

BCA313

Unit 2

Entity-Relationship Model

Dr Arpita Mathur
Assistant Professor, Faculty of Computer Science, LMCST,
Jodhpur

Entity-Relationship Model

- Entity
- Attributes
- Relationship
- Constraints
- E-R Diagram
- Keys
- Weak Entity Sets
- Specialization and Generalization
- Aggregation

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.
- 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. 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

ER Model

- The ER data mode was developed to facilitate database design by allowing specification of an **enterprise schema** that represents the overall logical structure of a database.
- The ER model is **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.
- The ER model also has an associated diagrammatic representation, the ER diagram, which can express the overall logical structure of a database graphically.

Entity Sets

- An **entity** is any real world object that 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, street, city, salary)
course=(course_id, title, credits)*
- Domain or value is a set of permitted values for each attribute.

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

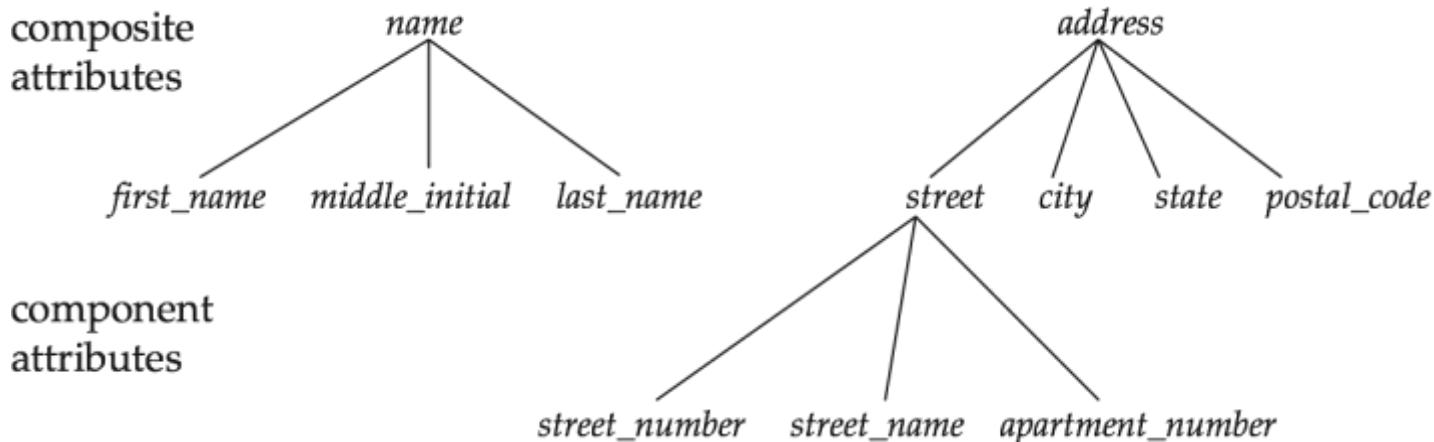
98988	Tanaka
12345	Shankar
00128	Zhang
76543	Brown
76653	Aoi
23121	Chavez
44553	Peltier

student

Types of 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



- An attribute can take a **null** value when an entity does not have a value for it. Null value indicate either the value does not exist or is unknown.

Relationship Sets

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

Example:

44553 (Amit) advisor 22222 (Nitin)
student entity relationship set instructor entity

- A **relationship set** is a set of relationships of the same type. It is a mathematical relation among $n \geq 2$ entities, each taken from entity sets

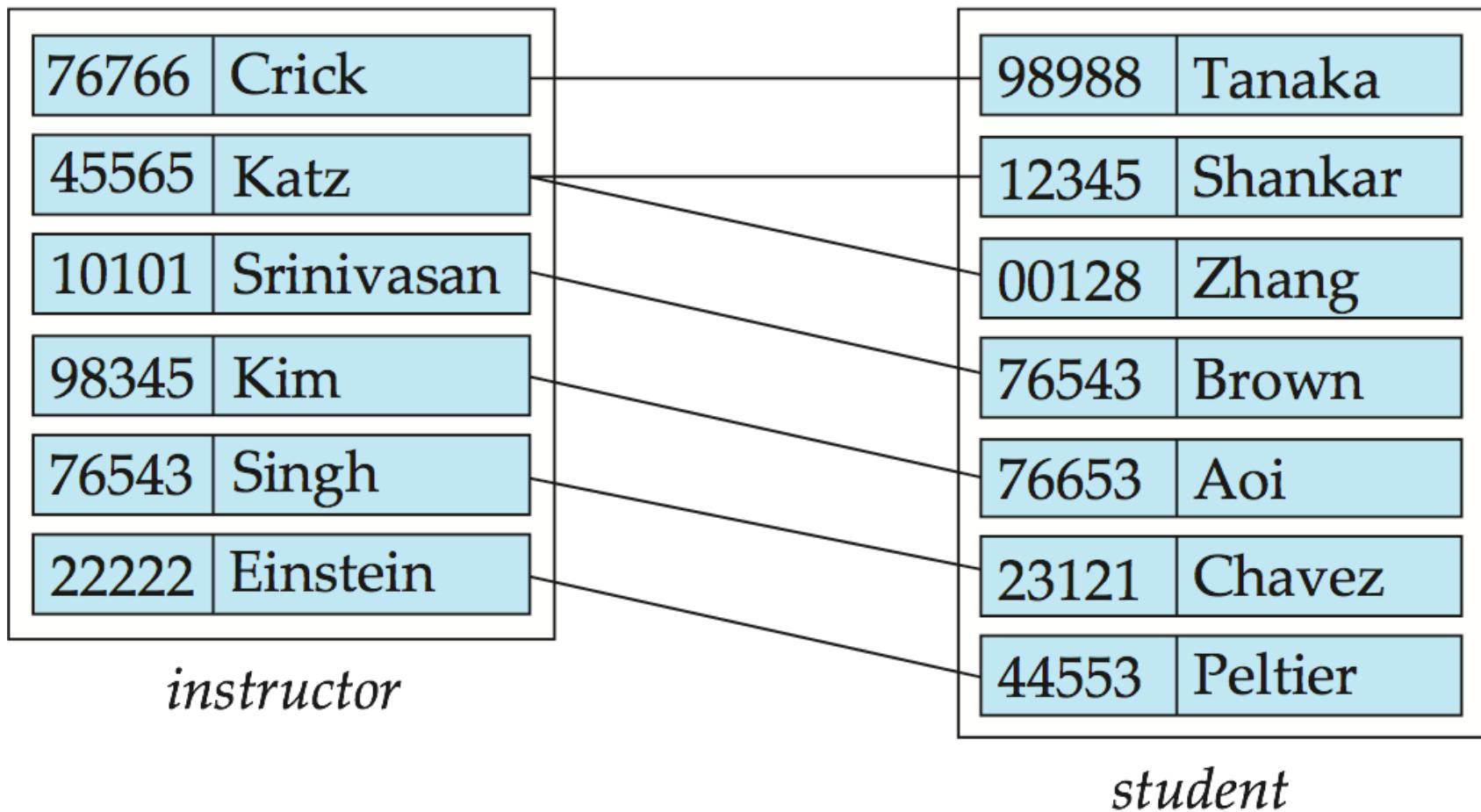
$$\{(e_1, e_2, \dots, e_n) \mid e_1 \in E_1, e_2 \in E_2, \dots, e_n \in E_n\}$$

where (e_1, e_2, \dots, e_n) is a relationship

- Example:

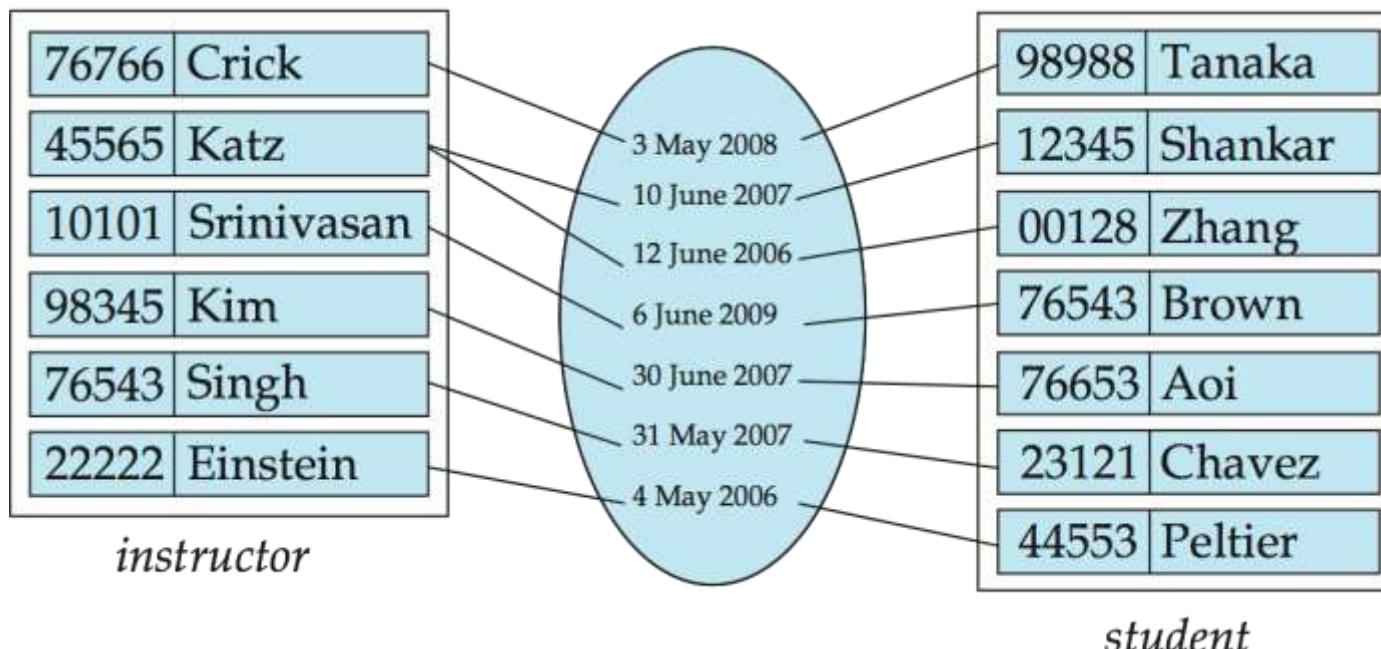
$$(44553, 22222) \in \text{advisor}$$

Relationship Set *advisor*



Relationship Sets (Cont.)

- An attribute can also be associated with a relationship set. It is known as **descriptive attribute**.
- 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.
- 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 *instructor*, *student*, and *project*

Constraints

An E R enterprise schema may define certain constraints to which the contents of a database must conform.

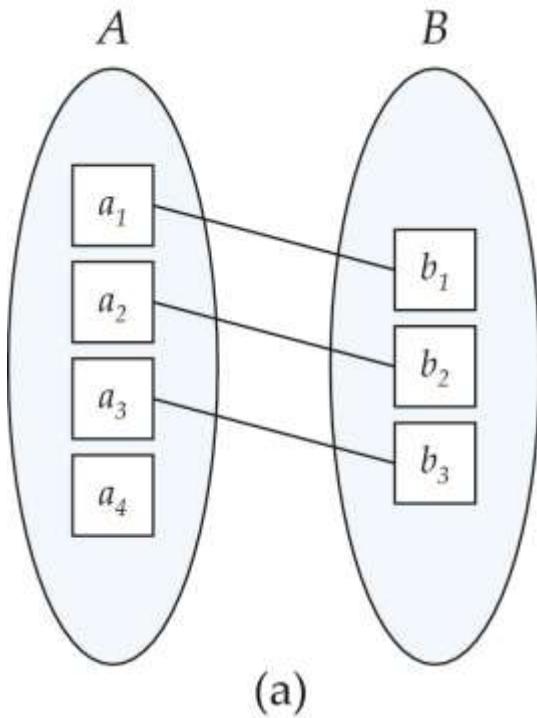
1. Mapping Cardinalities

- 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: An entity in A is associated with **at most one entity** in B.
 - One to many
 - Many to one
 - Many to many

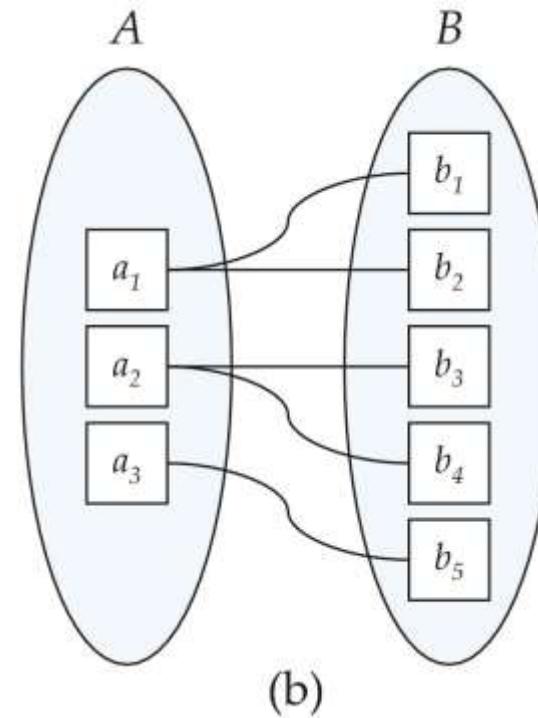
2. Participation constraints

- Total participation: The participation of an entity set E in a relationship set R is total if 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

Mapping Cardinalities



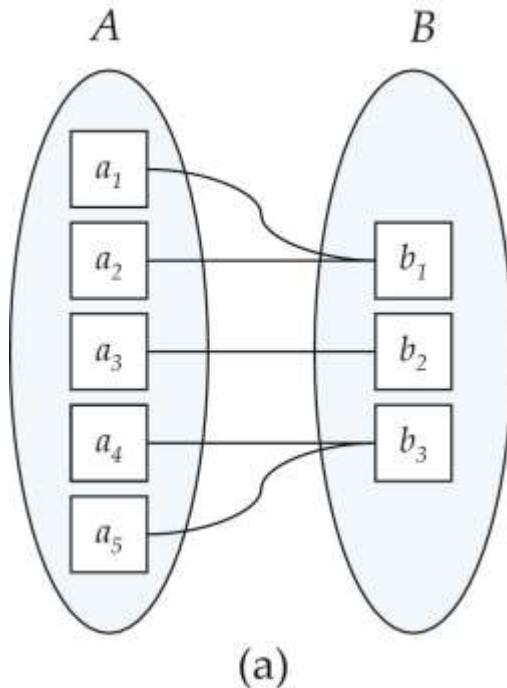
One to one



One to many

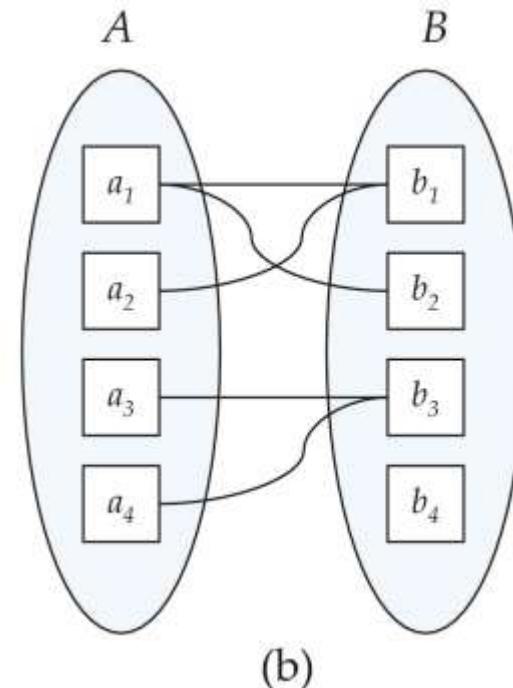
Note: Some elements in A and B may not be mapped to any elements in the other set

Mapping Cardinalities



(a)

Many to
one



(b)

Many to many

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

Keys

No two entities in an entity set are allowed to have exactly the same value for all attributes. A **key** allows us to identify a set of attributes that suffice to distinguish entities from each other. Keys also help uniquely identify relationships, and thus distinguish relationships from each other.

- **Super Key:** A superkey is a set of one or more attributes that, taken collectively, allow us to identify uniquely an entity in the entity set.
- **Candidate Key:** Minimal superkeys (superkeys for which no proper subset is a superkey) are called candidate keys.
- **Primary Key:** Candidate key that is chosen by the database designer as the principal means of identifying entities within an entity set is known as primary key. Remaining candidate keys are known as **alternate keys**.

A key (primary, candidate, and super) is a property of the entity set, rather than of the individual entities. Any two individual entities in the set are prohibited from having the same value on the key attributes at the same time. The designation of a key represents a constraint in the real-world enterprise being modeled.

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 *sec_course*; however, by doing so the relationship between *section* and *course* becomes implicit in an attribute, which is not desirable.

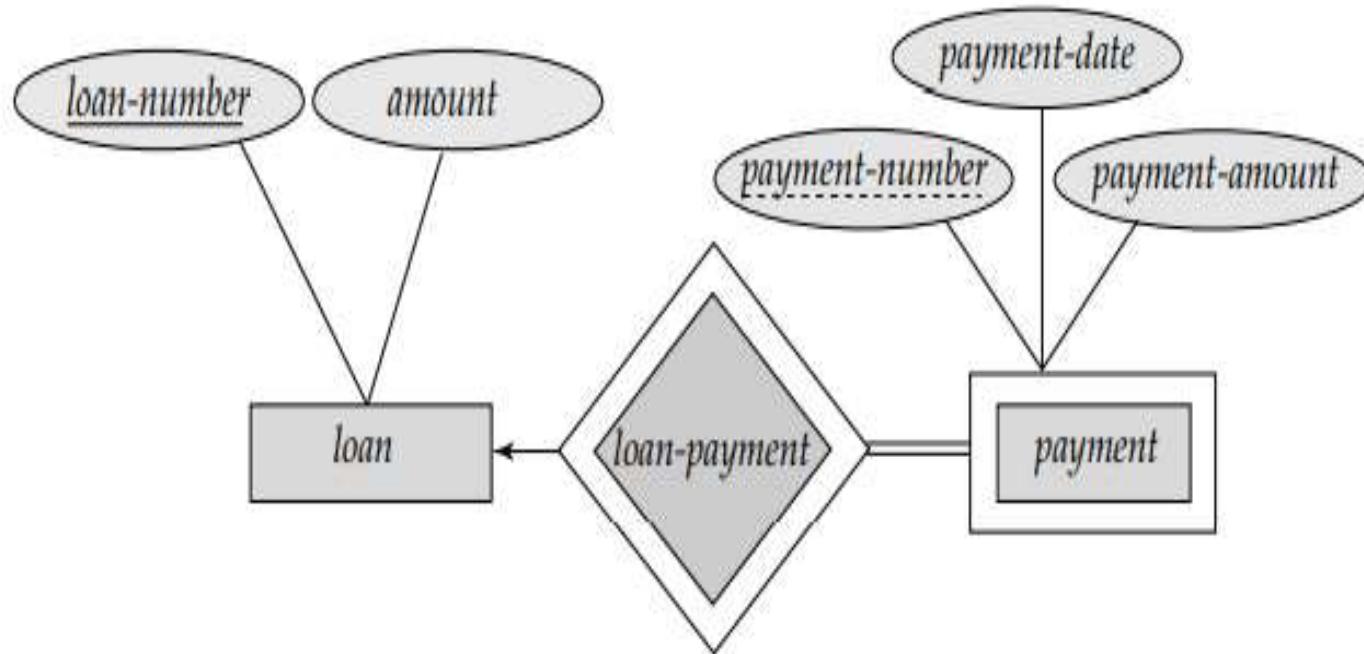
Weak Entity Sets (Cont.)

- 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; although each *section* entity is distinct, sections for different courses may share the same *section_id*, *year*, and *semester*.
- 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.
- The notion of **weak entity set** formalizes the above intuition. 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. An entity set that is not a weak entity set is termed a **strong entity set**.

Weak Entity Sets (Cont.)

- Every weak entity must be associated with an identifying entity; that is, the weak entity set is said to be **existence dependent** on the identifying entity set. The identifying entity set is said to **own** the weak entity set that it identifies. The relationship associating the weak entity set with the identifying entity set is called the **identifying relationship**.
- 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*.

Example of weak entity set



E-R Diagrams

Entity Sets

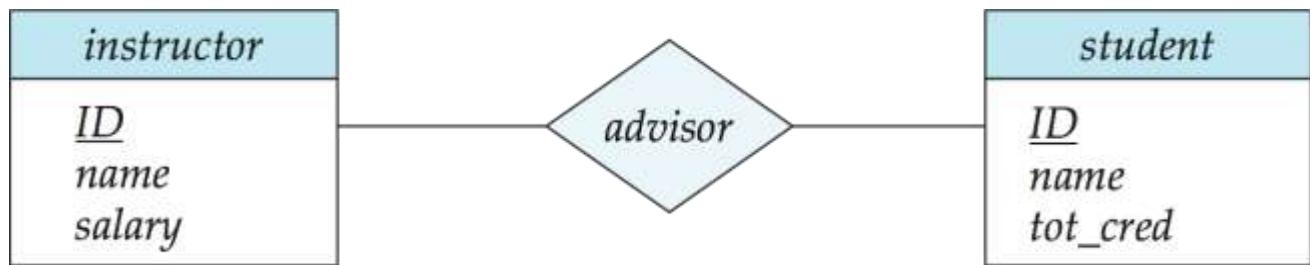
- Entities can be represented graphically as follows:
 - Rectangles represent entity sets.
 - Attributes listed inside entity rectangle
 - Underline indicates primary key attributes

<i>instructor</i>
<u>ID</u>
<i>name</i>
<i>salary</i>

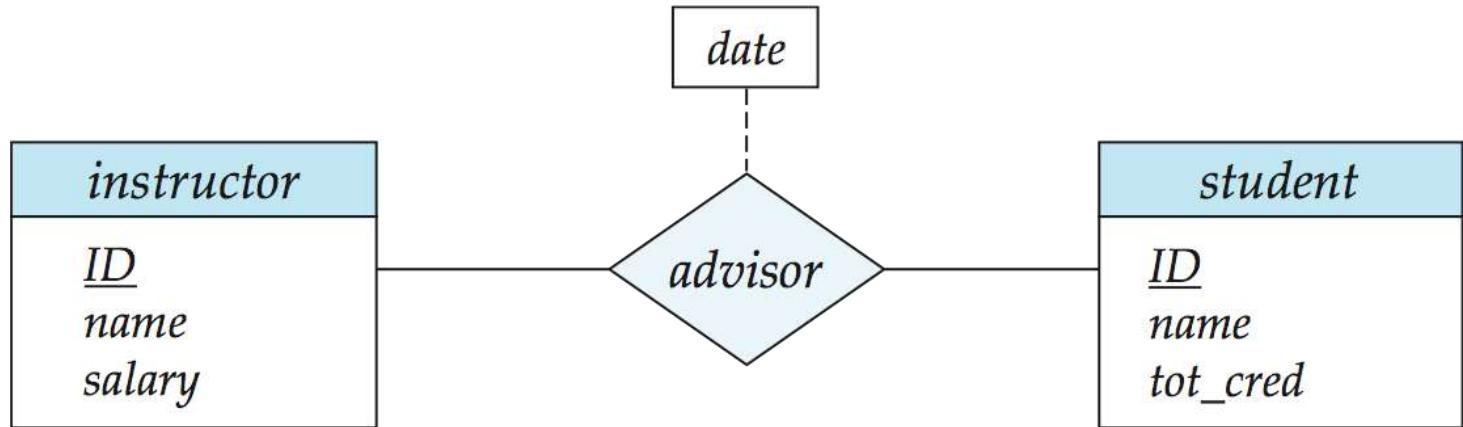
<i>student</i>
<u>ID</u>
<i>name</i>
<i>tot_cred</i>

Relationship Sets

- Diamonds represent relationship sets.

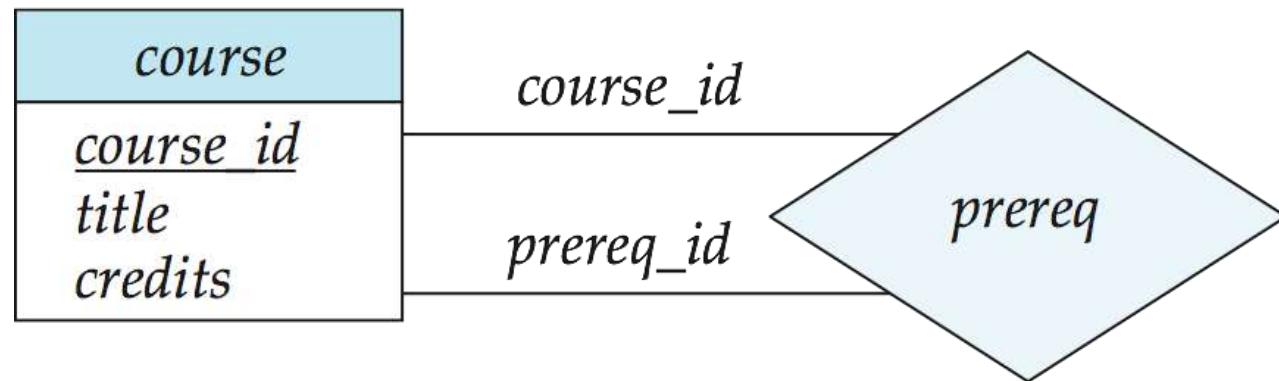


Relationship Sets with Attributes



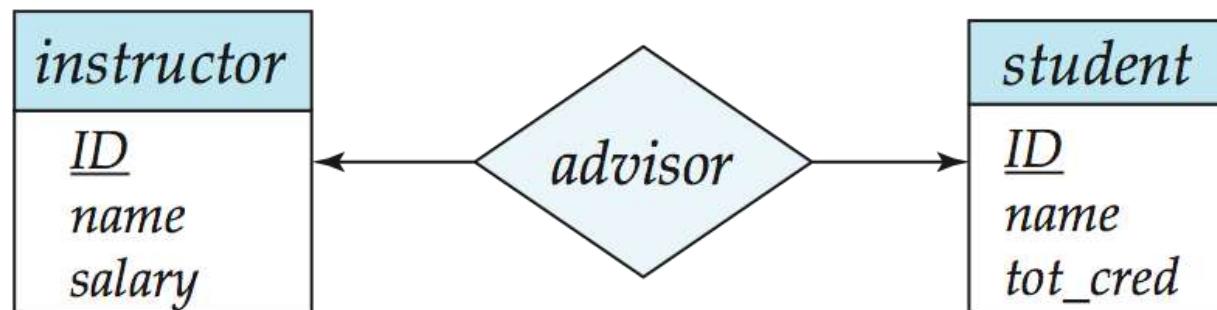
Roles

- 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**.



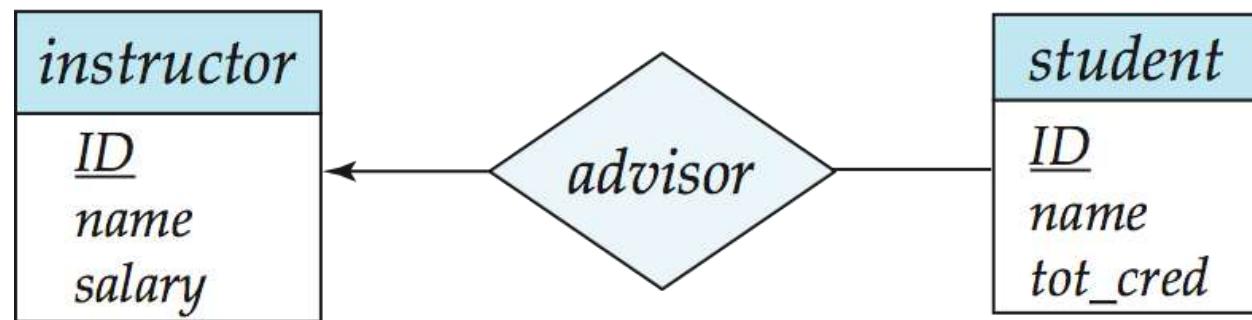
Cardinality Constraints

- 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*
 - A *student* is associated with at most one *department* via *stud_dept*



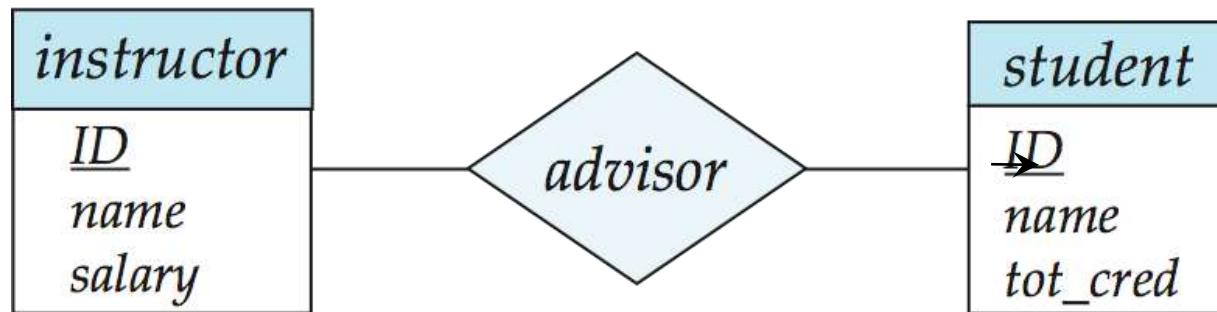
One-to-Many Relationship

- 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,



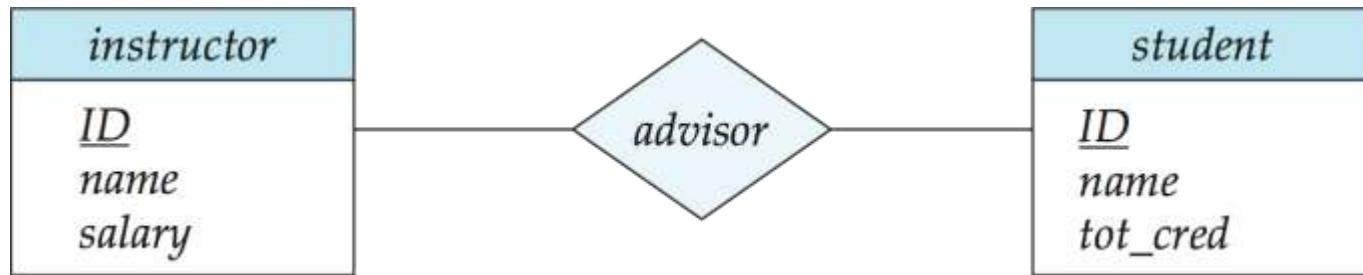
Many-to-One Relationships

- In a many-to-one relationship between an *instructor* and a *student*,
 - an *instructor* is associated with at most one *student* via *advisor*,
 - 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.



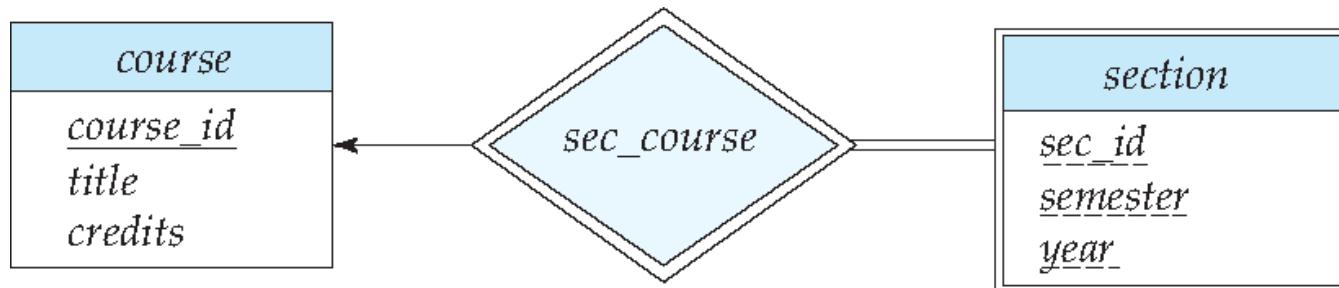
Instructor can advise 0 or more students. A student must have 1 advisor; cannot have multiple advisors

Notation to Express Entity with Complex Attributes

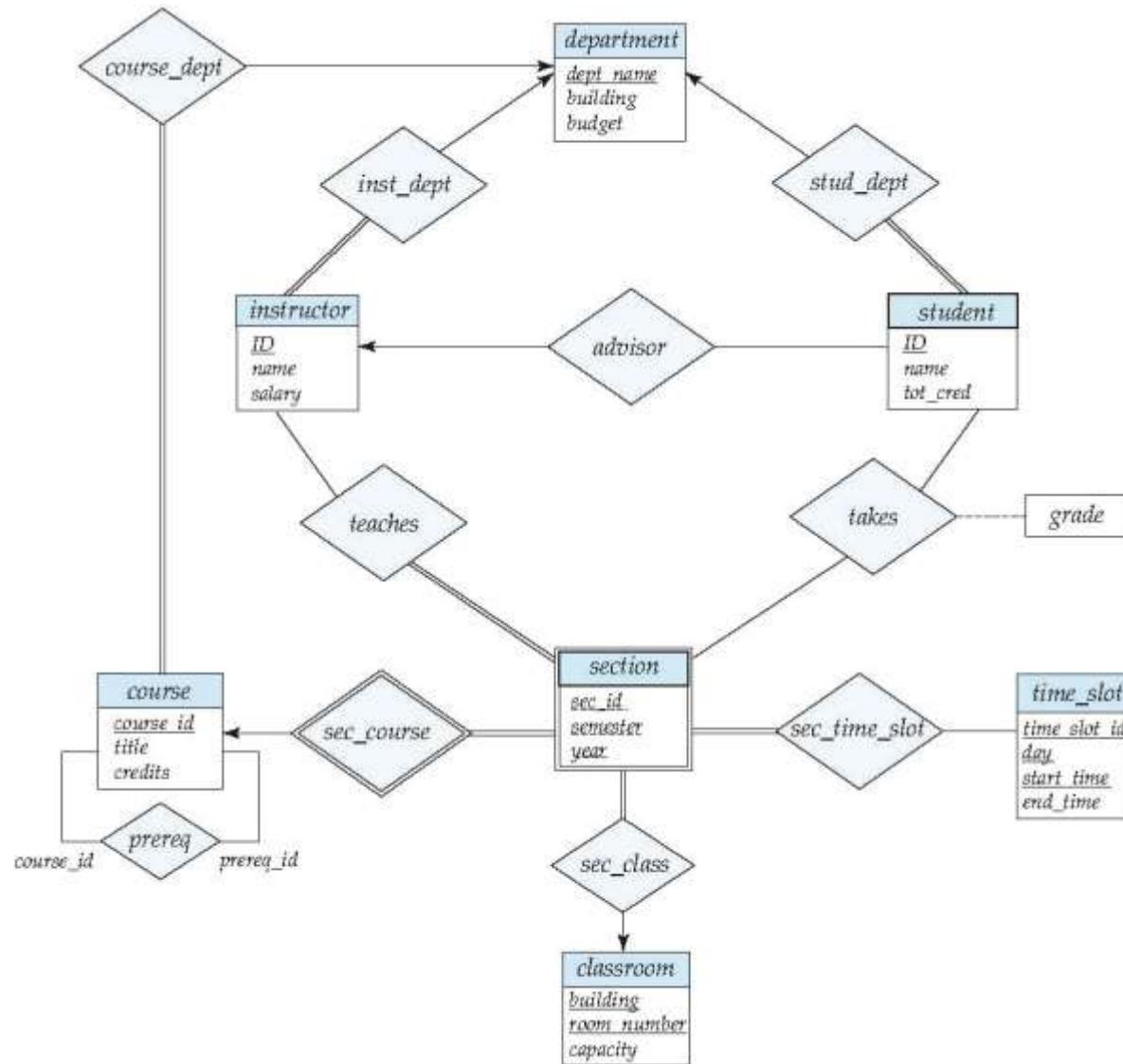
<i>instructor</i>
<i>ID</i>
<i>name</i>
<i>first_name</i>
<i>middle_initial</i>
<i>last_name</i>
<i>address</i>
<i>street</i>
<i>street_number</i>
<i>street_name</i>
<i>apt_number</i>
<i>city</i>
<i>state</i>
<i>zip</i>
{ <i>phone_number</i> }
<i>date_of_birth</i>
<i>age</i> ()

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*)



E-R Diagram for a University Enterprise



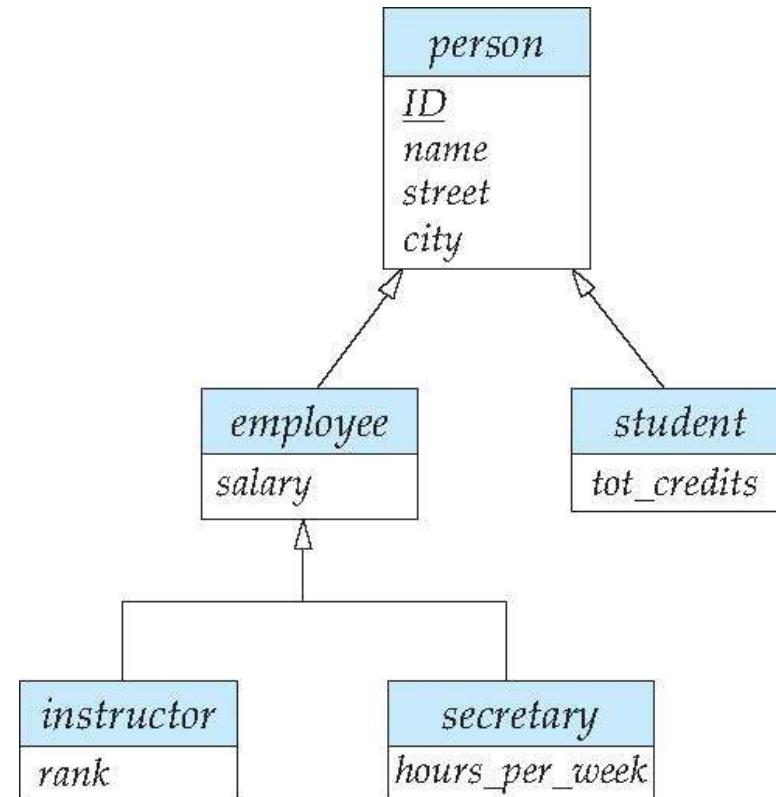
Specialization

- 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

- An entity set may include subgroupings of entities that are distinct in some way from other entities in the set. For instance, a subset of entities within an entity set may have attributes that are not shared by all the entities in the entity set. The E-R model provides a means for representing these distinctive entity groupings.
- The process of designating subgroupings within an entity set is called specialization

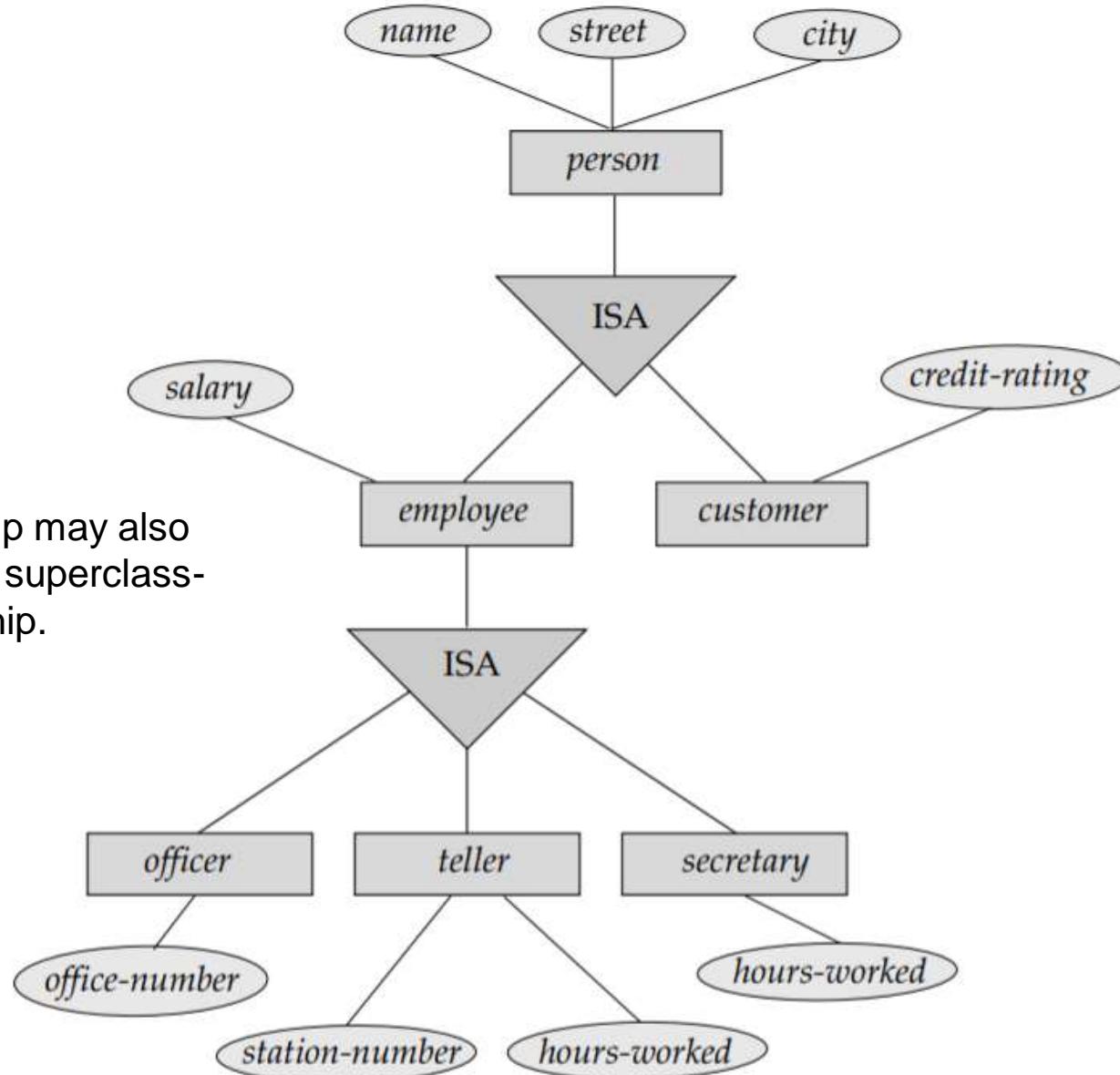
- **Overlapping** – *employee* and *student*
- **Disjoint** – *instructor* and *secretary*
- Total and partial



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.
- The refinement from an initial entity set into successive levels of entity subgroupings represents a top-down design process in which distinctions are made explicit. The design process may also proceed in a bottom-up manner, in which multiple entity sets are synthesized into a higher-level entity set on the basis of common features.
- This commonality can be expressed by generalization, which is a containment relationship that exists between a higher-level entity set and one or more lower-level entity sets.
- Higher- and lower-level entity sets also may be designated by the terms superclass and subclass, respectively

Another example



The ISA relationship may also be referred to as a superclass-subclass relationship.

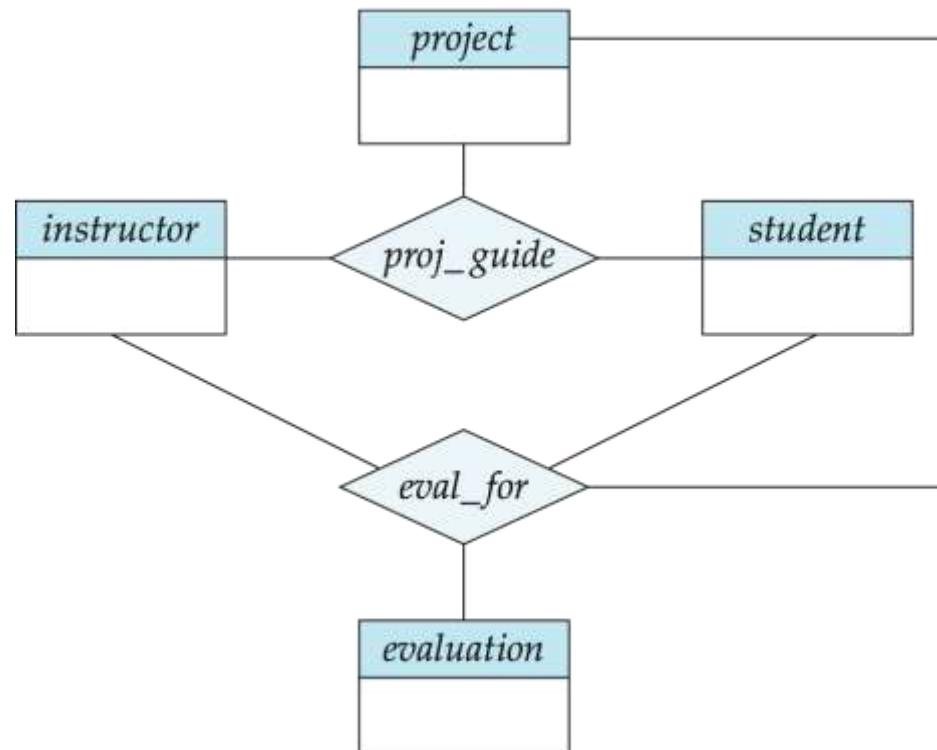
BASIS FOR COMPARISON	GENERALIZATION	SPECIALIZATION
Basic	It proceeds in a bottom-up manner.	It proceeds in a top-down manner.
Function	Generalization extracts the common features of multiple entities to form a new entity.	Specialization splits an entity to form multiple new entities that inherit some feature of the splitting entity.
Entities	The higher level entity must have lower level entities.	The higher level entity may not have lower level entities.
Size	Generalization reduces the size of a schema.	Specialization increases the size of a schema.
Application	Generalization entities on group of entities.	Specialization is applied on a single entity.
Result	Generalization results in forming a single entity from multiple entities.	Specialization results in forming the multiple entity from a single entity.

Cont..

- For all practical purposes, generalization is a simple inversion of specialization. We will apply both processes, in combination, in the course of designing the E-R schema for an enterprise.
- In terms of the E-R diagram itself, we do not distinguish between specialization and generalization. New levels of entity representation will be distinguished (specialization) or synthesized (generalization) as the design schema comes to express fully the database application and the user requirements of the database.
- Differences in the two approaches may be characterized by their starting point and overall goal. Specialization stems from a single entity set; it emphasizes differences among entities within the set by creating distinct lower-level entity sets. These lower-level entity sets may have attributes, or may participate in relationships, that do not apply to all the entities in the higher-level entity set. Indeed, the reason a designer applies specialization is to represent such distinctive features.
- Generalization proceeds from the recognition that a number of entity sets share some common features (namely, they are described by the same attributes and participate in the same relationship sets). On the basis of their commonalities, generalization synthesizes these entity sets into a single, higher-level entity set. Generalization is used to emphasize the similarities among lower-level entity sets and to hide the differences; it also permits an economy of representation in that shared attributes are not repeated.

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

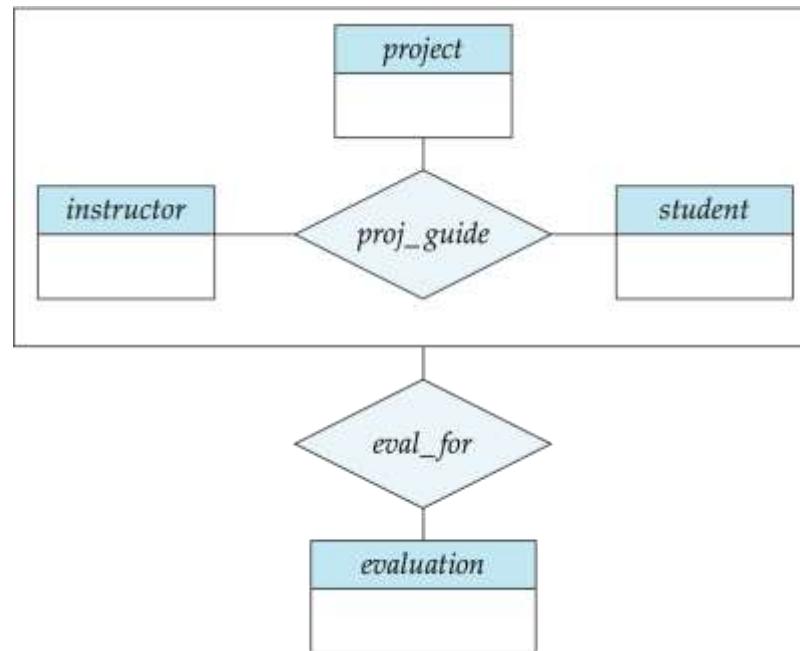


Aggregation (Cont.)

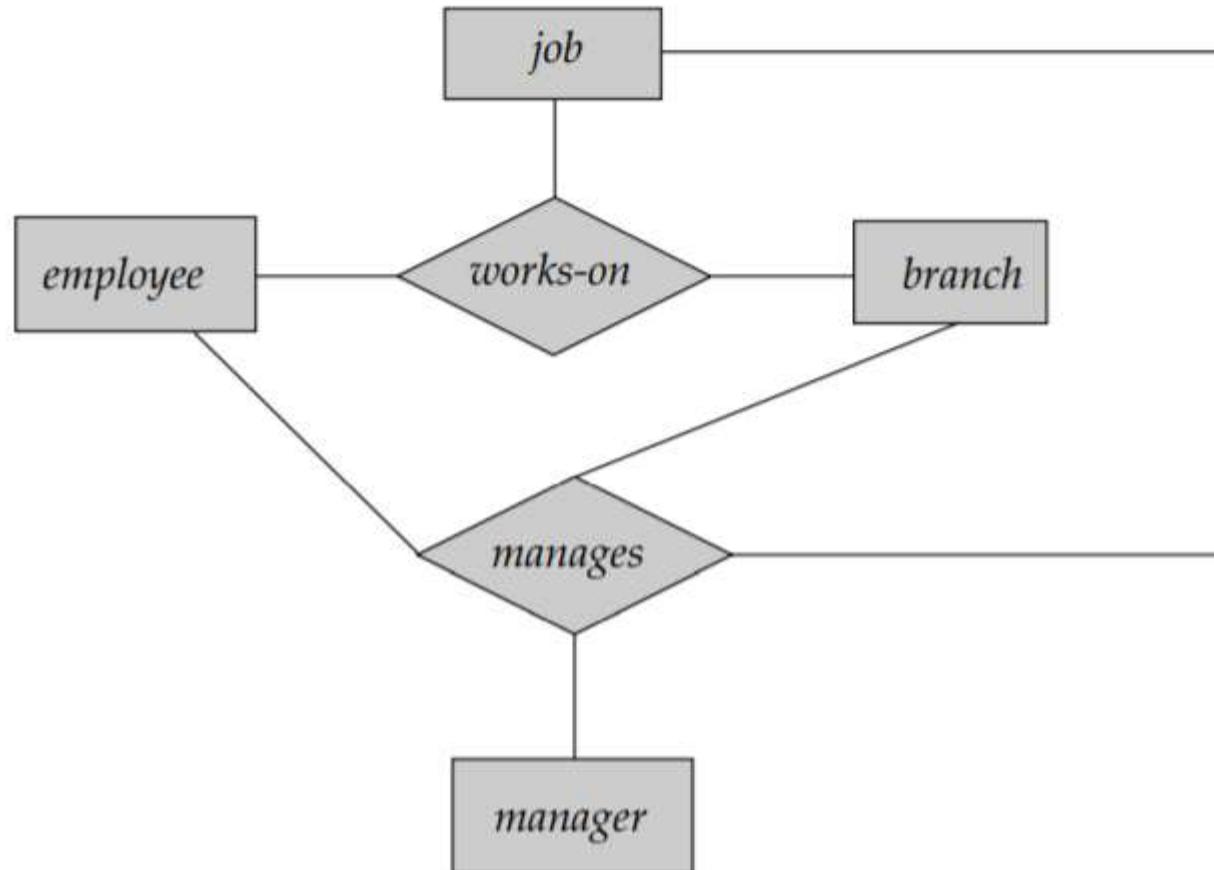
- Relationship sets *eval_for* and *proj_guide* represent overlapping information
 - Every *eval_for* relationship corresponds to a *proj_guide* relationship
 - However, some *proj_guide* relationships may not correspond to any *eval_for* relationships
 - ▶ So we can't discard the *proj_guide* relationship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

Aggregation (Cont.)

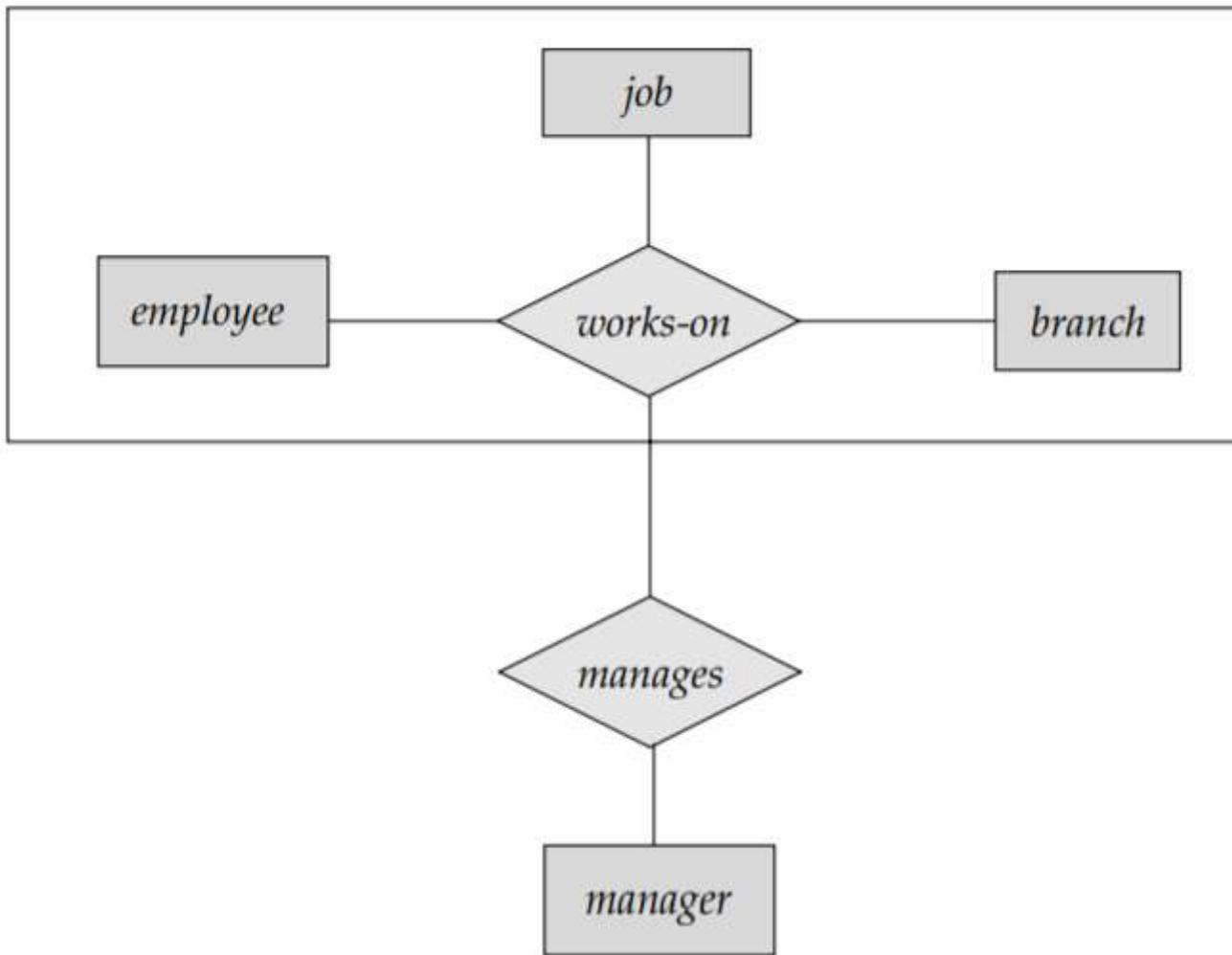
- Eliminate this redundancy via *aggregation* without introducing redundancy, the following diagram represents:
 - A student is guided by a particular instructor on a particular project
 - A student, instructor, project combination may have an associated evaluation



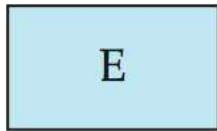
E-R diagram with redundant relationships



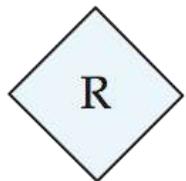
E-R diagram with aggregation



Summary of Symbols Used in E-R Notation



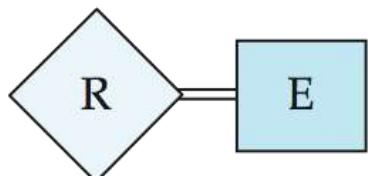
entity set



relationship set



identifying
relationship set
for weak entity set



total participation
of entity set in
relationship

E
A1
A2
A2.1
A2.2
{A3}
A4()

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

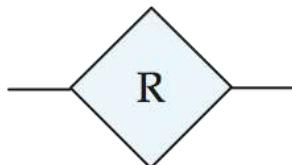
E
<u>A1</u>

primary key

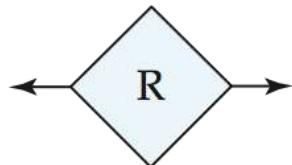
E
A1
.....

discriminating
attribute of
weak entity set

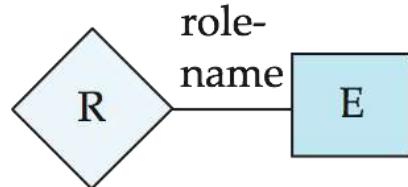
Symbols Used in E-R Notation (Cont.)



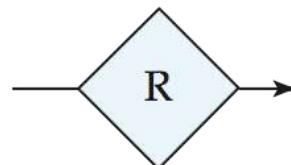
many-to-many
relationship



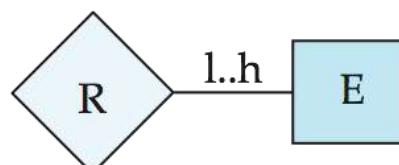
one-to-one
relationship



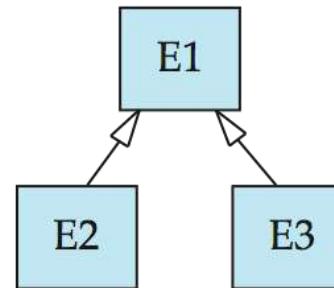
role indicator



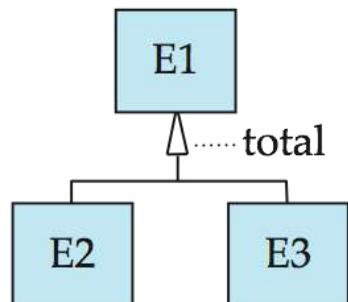
many-to-one
relationship



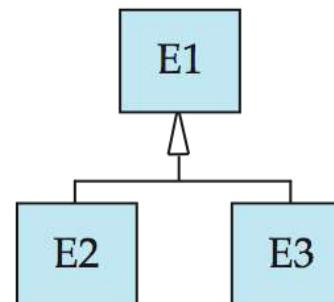
cardinality
limits



ISA: generalization
or specialization



total (disjoint)
generalization

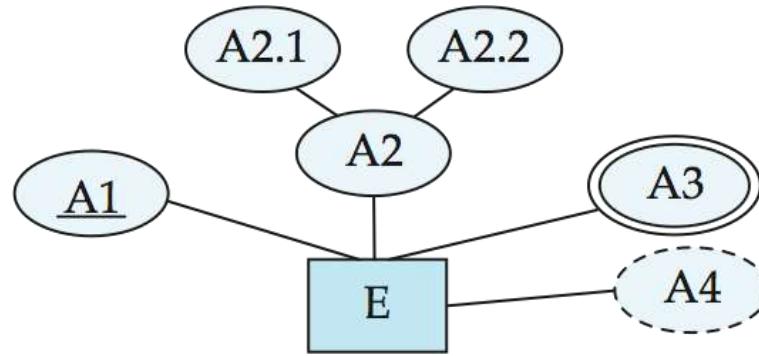


disjoint
generalization

Alternative ER Notations

- Chen, IDE1FX, ...

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



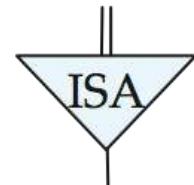
weak entity set



generalization



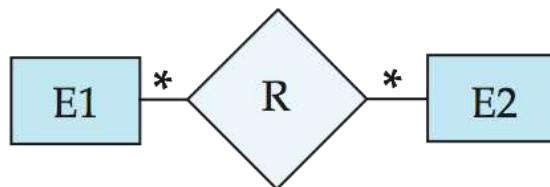
total
generalization



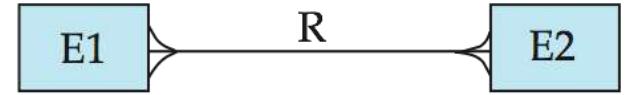
Alternative ER Notations

Chen

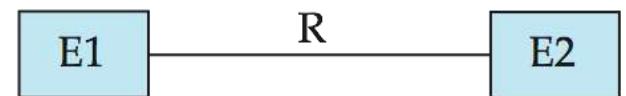
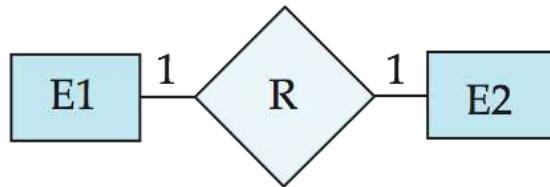
many-to-many
relationship



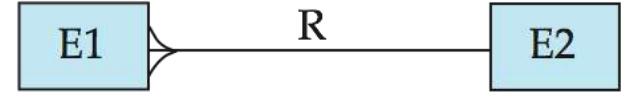
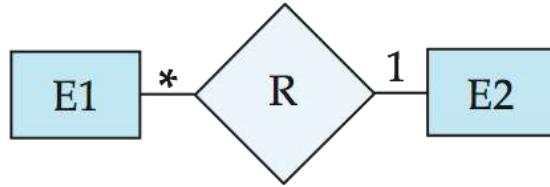
IDE1FX (Crows feet notation)



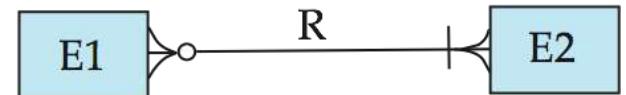
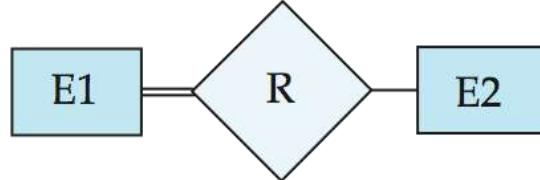
one-to-one
relationship



many-to-one
relationship



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



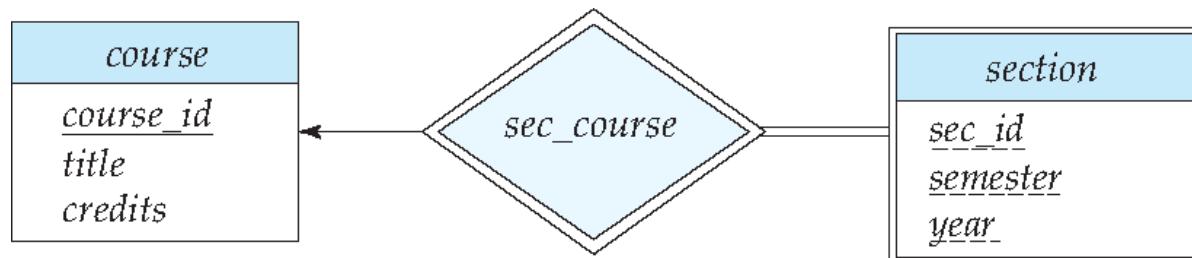
Reduction to Relation Schemas (not mentioned in syllabus but you must know)

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

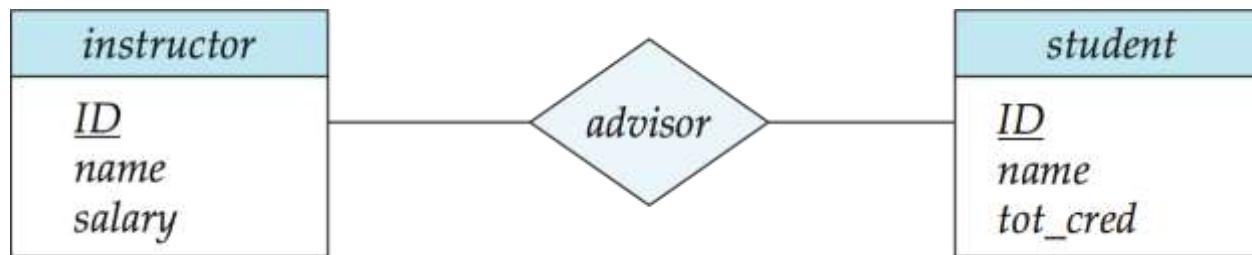
- A strong entity set reduces to a schema with the same attributes
 $\text{student}(\underline{ID}, \text{name}, \text{tot_cred})$
- A weak entity set becomes a table that includes a column for the primary key of the identifying strong entity set
 $\text{section}(\underline{\text{course_id}}, \underline{\text{sec_id}}, \text{sem}, \text{year})$



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)



Representation of Entity Sets with Composite Attributes

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

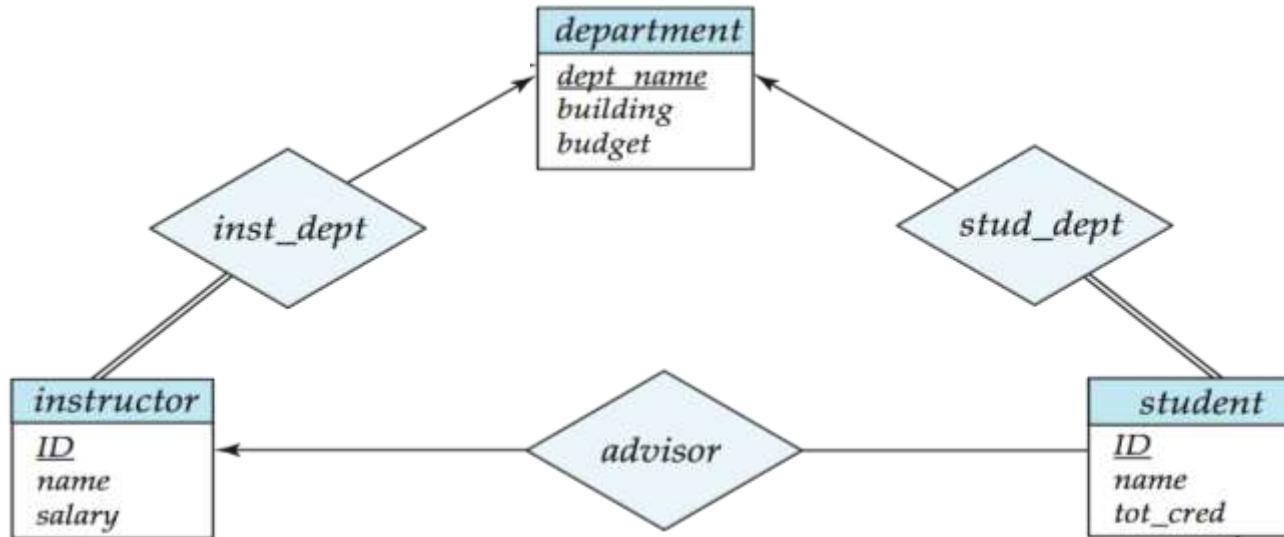
- Composite attributes are flattened out by creating a separate attribute for each component attribute
 - Example: given entity set *instructor* with composite attribute *name* with component attributes *first_name* and *last_name* the schema corresponding to the entity set has two attributes *name_first_name* 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)*

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:
 $inst_phone = (\underline{ID}, \underline{phone_number})$
- 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)

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*

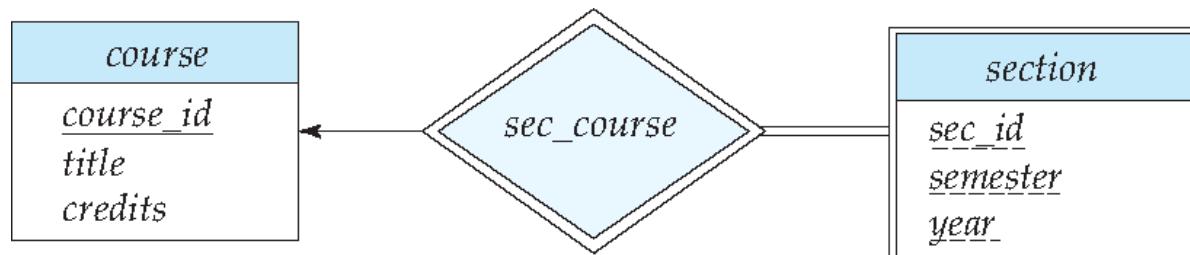


Redundancy of Schemas (Cont.)

- 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

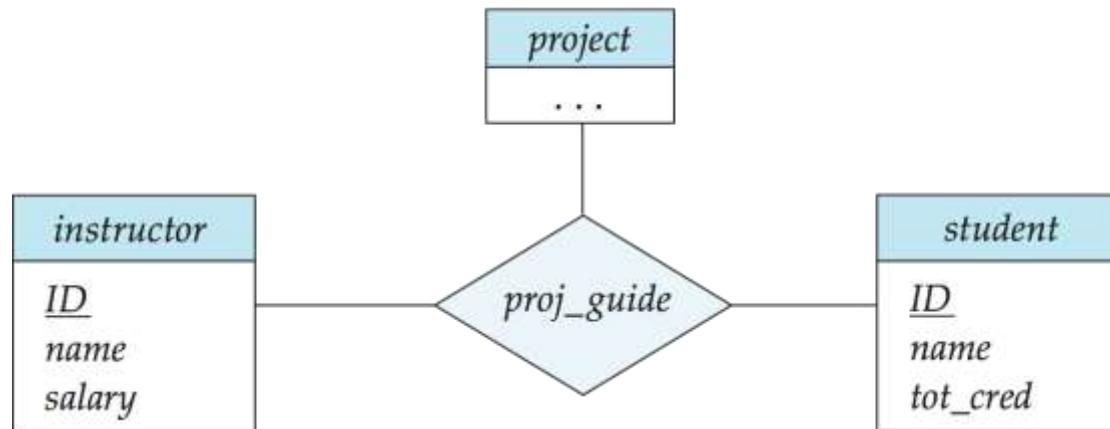
Redundancy of Schemas (Cont.)

- 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



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



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
- 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

Thanks

For any qureries you can watsapp or call

Dr Arpita Mathur

BCA313

Unit 2

Entity-Relationship Model

Dr Arpita Mathur
Assistant Professor, Faculty of Computer Science, LMCST,
Jodhpur

Entity-Relationship Model

- Entity
- Attributes
- Relationship
- Constraints
- E-R Diagram
- Keys
- Weak Entity Sets
- Specialization and Generalization
- Aggregation

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.
- 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. 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

ER Model

- The ER data mode was developed to facilitate database design by allowing specification of an **enterprise schema** that represents the overall logical structure of a database.
- The ER model is **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.
- The ER model also has an associated diagrammatic representation, the ER diagram, which can express the overall logical structure of a database graphically.

Entity Sets

- An **entity** is any real world object that 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, street, city, salary)
course=(course_id, title, credits)
- Domain or value is a set of permitted values for each attribute.

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

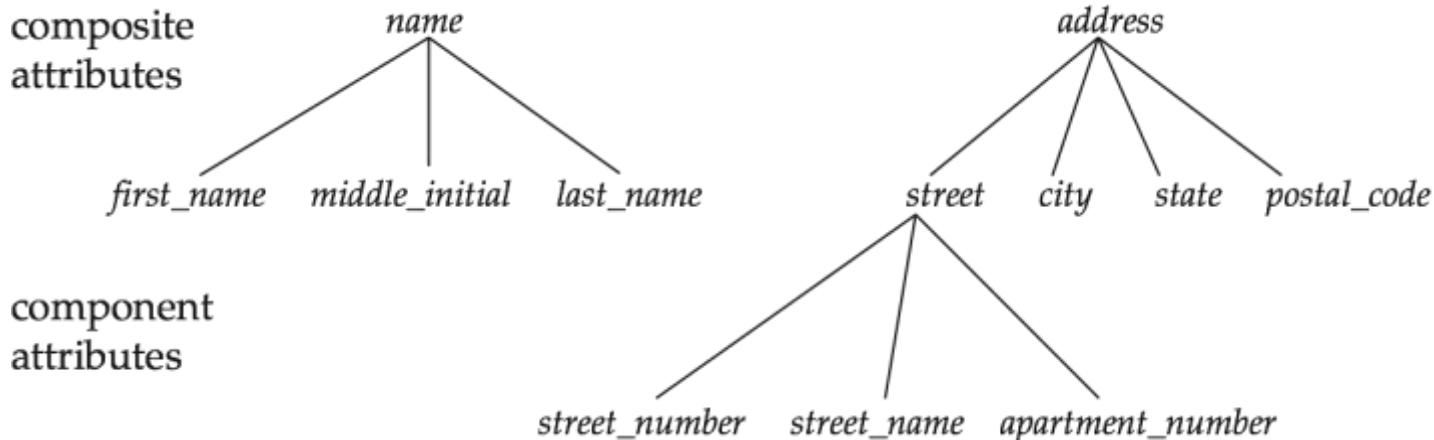
98988	Tanaka
12345	Shankar
00128	Zhang
76543	Brown
76653	Aoi
23121	Chavez
44553	Peltier

student

Types of 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*



- ## ■ An attribute can take a **null** value when an entity does not have a value for it. Null value indicate either the value does not exist or is unknown.

Relationship Sets

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

Example:

44553 (Amit) advisor 22222 (Nitin)
student entity relationship set instructor entity

- A **relationship set** is a set of relationships of the same type. It is a mathematical relation among $n \geq 2$ entities, each taken from entity sets

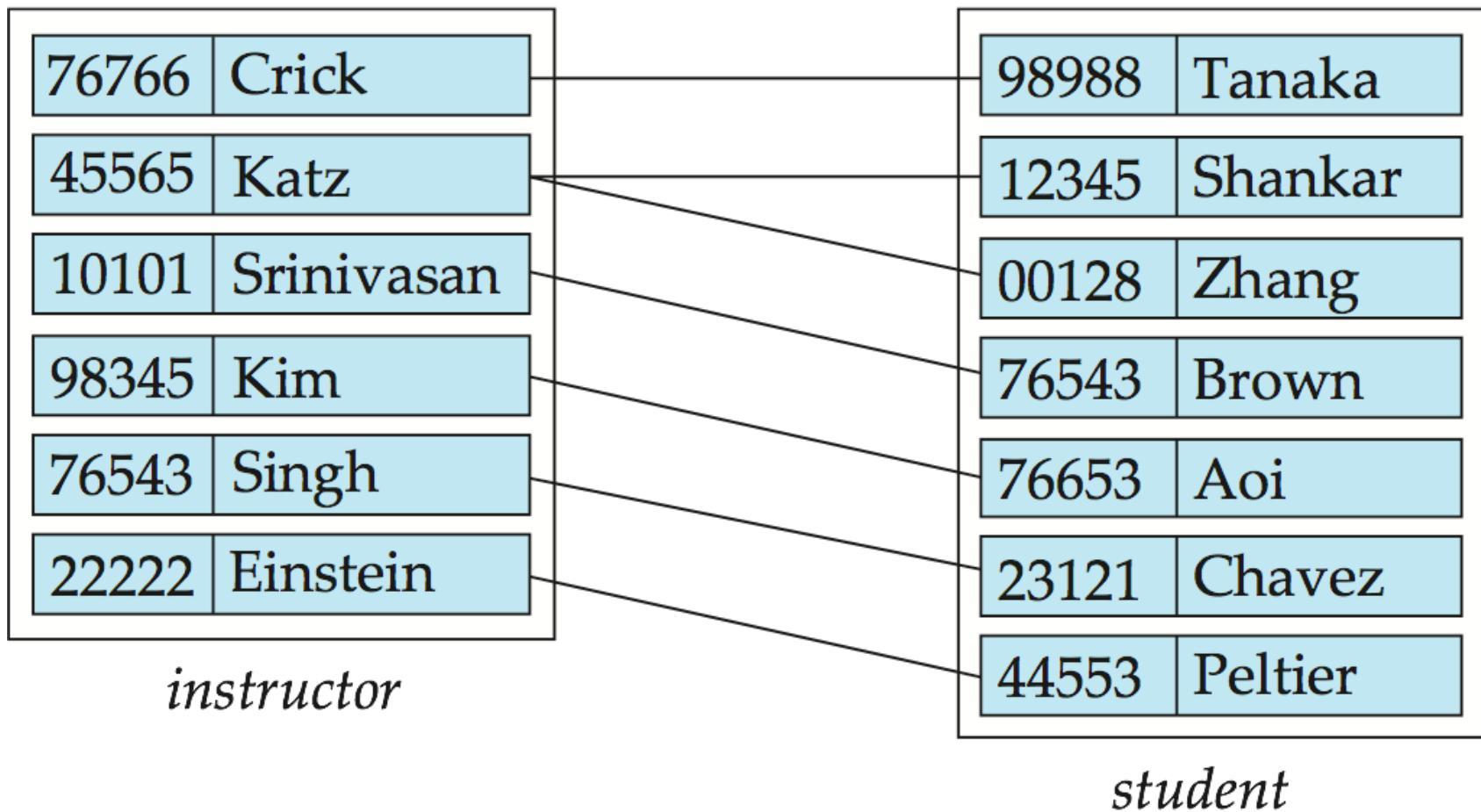
$$\{(e_1, e_2, \dots, e_n) \mid e_1 \in E_1, e_2 \in E_2, \dots, e_n \in E_n\}$$

where (e_1, e_2, \dots, e_n) is a relationship

- Example:

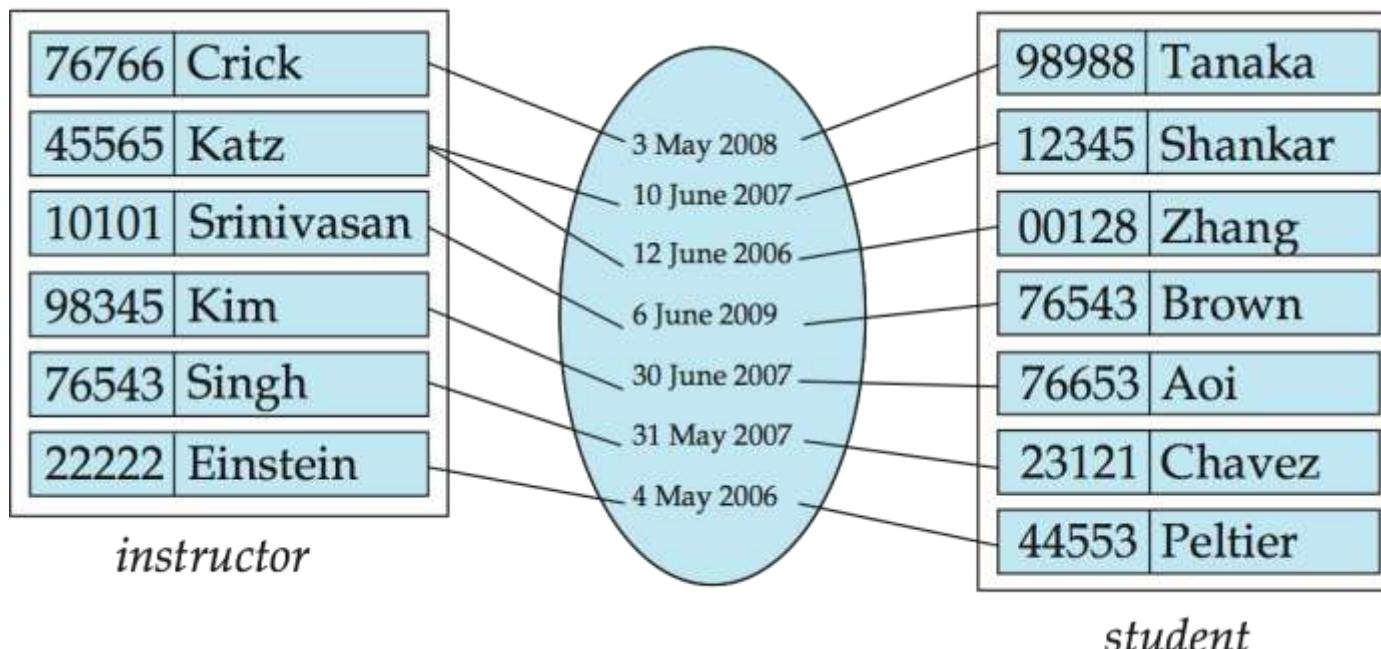
$$(44553, 22222) \in \text{advisor}$$

Relationship Set *advisor*



Relationship Sets (Cont.)

- An attribute can also be associated with a relationship set. It is known as **descriptive attribute**.
- 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.
- 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 *instructor*, *student*, and *project*

Constraints

An E R enterprise schema may define certain constraints to which the contents of a database must conform.

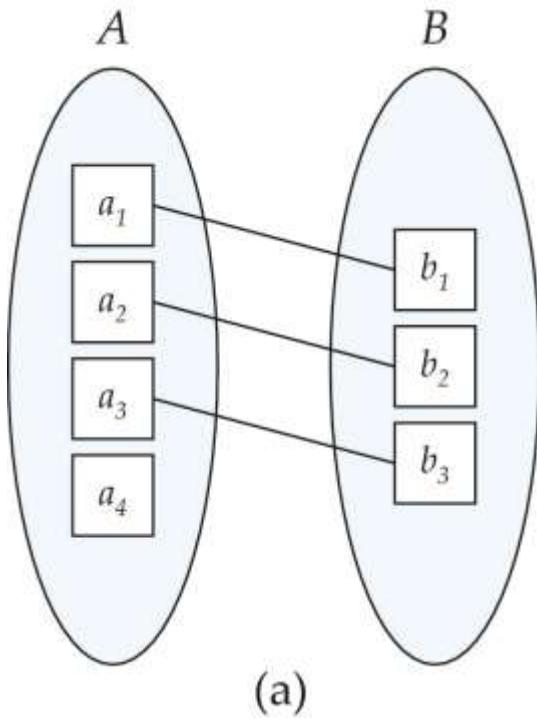
1. Mapping Cardinalities

- 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: An entity in A is associated with **at most one entity** in B.
 - One to many
 - Many to one
 - Many to many

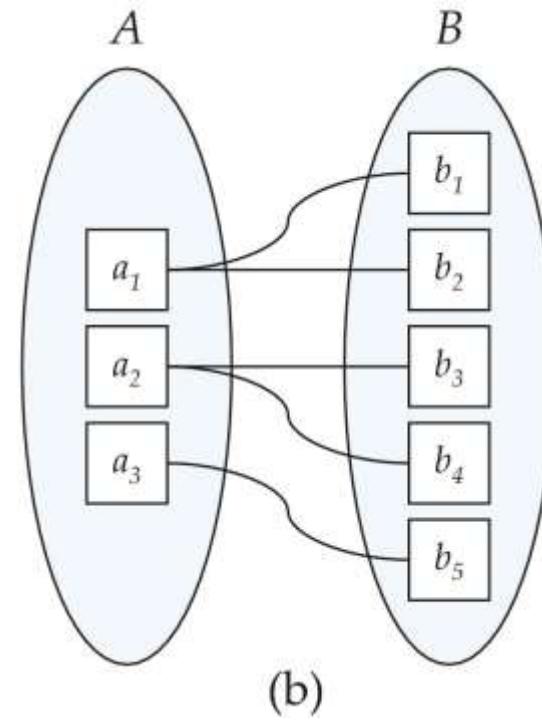
2. Participation constraints

- Total participation: The participation of an entity set E in a relationship set R is total if 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

Mapping Cardinalities



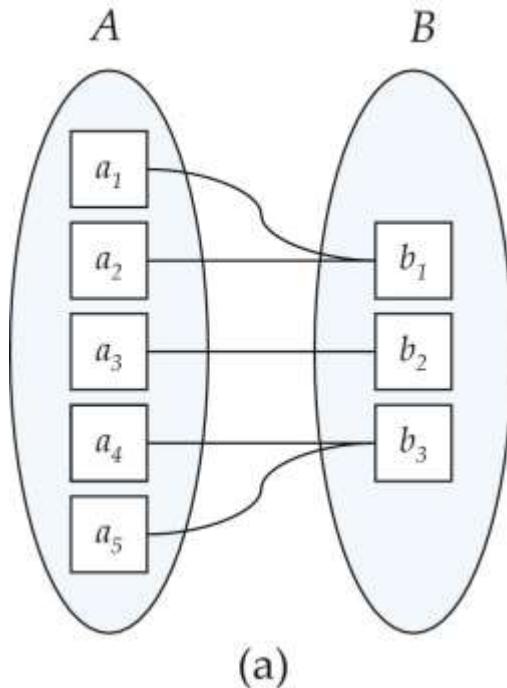
One to one



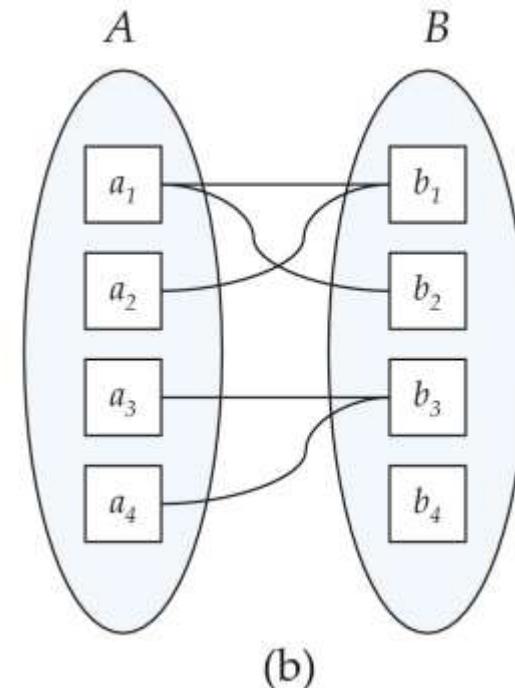
One to many

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

Keys

No two entities in an entity set are allowed to have exactly the same value for all attributes. A **key** allows us to identify a set of attributes that suffice to distinguish entities from each other. Keys also help uniquely identify relationships, and thus distinguish relationships from each other.

- **Super Key:** A superkey is a set of one or more attributes that, taken collectively, allow us to identify uniquely an entity in the entity set.
- **Candidate Key:** Minimal superkeys (superkeys for which no proper subset is a superkey) are called candidate keys.
- **Primary Key:** Candidate key that is chosen by the database designer as the principal means of identifying entities within an entity set is known as primary key. Remaining candidate keys are known as **alternate keys**.

A key (primary, candidate, and super) is a property of the entity set, rather than of the individual entities. Any two individual entities in the set are prohibited from having the same value on the key attributes at the same time. The designation of a key represents a constraint in the real-world enterprise being modeled.

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 *sec_course*; however, by doing so the relationship between *section* and *course* becomes implicit in an attribute, which is not desirable.

Weak Entity Sets (Cont.)

- 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; although each *section* entity is distinct, sections for different courses may share the same *section_id*, *year*, and *semester*.
- 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.
- The notion of **weak entity set** formalizes the above intuition. 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. An entity set that is not a weak entity set is termed a **strong entity set**.

Weak Entity Sets (Cont.)

- Every weak entity must be associated with an identifying entity; that is, the weak entity set is said to be **existence dependent** on the identifying entity set. The identifying entity set is said to **own** the weak entity set that it identifies. The relationship associating the weak entity set with the identifying entity set is called the **identifying relationship**.
- 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*.

E-R Diagrams

Entity Sets

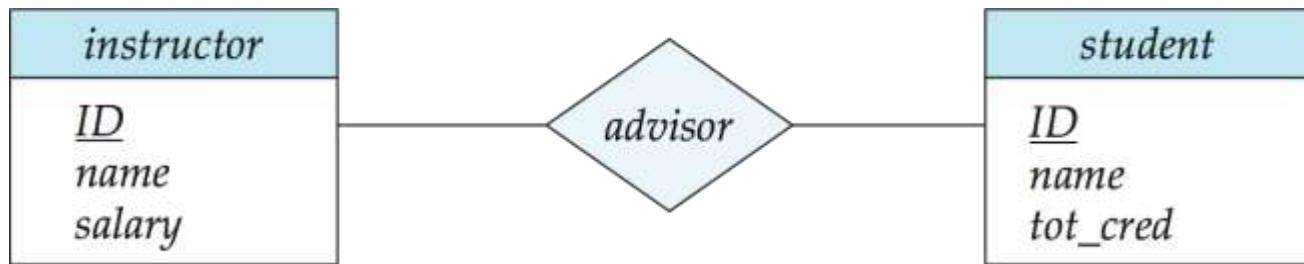
- Entities can be represented graphically as follows:
 - Rectangles represent entity sets.
 - Attributes listed inside entity rectangle
 - Underline indicates primary key attributes

<i>instructor</i>
<u>ID</u>
<i>name</i>
<i>salary</i>

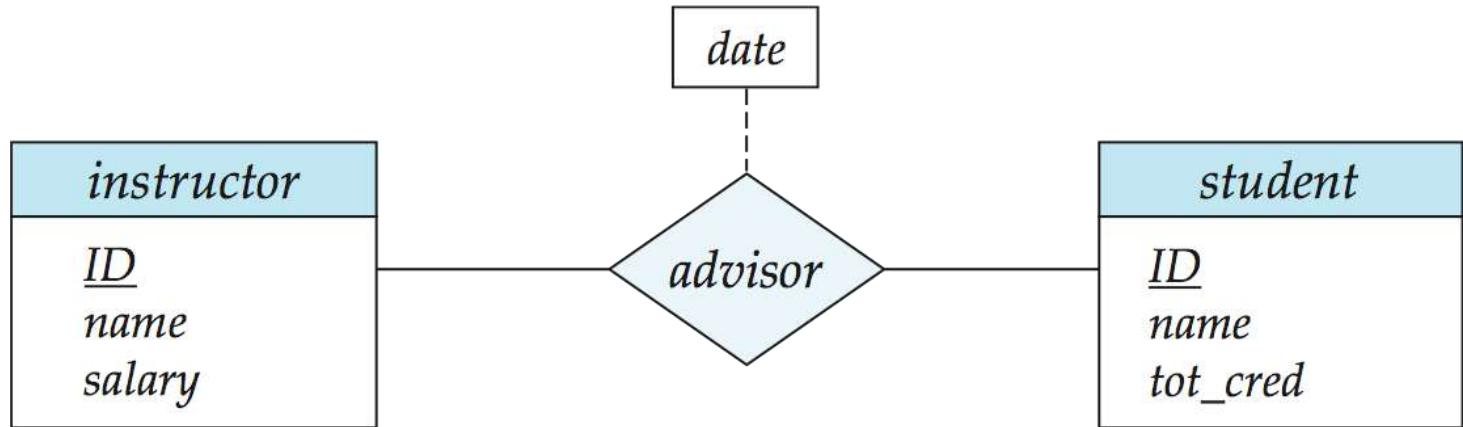
<i>student</i>
<u>ID</u>
<i>name</i>
<i>tot_cred</i>

Relationship Sets

- Diamonds represent relationship sets.

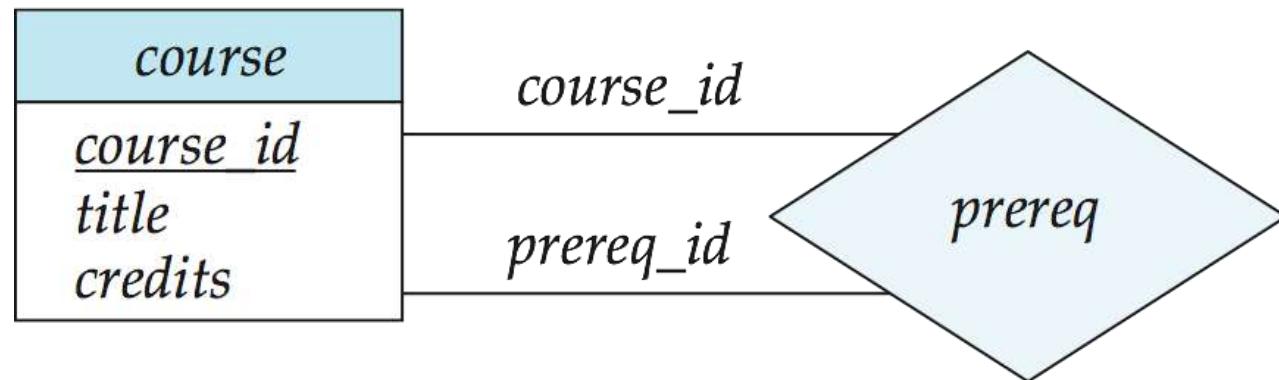


Relationship Sets with Attributes



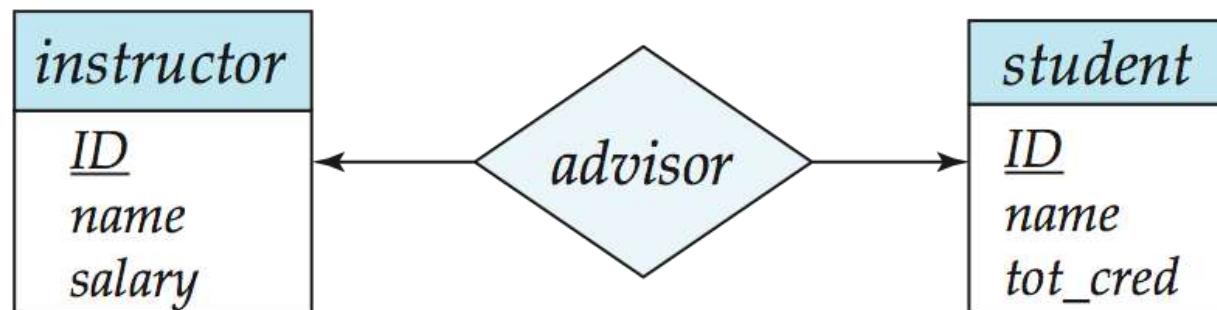
Roles

- 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**.



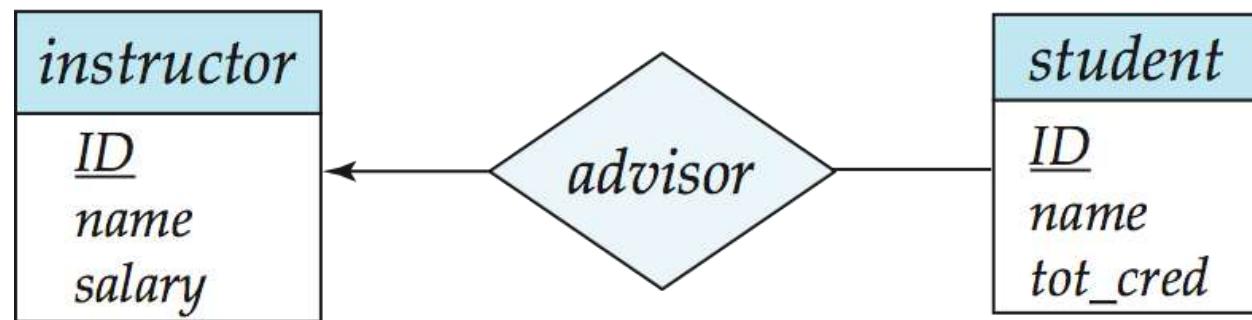
Cardinality Constraints

- 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*
 - A *student* is associated with at most one *department* via *stud_dept*



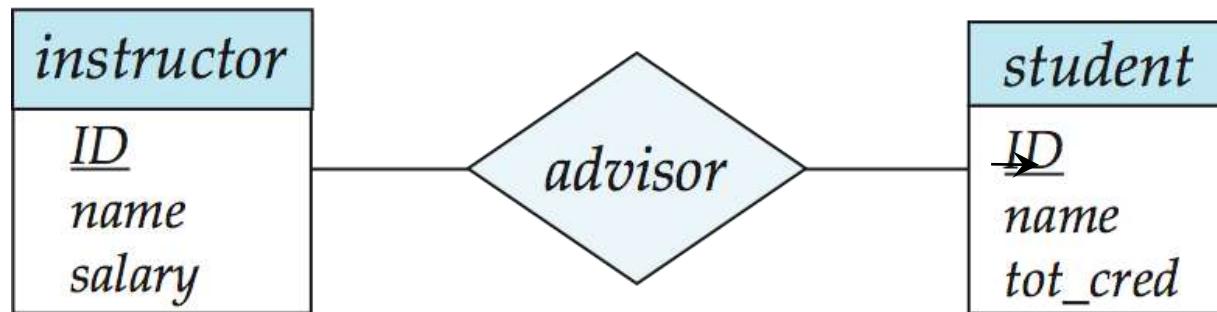
One-to-Many Relationship

- 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,



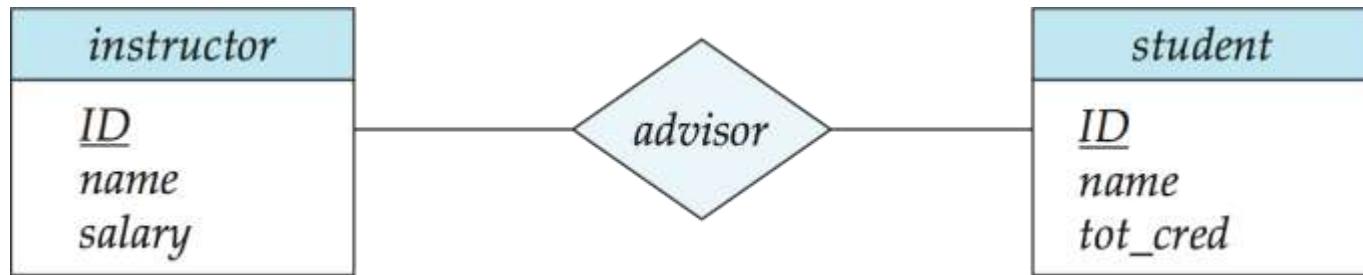
Many-to-One Relationships

- In a many-to-one relationship between an *instructor* and a *student*,
 - an *instructor* is associated with at most one *student* via *advisor*,
 - 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.



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

Notation to Express Entity with Complex Attributes

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