- Can an entity set itself be made a weak entity set through roles?



E-R Diagram for a University Enterprise

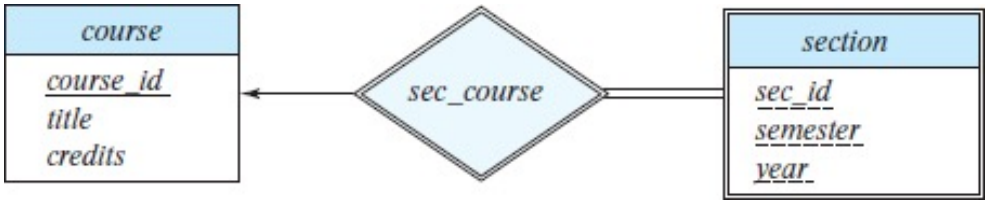


Reduction to Relation Schemas

- Entity sets and relationship sets can be expressed uniformly as *relation schemas* that represent the contents of the database
- A database which conforms to an E-R diagram can be represented by a collection of 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

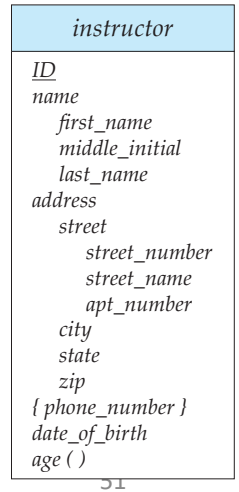
Representing Entity Sets

- A strong entity set reduces to a schema with the same attributes
  student(ID, 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 (course_id, sec_id, sem, year)



Representation of Entity Sets with Composite Attributes

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



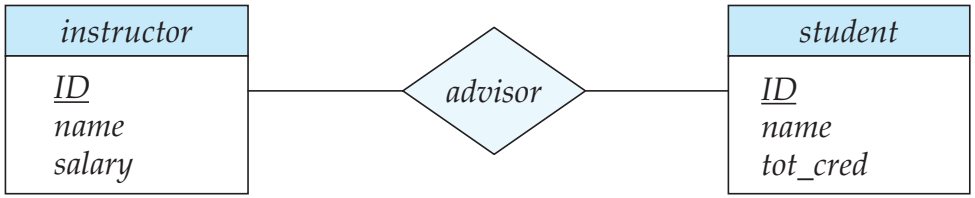
Representation of Entity Sets with Multivalued Attributes

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

Representing Relationship Sets

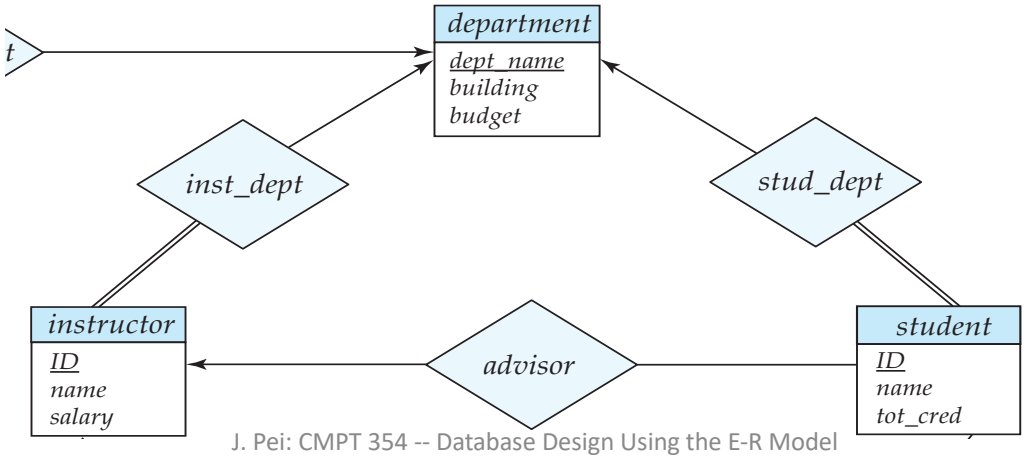
- A many-to-many relationship set is represented as a schema with attributes for the primary keys of the two participating entity sets, and any descriptive attributes of the relationship set
- Example: schema for relationship set *advisor*

  advisor = (*s_id*, *i_id*)



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*



Redundancy of Schemas

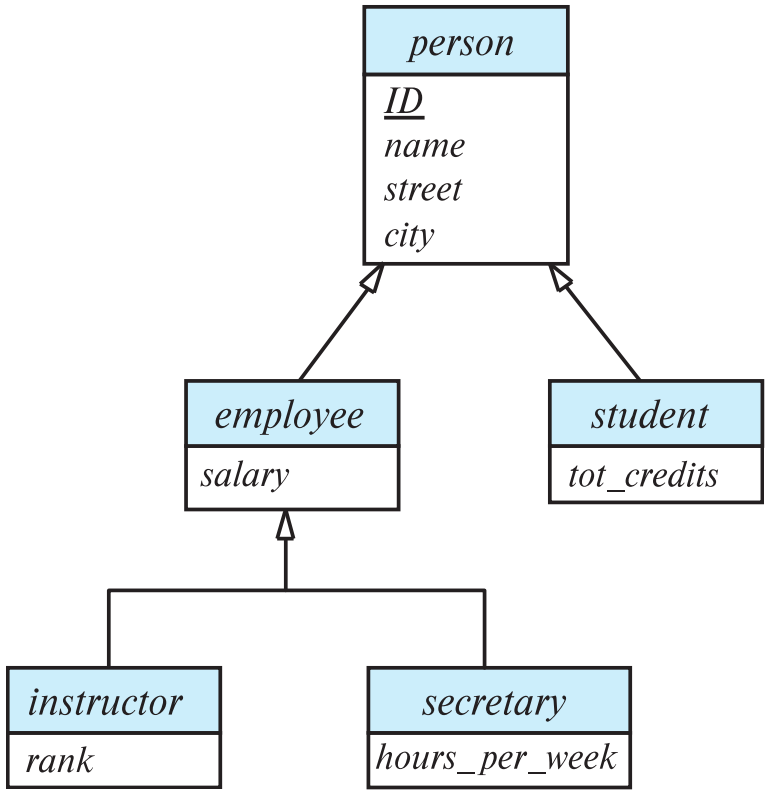
- For one-to-one relationship sets, either side can be chosen to act as the "many" side
  - That is, an extra attribute can be added to either of the tables corresponding to the two entity sets
- If participation is partial on the "many" side, replacing a schema by an extra attribute in the schema corresponding to the "many" side could result in null values
- 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

Specialization

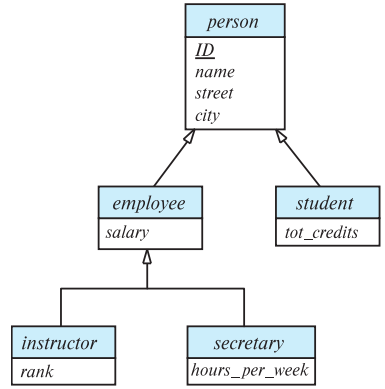
- A top-down design process: we designate sub-groupings within an entity set that are distinctive from other entities in the set
- These sub-groupings become lower-level entity sets that have attributes or participate in relationships that do not apply to the higher-level entity set
- Depicted by a triangle component labeled ISA (e.g., instructor “is a” person)
- Attribute inheritance – a lower-level entity set inherits all the attributes and relationship participation of the higher-level entity set to which it is linked

Specialization Example

- Overlapping – *employee* and *student*
- Disjoint – *instructor* and secretary
- Total and partial



Representing Specialization via Schemas



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

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

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

Representing Specialization as Schemas

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

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

- Drawback: name, street and city may be stored redundantly for people who are both students and employees

Generalization

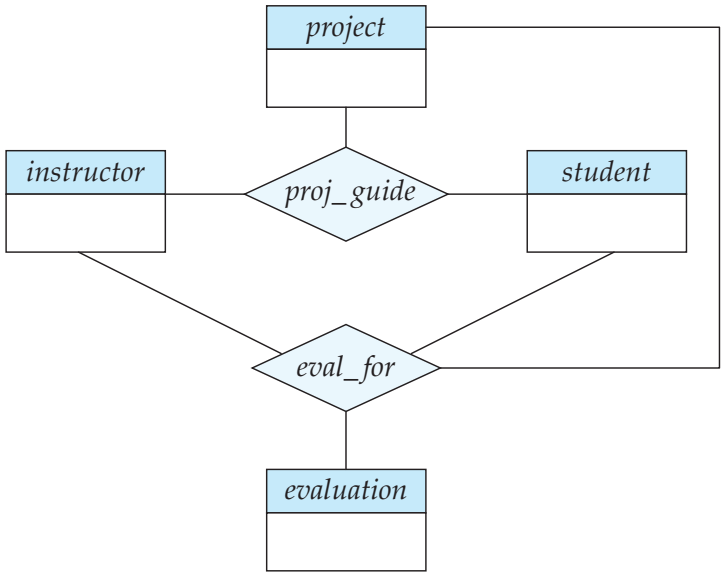
- A bottom-up design process – combine a number of entity sets that share the same features into a higher-level entity set
- Specialization and generalization are simple inversions of each other, and are represented in an E-R diagram in the same way
- The terms specialization and generalization are used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

Completeness Constraints

- A completeness constraint specifies whether an entity in the higher-level entity set must belong to at least one of the lower-level entity sets within a generalization
  - Total: an entity must belong to one of the lower-level entity sets
  - Partial: an entity need not belong to one of the lower-level entity sets
- Partial generalization is the default
- We can specify total generalization in an ER diagram by adding the keyword total in the diagram and drawing a dashed line from the keyword to the corresponding hollow arrow-head to which it applies (for a total generalization), or to the set of hollow arrow-heads to which it applies (for an overlapping generalization)
- The student generalization is total: all student entities must be either graduate or undergraduate. Because the higher-level entity set arrived at through generalization is generally composed of only those entities in the lower-level entity sets, the completeness constraint for a generalized higher-level entity set is usually total

Aggregation

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

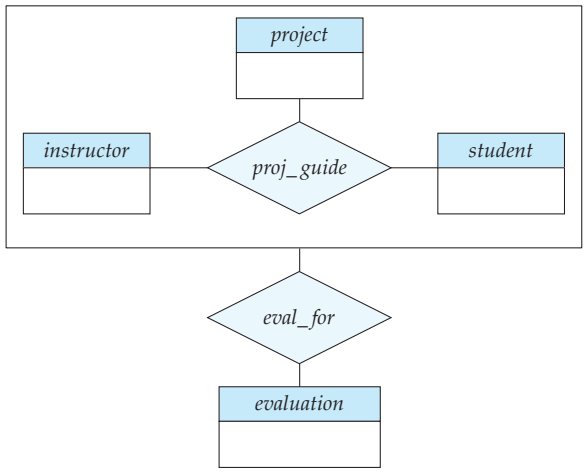


Aggregation

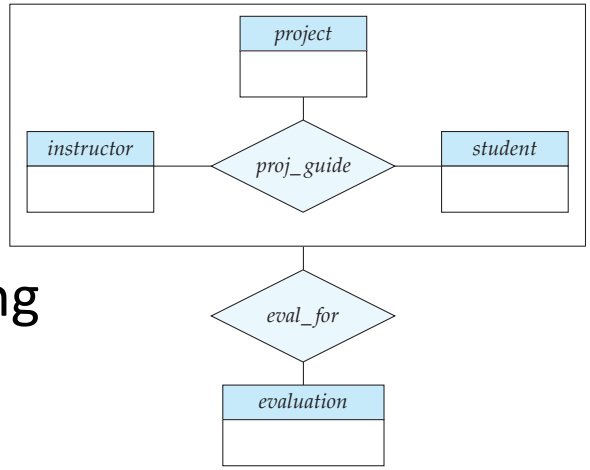
- Relationship sets *eval_for* and *proj_guide* represent overlapping information
  - Every *eval_for* relationship corresponds to a *proj_guide* relationship
  - However, some *proj_guide* relationships may not correspond to any *eval_for* relationships
    - So, we cannot discard the *proj_guide* relationship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

Aggregation

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

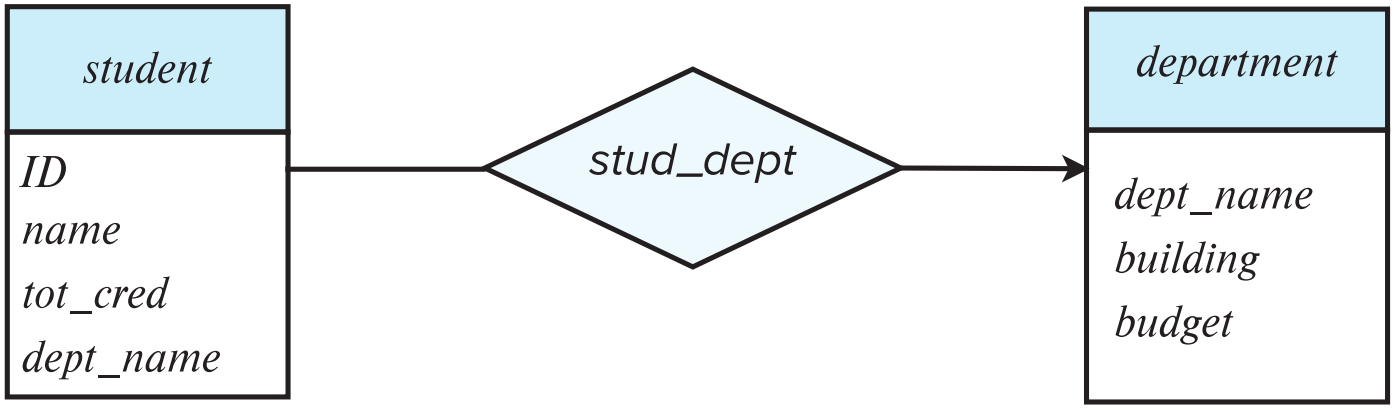


Reduction to Relational Schemas

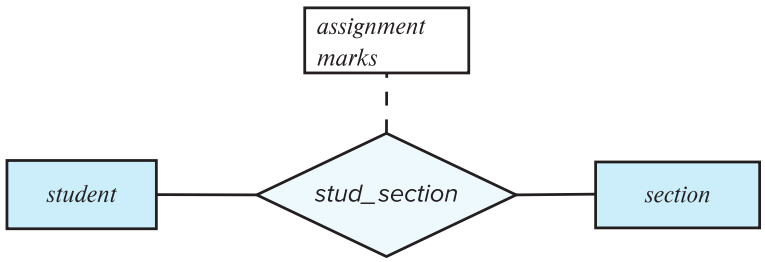


- To represent aggregation, create a schema containing
  - Primary key of the aggregated relationship
  - The primary key of the associated entity set
  - Any descriptive attributes
- In our example:
  - The schema *eval_for* is *eval_for* (*s_ID*, *project_id*, *i_ID*, *evaluation_id*)
  - The schema *proj_guide* is redundant

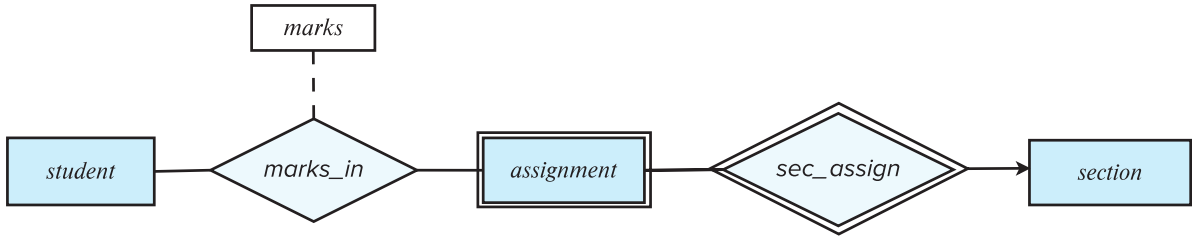
Common Mistakes in E-R Diagrams



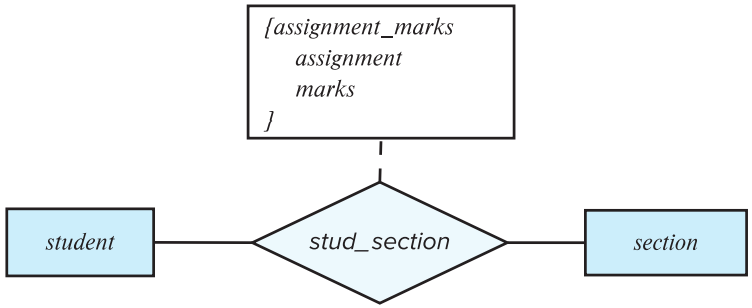
Common Mistakes in E-R Diagrams



(b) Erroneous use of relationship attributes



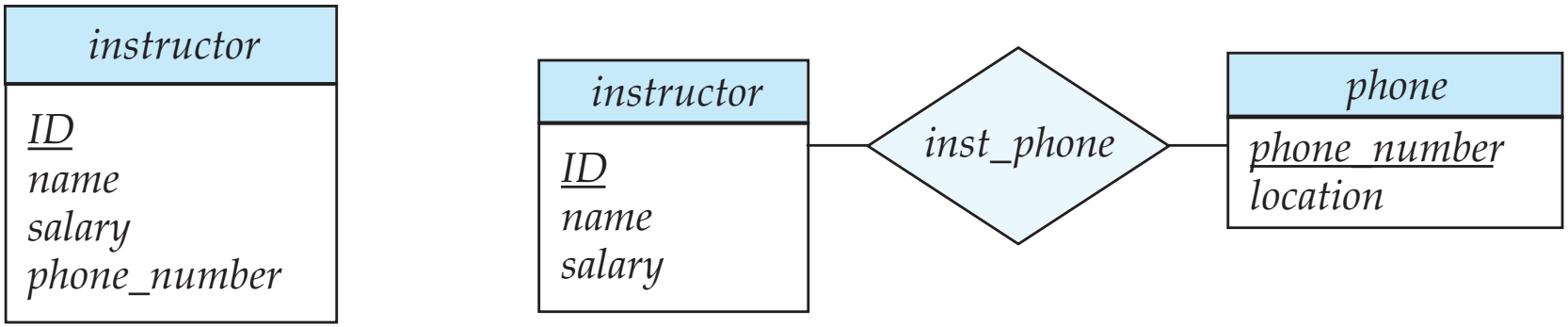
(c) Correct alternative to erroneous E-R diagram (b)



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

Entities vs. Attributes

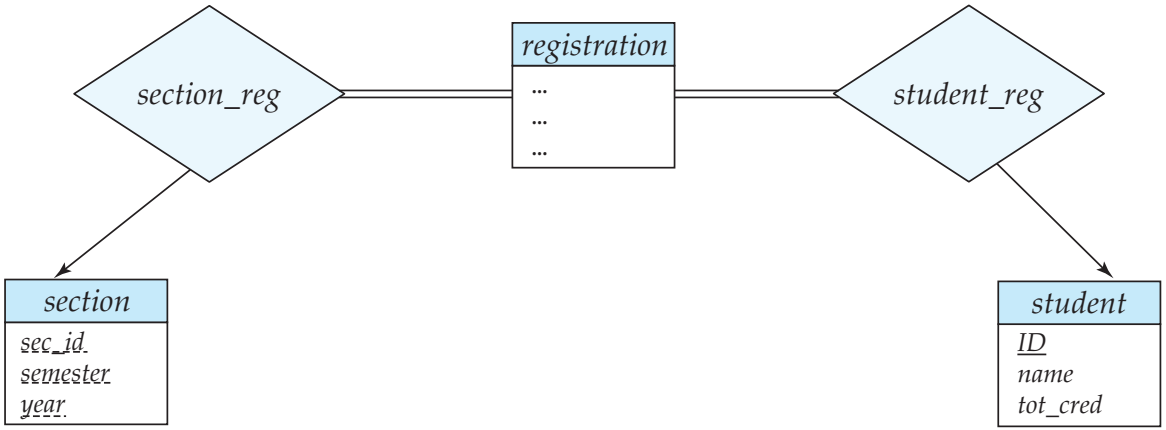
- Use of entity sets vs. attributes



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

Entities vs. Relationship sets

- Use of entity sets vs. relationship sets: a possible guideline is to designate a relationship set to describe an action that occurs between entities



- Placement of relationship attributes: for example, attribute *date* as attribute of *advisor* or as attribute of *student*

Binary Vs. Non-Binary Relationships

- Although it is possible to replace any non-binary (*n*-ary, for $n > 2$) relationship set by a number of distinct binary relationship sets, an *n*-ary relationship set shows more clearly that several entities participate in a single relationship
- Some relationships that appear to be non-binary may be better represented using binary relationships
  - For example, a ternary relationship *parents*, relating a child to his/her father and mother, is best replaced by two binary relationships, *father* and *mother*
    - Using two binary relationships allows partial information (e.g., only mother being known)
  - But there are some relationships that are naturally non-binary
    - Example: *proj_guide*

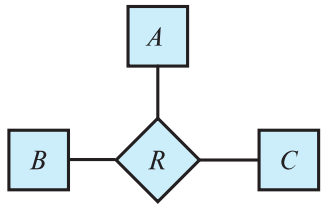
Converting Non-Binary Relationships to Binary Form

- In general, any non-binary relationship can be represented using binary relationships by creating an artificial entity set.
  - Replace *R* between entity sets A, B and C by an entity set *E*, and three relationship sets:

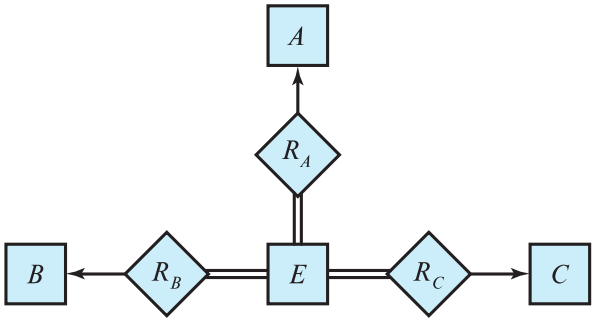
    1. R_A, relating *E* and *A* 2. R_B, relating *E* and *B*
    3. R_C, relating *E* and *C*
  - Create an identifying attribute for *E and* add any attributes of *R* to *E*
  - For each relationship (a_i , b_i , c_i) in *R,* create

1. a new entity e_i in the entity set *E* 2. add (e_i , a_i) to R_A

3. add (e_i , b_i) to R_B 4. add (e_i , c_i) to R_C



(a)



(b)

Summary

- Overview of the Design Process
- The Entity-Relationship Model
- Complex Attributes
- Mapping Cardinalities
- Primary Key
- Removing Redundant Attributes in Entity Sets
- Reducing ER Diagrams to Relational Schemas
- Extended E-R Features
- Entity-Relationship Design Issues
- Alternative Notations for Modeling Data
- Other Aspects of Database Design