



Chapter 7: Entity-Relationship Model

Database System Concepts, 6th Ed.

©Silberschatz, Korth and Sudarshan

See www.db-book.com for conditions on re-use



Chapter 7: Entity-Relationship Model

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



Modeling

- A *database* can be modeled as:
 - a collection of entities,
 - relationship among entities.
- An **entity** is an object that exists and is distinguishable from other objects.
 - Example: specific person, company, event, plant
- Entities have **attributes**
 - Example: people have *names* and *addresses*
- An **entity set** is a set of entities of the same type that share the same properties.
 - Example: set of all persons, companies, trees, holidays



Entity Sets *instructor* and *student*

ID name

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

instructor

ID name

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

student



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

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

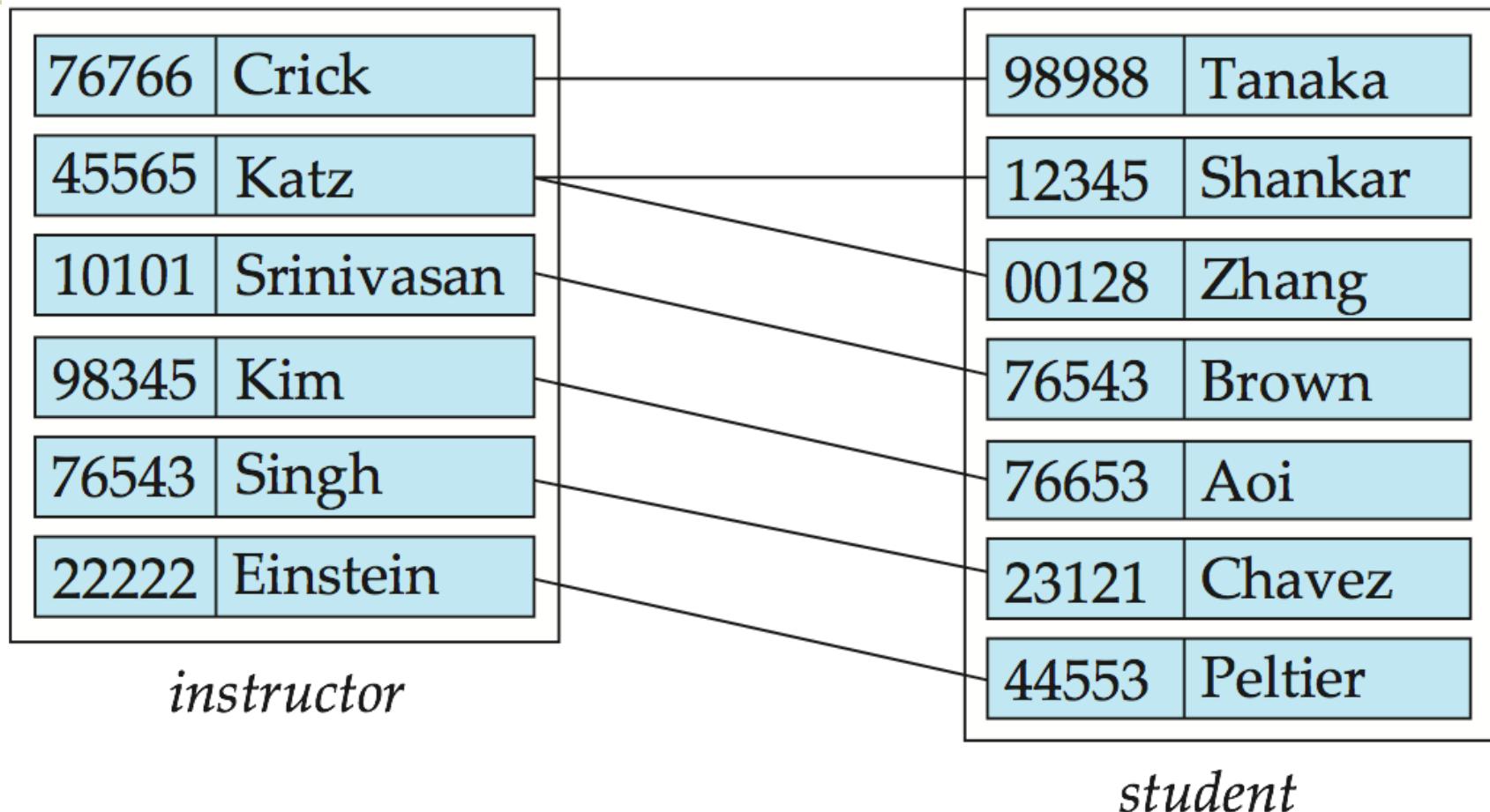
where (e_1, e_2, \dots, e_n) is a relationship

- Example:

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

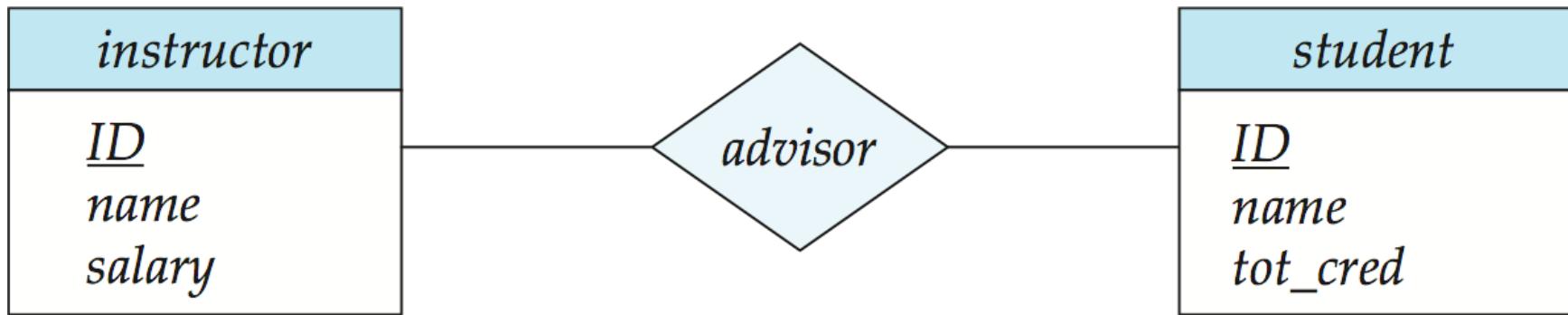


Relationship Set *advisor*





E-R Diagrams



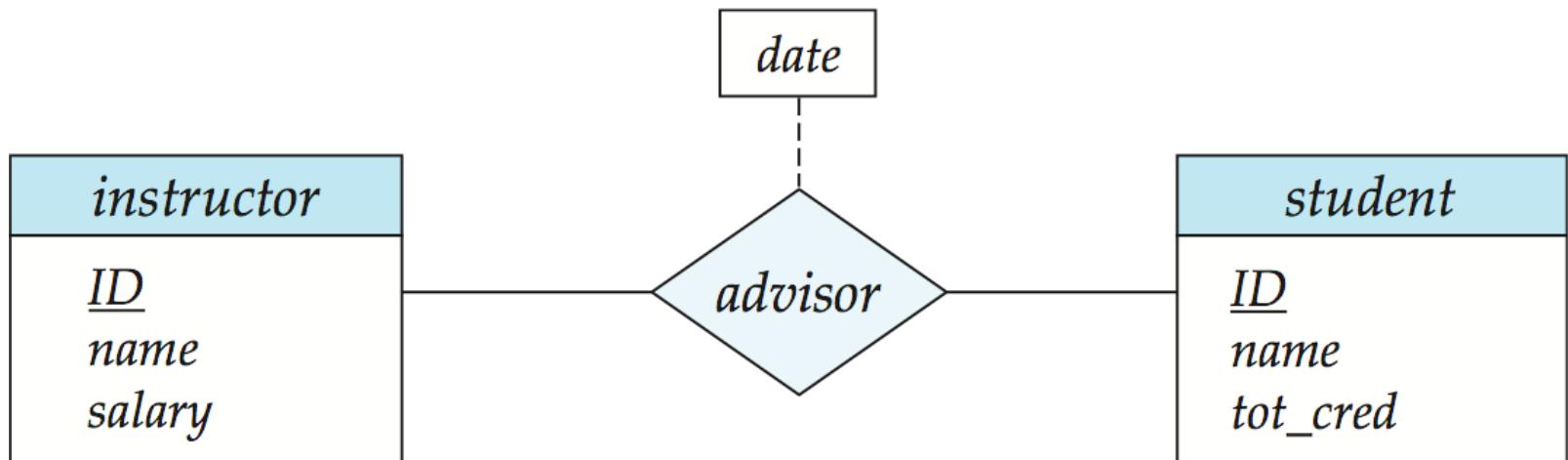
- Rectangles represent entity sets.
- Diamonds represent relationship sets.
- Attributes listed inside entity rectangle
- Underline indicates primary key attributes

Note: ER diagram notation in 6th edition of Database System Concepts changed from earlier editions; now based on UML class diagram notation with some modifications.



Relationship Sets with Attributes

- An attribute can also be property of a relationship set.
- For instance, the advisor relationship set between entity sets instructor and student may have the attribute date
 - E.g. date may track when the student started being associated with the advisor



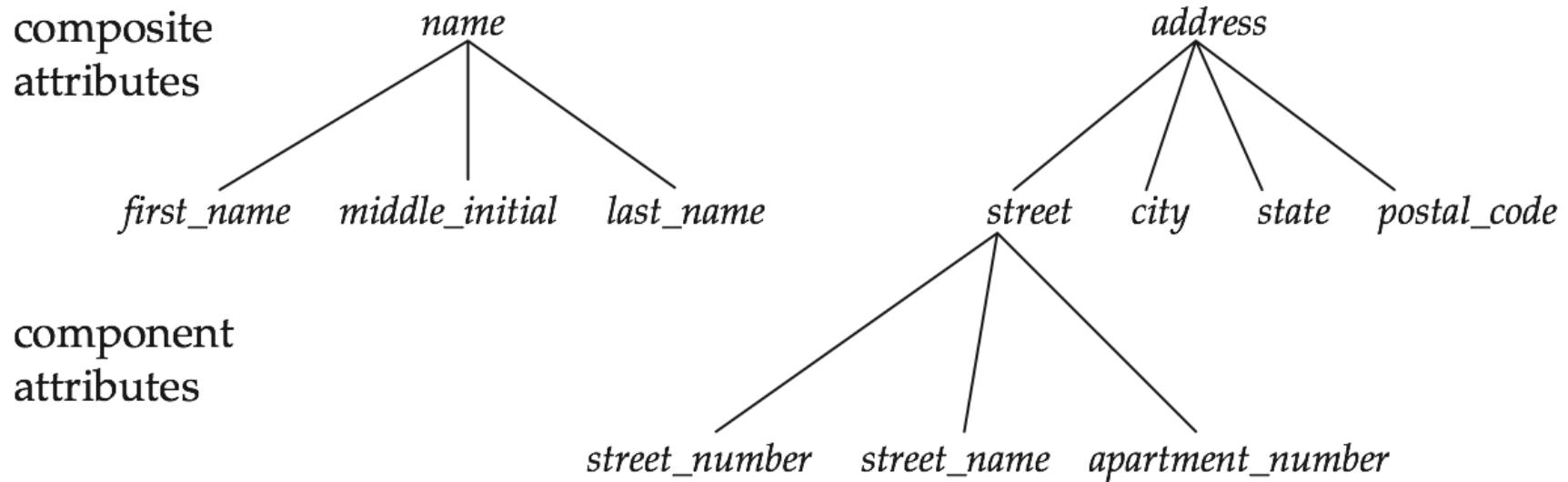


Attributes

- An entity is represented by a set of attributes, that is descriptive properties possessed by all members of an entity set.
 - Example:
 $\text{instructor} = (\text{ID}, \text{name}, \text{street}, \text{city}, \text{salary})$
 $\text{course} = (\text{course_id}, \text{title}, \text{credits})$
- **Domain** – the set of permitted values for each attribute
- 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

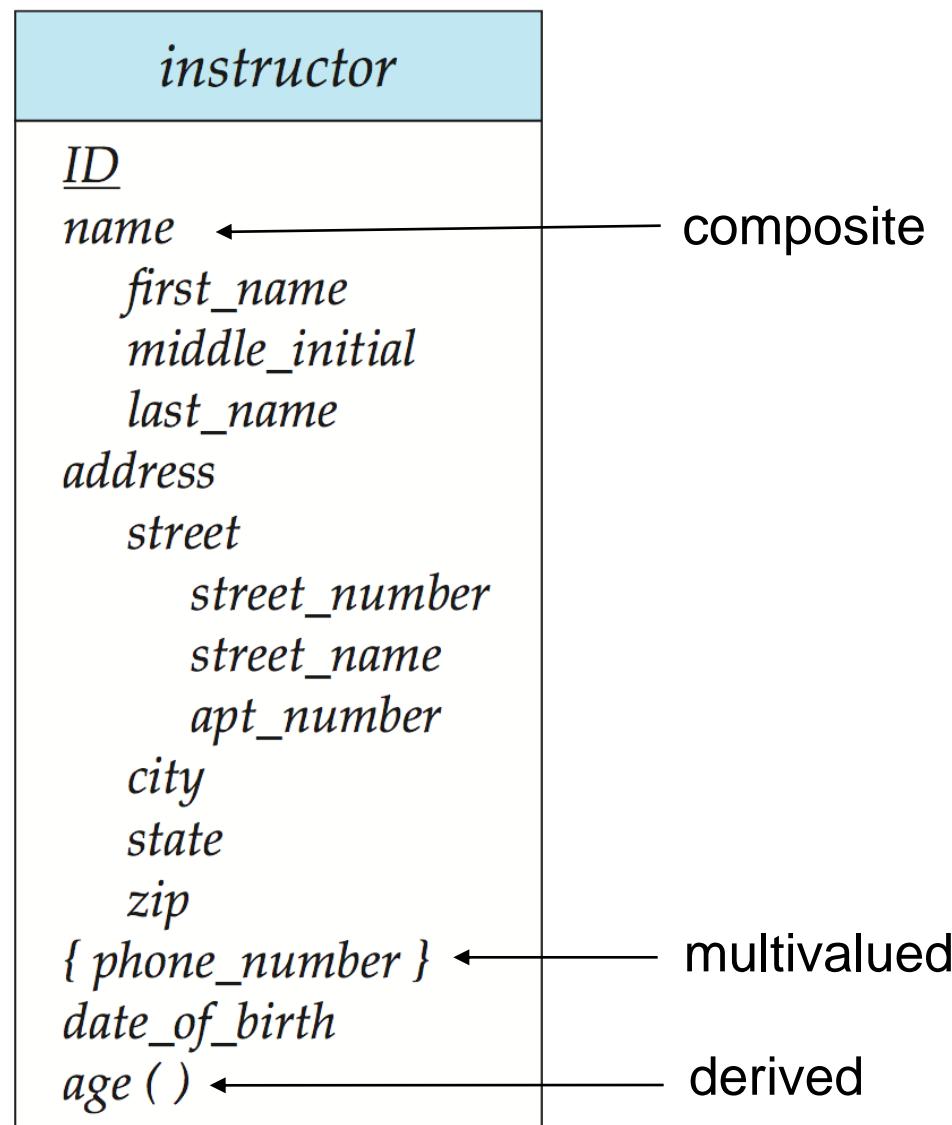


Composite Attributes





Entity With Composite, Multivalued, and Derived Attributes





Degree of a Relationship Set

■ binary relationship

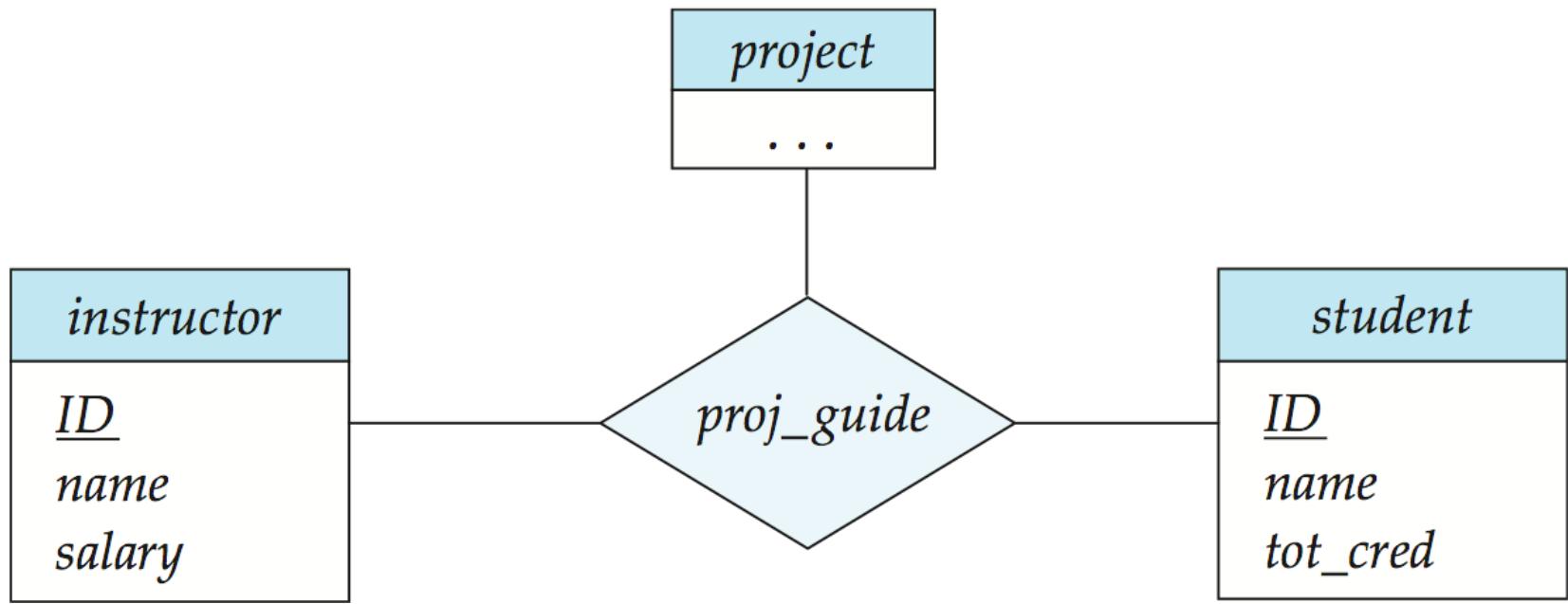
- involve two entity sets (or degree two).
- most relationship sets in a database system are binary.

■ 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*



E-R Diagram with a Ternary Relationship





Quiz Time

Quiz Q1: Suppose are given a person entity set, and we wish to represent the relationship between people and their father and mother. Which of these would be an appropriate representation, if for some people we only know either their father or their mother but not both?

- (1) two binary relationships, father and mother, between persons
- (2) a ternary relationship between persons, linking a person to his/her father and mother
- (3) an entity set parent, linked by three relationships to person, representing person, father and mother.
- (4) an entity set with attributes person, father and mother

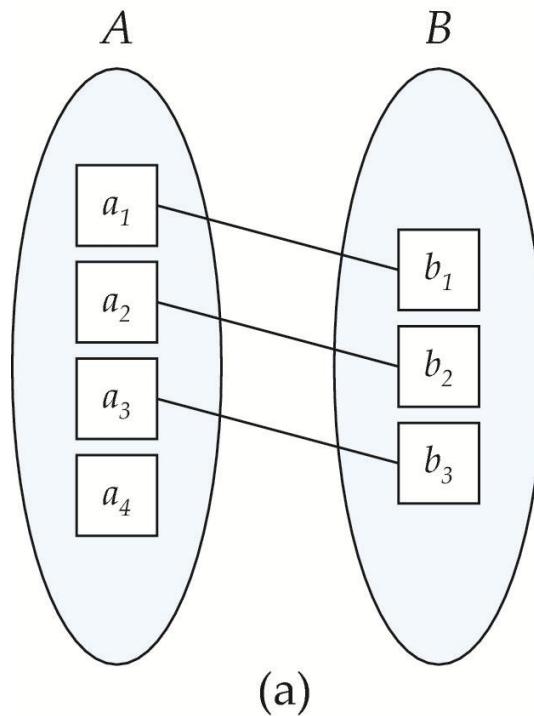


Mapping Cardinality Constraints

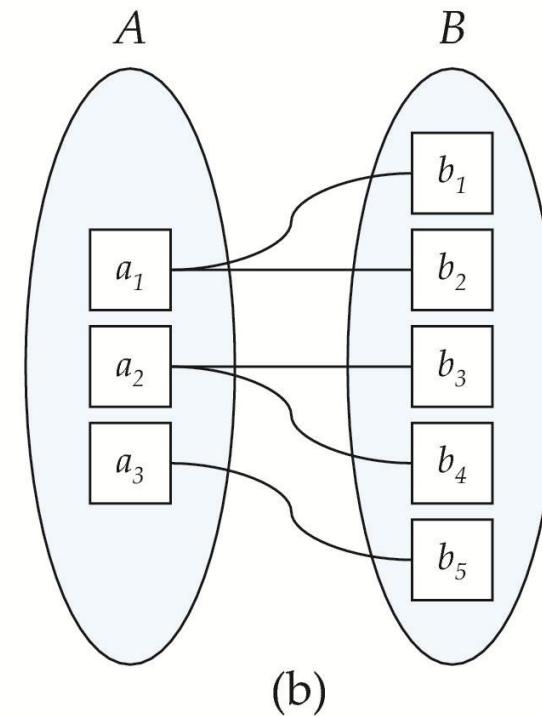
- Express the number of entities to which another entity can be associated via a relationship set.
- Most useful in describing binary relationship sets.
- For a binary relationship set the mapping cardinality must be one of the following types:
 - One to one
 - One to many
 - Many to one
 - Many to many



Mapping Cardinalities



One to one

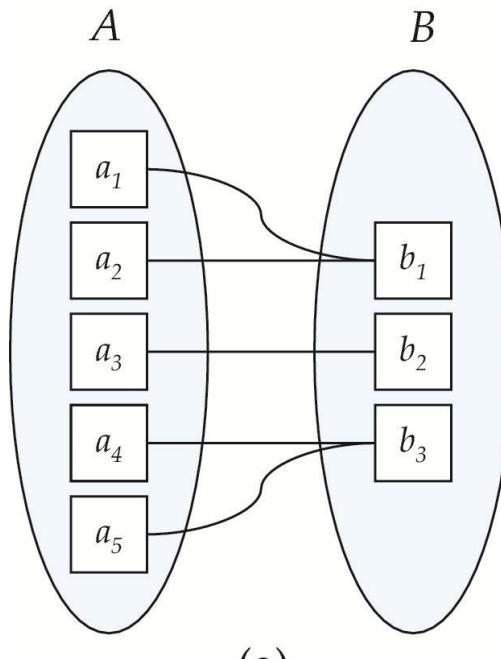


One to many

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

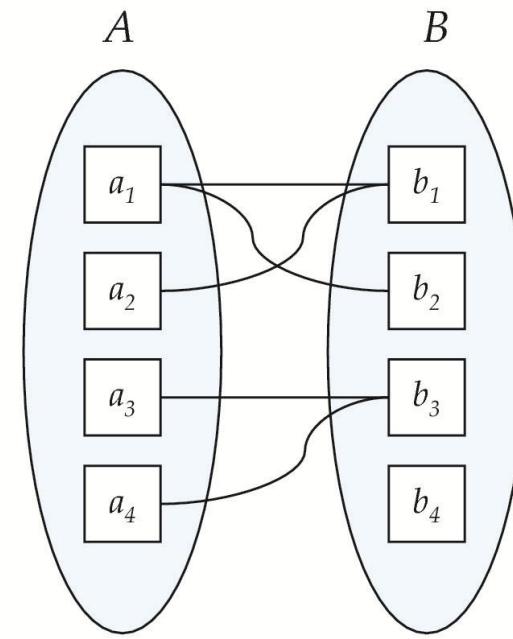


Mapping Cardinalities



(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



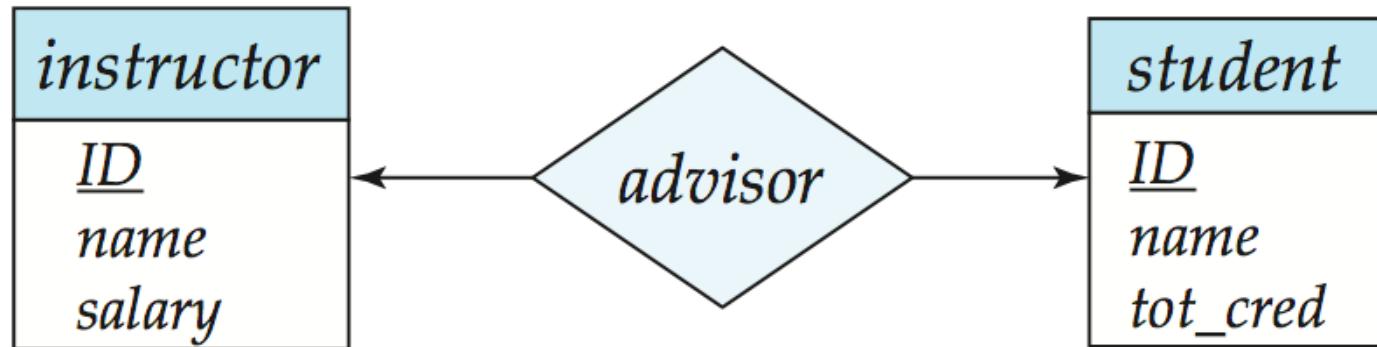
ER Notation for Cardinality Constraints

- 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 relationships:
 - 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-One Relationship

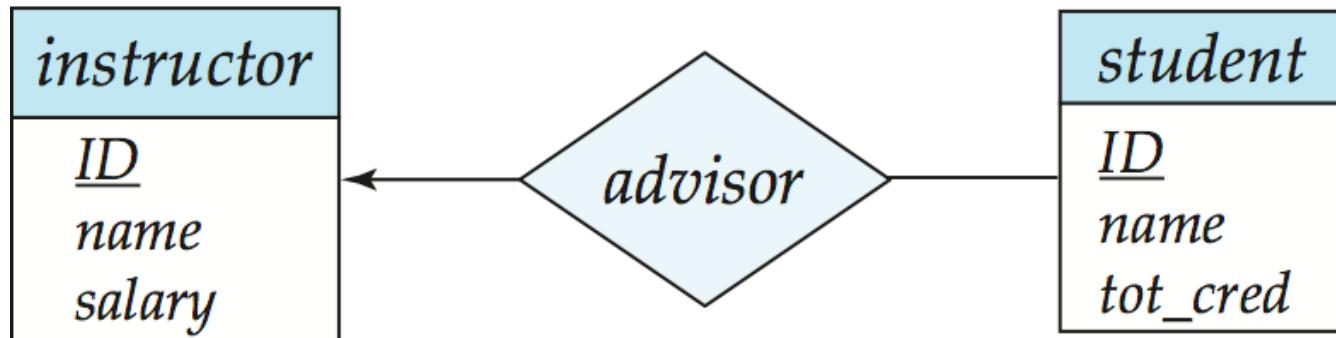
- one-to-one relationship between an *instructor* and a *student*
 - an instructor is associated with at most one student via *advisor*
 - and a student is associated with at most one instructor via *advisor*





One-to-Many Relationship

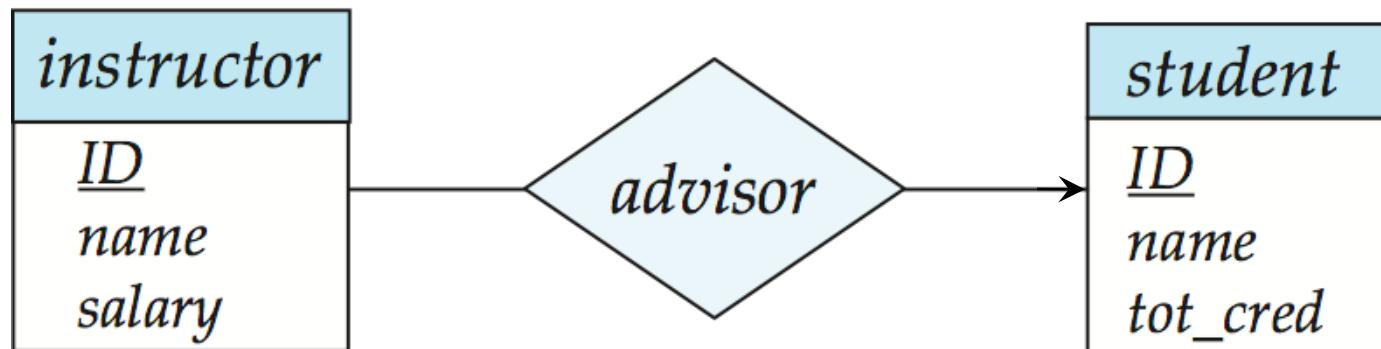
- one-to-many relationship between an *instructor* and a *student*
 - an instructor is associated with several (including 0) students via *advisor*
 - a student is associated with at most one instructor via *advisor*





Many-to-One Relationships

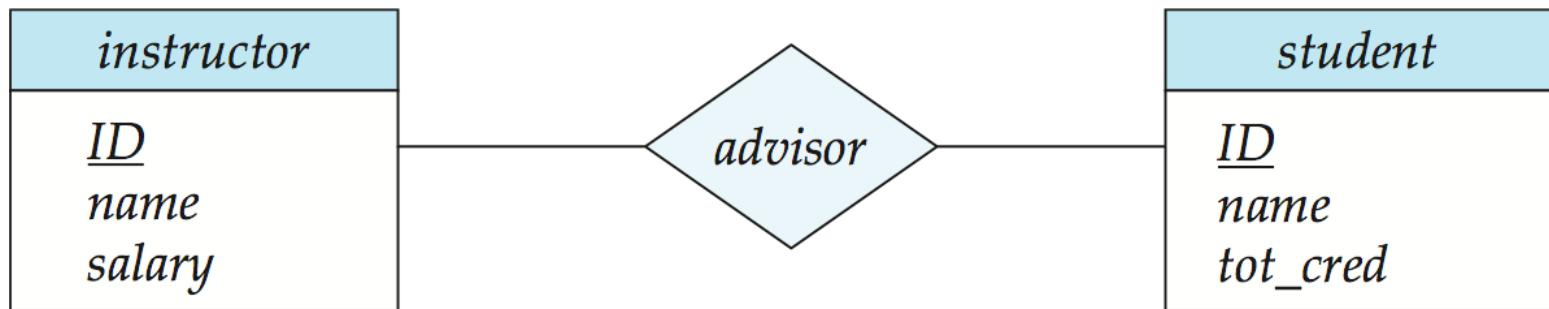
- In a many-to-one relationship between an *instructor* and a *student*,
 - an *instructor* is associated with at most one *student* via *advisor*,
 - and a *student* is associated with several (including 0) *instructors* via *advisor*





Many-to-Many Relationship

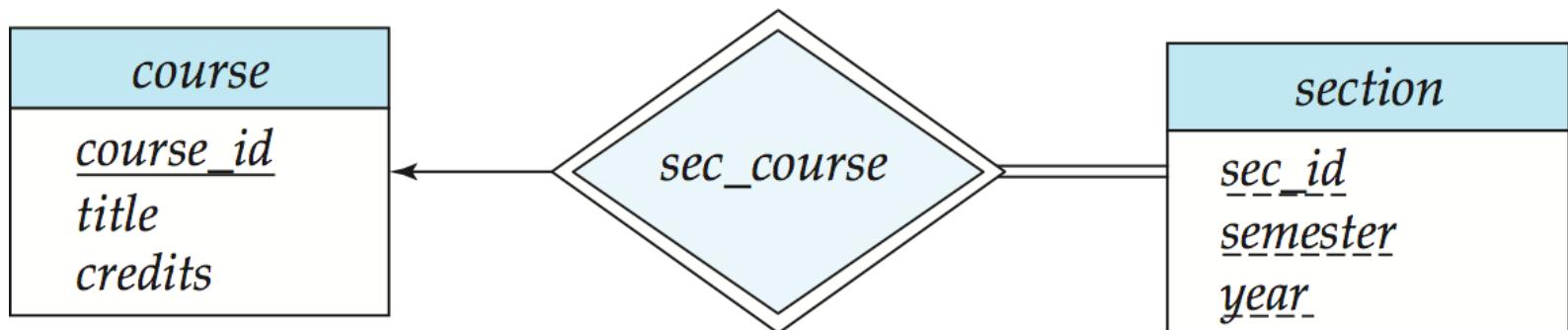
- An instructor is associated with several (possibly 0) students via *advisor*
- A student is associated with several (possibly 0) instructors via *advisor*





Participation of an Entity Set in a Relationship Set

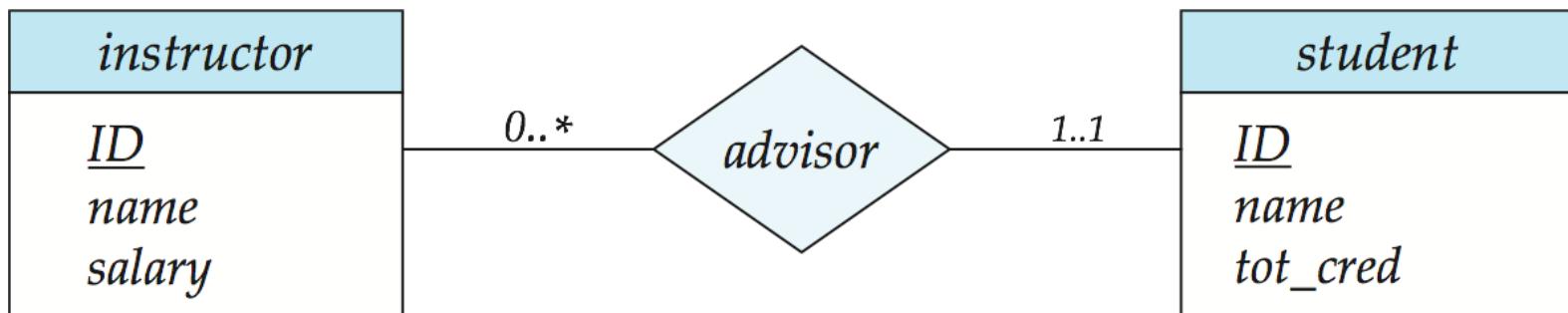
- Total participation (indicated by double line): every entity in the entity set participates in at least one relationship in the relationship set
 - E.g. participation of *section* in *sec_course* is total
 - ▶ every must have an associated course
- Partial participation: some entities may not participate in any relationship in the relationship set
 - Example: participation of *instructor* in *advisor* is partial





Alternative Notation for Cardinality Limits

- Cardinality limits can also express participation constraints
 - Number on line represents how many times the entity can participate in the relationship



Quiz Q2: The above relationship is

- (1) Many to one from instructor to student
- (2) One to many from instructor to student
- (3) One to one
- (4) many to many



Keys for Relationship Sets

- The combination of primary keys of the participating entity sets forms a super key of a relationship set.
 - (s_id, i_id) is the super key of *advisor*
 - ***NOTE: this means a pair of entity sets can have at most one relationship in a particular relationship set.***
 - ▶ Example: if we wish to track multiple meeting dates between a student and her advisor, we cannot have separate relationship instances for each meeting.
 - ▶ We can use a multivalued attribute though
- Must consider the mapping cardinality of the relationship set when deciding what are the candidate keys
- Need to consider semantics of relationship set in selecting the *primary key* in case of more than one candidate key



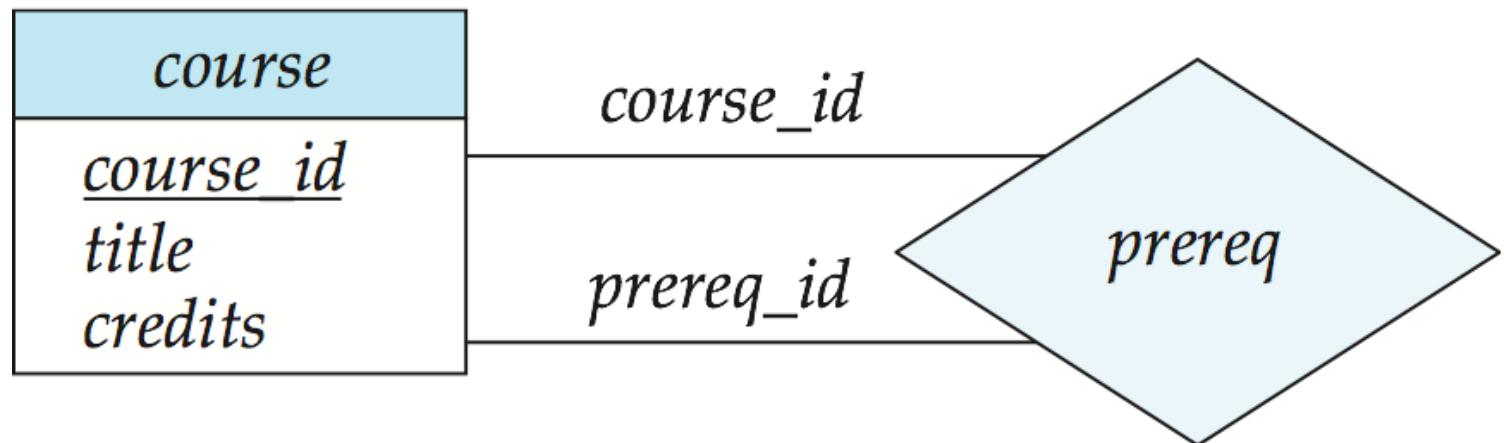
Redundant Attributes

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



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**.





**How about doing an ER design
interactively on the board?
Suggest an application to be modeled.**

Database System Concepts, 6th Ed.

©Silberschatz, Korth and Sudarshan

See www.db-book.com for conditions on re-use



End of Part 1

Database System Concepts, 6th Ed.

©Silberschatz, Korth and Sudarshan

See www.db-book.com for conditions on re-use



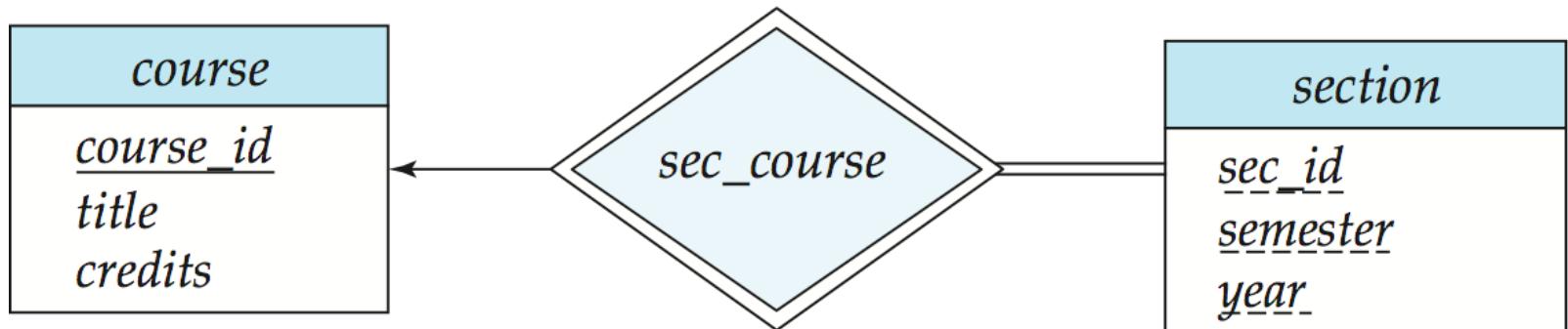
Weak Entity Sets

- 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 **identifying entity set**
 - it must relate to the identifying entity set via a total, one-to-many relationship set from the identifying to the weak entity set
 - **Identifying relationship** depicted using a double diamond
- The **discriminator** (*or partial key*) of a weak entity set is the set of attributes that distinguishes among all the entities of a weak entity set.
- The primary key of a weak entity set is formed by the primary key of the strong entity set on which the weak entity set is existence dependent, plus the weak entity set's discriminator.



Weak Entity Sets (Cont.)

- We underline the discriminator of a weak entity set with a dashed line.
- We put the identifying relationship of a weak entity in a double diamond.
- Discriminator of the weak entity set is underlined by dashed lines
- Primary key for *section* – (*course_id*, *sec_id*, *semester*, *year*)



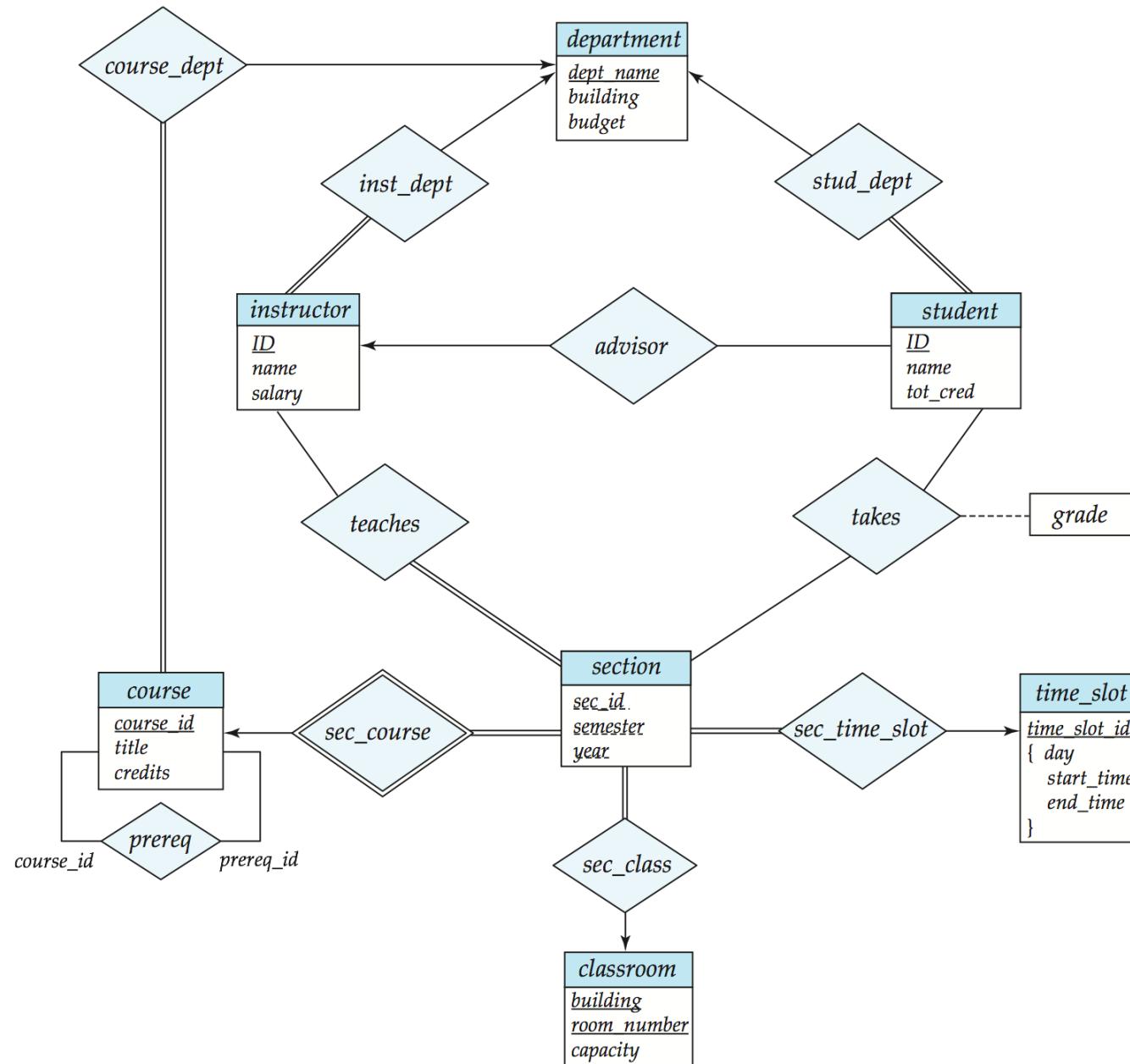


Weak Entity Sets (Cont.)

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



E-R Diagram for a University Enterprise



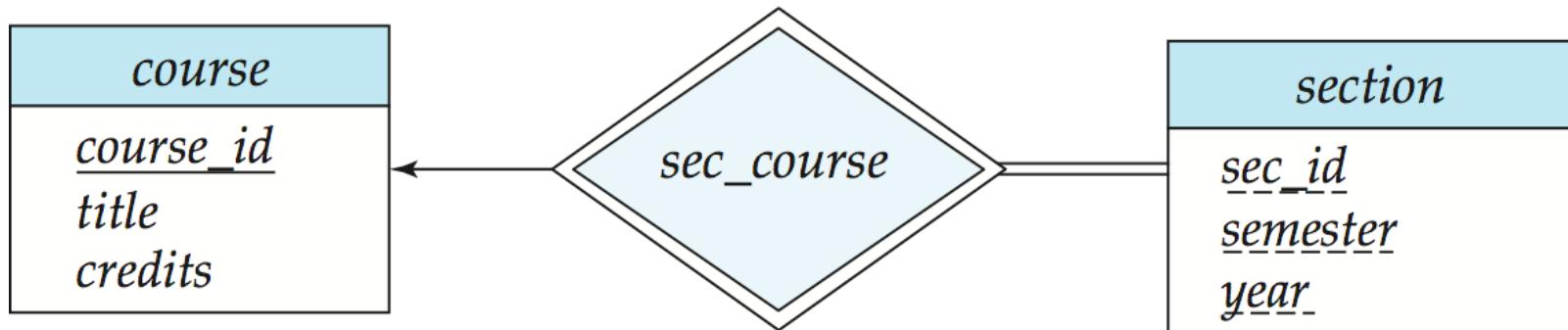


Reduction to Relational Schemas



Representing Entity Sets with Simple Attributes

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





Representing Relationship Sets

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

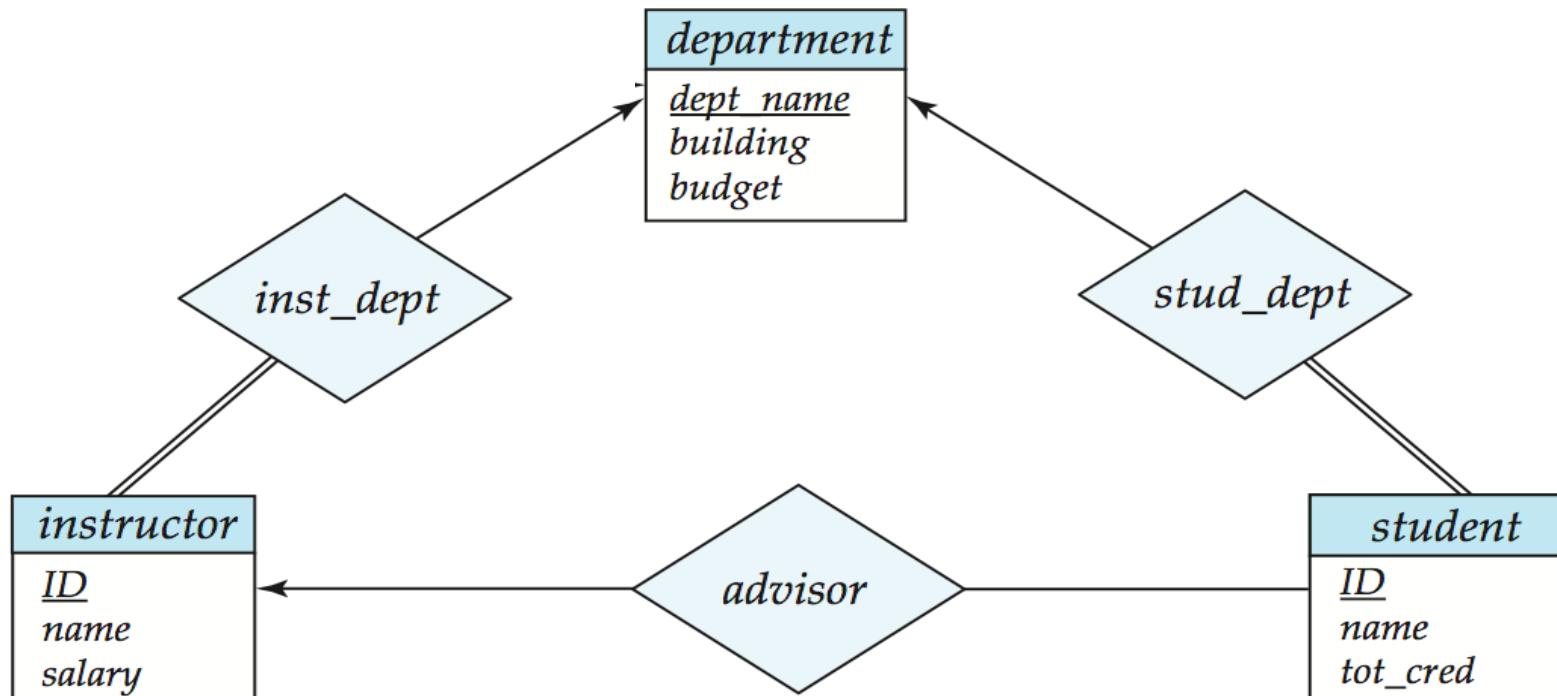
advisor = (s_id, i_id)





Redundancy of Schemas

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





Redundancy of Schemas (Cont.)

- For one-to-one relationship sets, either side can be chosen to act as the “many” side
 - That is, extra attribute can be added to either of the tables corresponding to the two entity sets
- If participation is *partial* on the “many” side, replacing a schema by an extra attribute in the schema corresponding to the “many” side could result in null values
- The schema corresponding to a relationship set linking a weak entity set to its identifying strong entity set is redundant.
 - Example: The *section* schema already contains the attributes that would appear in the *sec_course* schema



Composite and Multivalued Attributes

<i>instructor</i>
<u>ID</u>
<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
- 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)



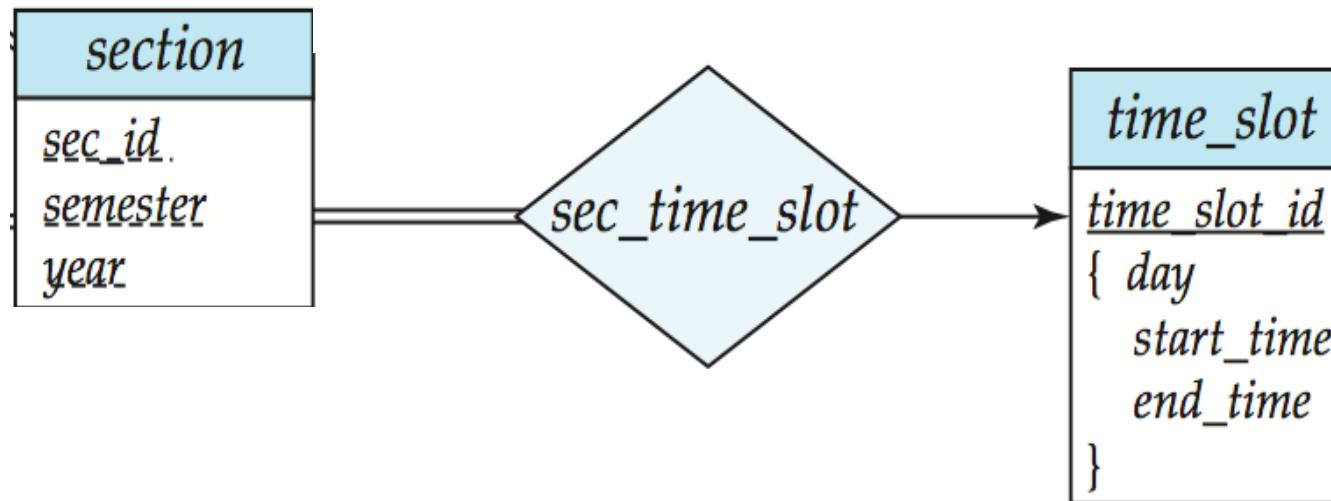
Composite and Multivalued Attributes

- 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)



Multivalued Attributes (Cont.)

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





Extended ER Features

- Specialization/Generalization
- Aggregation (see book for details)

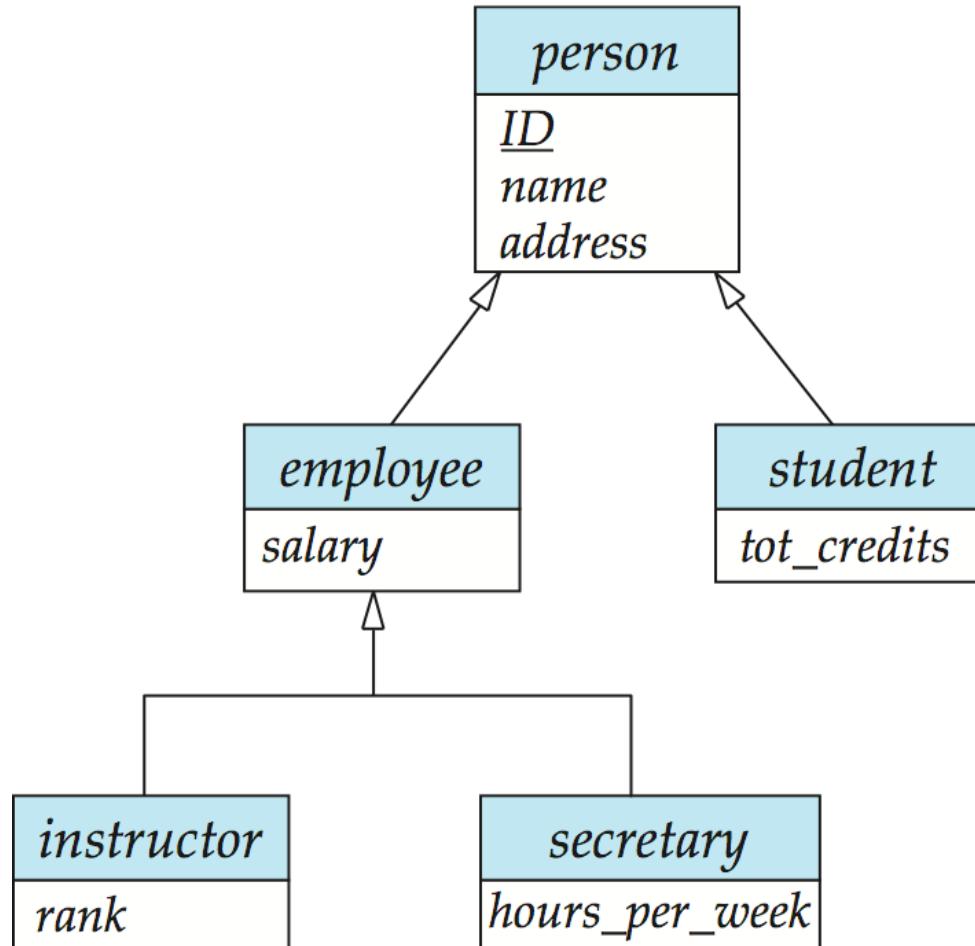


Extended E-R Features: Specialization

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





Design Constraints on a Specialization/Generalization

- Constraint on which entities can be members of a given lower-level entity set.
 - condition-defined
 - ▶ Example: all customers over 65 years are members of *senior-citizen* entity set; *senior-citizen* ISA *person*.
 - user-defined
- Constraint on whether or not entities may belong to more than one lower-level entity set within a single generalization.
 - **Disjoint**
 - ▶ an entity can belong to only one lower-level entity set
 - ▶ Noted in E-R diagram by having multiple lower-level entity sets link to the same triangle
 - **Overlapping**
 - ▶ an entity can belong to more than one lower-level entity set



Design Constraints on a Specialization/Generalization (Cont.)

- **Completeness constraint** -- specifies whether or not an entity in the higher-level entity set must belong to at least one of the lower-level entity sets within a generalization.
 - **total** : an entity must belong to one of the lower-level entity sets
 - **partial**: an entity need not belong to one of the lower-level entity sets



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
<i>person</i>	<i>ID, name, street, city</i>
<i>student</i>	<i>ID, tot_cred</i>
<i>employee</i>	<i>ID, salary</i>

- 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
<i>person</i>	<i>ID, name, street, city</i>
<i>student</i>	<i>ID, name, street, city, tot_cred</i>
<i>employee</i>	<i>ID, name, street, city, salary</i>

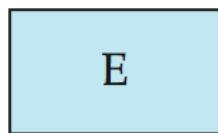
- If specialization is total, the schema for the generalized entity set (*person*) not required to store information
 - Can be defined as a “view” relation containing union of specialization relations
 - But explicit schema may still be needed for foreign key constraints
- Drawback: *name, street* and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



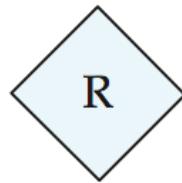
How about doing another ER design interactively on the board?



Summary of Symbols Used in E-R Notation



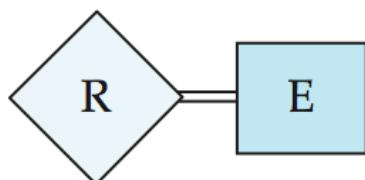
entity set



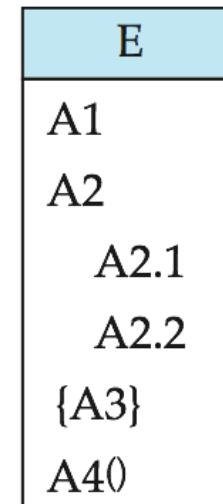
relationship set



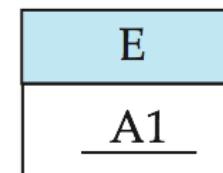
identifying
relationship set
for weak entity set



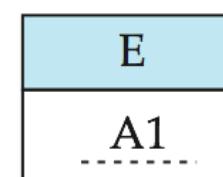
total participation
of entity set in
relationship



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



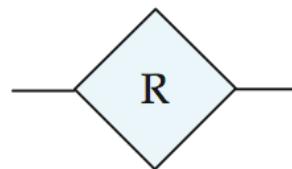
primary key



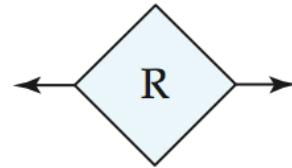
discriminating
attribute of
weak entity set



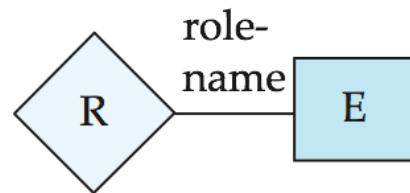
Symbols Used in E-R Notation (Cont.)



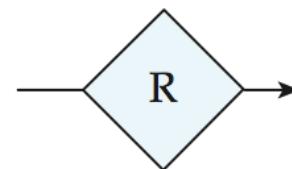
many-to-many
relationship



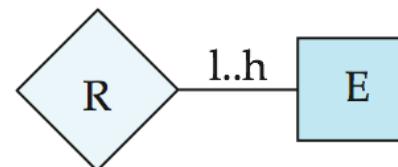
one-to-one
relationship



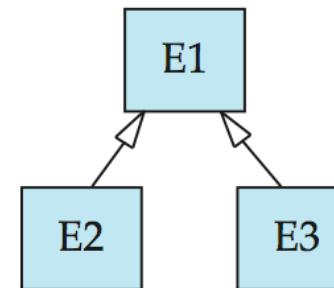
role indicator



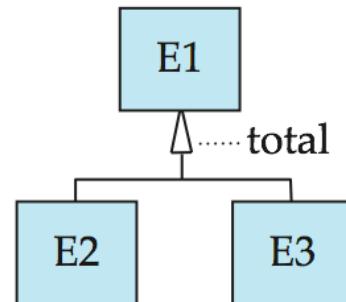
many-to-one
relationship



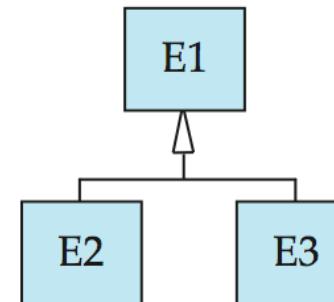
cardinality
limits



ISA: generalization
or specialization



total (disjoint)
generalization



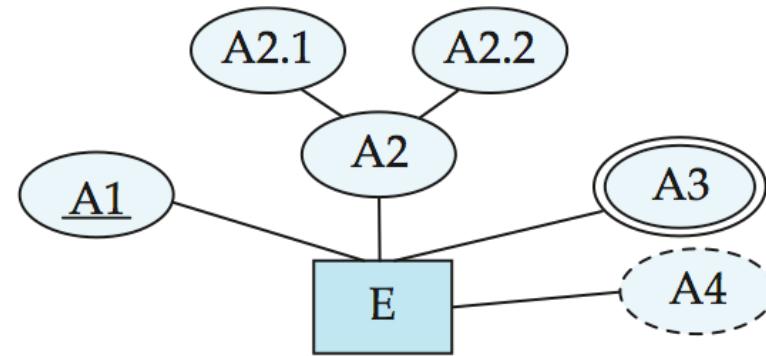
disjoint
generalization



Alternative ER Notations

- Chen, IDE1FX, ...

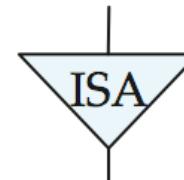
entity set E with
simple attribute A1,
composite attribute A2,
multivalued attribute A3,
derived attribute A4,
and primary key A1



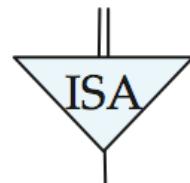
weak entity set



generalization



total
generalization

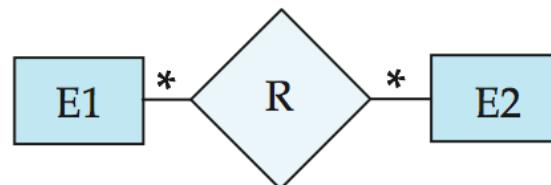




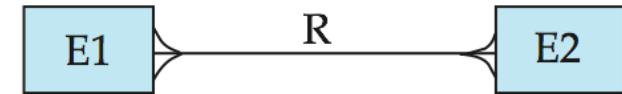
Alternative ER Notations

Chen

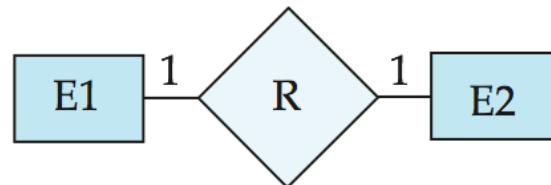
many-to-many
relationship



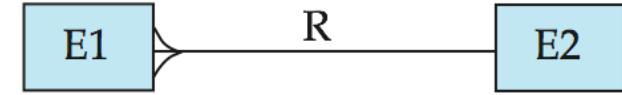
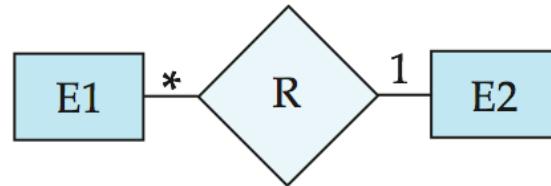
IDE1FX (Crows feet notation)



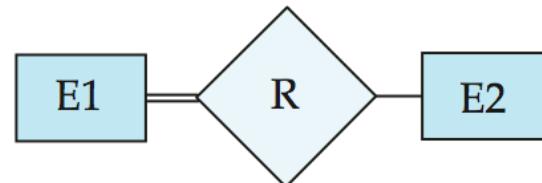
one-to-one
relationship



many-to-one
relationship



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





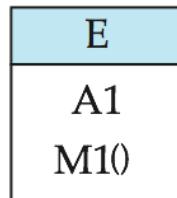
UML

- **UML**: Unified Modeling Language
- UML has many components to graphically model different aspects of an entire software system
- UML Class Diagrams correspond to E-R Diagram, but several differences.

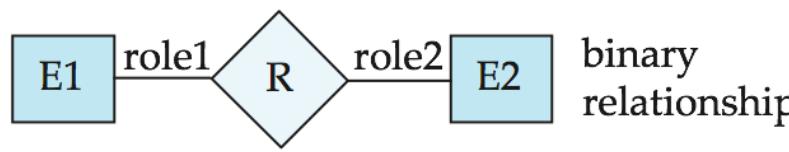


ER vs. UML Class Diagrams

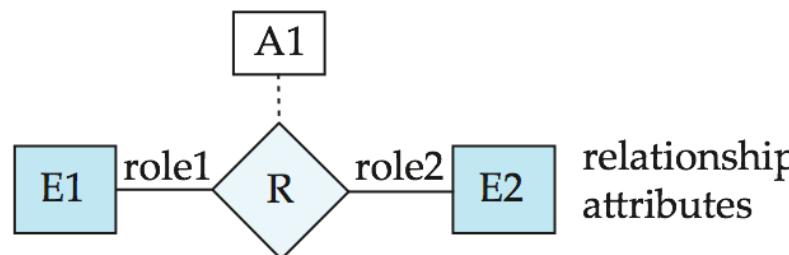
ER Diagram Notation



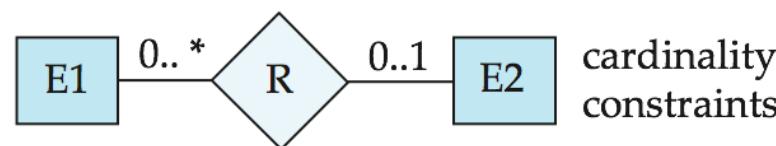
entity with attributes (simple, composite, multivalued, derived)



binary relationship

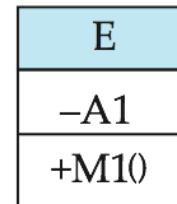


relationship attributes

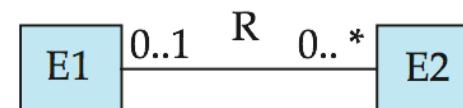
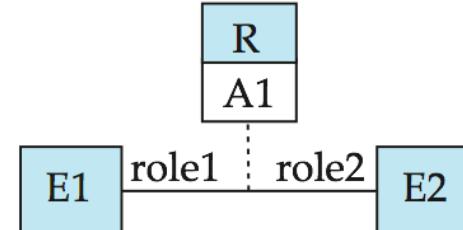
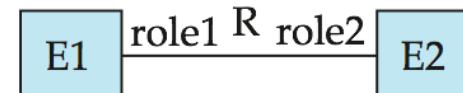


cardinality constraints

Equivalent in UML



class with simple attributes and methods (attribute prefixes: + = public, - = private, # = protected)

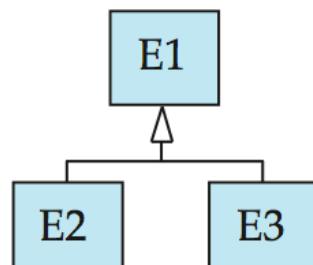
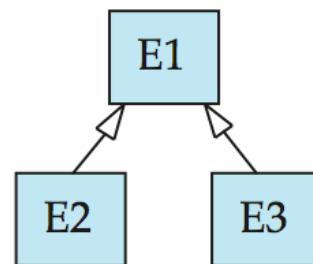
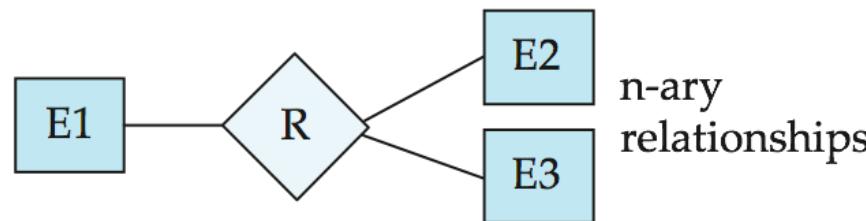


*Note reversal of position in cardinality constraint depiction

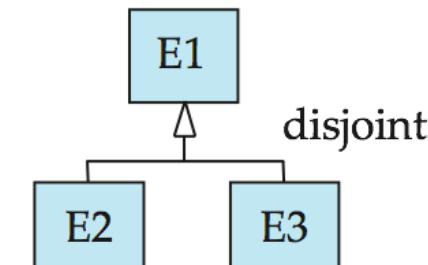
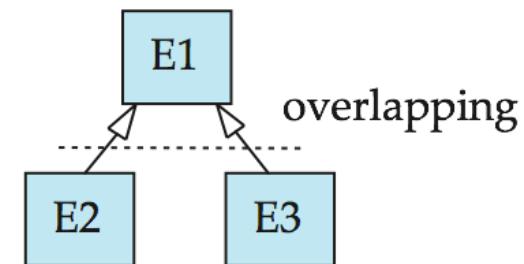
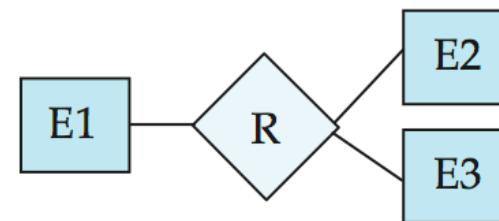


ER vs. UML Class Diagrams

ER Diagram Notation



Equivalent in UML



*Generalization can use merged or separate arrows independent of disjoint/overlapping

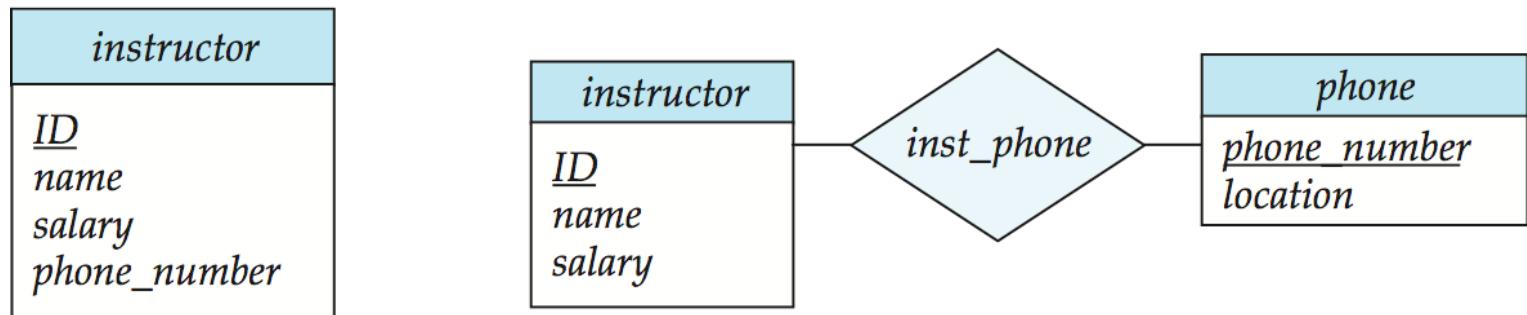


Extra Slides



Design Issues

■ Use of entity sets vs. attributes

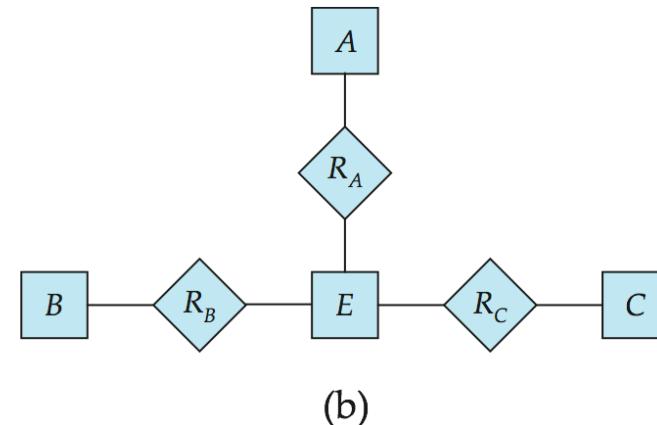
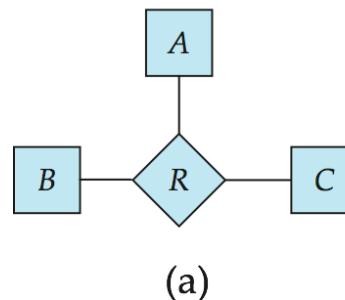


- Use of phone as an entity allows extra information about phone numbers (plus multiple phone numbers)



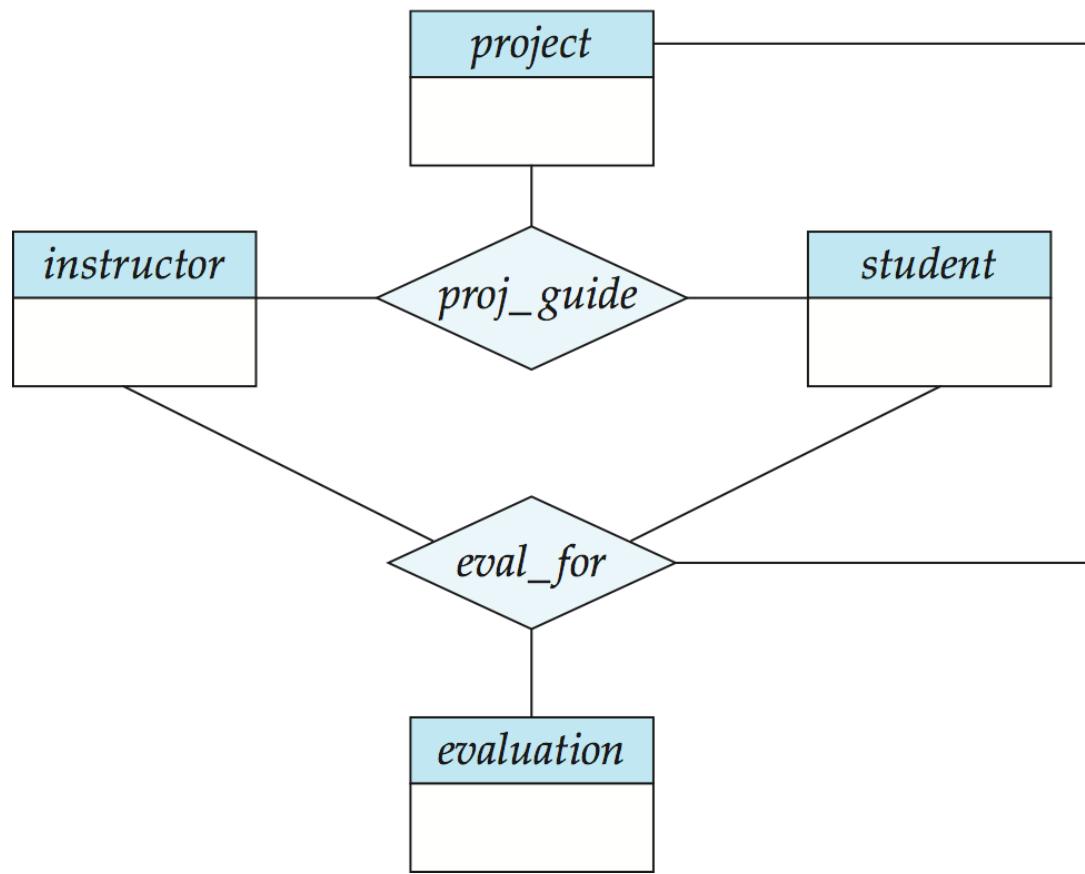
Converting Non-Binary Relationships to Binary Form

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





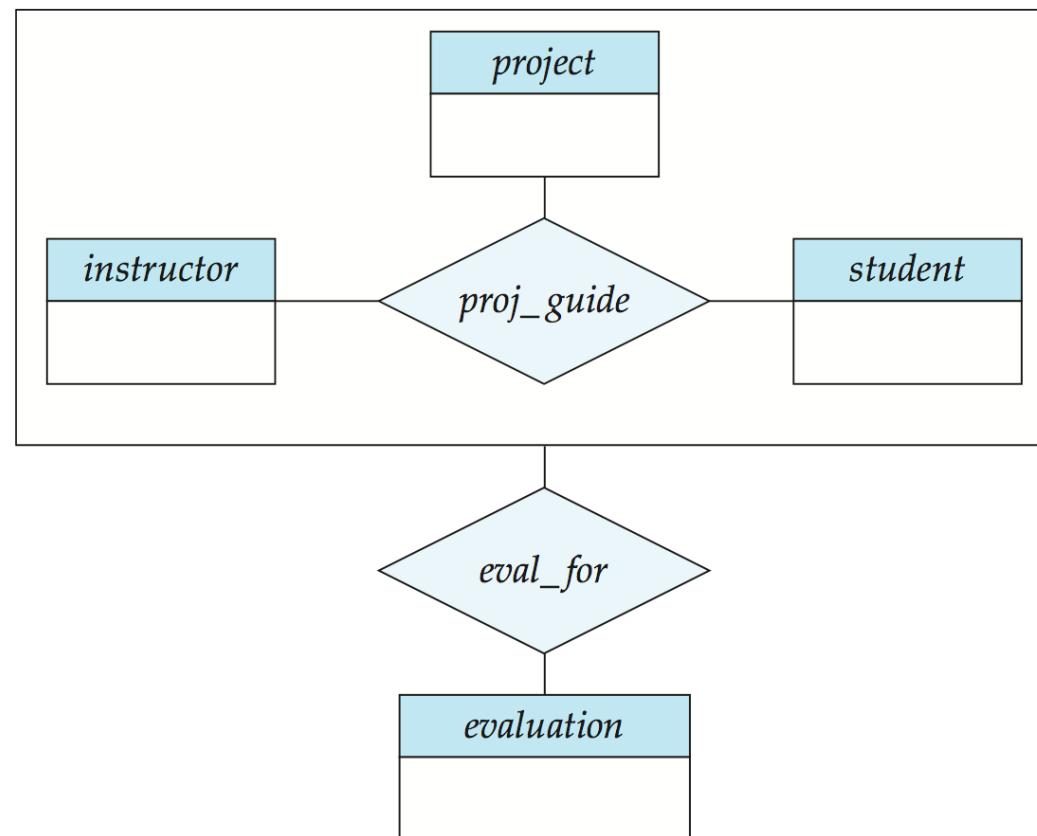
Motivation for Aggregation





Aggregation (Cont.)

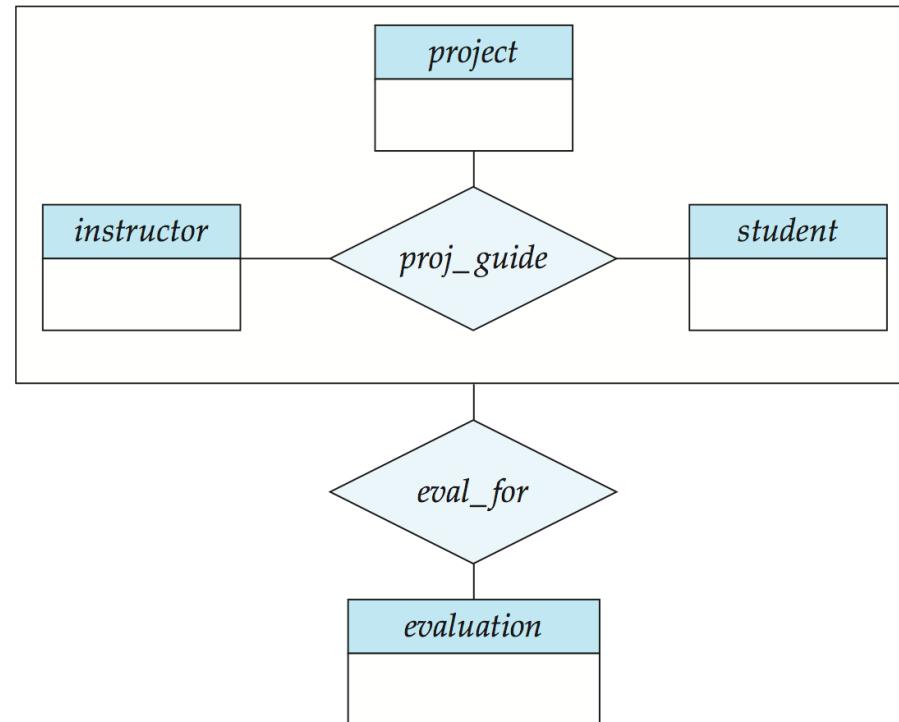
- Without introducing redundancy, the following diagram represents:
 - A student is guided by a particular instructor on a particular project
 - A student, instructor, project combination may have an associated evaluation





Relational Schemas Corresponding to Aggregation (Cont.)

- For example, to represent aggregation manages between relationship works_on and entity set manager, create a schema `eval_for (s_ID, project_id, i_ID, evaluation_id)`
- Schema `proj_guide` is redundant provided we are willing to store null values for attribute `manager_name` in relation on schema `manages`





End of Chapter



Keys

- A **super key** of an entity set is a set of one or more attributes whose values uniquely determine each entity.
- A **candidate key** of an entity set is a minimal super key
 - *ID* is candidate key of *instructor*
 - *course_id* is candidate key of *course*
- Although several candidate keys may exist, one of the candidate keys is selected to be the **primary key**.



Relationship Sets With Attributes

- An attribute can also be property of a relationship set.
- For instance, the advisor relationship set between entity sets instructor and student may have the attribute date
 - E.g. date may track when the student started being associated with the advisor

