ELSEWEDY UNIVERSITY OF TECHNOLOGY
- POLYTECHNIC OF EGYPT -

Database Management Systems

Lecture 5: Database Design Using the E-R Model

© Spring 2023 – **Dr. Alyaa Hamza**
alyaa.hamza@sut.edu.eg

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
- Extended E-R Features
- Entity-Relationship Design Issues
- Alternative Notations for Modeling Data
- Other Aspects of 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 Phases

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

Design Phases

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

Outline of the ER Model

Entity Sets

- An **entity** is an object that exists and is distinguishable from other objects.
  - Example: specific person, company, event, plant
- An **entity set** is a set of entities of the same type that share the same properties.
  - Example: set of all persons, companies, trees, holidays
- An entity is represented by a set of **attributes**; i.e., descriptive properties possessed by all members of an entity set.
  - Example:

    instructor = (*ID, name, salary*)

    course= (*course_id, title, credits*)
- A subset of the attributes form a **primary key** of the entity set; i.e., uniquely identifying each member of the set.

Entity Sets -- *instructor* and *student*

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

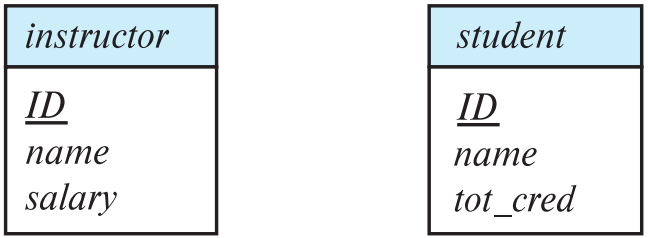
instructor

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 as follows:
  - Rectangles represent entity sets.
  - Attributes listed inside entity rectangle
  - **Underline** indicates primary key attributes



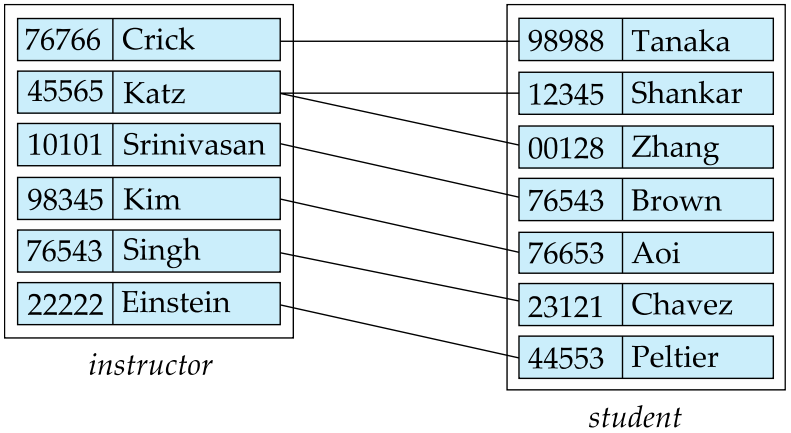
Relationship Sets

- A **relationship** is an association among several entities
  Example:
  44553 (Peltier) *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:
    $(44553,22222) \in advisor$

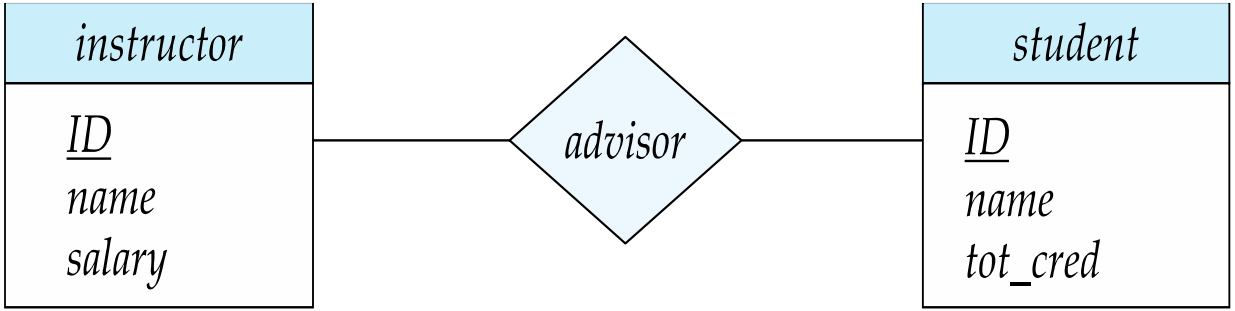
Relationship Sets (Cont.)

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



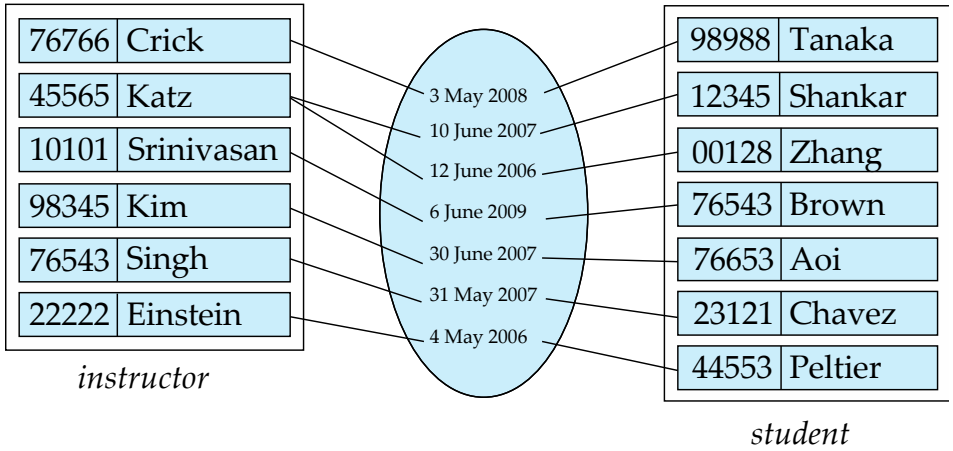
Representing Relationship Sets via ER Diagrams

- Diamonds represent relationship sets.

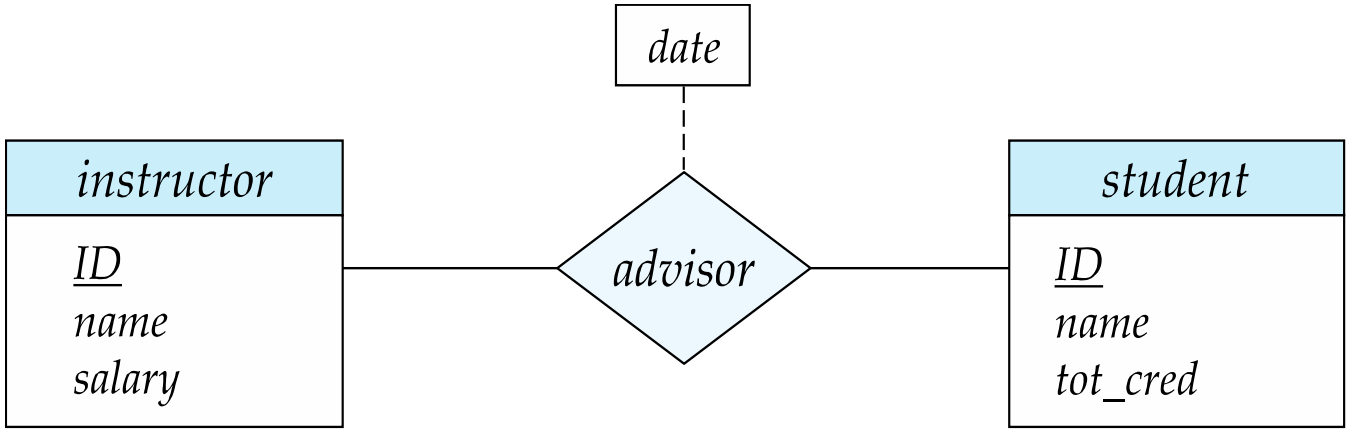


Relationship Sets (Cont.)

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

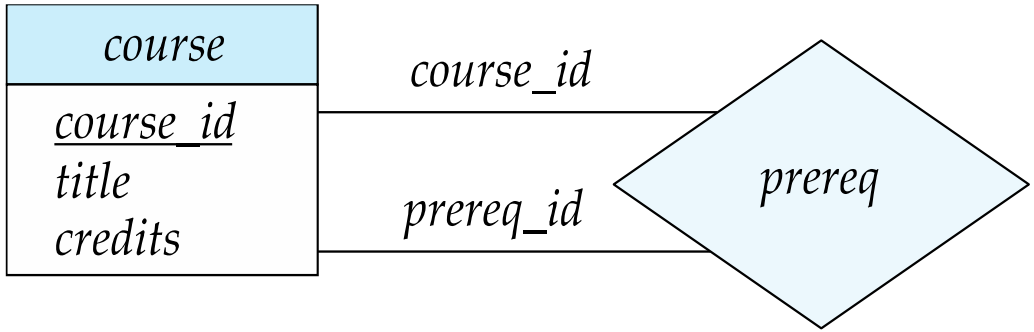


Relationship Sets with Attributes



Roles

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

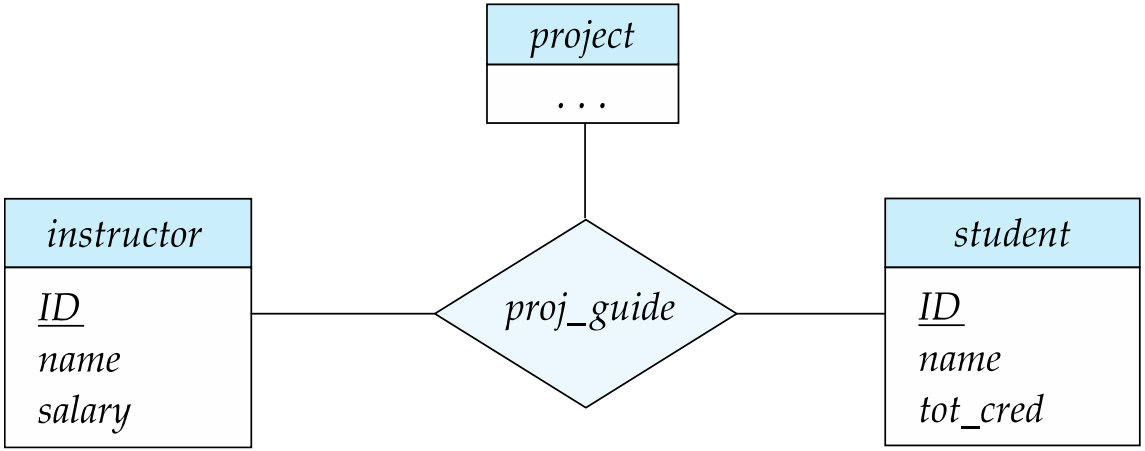


Degree of a Relationship Set

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

Non-binary Relationship Sets

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

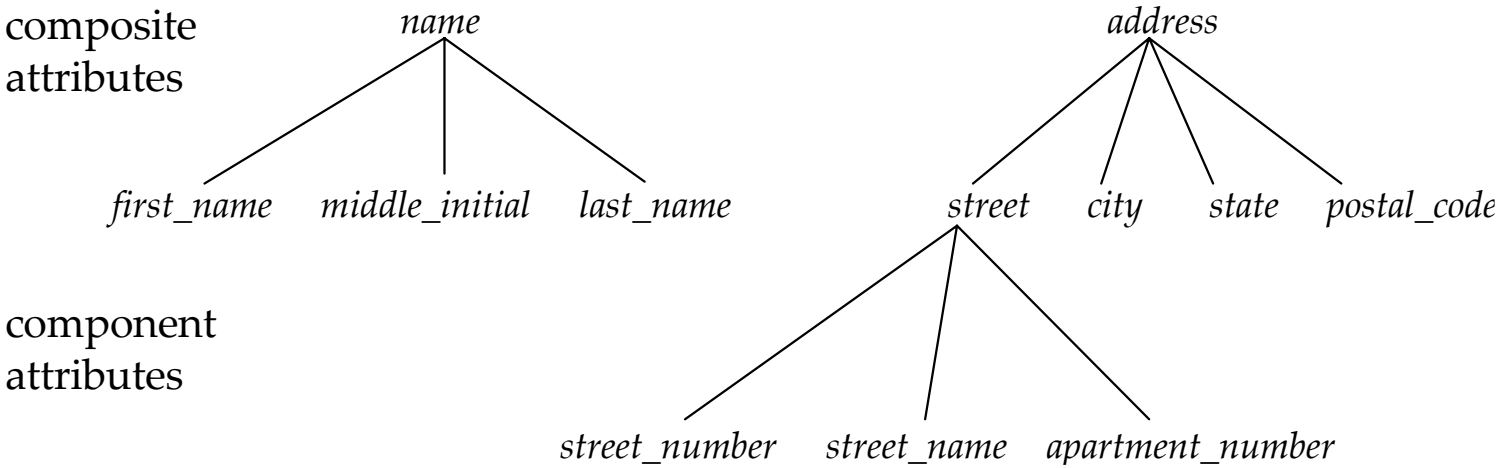


Complex Attributes

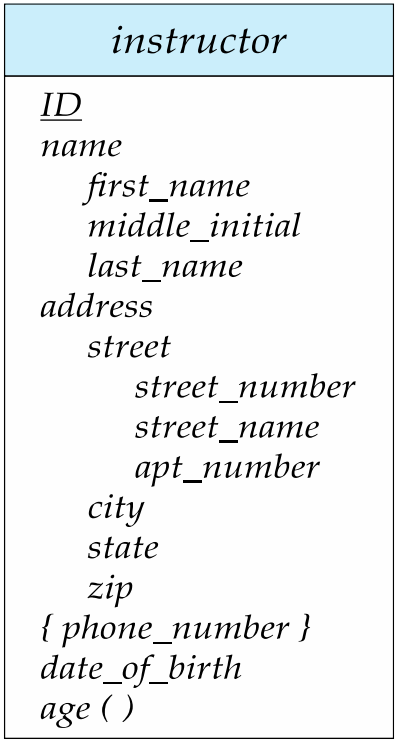
- Attribute types:
  - **Simple** and **composite** attributes.
  - **Single-valued** and **multivalued** attributes
    - Example: multivalued attribute: *phone_numbers*
  - **Derived** attributes
    - Can be computed from other attributes
    - Example: age, given date_of_birth
- **Domain** – the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

Composite Attributes

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



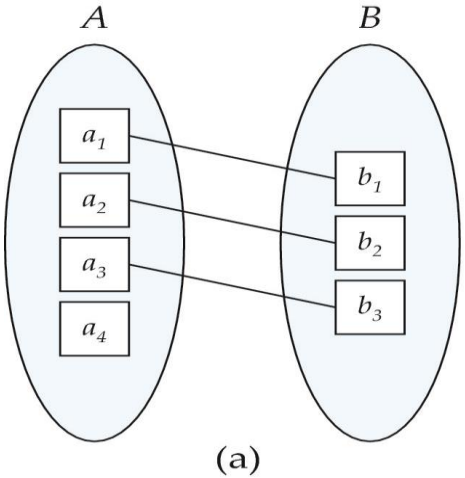
Representing Complex Attributes in ER Diagram



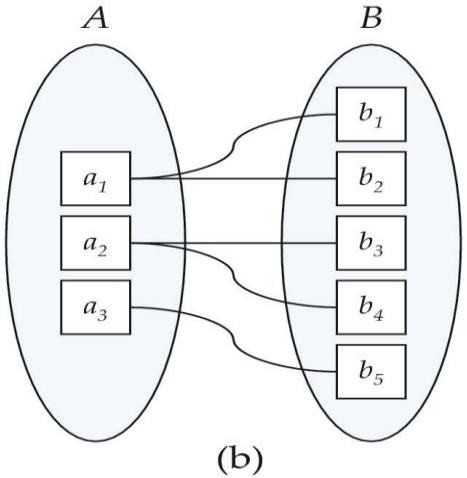
Mapping Cardinality Constraints

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

Mapping Cardinalities



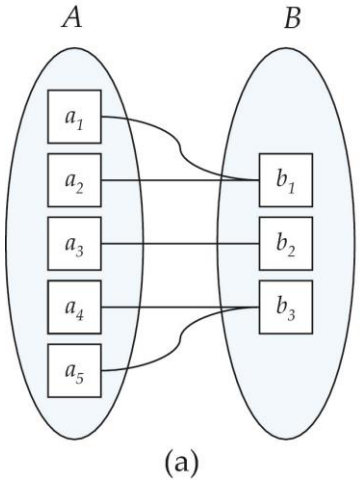
One to one



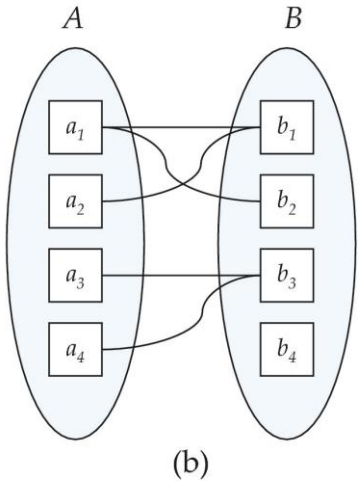
One to many

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

Mapping Cardinalities



Many to one

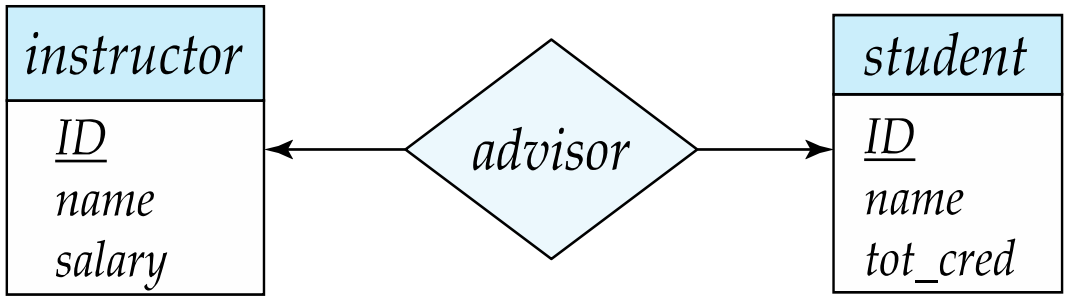


Many to many

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

Representing Cardinality Constraints in ER Diagram

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



One-to-Many Relationship

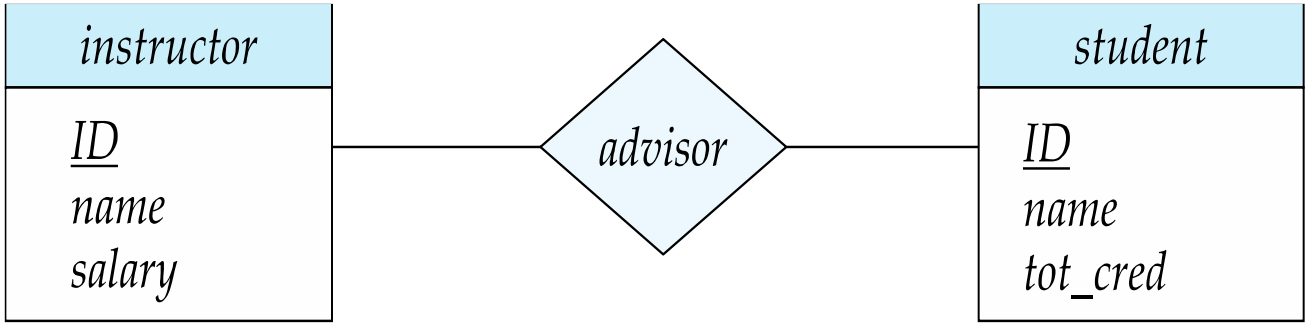
- **one-to-many relationship** between an *instructor* and a *student*
  - an instructor is associated with several (including 0) students via *advisor*
  - a student is associated with at most one instructor via advisor,

Many-to-One Relationships

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

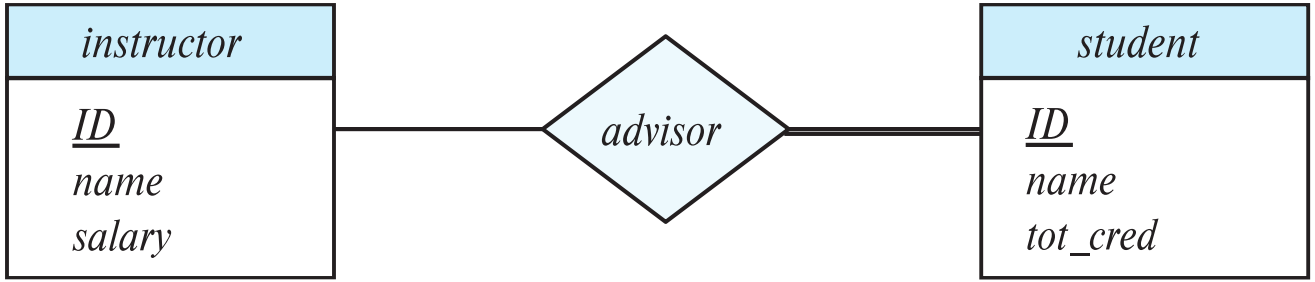
Many-to-Many Relationship

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



Total and Partial Participation

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



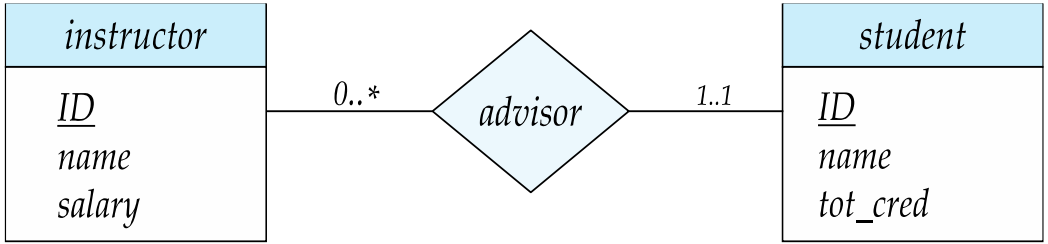
participation of *student* in *advisor* relation is total

  - every *student* must have an associated instructor

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

Notation for Expressing More Complex Constraints

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



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

Primary Key

- Primary keys provide a way to specify how entities and relations are distinguished. We will consider:
  - Entity sets
  - Relationship sets.
  - Weak entity sets

Primary key for Entity Sets

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

Primary Key for Relationship Sets

- To distinguish among the various relationships of a relationship set we use the individual primary keys of the entities in the relationship set.
  - Let *R* be a relationship set involving entity sets E1, E2, .. En
  - The primary key for R is consists of the union of the primary keys of entity sets E1, E2, ..En
  - If the relationship set *R* has attributes a1, a2, .., am associated with it, then the primary key of *R* also includes the attributes a1, a2, .., am
- Example: relationship set "**advisor**".
  - The primary key consists of ***instructor.ID*** and ***student.ID***
- The choice of the primary key for a relationship set depends on the **mapping cardinality of the relationship set.**

Choice of Primary key for Binary Relationship

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

E-R Diagram Components

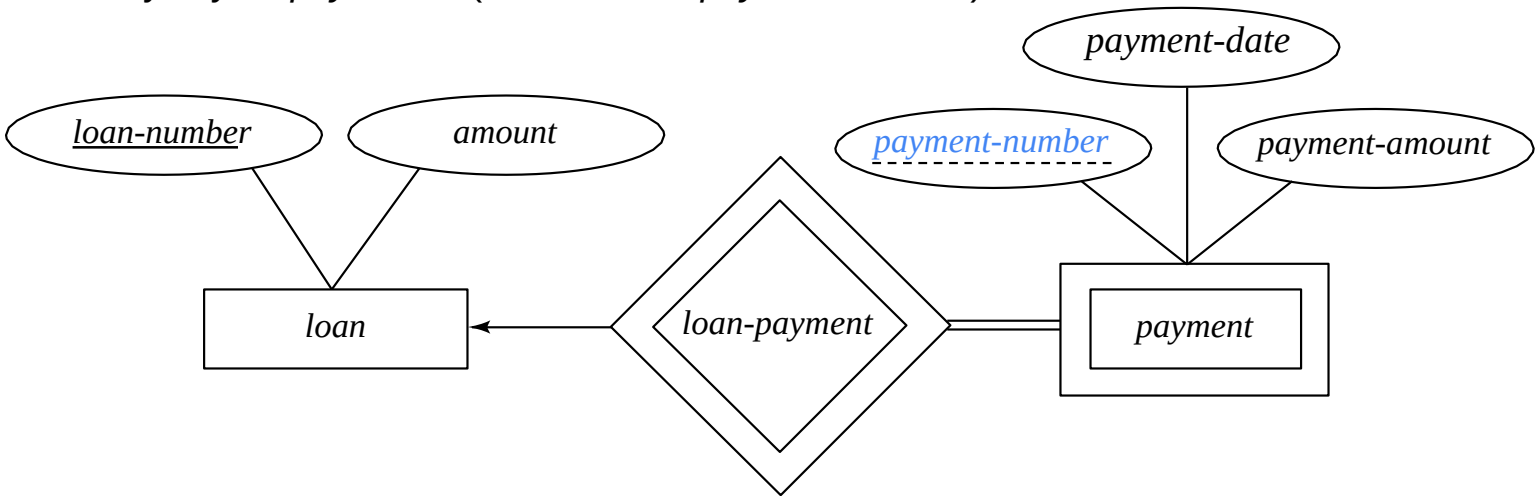
- **Rectangles** represent entity sets.
- **Ellipses** represent attributes.
- **Diamonds** represent relationship sets.
- **Lines** link attributes to entity sets and entity sets to relationship sets.
- **Double ellipses** represent multivalued attributes.
- **Dashed ellipses** denote derived attributes.
- **Primary key** attributes are underlined.

Weak Entity Sets

- An entity set that does not have a primary key is referred to as a ***weak entity*** *set*.
- The existence of a weak entity set depends on the existence of a strong entity set; it must relate to the strong set via a
  - one-to-many relationship set.
- 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.

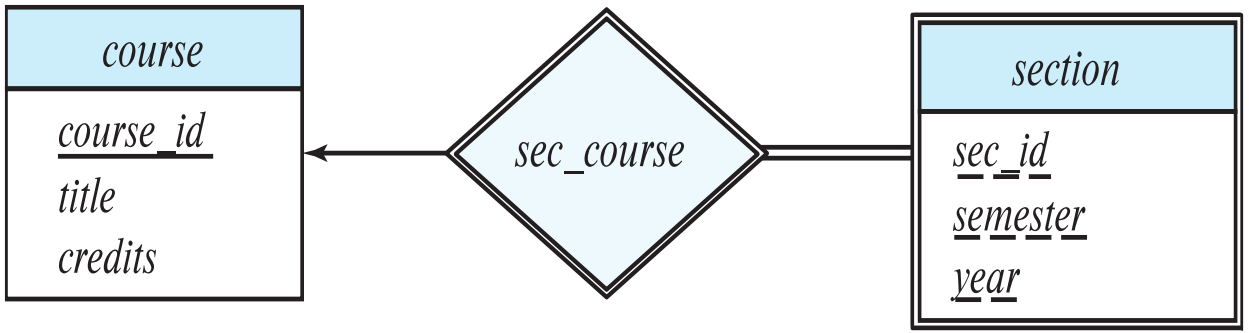
Weak Entity Sets (Cont.)

- We depict a weak entity set by double rectangles.
- We underline the discriminator of a weak entity set with a dashed line.
- *payment-number* – discriminator of the *payment* entity set
- Primary key for *payment* – (*loan-number, payment-number*)



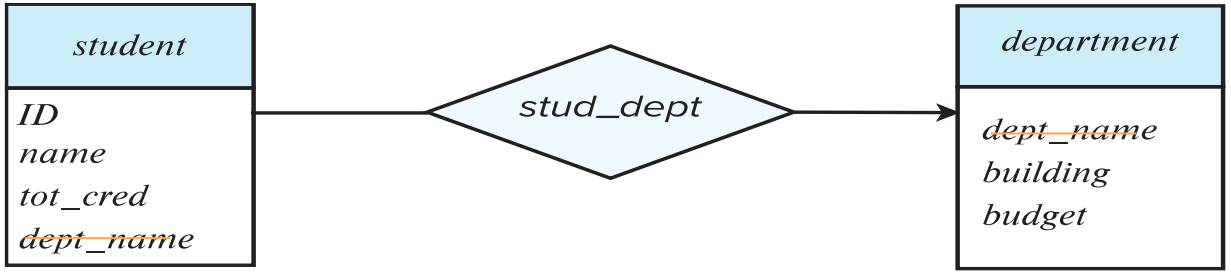
Expressing Weak Entity Sets

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



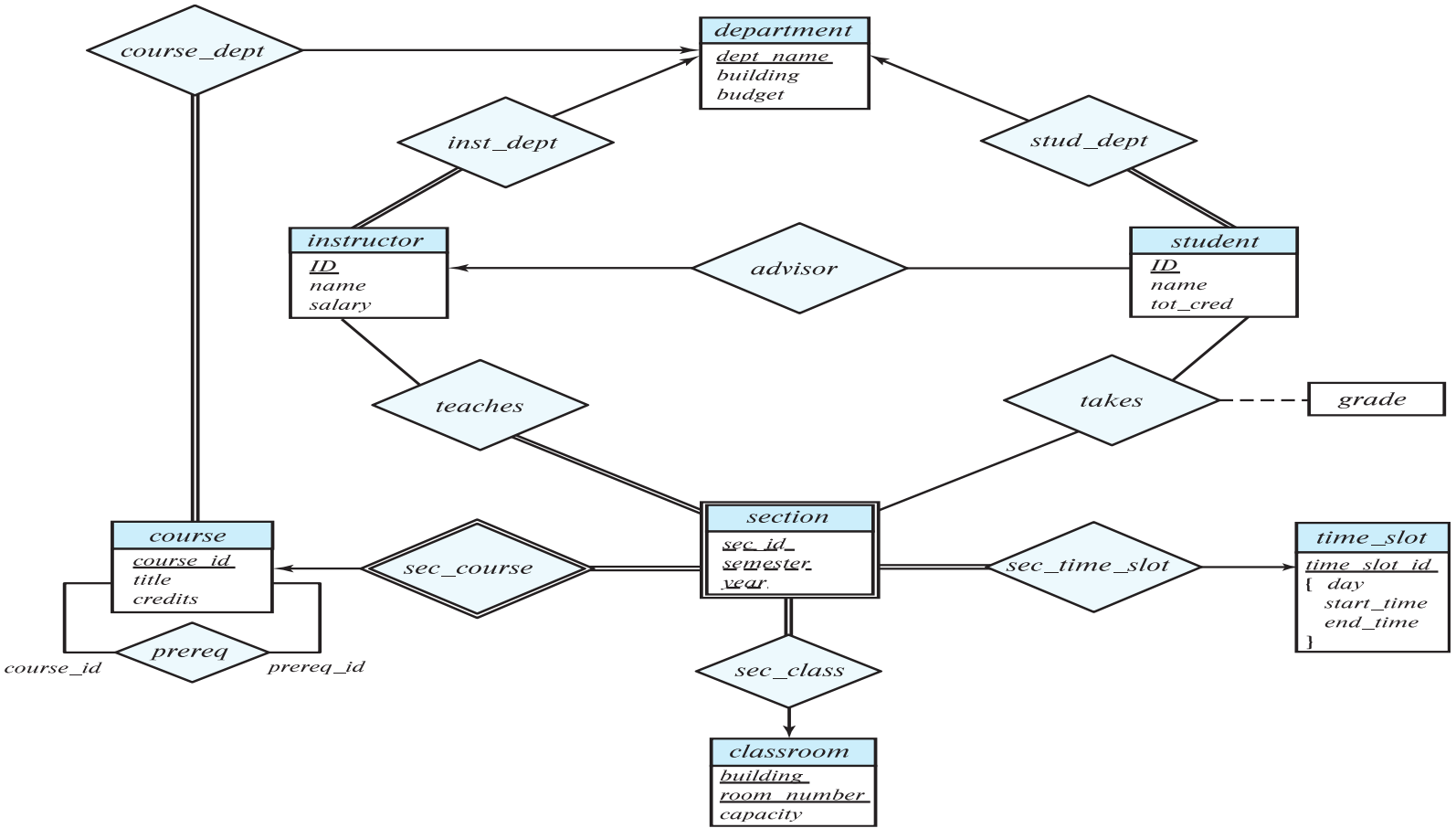
Redundant Attributes

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



(a) Incorrect use of attribute

E-R Diagram for a University Enterprise



Reduction to Relation Schemas

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

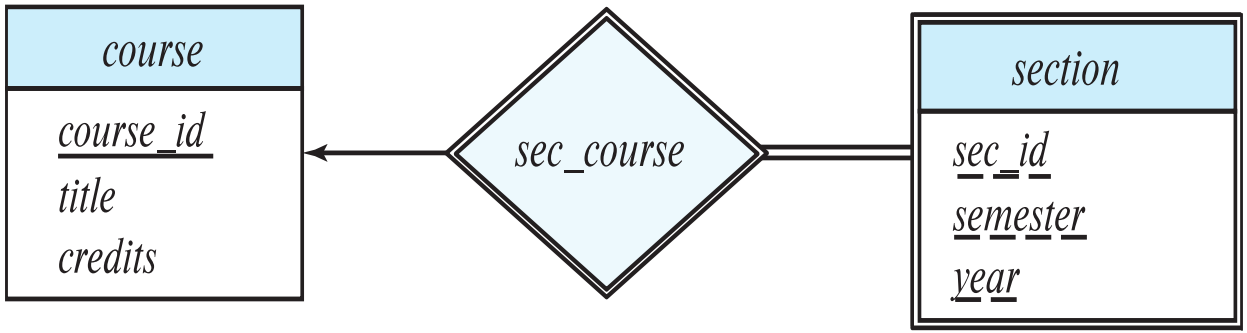
- A **strong entity** set reduces to a schema with the same attributes

  Course (course_id, title, credits)

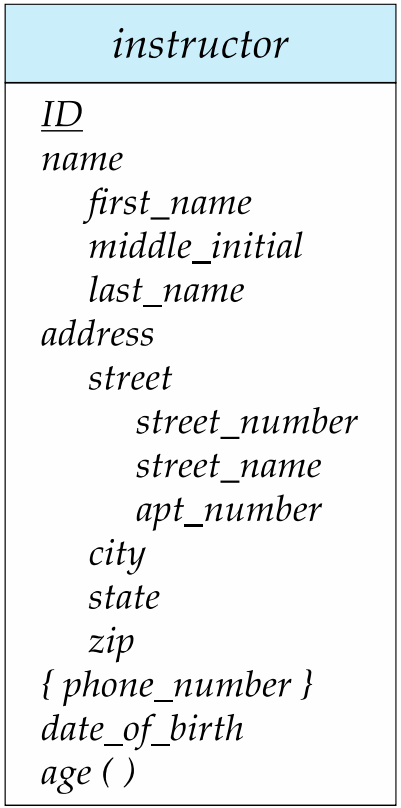
- A **weak entity** set becomes a table that includes a column for the primary key of the identifying strong entity set

  section (course_id, sec_id, semester, year)

- **Example**



Representation of Entity Sets with Composite Attributes

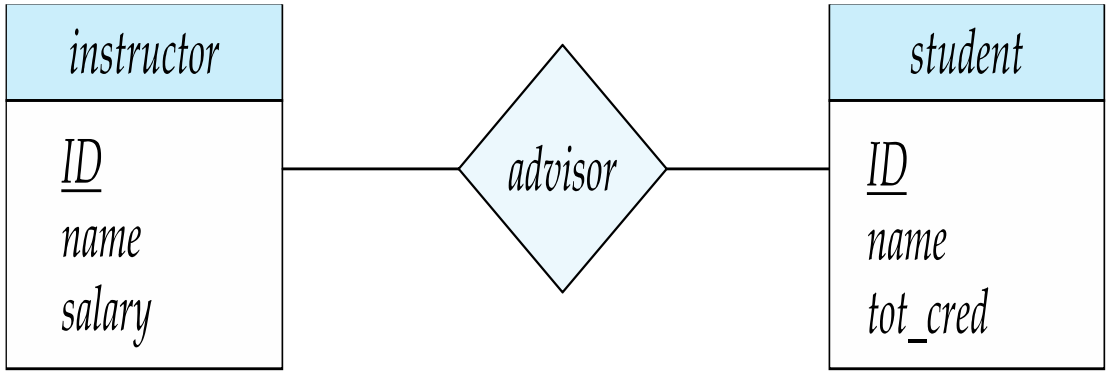


- Composite attributes are flattened out by creating a separate attribute for each component attribute
  - Example: given entity set *instructor* with composite attribute *name* with component attributes *first_name* and *last_name* the schema corresponding to the entity set has two attributes *name_first_name*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

Representing Relationship Sets

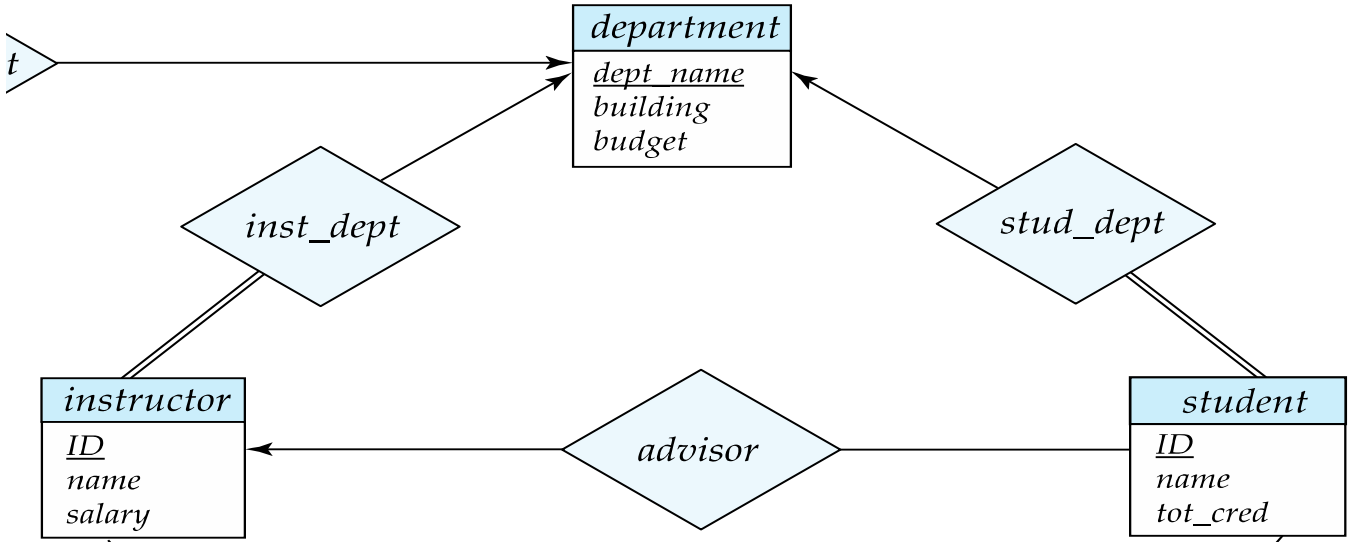
- A many-to-many relationship set is represented as a schema with attributes for the primary keys of the two participating entity sets, and any descriptive attributes of the relationship set.
- Example: schema for relationship set *advisor*

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*
- **Example**



Redundancy of Schemas (Cont.)

- 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

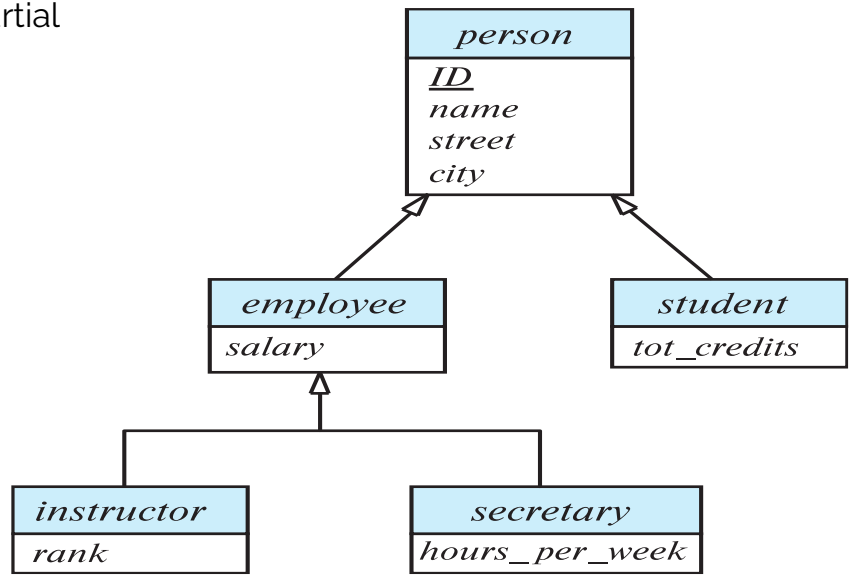
Extended E-R Features

Specialization

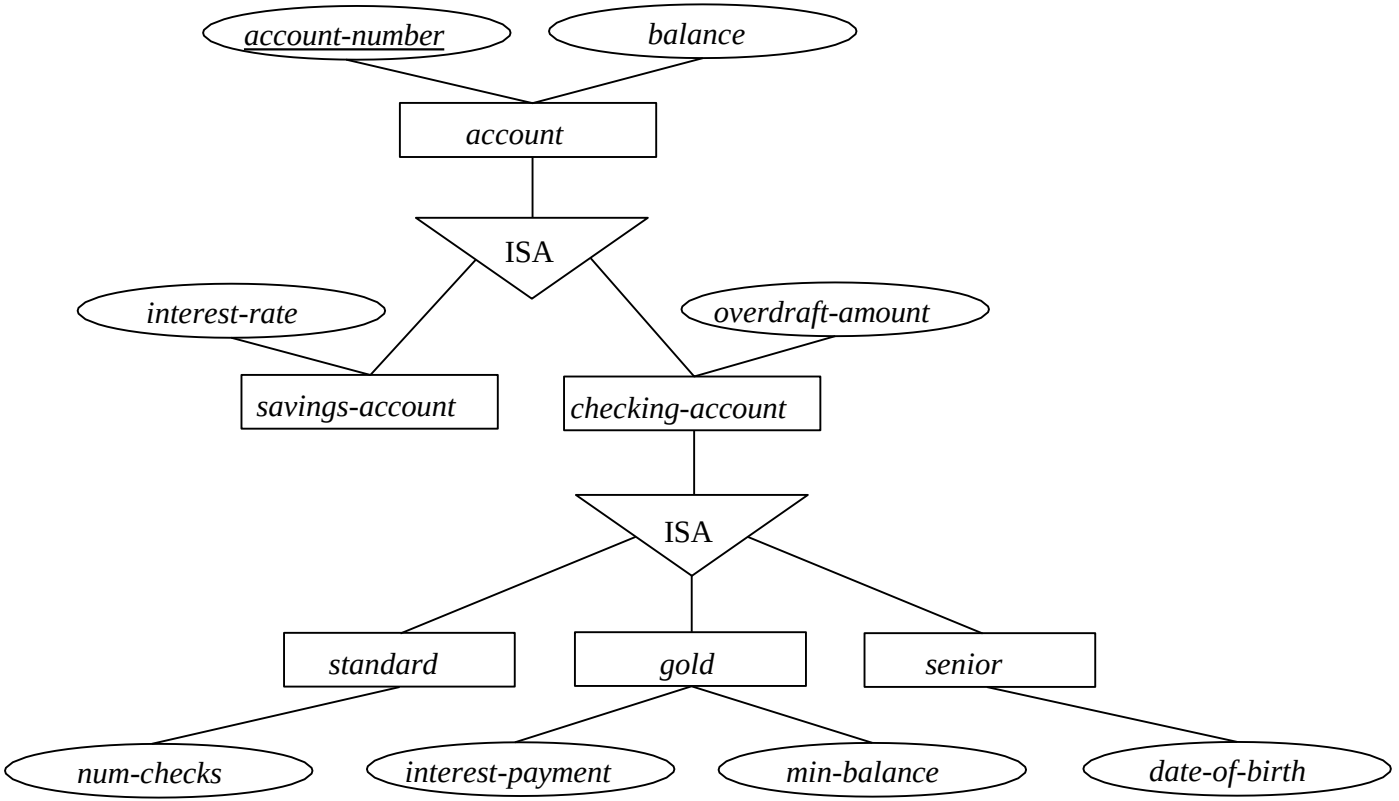
- Top-down design process; we designate sub-groupings within an entity set that are distinctive from other entities in the set.
- These sub-groupings become lower-level entity sets that have attributes or participate in relationships that do not apply to the higher-level entity set.
- Depicted by a *triangle* 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



Specialization Example 2



Representing Specialization via Schemas

- **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

Representing Specialization as Schemas (Cont.)

- **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

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

Generalization

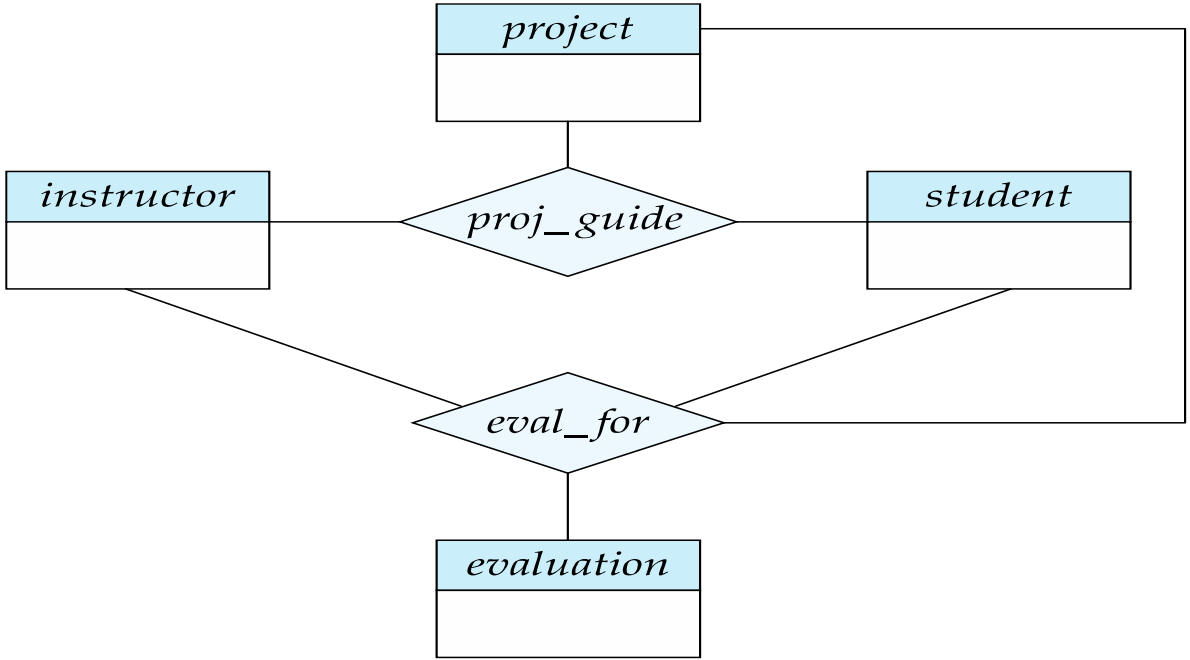
- **A bottom-up design process** – combine a number of entity sets that share the same features into a higher-level entity set.
- Specialization and generalization are simple inversions of each other; they are represented in an E-R diagram in the same way.
- The terms specialization and generalization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

Design Constraints on a Generalization

- Constraint on which entities can be members of a given lower-level entity set.
  - condition-defined
  - user-defined
- Constraint on whether or not entities may belong to more than one lower-level entity set within a single generalization.
  - disjoint
  - overlapping
- 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
  - partial

Aggregation

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

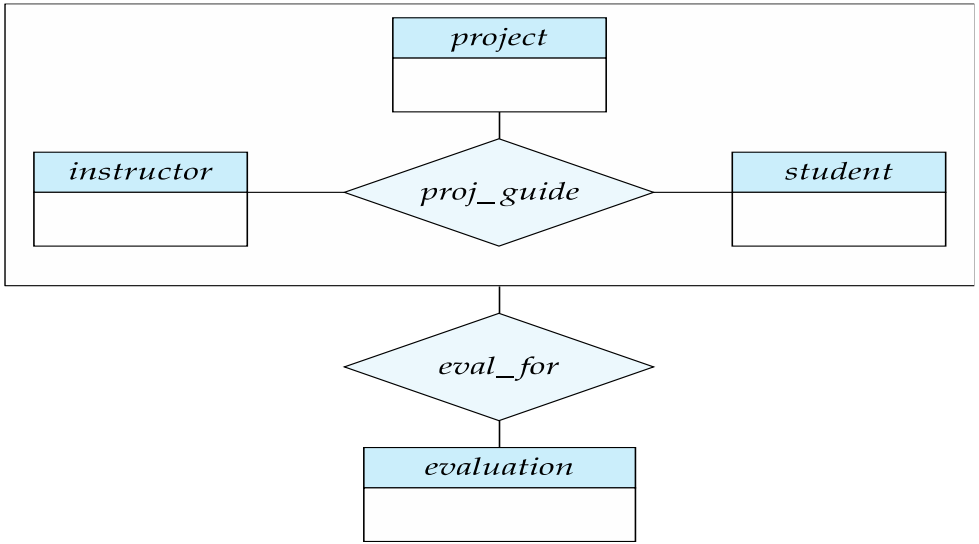


Aggregation (Cont.)

- 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

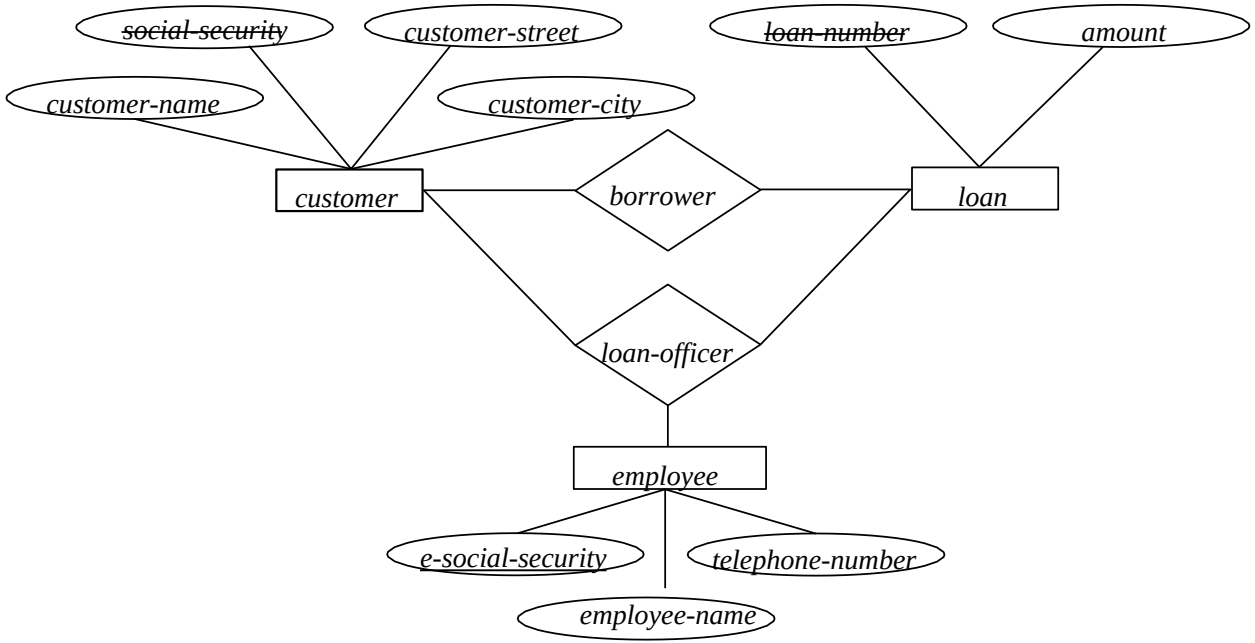
Aggregation (Cont.)

- Eliminate this redundancy via *aggregation* 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



Aggregation Example 2

- Loan customers may be advised by a loan-officer.



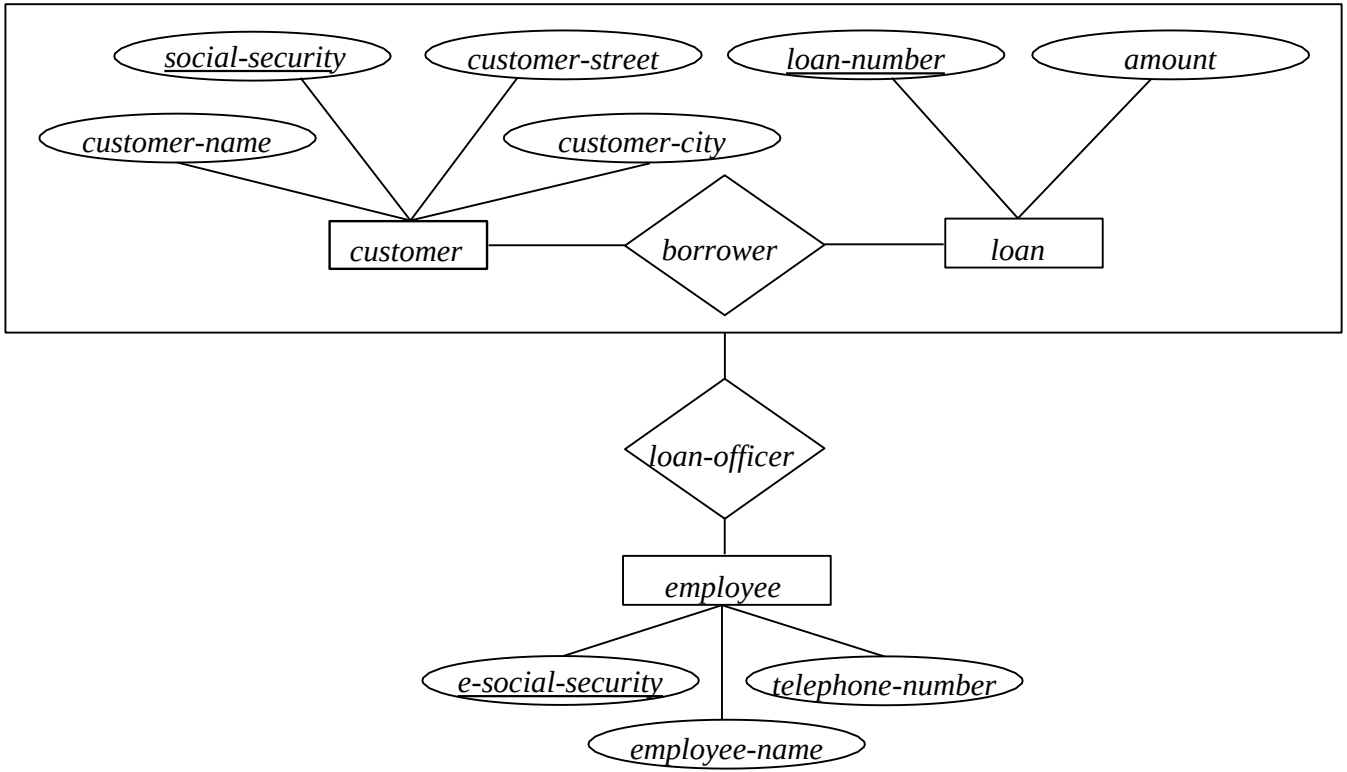
Aggregation Example 2

- Relationship sets *borrower* and *loan-officer* represent the same information
- Eliminate this redundancy via *aggregation*
  - Treat relationship as an abstract entity
  - Allows relationships between relationships
  - Abstraction of relationship into new entity

Without introducing redundancy, the following diagram represents that:

- A customer takes out a loan
- An employee may be a loan officer for a *customer-loan* pair

Aggregation Example 2



Reduction to Relational Schemas

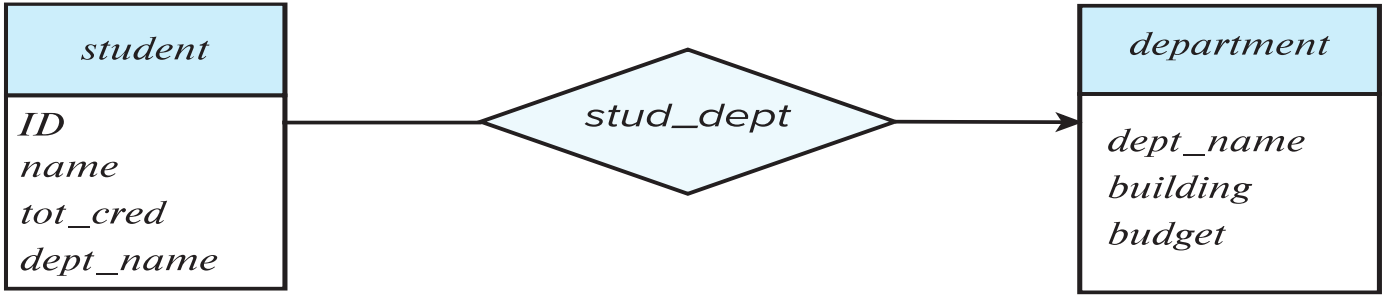
- To represent aggregation, create a schema containing
  - Primary key of the aggregated relationship,
  - The primary key of the associated entity set
  - Any descriptive attributes
- In our example:
  - The schema *eval_for* is:

    eval_for (*s_ID, project_id, i_ID, evaluation_id*)
  - The schema *proj_guide* is redundant.

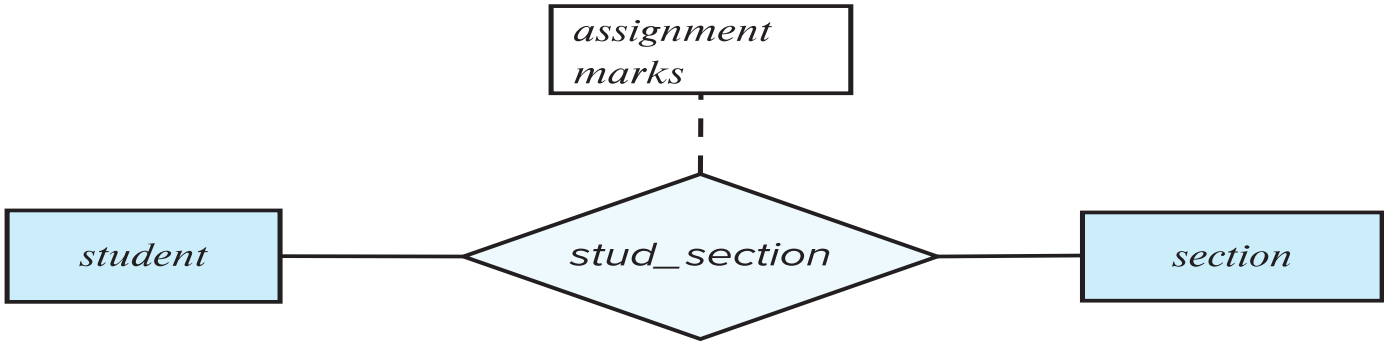
ELSEWEDY UNIVERSITY OF TECHNOLOGY
- POLYTECHNIC OF EGYPT -

Design Issues

Common Mistakes in E-R Diagrams

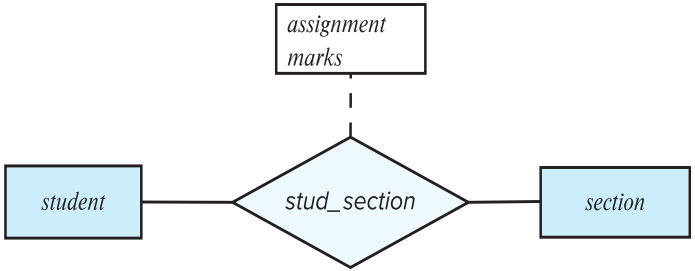


(a) Incorrect use of attribute

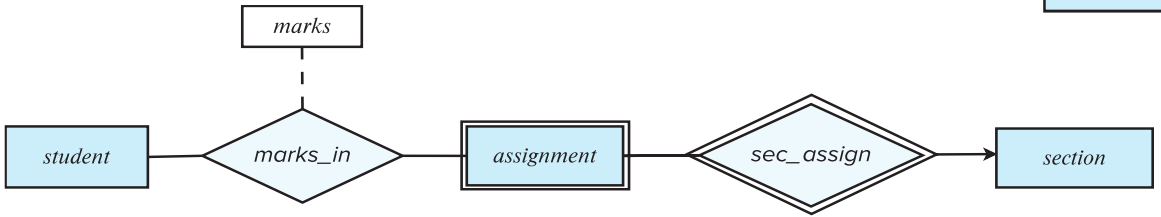


(b) Erroneous use of relationship attributes

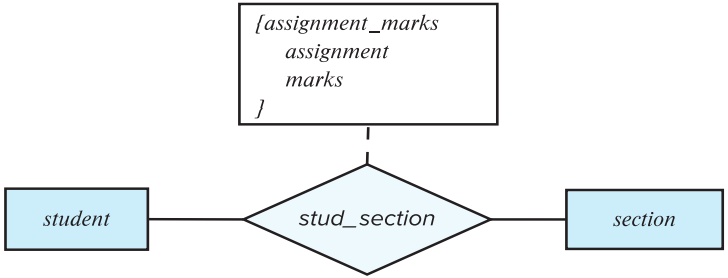
Common Mistakes in E-R Diagrams (Cont.)



(b) Erroneous use of relationship attributes



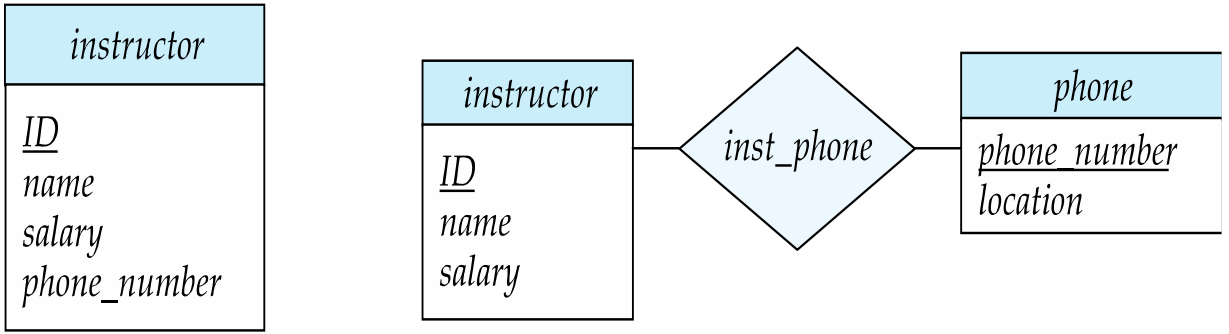
(c) Correct alternative to erroneous E-R diagram (b)



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

Entities vs. Attributes

- Use of entity sets vs. attributes

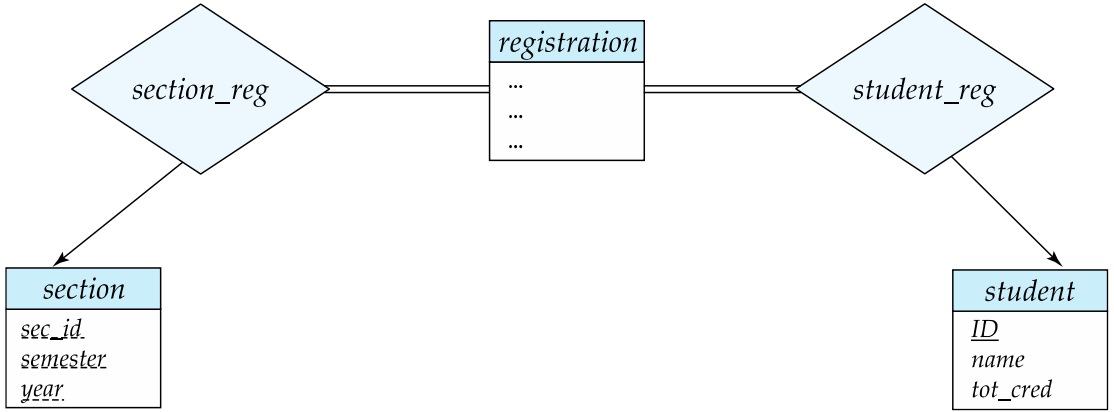


- Use of phone as an entity allows extra information about phone numbers (plus multiple phone numbers)

Entities vs. Relationship sets

- **Use of entity sets vs. relationship sets**

  Possible guideline is to designate a relationship set to describe an action that occurs between entities



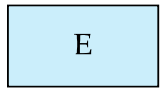
- **Placement of relationship attributes**

  For example, attribute date as attribute of advisor or as attribute of student

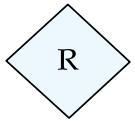
E-R Design Decisions

- 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.

Summary of Symbols Used in E-R Notation

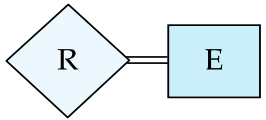


entity set

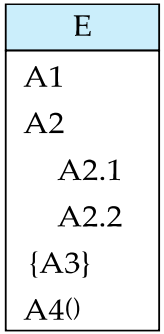


relationship set

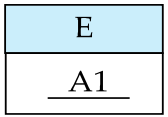
identifying relationship set for weak entity set



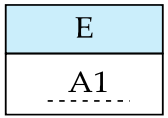
total participation of entity set in relationship



attributes:
simple (A1),
composite (A2) and
multivalued (A3)
derived (A4)

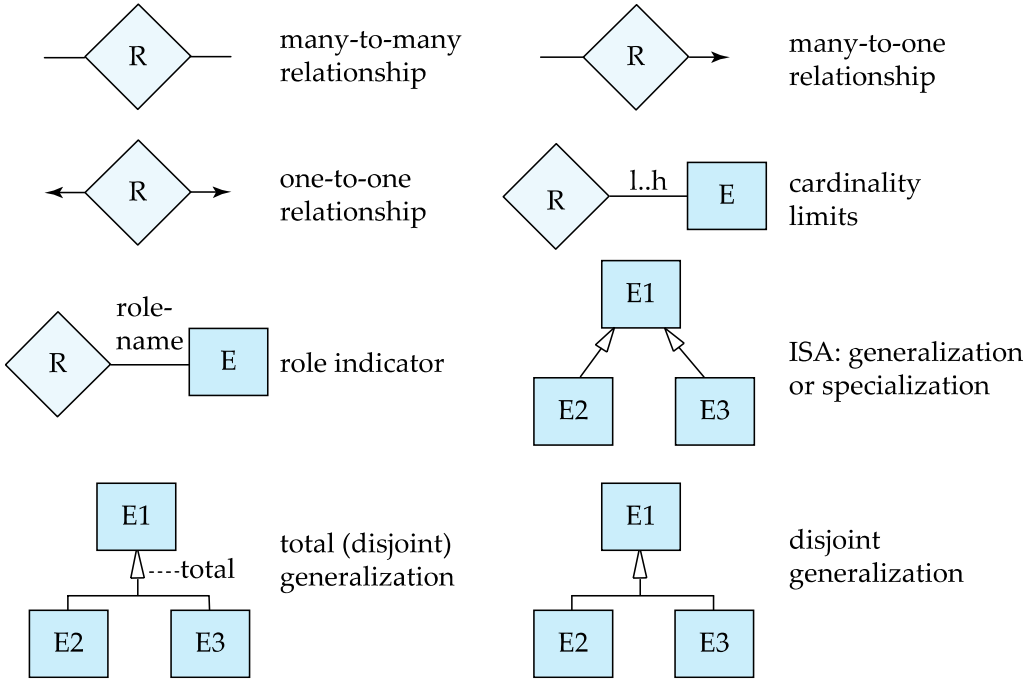


primary key



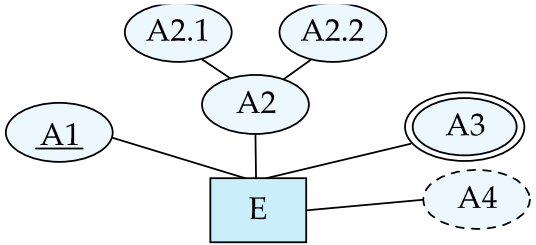
discriminating attribute of weak entity set

Symbols Used in E-R Notation (Cont.)



Alternative ER Notations

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

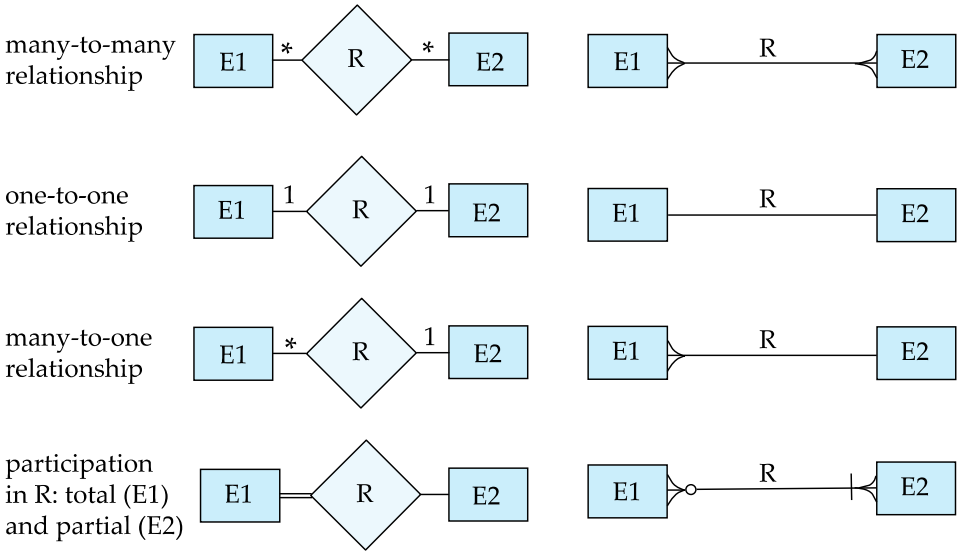


weak entity set 

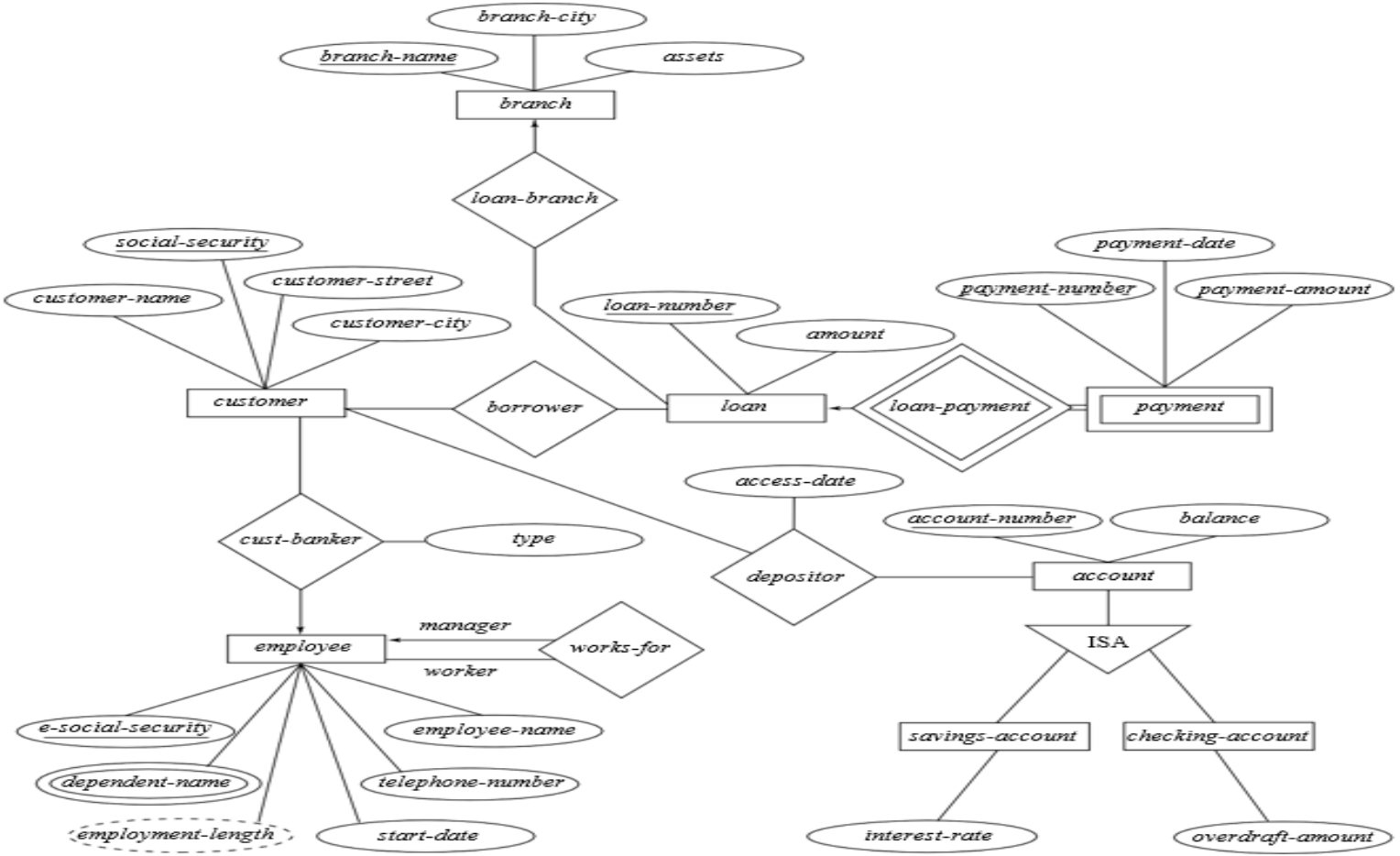
generalization 

total
generalization 

Alternative ER Notations



E-R Diagram for Banking Enterprise



Reduction of an E-R Schema to Tables (rational model)

- Primary keys allow entity sets and relationship sets to be expressed uniformly as *tables* which represent the contents of the database.
- A database which conforms to an E-R diagram can be represented by a collection of tables.
- For each entity set and relationship set there is a unique table which is assigned the name of the corresponding entity set or relationship set.
- Each table has a number of columns (generally corresponding to attributes), which have unique names.
- Converting an E-R diagram to a table format is the basis for deriving a relational database design from an E-R diagram.

Representing Entity Sets as Tables

- A strong entity set reduces to a table with the same attributes.

customer-name	*social-security*	*c-street*	*c-city*
Jones	321-12-3123	Main	Harrison
Smith	019-28-3746	North	Rye
Hayes	677-89-9011	Main	Harrison

The *customer* table

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

loan-number	*payment-number*	*payment-date*	*payment-amount*
L-17	5	10 May 1996	50
L-23	11	17 May 1996	75
L-15	22	23 May 1996	300

The *payment* table

Representing Entity Sets as Tables

- A many-to-many relationship set is represented as a table with columns for the primary keys of the two participating entity sets, and any descriptive attributes of the relationship set.

social-security	*account-number*	*access-date*
...	...	...

The *depositor* table

- The table corresponding to a relationship set linking a weak entity set to its identifying strong entity set is redundant.

  The *payment* table already contains the information that would appear in the *loan-payment* table (i.e., the columns
  loan-number and *payment-number*).

Representing Entity Sets as Tables

- Method 1: Form a table for the generalized entity ***account*** Form a table for each entity set that is generalized (include primary key of generalized entity set)

table	table attributes
account	*account-number, balance, account-type*
savings-account	*account-number, interest-rate*
checking-account	*account-number, overdraft-amount*

- Method 2: Form a table for each entity set that is generalized

table	table attributes
savings-account	*account-number, balance, interest-rate*
checking-account	*account-number, balance, overdraft-amount*

Method 2 has no table for generalized entity *account*

Relations Corresponding to Aggregation

customer

customer-name	*cust-social-security*	*customer-street*	*customer-city*

loan

loan-number	*amount*

borrower

cust-social-security	*loan-number*

employee

emp-social-security	*employee-name*	*phone-number*

loan-officer

emp-social-security	*cust-social-security*	*loan-number*


Question
&
Answer