



Chapter 7: Entity-Relationship Model

Database System Concepts, 6th Ed.

©Silberschatz, Korth and Sudarshan

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



Database System Concepts

- Chapter 1: Introduction
- **Part 1: Relational databases**
 - Chapter 2: Introduction to the Relational Model
 - Chapter 3: Introduction to SQL
 - Chapter 4: Intermediate SQL
 - Chapter 5: Advanced SQL
 - Chapter 6: Formal Relational Query Languages
- **Part 2: Database Design**
 - [Chapter 7: Database Design: The E-R Approach](#)
 - Chapter 8: Relational Database Design
 - Chapter 9: Application Design
- **Part 3: Data storage and querying**
 - Chapter 10: Storage and File Structure
 - Chapter 11: Indexing and Hashing
 - Chapter 12: Query Processing
 - Chapter 13: Query Optimization
- **Part 4: Transaction management**
 - Chapter 14: Transactions
 - Chapter 15: Concurrency control
 - Chapter 16: Recovery System
- **Part 5: System Architecture**
 - Chapter 17: Database System Architectures
 - Chapter 18: Parallel Databases
 - Chapter 19: Distributed Databases
- **Part 6: Data Warehousing, Mining, and IR**
 - Chapter 20: Data Mining
 - Chapter 21: Information Retrieval
- **Part 7: Specialty Databases**
 - Chapter 22: Object-Based Databases
 - Chapter 23: XML
- **Part 8: Advanced Topics**
 - Chapter 24: Advanced Application Development
 - Chapter 25: Advanced Data Types
 - Chapter 26: Advanced Transaction Processing
- **Part 9: Case studies**
 - Chapter 27: PostgreSQL
 - Chapter 28: Oracle
 - Chapter 29: IBM DB2 Universal Database
 - Chapter 30: Microsoft SQL Server
- **Online Appendices**
 - Appendix A: Detailed University Schema
 - Appendix B: Advanced Relational Database Model
 - Appendix C: Other Relational Query Languages
 - Appendix D: Network Model
 - Appendix E: Hierarchical Model



Chapter 7: Entity-Relationship Model

- 7.1 Overview of the Design Process
- 7.2 The Entity-Relationship Model
- 7.3 Constraints
- 7.4 Removing Redundant Attributes in Entity Sets
- 7.5 Entity-Relationship Diagrams
- 7.6 Reduction to Relational Schemas
- 7.7 Entity-Relationship Design Issues
- 7.8 Extended E-R Features
- 7.9 Alternative Notation for Modeling Data
- 7.10 Other Aspects of Database Design



Design Process

- **Creating a database application**
 - Design of the database schema
 - Design of the programs that access and update the data
 - Design of a security scheme to control access to data
- **The phases of the database design**
 - Understanding the needs of users and enterprises
 - Conceptual design using the abstract model like ER Model
 - Specification of functional requirements (operations & transactions)
 - Converting the abstract model to implementation details
 - ▶ Logical design phase: convert ER to Tables
 - ▶ Physical design phase: file organization, storage structure, index creation
- **Design alternatives for avoiding two major pitfalls**
 - Redundancy
 - Incompleteness



Chapter 7: Entity-Relationship Model

- 7.1 Overview of the Design Process
- 7.2 The Entity-Relationship Model
- 7.3 Constraints
- 7.4 Removing Redundant Attributes in Entity Sets
- 7.5 Entity-Relationship Diagrams
- 7.6 Reduction to Relational Schemas
- 7.7 Entity-Relationship Design Issues
- 7.8 Extended E-R Features
- 7.9 Alternative Notation for Modeling Data
- 7.10 Other Aspects of Database Design



ER Modeling

- Peter Chen, 1976, “The ER Model: Toward a Unified View of Data”, ACM Transactions of Database Systems (TODS)
- A *database* can be modeled as a collection of entities and 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: Entity “people” have attributes “*name*” and “*address*”
- 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

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

instructor

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

student



Relationship Sets [1/2]

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

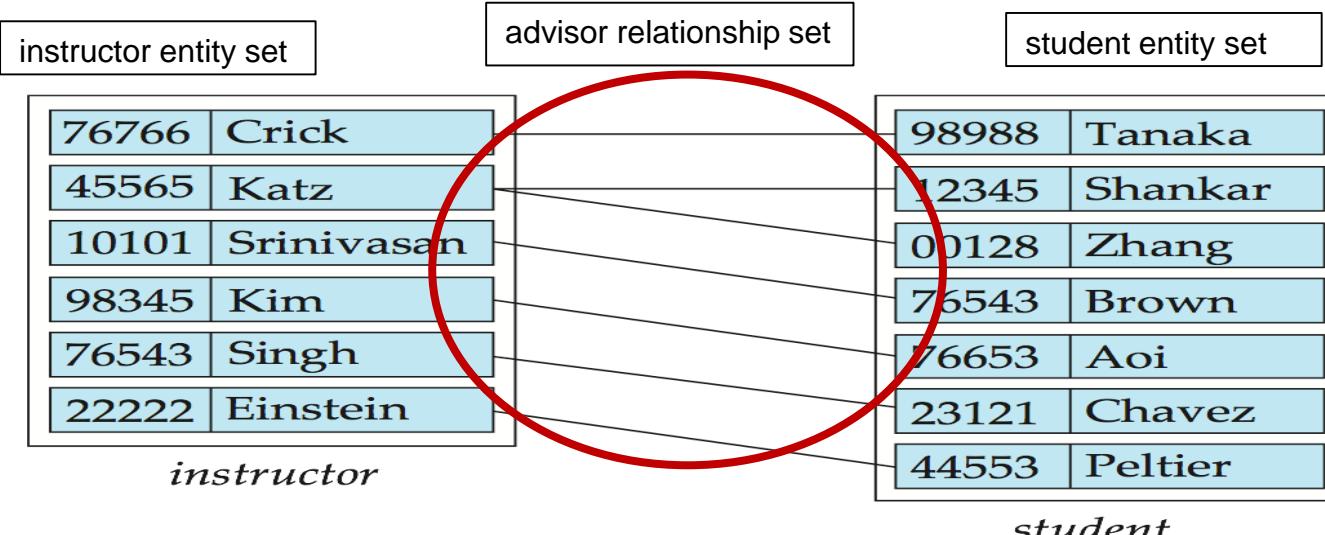
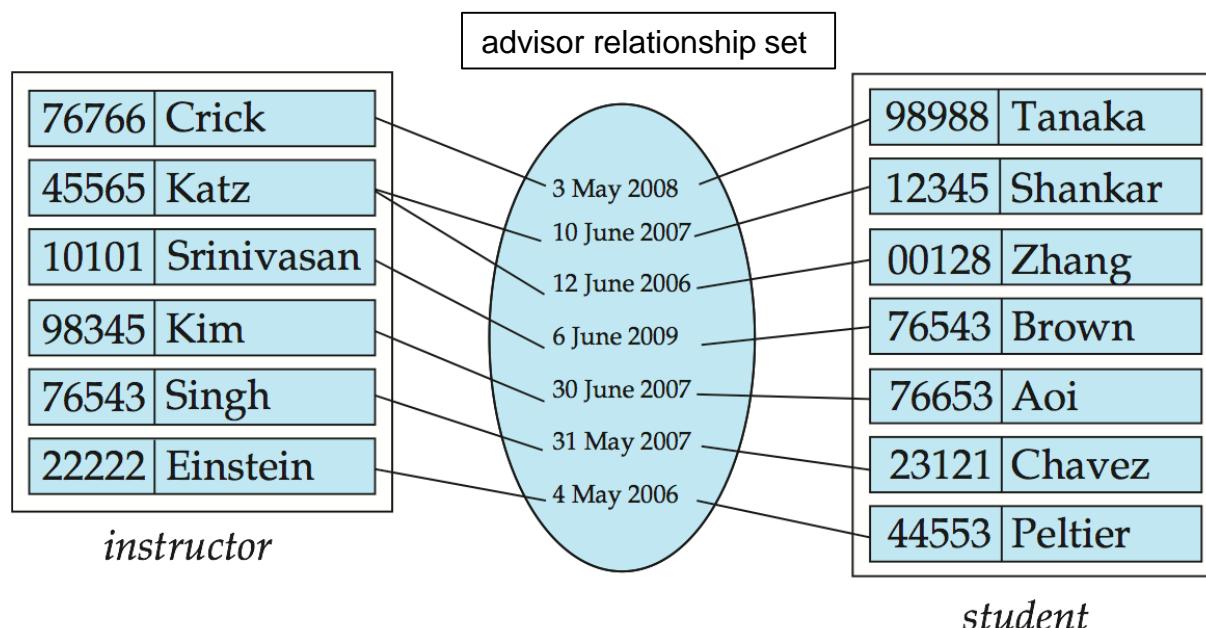




Fig 7.03: Relationship Sets [2/2]

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



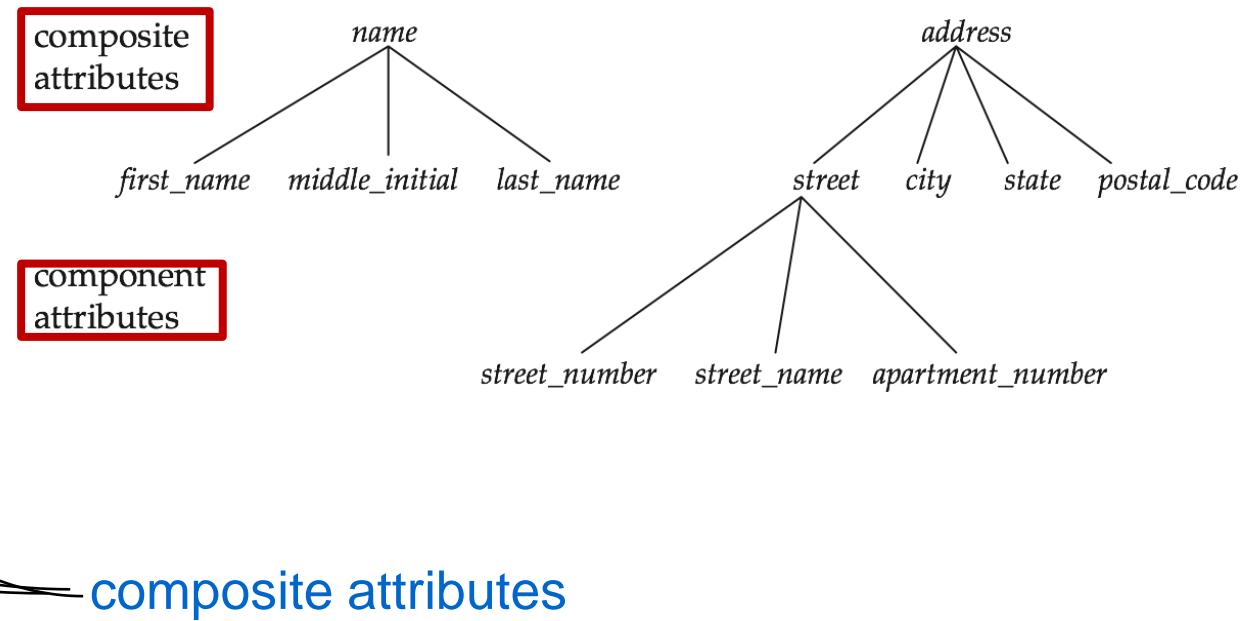
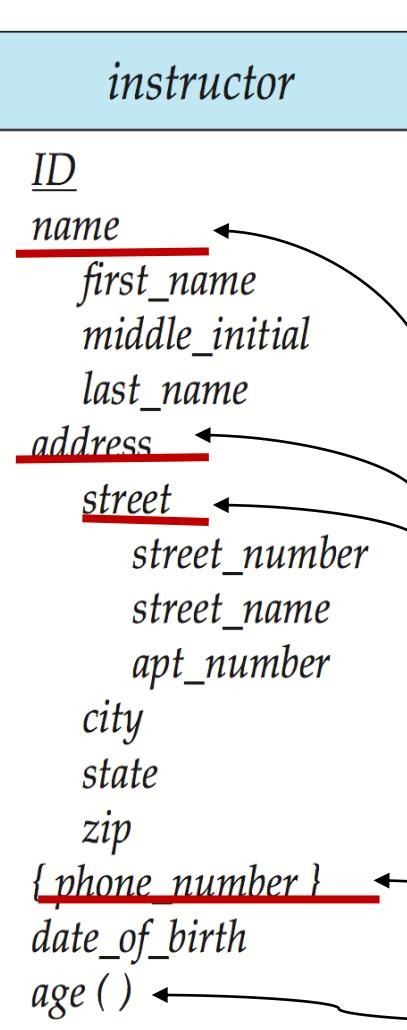


Attributes

- 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 and Multivalued Attributes



multivalued attributes

derived attributes



Chapter 7: Entity-Relationship Model

- 7.1 Overview of the Design Process
- 7.2 The Entity-Relationship Model
- 7.3 Constraints
- 7.4 Removing Redundant Attributes in Entity Sets
- 7.5 Entity-Relationship Diagrams
- 7.6 Reduction to Relational Schemas
- 7.7 Entity-Relationship Design Issues
- 7.8 Extended E-R Features
- 7.9 Alternative Notation for Modeling Data
- 7.10 Other Aspects of Database Design



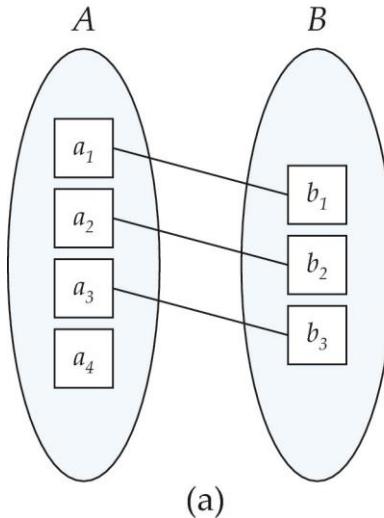
Mapping Cardinality Constraints

- 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

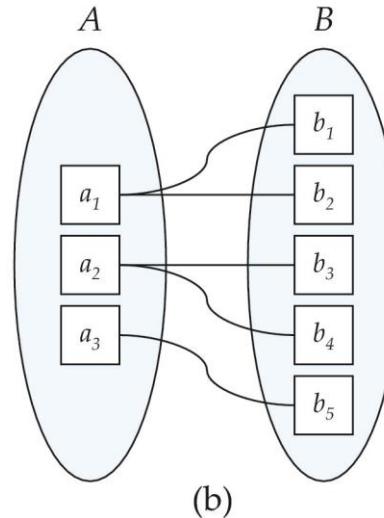


Fig 7.05 & 7.06: Mapping Cardinalities

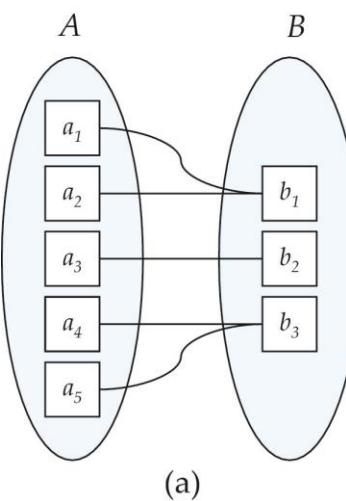
One to one



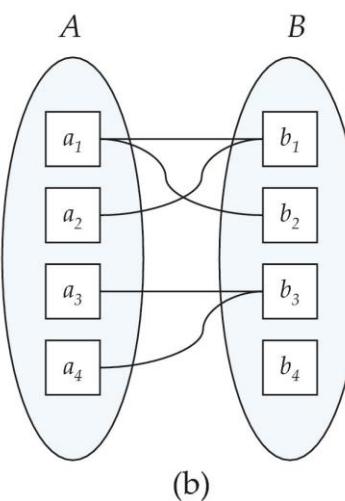
One to many



Many to one



Many to many



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



Keys of Entity Sets

- 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**

instructor = (ID, name, street, city, salary)

course= (course_id, title, credits)

- *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**.



Keys for Relationship Sets

- The combination of primary keys of the participating entity sets forms a super key of a relationship set.
 - The relationship set “advisor” involves the 2 entity sets “instructor (i_id, instructor_name)” and “student (s-id, student_name)”
 - (s_id, i_id) is the super key of advisor relationship
 - **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 assume a relationship for each meeting. We can use a multivalued attribute “date” though
 - ▶ Multivalued attribute “date” having a set value such as { 13-March-2016, 16-April-2016, 4-May-2016}
- 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



Chapter 7: Entity-Relationship Model

- 7.1 Overview of the Design Process
- 7.2 The Entity-Relationship Model
- 7.3 Constraints
- 7.4 Removing Redundant Attributes in Entity Sets
- 7.5 Entity-Relationship Diagrams
- 7.6 Reduction to Relational Schemas
- 7.7 Entity-Relationship Design Issues
- 7.8 Extended E-R Features
- 7.9 Alternative Notation for Modeling Data
- 7.10 Other Aspects of Database Design



Redundant Attributes

- Suppose we have entity sets
 - *Instructor* (*ID*, *name*, *dept_name*, *salary*) where *ID* is the primary key
 - *Department* (*dept_name*, *building*, *budget*) where *dept_name* is the primary key
- and a relationship set
 - *inst_dept* (*ID*, *dept_name*) relating *instructor* and *department*
- The attribute *dept_name* in entity *instructor* is redundant since there is an explicit relationship *inst_dept* which relates instructors to departments
 - The attribute *dept_name* replicates information present in the relationship *inst_dept*, and should be removed from *instructor*
 - BUT: when converting back to tables, in some cases the attribute gets reintroduced, as we will see.

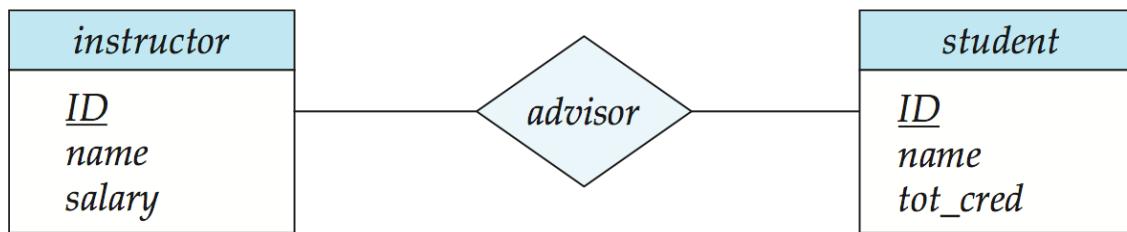


Chapter 7: Entity-Relationship Model

- 7.1 Overview of the Design Process
- 7.2 The Entity-Relationship Model
- 7.3 Constraints
- 7.4 Removing Redundant Attributes in Entity Sets
- 7.5 Entity-Relationship Diagrams
- 7.6 Reduction to Relational Schemas
- 7.7 Entity-Relationship Design Issues
- 7.8 Extended E-R Features
- 7.9 Alternative Notation for Modeling Data
- 7.10 Other Aspects of Database Design

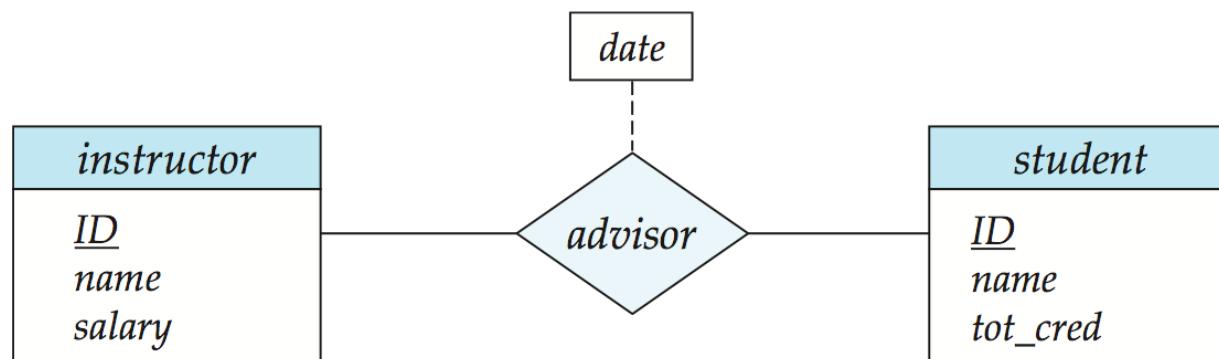


E-R Diagrams based on UML-like Notations



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

Fig 7.08: Relationship Sets with Attributes





Cardinality Constraints

- A directed line (\rightarrow), signifying “one,” or an undirected line ($-$), signifying “many,” between the relationship set and the entity set.

Fig 7.9a: 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*

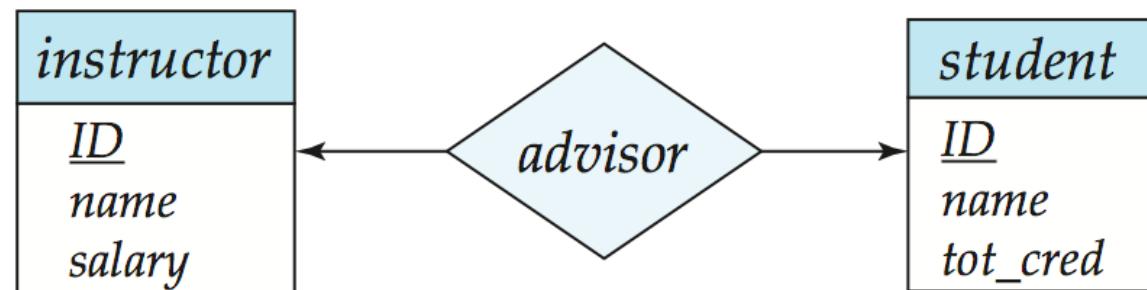




Fig 7.9b: One-to-Many Relationship

- 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*,
- 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*

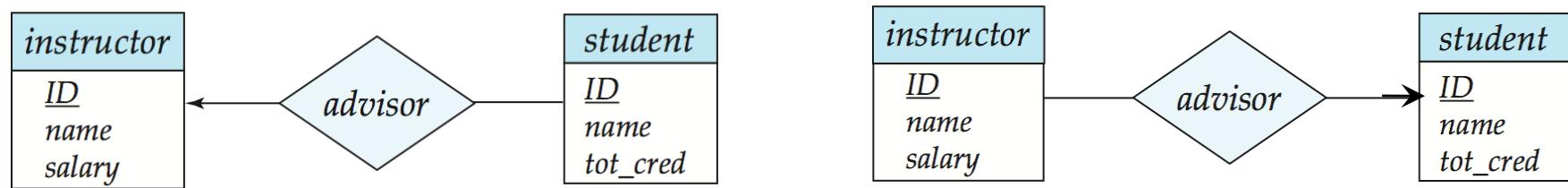


Fig 7.8c: Many-to-Many Relationship

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

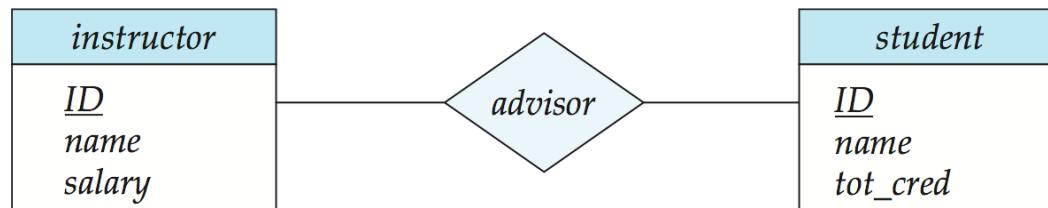




Fig 7.14: Participation of an Entity Set in a Relationship Set

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

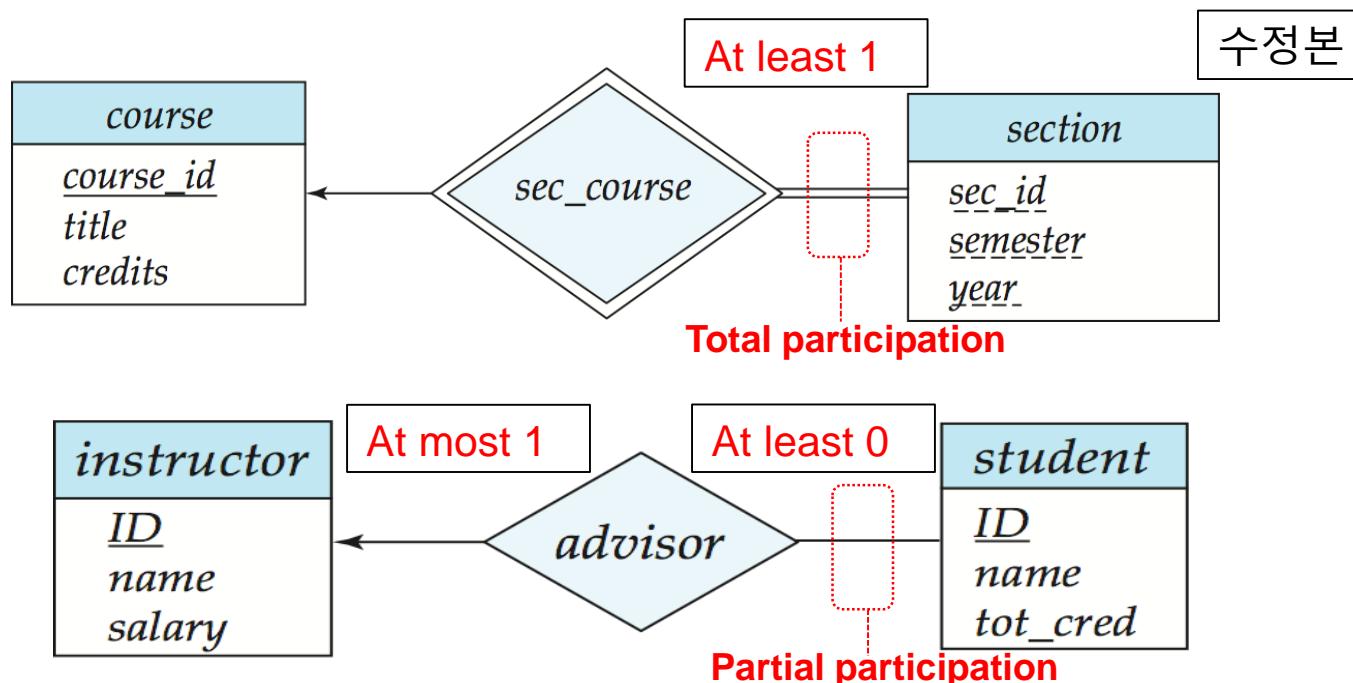




Fig 7.10: Alternative Notation for Cardinality Limits

- Cardinality limits can also express participation constraints

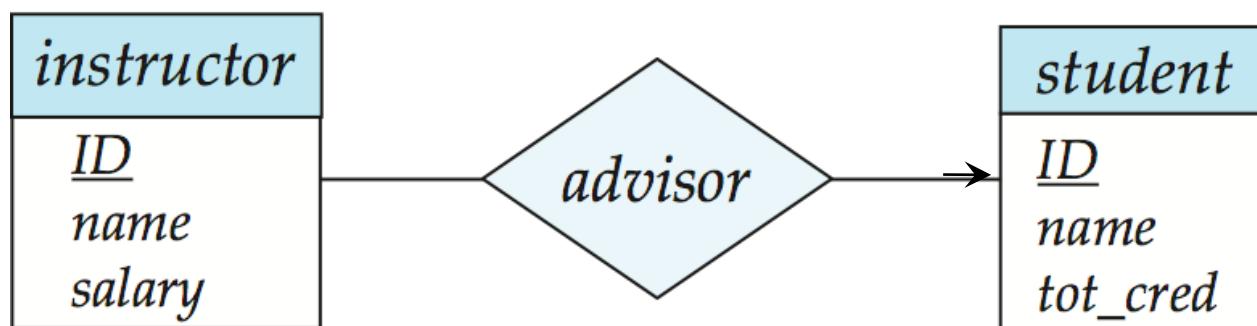
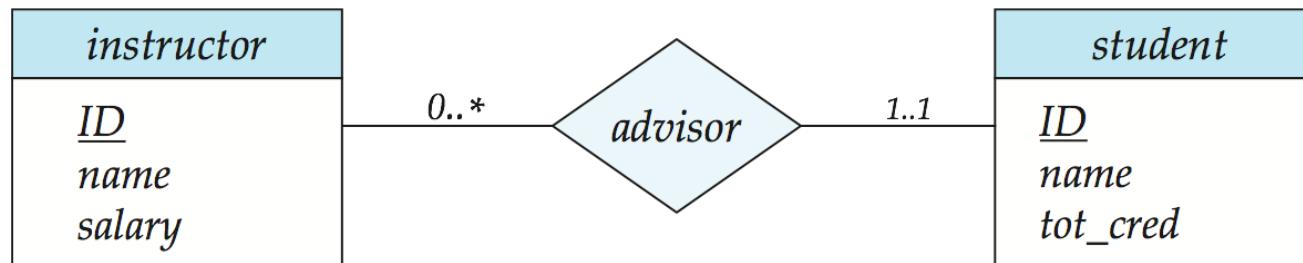
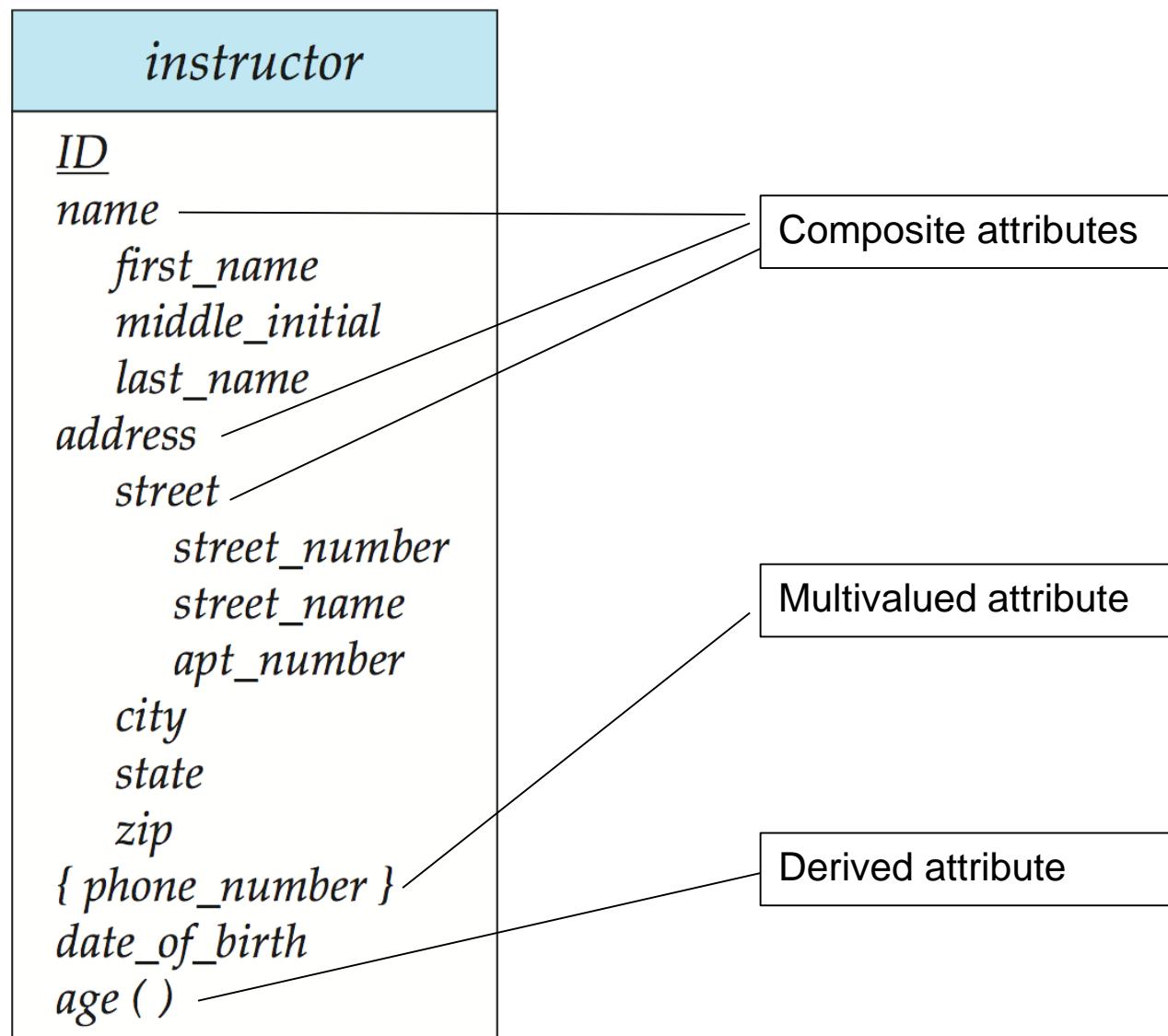




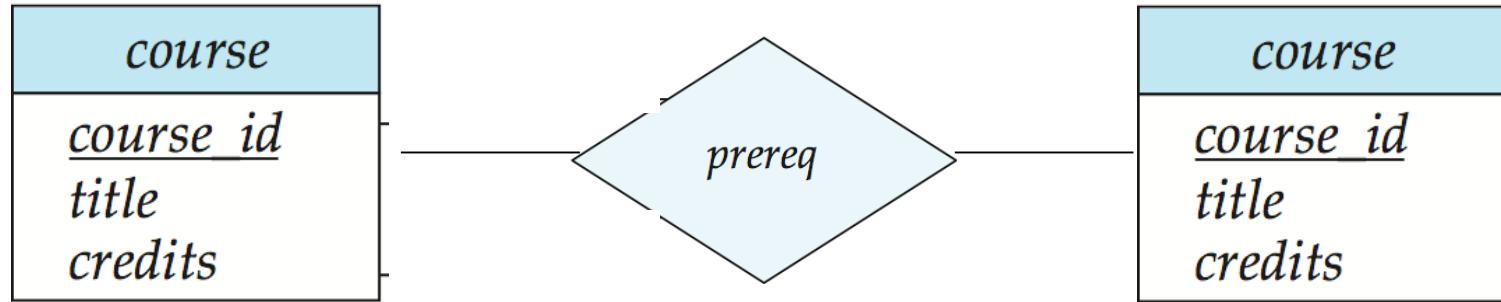
Fig 7.11: Entity With Composite, Multivalued, and Derived Attributes



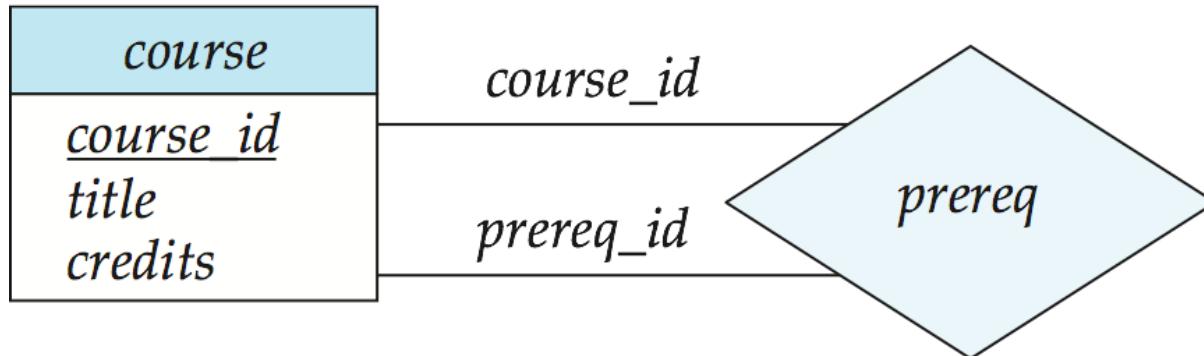


Role Indicator in Multiple Same Entities

- Entity sets of a relationship **need not be distinct (may appear twice)**
 - Each occurrence of an entity set plays a “role” in the relationship



- The labels “*course_id*” and “*prereq_id*” are called **roles (role indicator)**





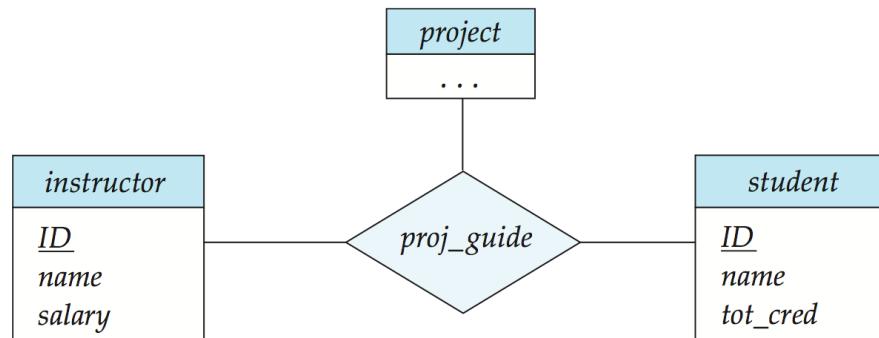
Non Binary Relationships

■ 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.

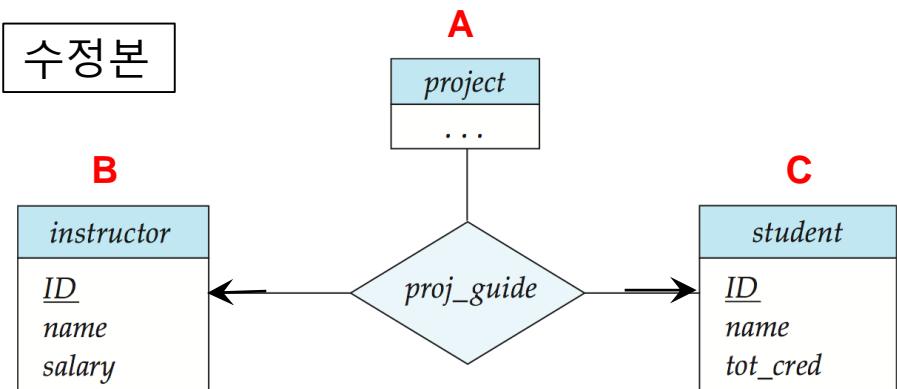
- Example: *students* work on research *projects* under the guidance of an *instructor*.
- The *proj_guide* is a ternary relationship between *instructor*, *student*, and *project*





Cardinality Constraints on Ternary Relationship

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

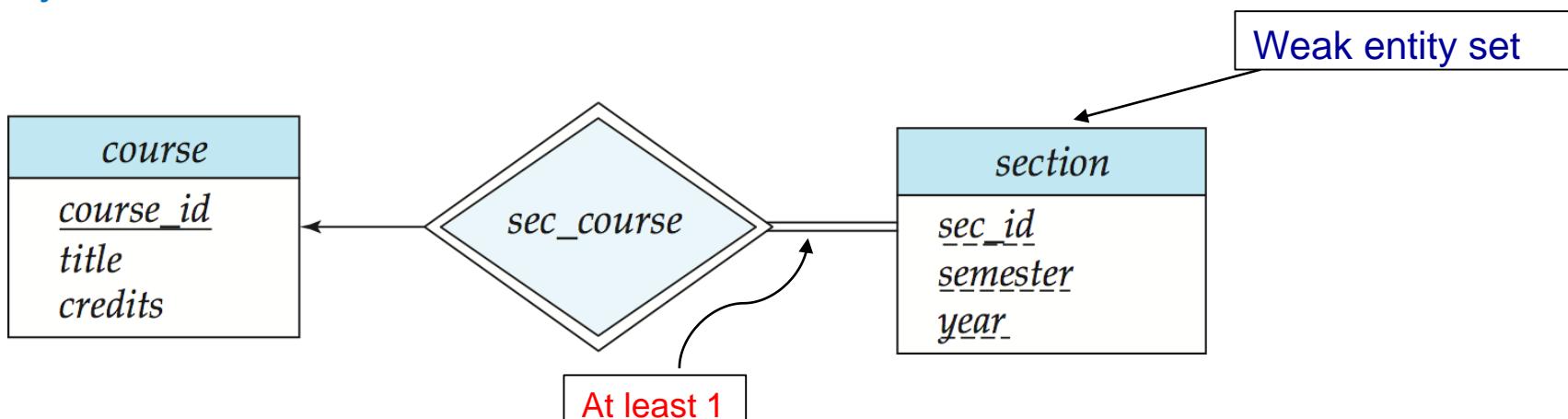


- One project is associated with a unique instructor and unique student
- Each pair of (project, instructor) is associated with a unique student and each pair of (project, student) is associated with a unique instructor



Weak Entity Sets [1/2]

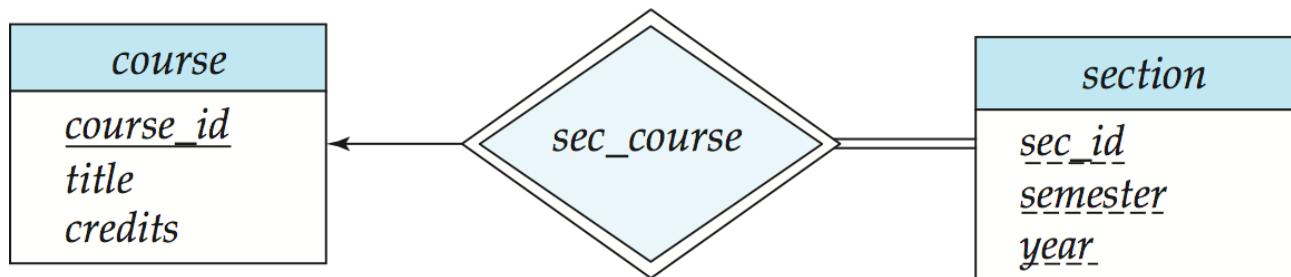
- An entity set that does not have a primary key is referred to as a **weak entity set**.
- The existence of a weak entity set depends on the existence of a **identifying entity set**
 - It must relate to the identifying entity set via a total, one-to-many relationship set from the identifying to the 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 [2/2]

- 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**.
- Primary key for *section* – (*course_id*, *sec_id*, *semester*, *year*)



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



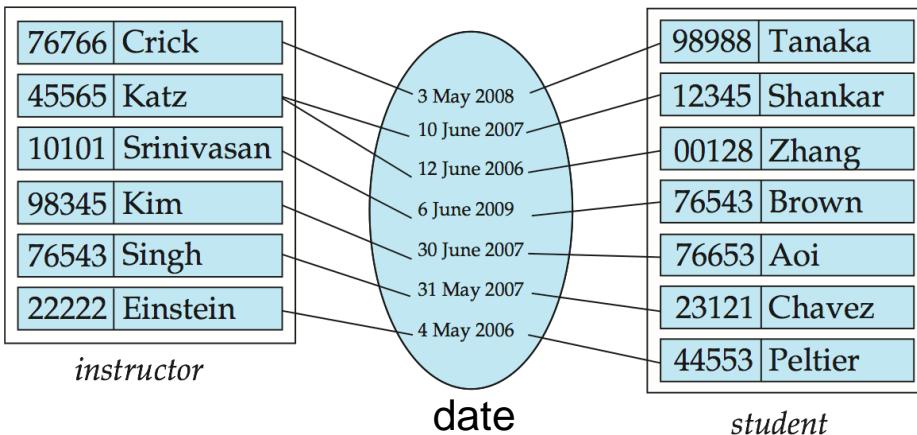
Chapter 7: Entity-Relationship Model

- 7.1 Overview of the Design Process
- 7.2 The Entity-Relationship Model
- 7.3 Constraints
- 7.4 Removing Redundant Attributes in Entity Sets
- 7.5 Entity-Relationship Diagrams
- 7.6 Reduction to Relational Schemas
- 7.7 Entity-Relationship Design Issues
- 7.8 Extended E-R Features
- 7.9 Alternative Notation for Modeling Data
- 7.10 Other Aspects of Database Design



Reduction ER Diagram to Relation DB Schemas

- A database which conforms to an E-R diagram can be represented by a collection of relation schemas.
- Entity sets and relationship sets can be expressed uniformly as *relation schemas* that represent the contents of the database.
 - For each entity set and relationship set there is a unique relation schema that is assigned the name of the corresponding entity set or relationship set.
 - Each relation schema has a number of columns (generally corresponding to attributes), which have unique names.

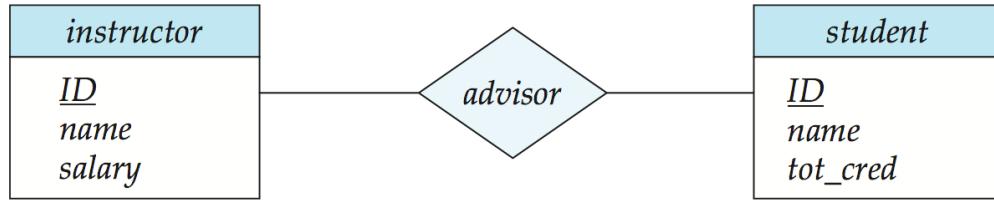


Instructor(ID, name)		
student(ID, name)		
advice_meeting(Instructor.ID, Student.ID, Date)		
76766	Crick	
45565	Katz	
10101	Srinivasan	
98345	Kim	
76543	Singh	
22222	Einstein	
98988	Tanaka	
12345	Shankar	
00128	Zhang	
76543	Brown	
76653	Aoi	
23121	Chavez	
44553	Peltier	
76766	12345	3 May 2008
45565	00128	10 June 2007
.....

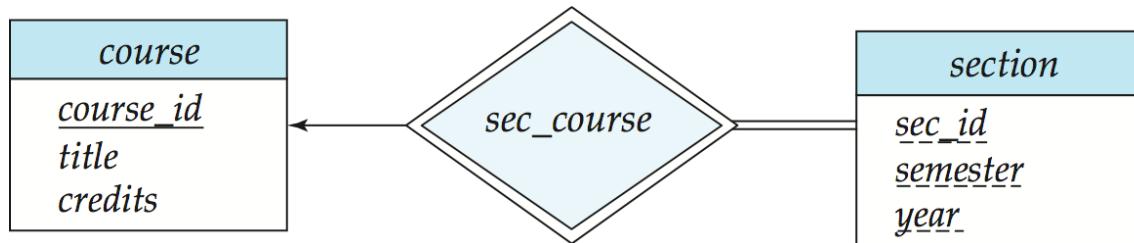


Representing Entity Sets With Simple Attributes

- A strong entity set reduces to a RDB schema with the same attributes
instructor(ID, name, salary), *student(ID, name, tot_cred)*



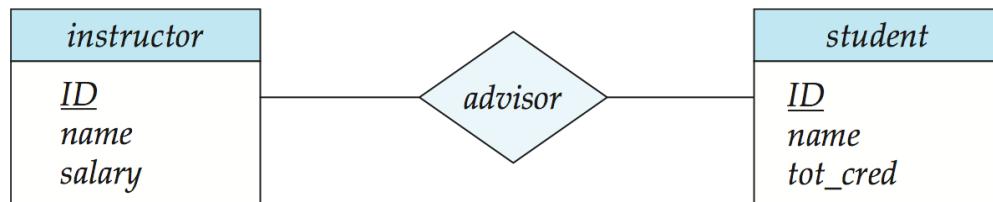
- A weak entity set becomes a table that includes a column for the primary key of the identifying strong entity set
section (course_id, sec_id, sem, year)





Representing Relationship Sets to RDB Schema

- 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: RDB schema for relationship set *advisor*
advisor (s_id, i_id)



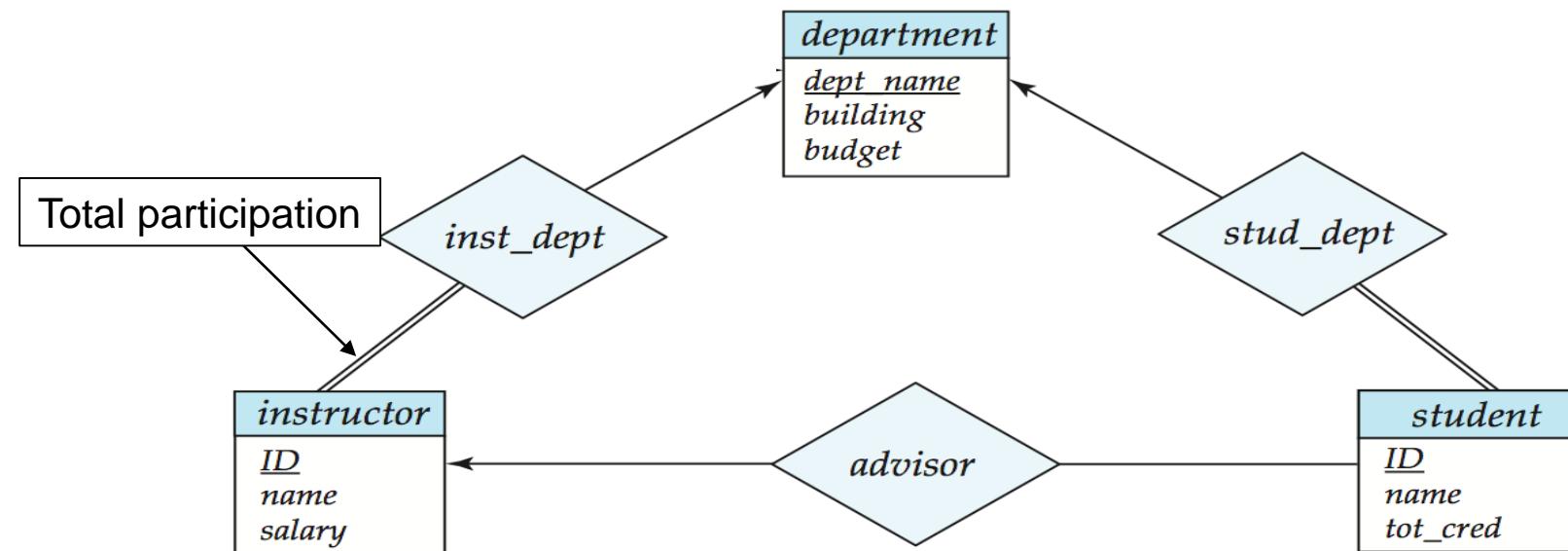
- Sometimes, we can skip a relation schema for certain relationship set! (next slide)



Redundancy of Schemas

[1/2]

- 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 entity**, containing the primary key of the “one” side
- Example: Add an attribute *dept_name* to the schema arising from entity set *instructor*
instructor (dept_name, ID, name, salary)
inst_dept(dept_name, ID) → **redundant schema, so we can skip this relationship set**

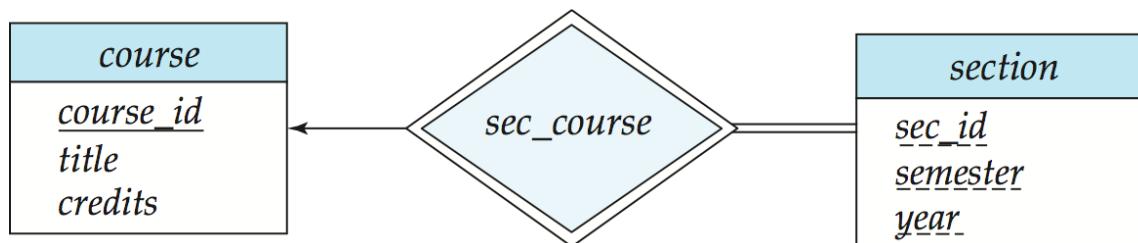




Redundancy of Schemas

[2/2]

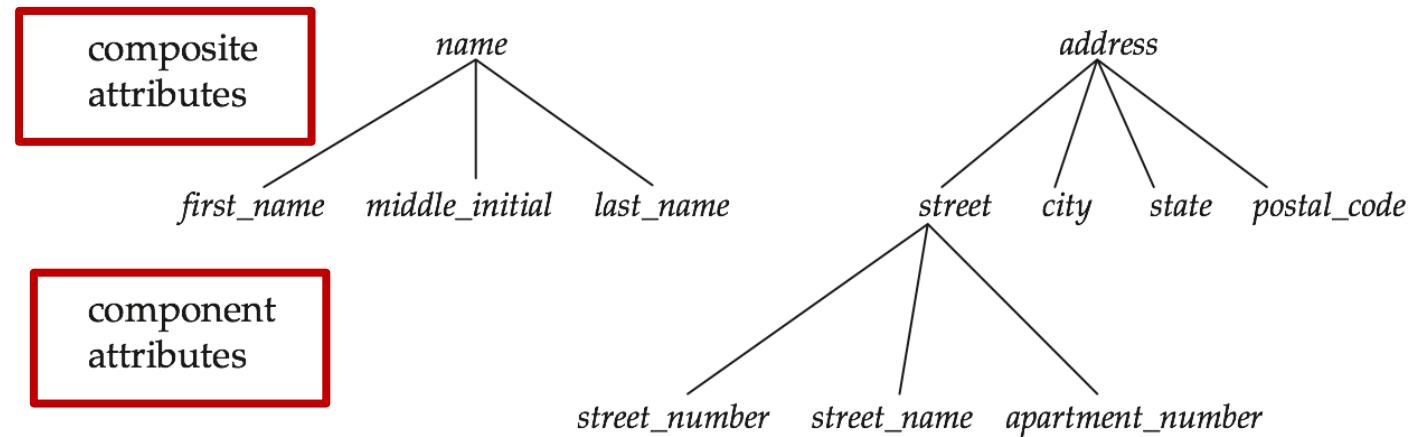
- If participation is *partial* on the “many” side, replacing a schema by an extra attribute in the schema corresponding to the “many” side could result in null values
 - section(course_id, sec_id, semester, year)
 - course(course_id, title, credits)
 - sec_course(course_id, sec_id) → redundant schema, so we can skip it!
- 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

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



Idea for composite attributes → Flattening the tree structure

Idea for multivalued attributes → Extra relation



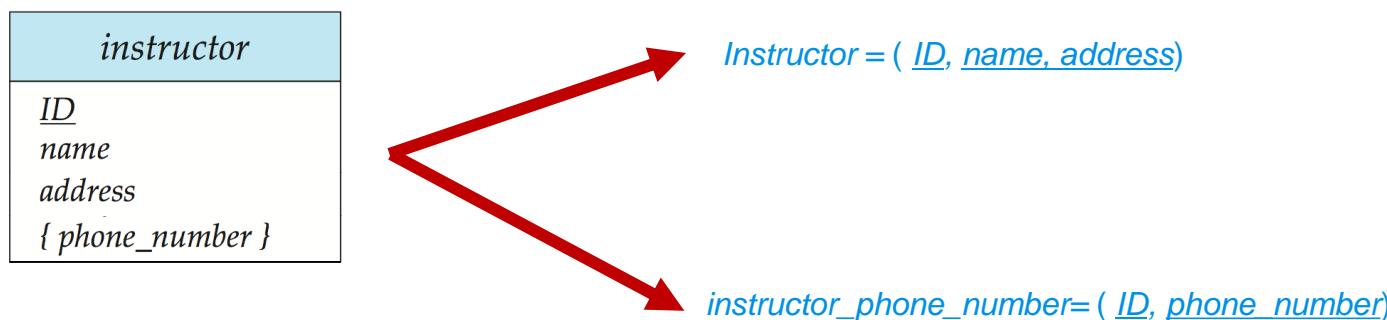
Composite Attributes into RDB Schema

- 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)



Multivalued Attributes into RDB Schemas [1/2]

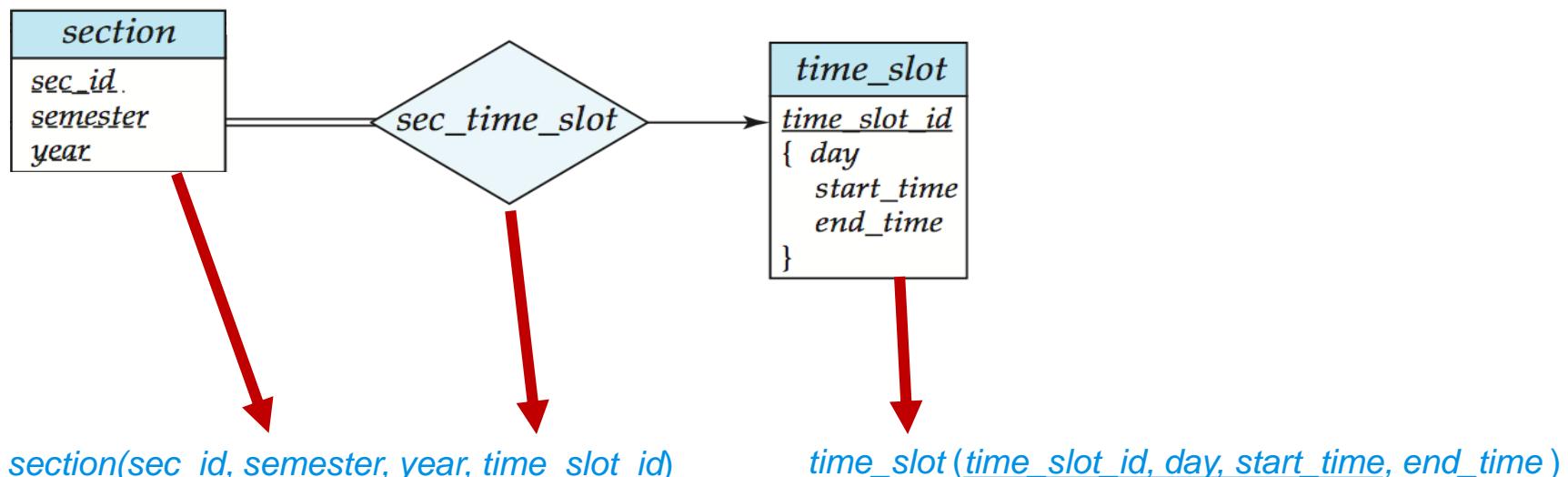
- A multivalued attribute M of an entity E is represented by a separate schema EM
 - Schema EM has attributes corresponding to the primary key of E and a multivalued attribute M
 - Example: Multivalued attribute *phone_number* of *instructor* is represented by a schema: *instructor_phone_number*= (*ID*,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





Multivalued Attributes into RDB Schemas [2/2]

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





Chapter 7: Entity-Relationship Model

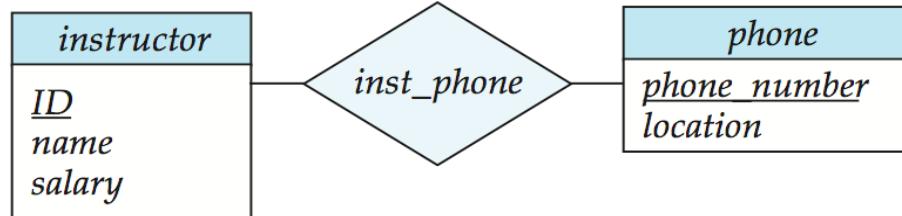
- 7.1 Overview of the Design Process
- 7.2 The Entity-Relationship Model
- 7.3 Constraints
- 7.4 Removing Redundant Attributes in Entity Sets
- 7.5 Entity-Relationship Diagrams
- 7.6 Reduction to Relational Schemas
- 7.7 Entity-Relationship Design Issues
- 7.8 Extended E-R Features
- 7.9 Alternative Notation for Modeling Data
- 7.10 Other Aspects of Database Design



ER Design Issue: Entity sets vs. Attributes

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

OR



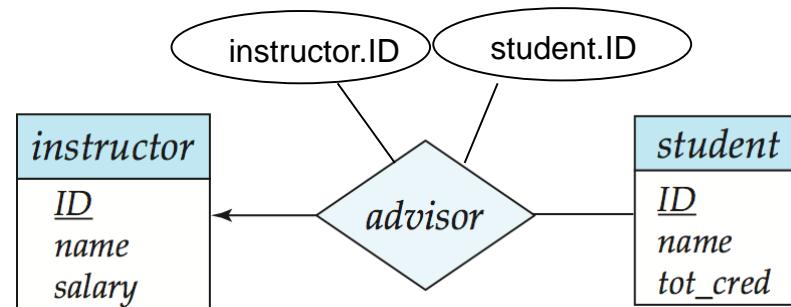
OR

<i>instructor</i>
<u>ID</u>
<u>name</u>
<u>salary</u>
{ <u>phone_number</u> }

■ What constitutes an attribute? Vs What constitutes an entity set?

- There is no simple answers
- Depends on the structure and semantics of real-world enterprise

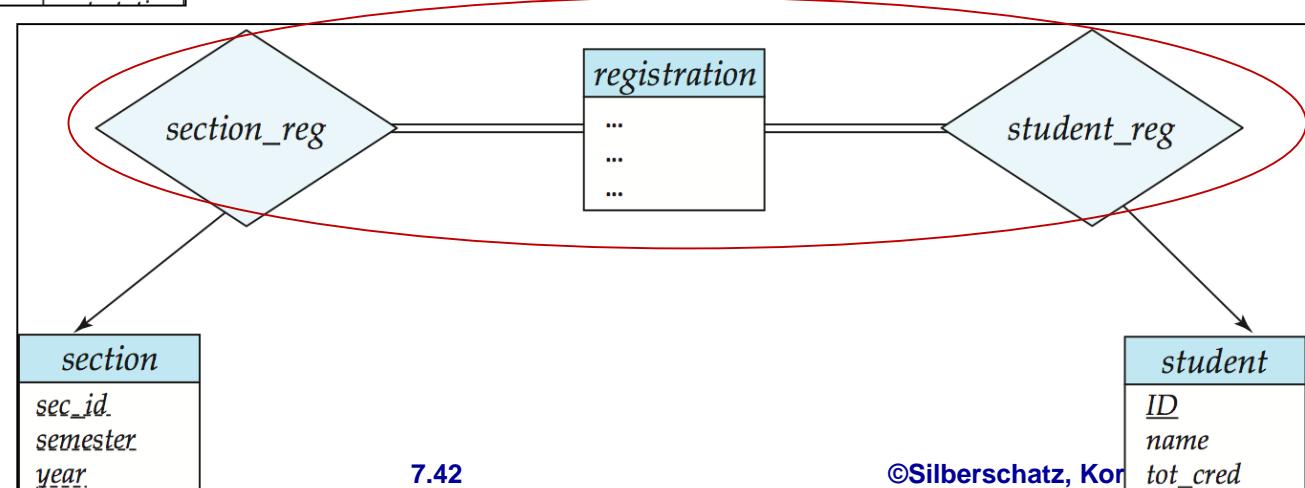
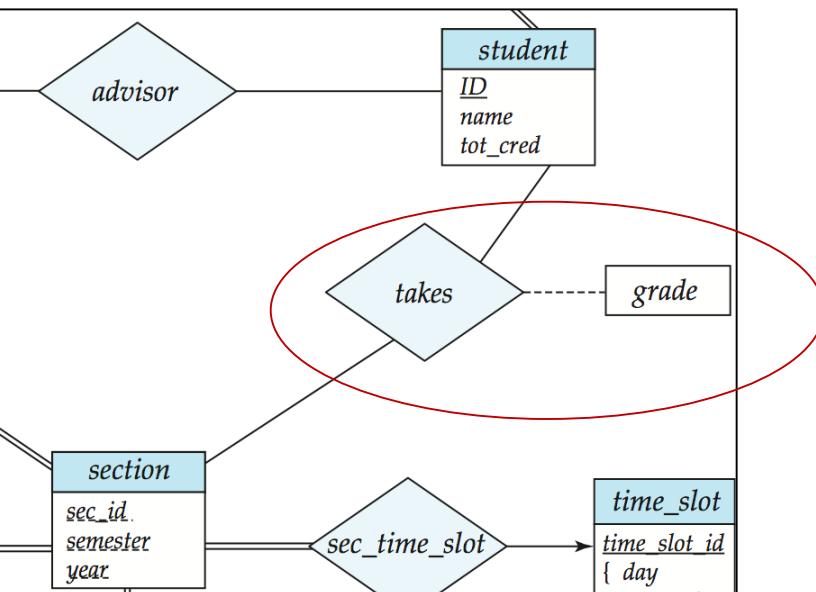
■ Another frequent mistake





ER Design Issue: Entity sets vs. Relationship Sets

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





ER Design Issue: Binary vs N-ary Relationship Set

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



Converting Non-Binary Relationships to Binary Form [1/2]

- 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 3 relationship sets**:
 1. R_A , relating E and A
 2. R_B , relating E and B
 3. R_C , relating E and C
 - Create **a special identifying attribute for E**
 - Add any attributes of R to E
 - For each relationship (a_i, b_i, c_i) in R , create
 1. a new entity e_i in the entity set E
 2. add (e_i, a_i) to R_A
 3. add (e_i, b_i) to R_B
 4. add (e_i, c_i) to R_C

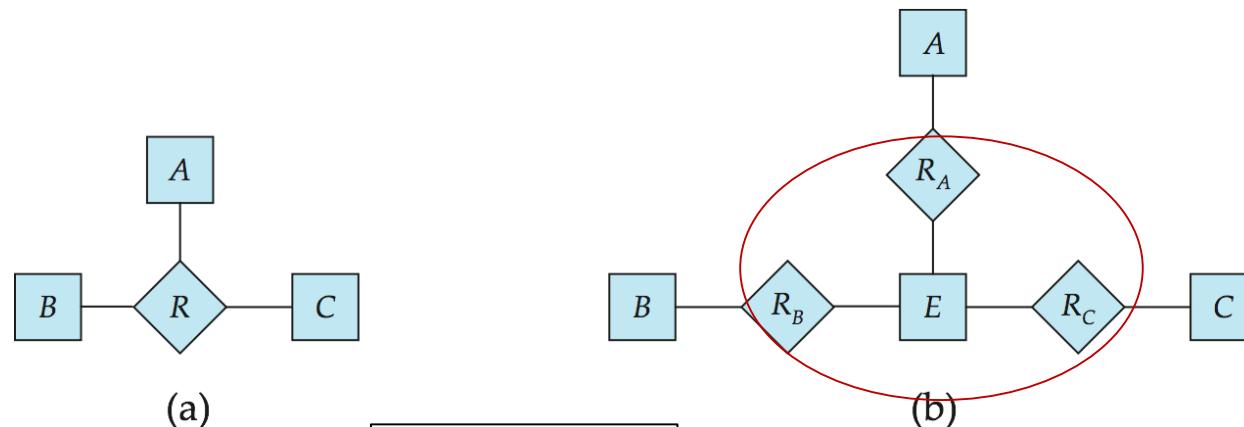
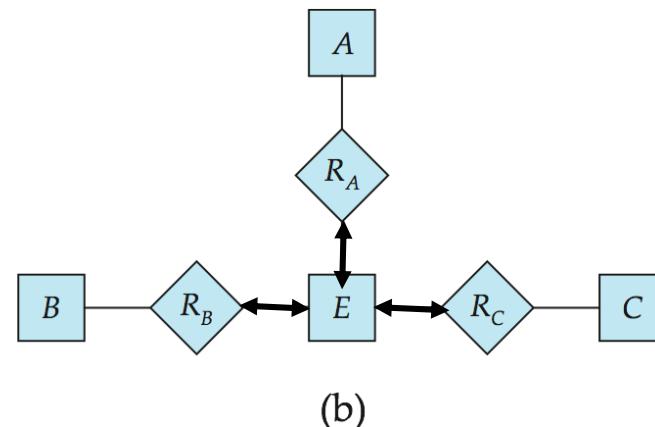
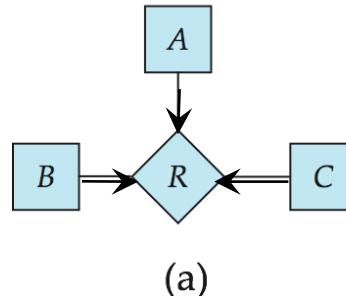


Fig 7.19



Converting Non-Binary Relationships to Binary Form [2/2]

- Not always possible to translate constraints in non-binary relationships into constraints on the binary relationships
 - Suppose R is many-to-one from A, B, and C
 - Exercise: add constraints to the relationships R_A , R_B and R_C to ensure that a newly created entity corresponds to exactly one entity in each of entity sets A, B and C





Chapter 7: Entity-Relationship Model

- 7.1 Overview of the Design Process
- 7.2 The Entity-Relationship Model
- 7.3 Constraints
- 7.4 Removing Redundant Attributes in Entity Sets
- 7.5 Entity-Relationship Diagrams
- 7.6 Reduction to Relational Schemas
- 7.7 Entity-Relationship Design Issues
- 7.8 Extended E-R Features
- 7.9 Alternative Notation for Modeling Data
- 7.10 Other Aspects of Database Design



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.

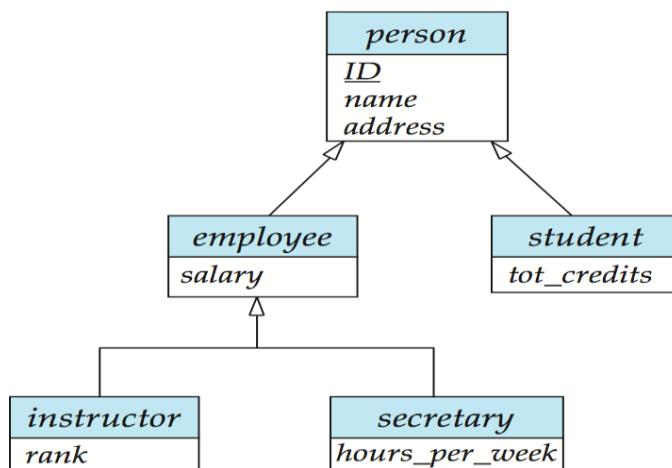


Fig 7.21: Specialization Example



Extended ER Features: Generalization

- **A bottom-up design process** – combine a number of entity sets that share the same features into a higher-level entity set.
- Specialization and generalization are simple inversions of each other
- The terms specialization and generalization are used interchangeably.
- Can have **multiple specializations** of an entity set based on different features.
 - E.g., *permanent_employee* vs. *temporary_employee*, in addition to *instructor* vs. *secretary*
- Each particular employee would be
 - a member of one of *permanent_employee* or *temporary_employee*,
 - and also a member of one of *instructor*, *secretary*
- The ISA relationship also referred to as **superclass - subclass** relationship



Design Constraints on a Specialization/Generalization

- Constraint on which entities can be members of a given lower-level entity set.
 - **condition-defined** (condition or predicate based)
 - ▶ Example: all customers over 65 years are members of *senior-citizen* entity set; *senior-citizen* ISA *person*.
 - **user-defined**
 - ▶ Example: a new employee is assigned to one of teams in the company
- Constraint on whether entities may belong to more than one lower-level entity set
 - **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
- **Completeness constraint** on 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

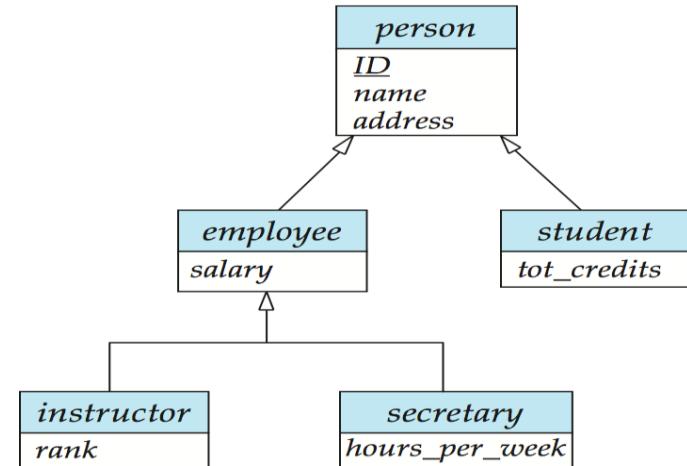


Relation Schemas for Specialization/Generalization

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

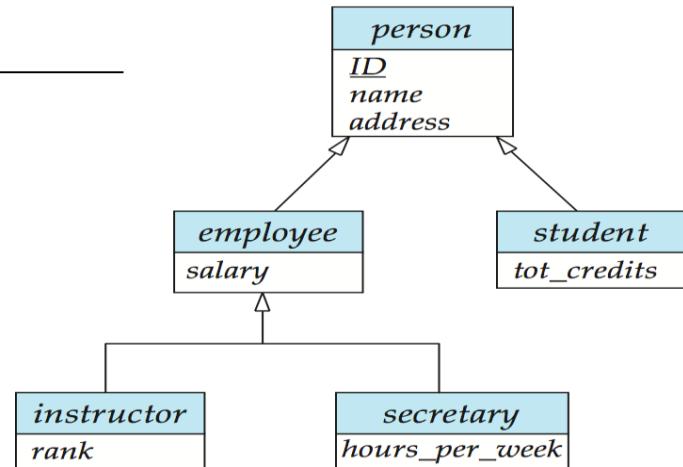


Relation Schemas for Specialization/Generalization (cont.)

Method 2:

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

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



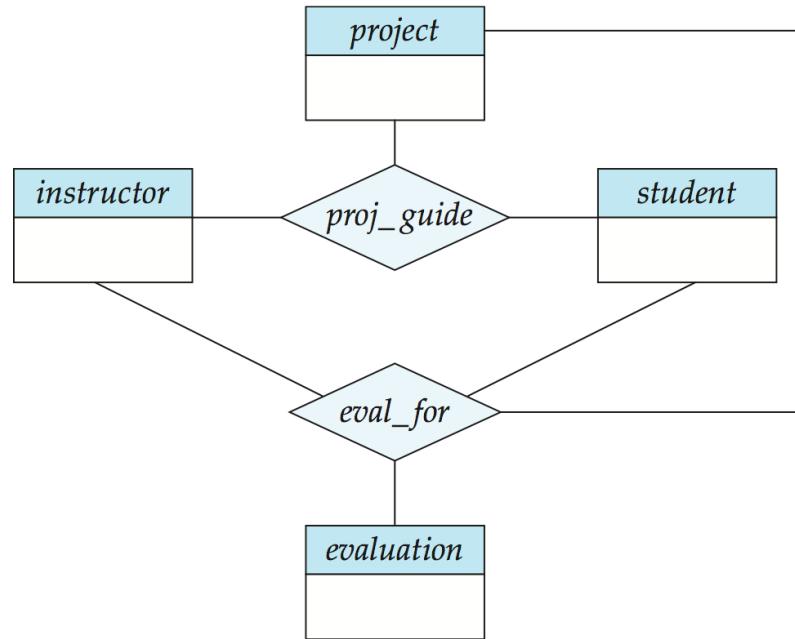
- If specialization is total, the schema for the generalized entity set (*person*) not required to store information
 - Can be defined as a “view” relation containing union of specialization relations
 - But explicit schema may still be needed for foreign key constraints
- Drawback: *name, street* and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



Extended ER Feature: 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

Fig 7.22



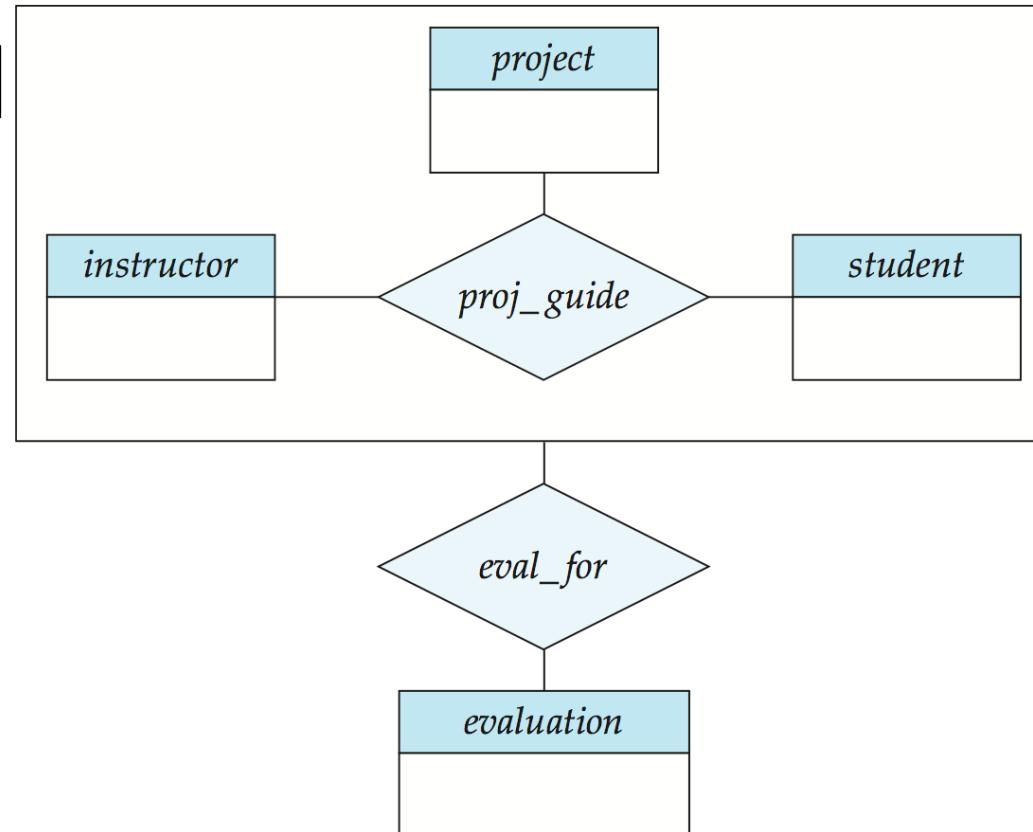
- Relationship sets *eval_for* and *proj_guide* represent overlapping information
- Eliminate this redundancy via *aggregation*
 - Treat relationship as an abstract entity
 - Allows relationships between relationships
 - Abstraction of relationship into new entity



Extended ER Features: 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](#)

Fig 7.23

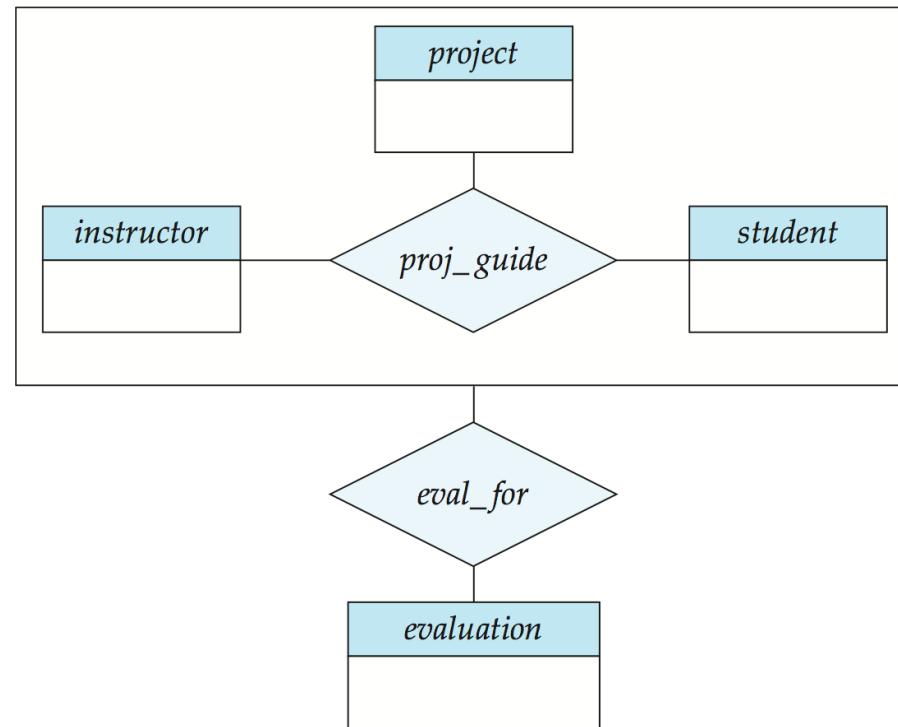




Relation Schemas for Aggregation

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

eval_for (student_ID, project_id, instructor_ID, evaluation_id)





E-R Design Decisions: Summary

- 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.
 - Noun → entity set
 - Verb → relationship set
 - Adjective → attribute
- 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.

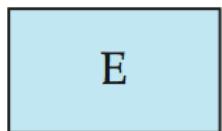


Chapter 7: Entity-Relationship Model

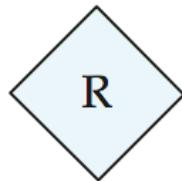
- 7.1 Overview of the Design Process
- 7.2 The Entity-Relationship Model
- 7.3 Constraints
- 7.4 Removing Redundant Attributes in Entity Sets
- 7.5 Entity-Relationship Diagrams
- 7.6 Reduction to Relational Schemas
- 7.7 Entity-Relationship Design Issues
- 7.8 Extended E-R Features
- 7.9 Alternative Notation for Modeling Data
- 7.10 Other Aspects of Database Design



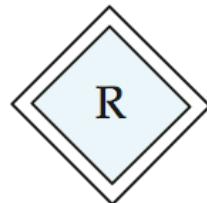
UML-like E-R Notation in the Book [1/2]



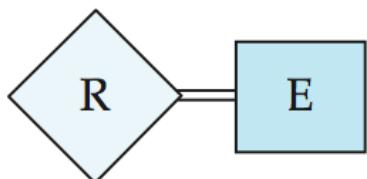
entity set



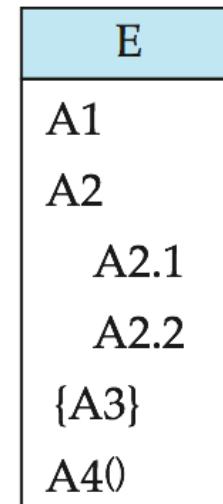
relationship set



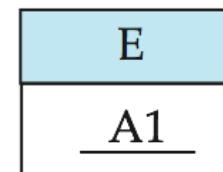
identifying
relationship set
for weak entity set



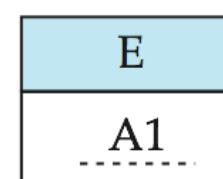
total participation
of entity set in
relationship



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



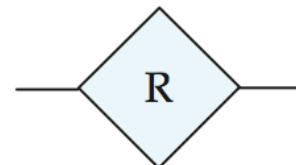
primary key



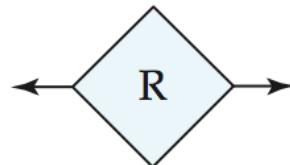
discriminating
attribute of
weak entity set



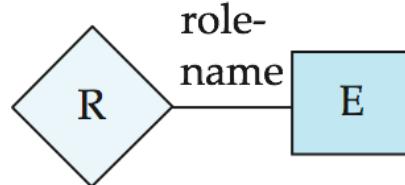
UML-like E-R Notation in the Book [2/2]



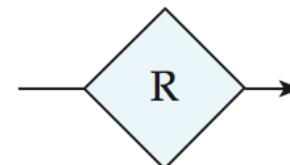
many-to-many
relationship



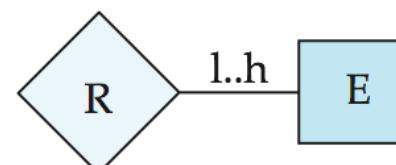
one-to-one
relationship



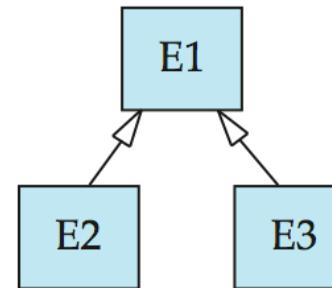
role indicator



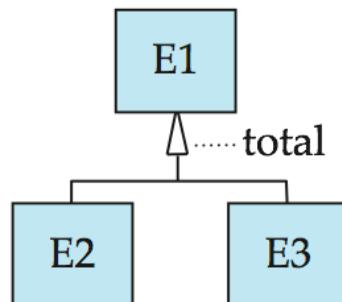
many-to-one
relationship



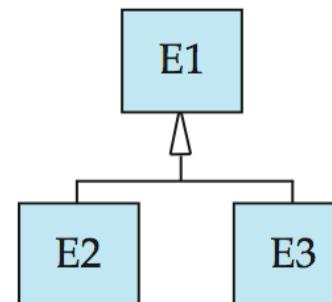
cardinality
limits



ISA: generalization
or specialization



total (disjoint)
generalization



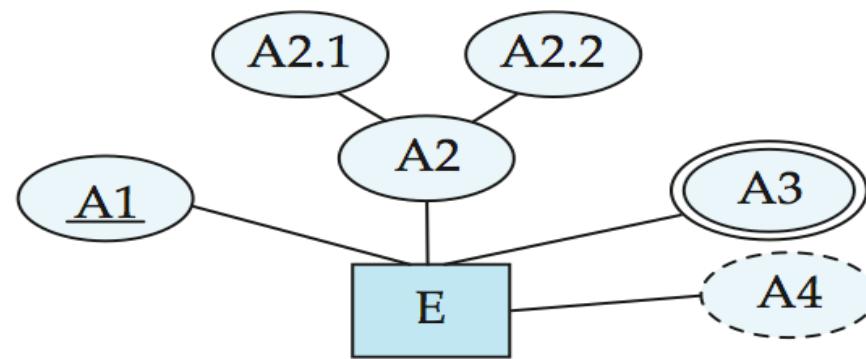
disjoint
generalization



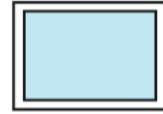
Alternative ER Notations [1/2]

- Chen's ER notation: P.P. Chen, 1976, "The ER Model: Toward a Unified View of Data, ACM Transactions of Database Systems
- IDE1FX: The US National Institute of Standards and Technology (NIST), 1993

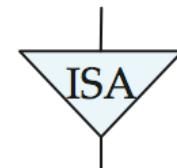
entity set E with
simple attribute A1,
composite attribute A2,
multivalued attribute A3,
derived attribute A4,
and primary key A1



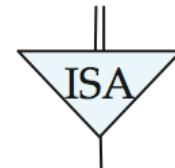
weak entity set



generalization



total
generalization

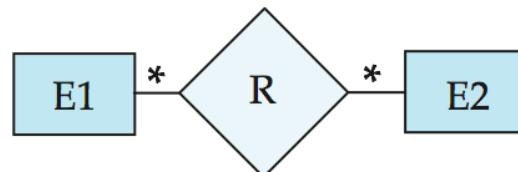




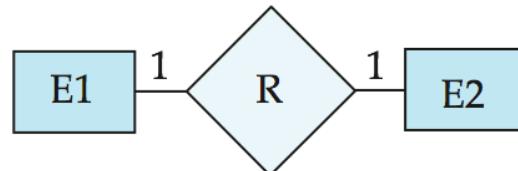
Alternative ER Notations [2/2]

Chen's ER Notation

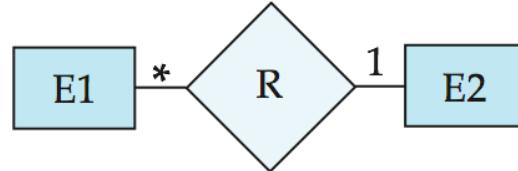
many-to-many
relationship



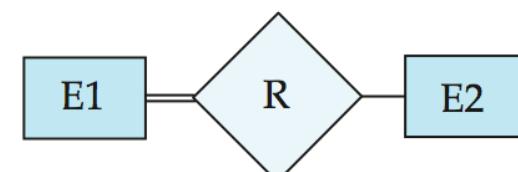
one-to-one
relationship



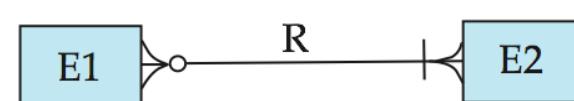
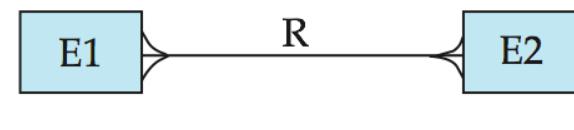
many-to-one
relationship



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



IDE1FX (Crows feet notation)





UML (Unified Modeling Language) [1/2]

- Standard by [OMG](#) (Object Management Group)
 - Long history of various modeling languages
 - ▶ Rumbaugh's OMT-1, OMT-2
 - ▶ Booch92 Method
 - ▶ Jacobson OOSE Method
 - Now a “de facto” standard
- UML has many components (10 Diagrams) to graphically model different aspects of an entire software system
 - [Use case Diagram](#)
 - [Class/Object diagram](#)
 - [Sequence/Collaboration Diagram](#)
 - [State/Activity Diagram](#)
 - [Component/Deployment Diagram](#)
 -

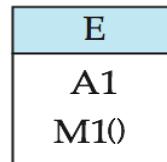


UML (Unified Modeling Language) [2/2]

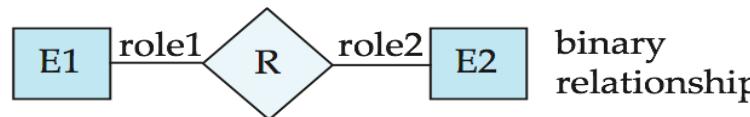
- Various Application besides Database Modeling
 - Software design
 - Hardware design
 - ▶ Computer hardware
 - ▶ Ship building
 - ▶ Mechanical design
 - ▶
- In our textbook, ER diagram is drawn with UML-like notations



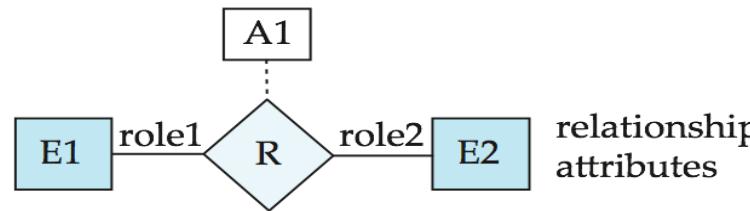
Our Book UML-like ER Diagram Notations



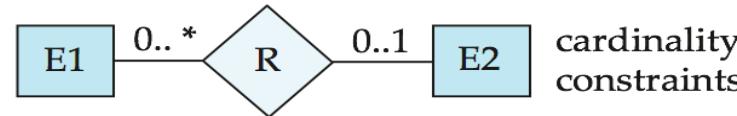
entity with attributes (simple, composite, multivalued, derived)



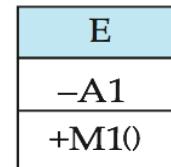
binary relationship



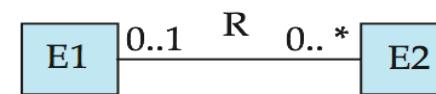
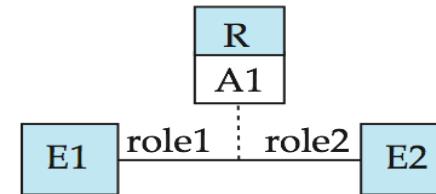
relationship attributes



cardinality constraints



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



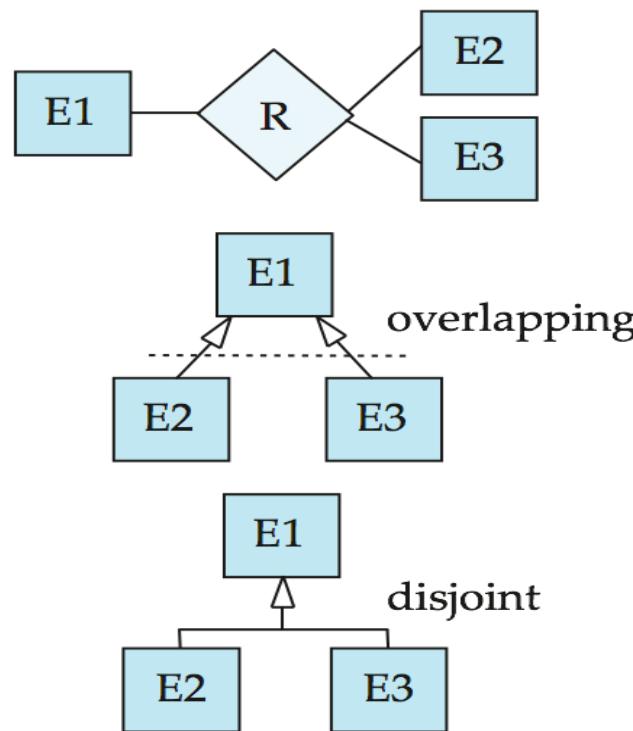
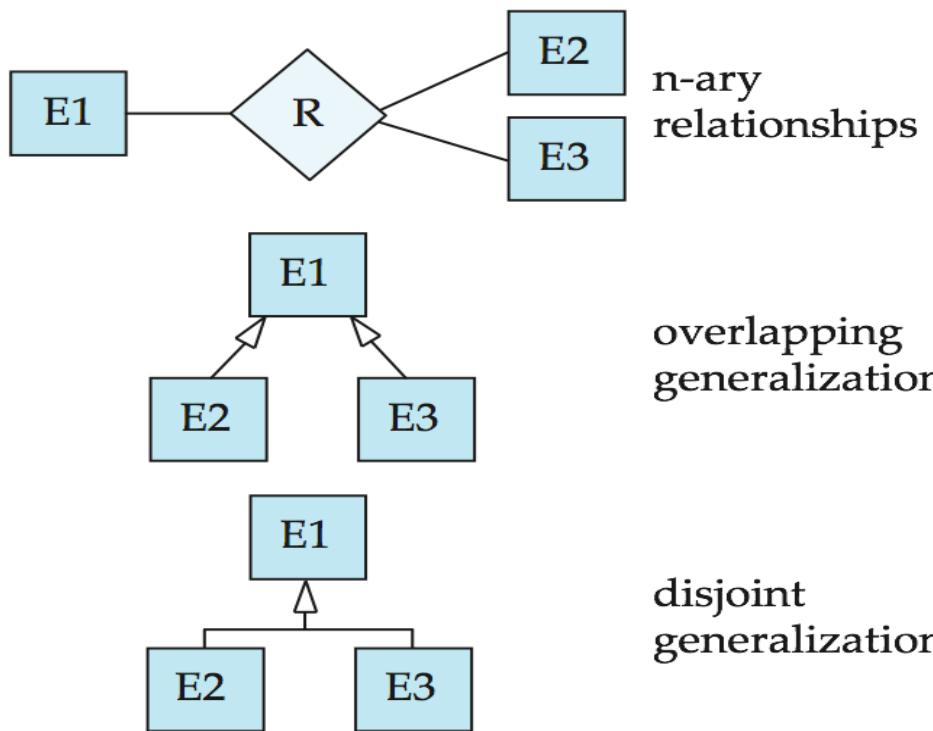
*Note reversal of position in cardinality constraint depiction

- **Binary relationship sets** are represented in UML by just drawing a line connecting the entity sets. The relationship set name is written adjacent to the line.
- **The role** played by an entity set in a relationship set may also be specified by writing the role name on the line, adjacent to the entity set.
- **The relationship set name** may alternatively be written in a box, along with attributes of the relationship set, and the box is connected, using a dotted line, to the line depicting the relationship set.



Our Book UML-like ER Diagram Notations

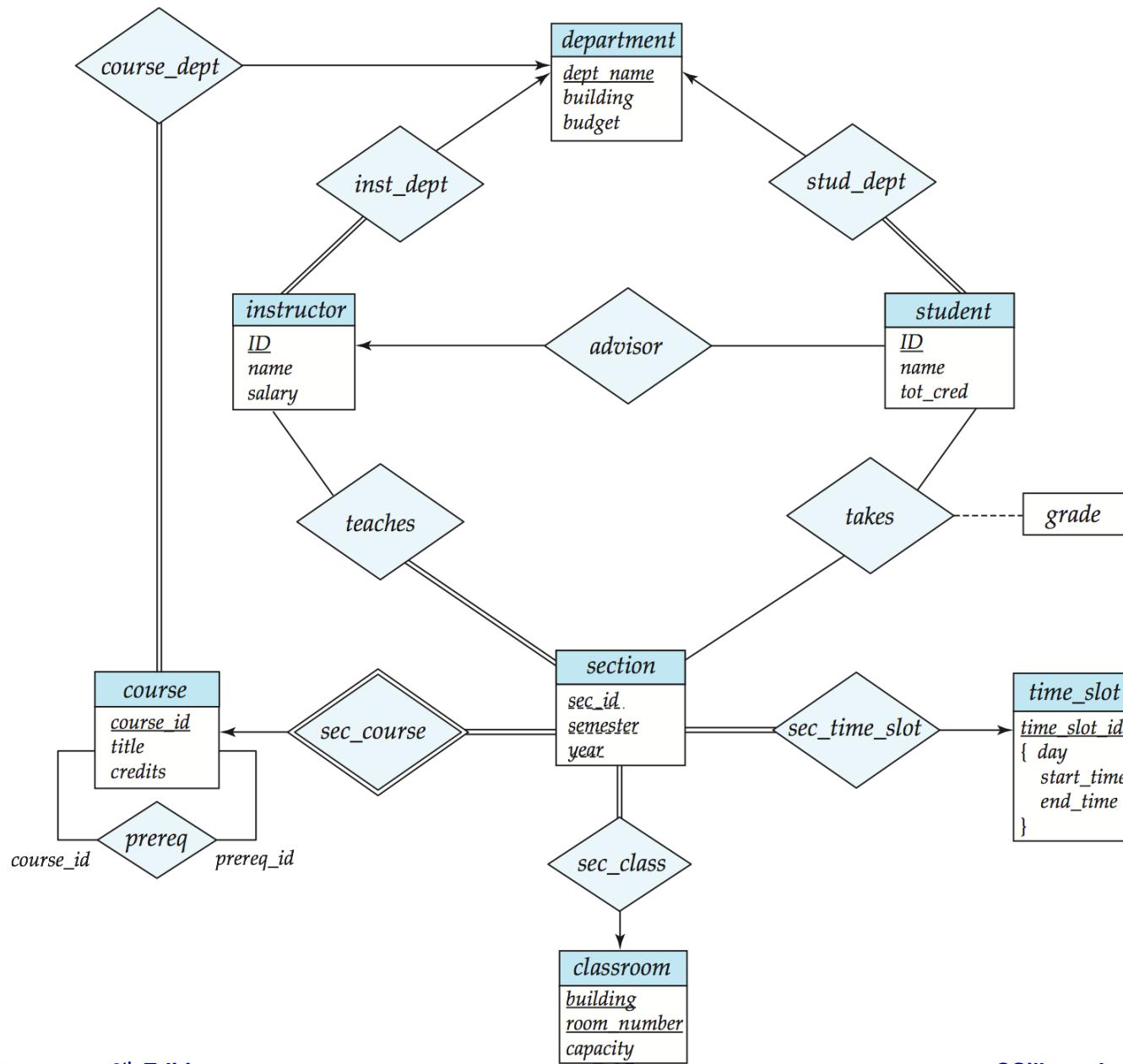
vs. Original UML Class Diagrams



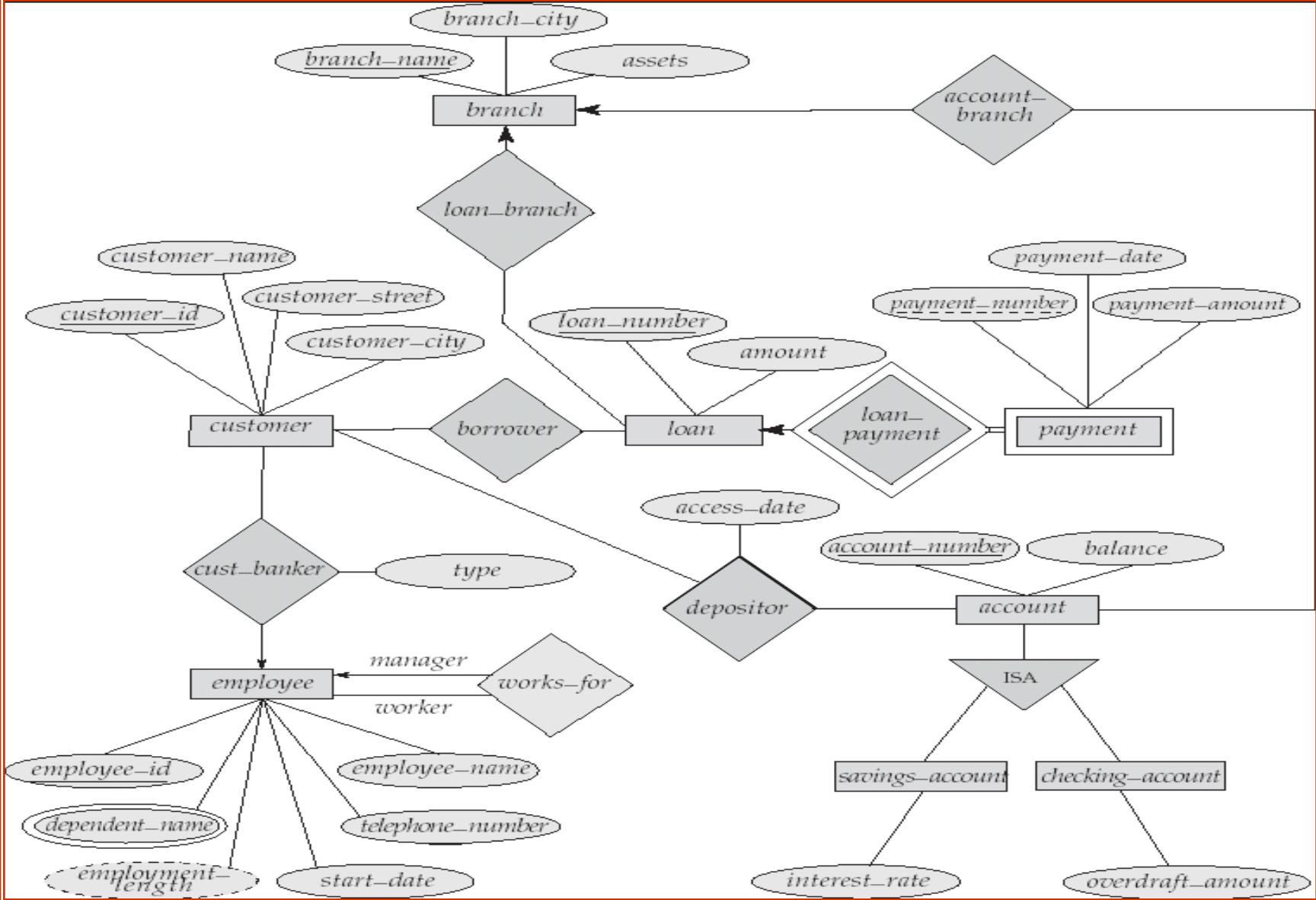
*Generalization can use merged or separate arrows independent of disjoint/overlapping



UML-like ER Diagram for University Database



Chen's ER Notation for Bank Database





Chapter 7: Entity-Relationship Model

- 7.1 Overview of the Design Process
- 7.2 The Entity-Relationship Model
- 7.3 Constraints
- 7.4 Removing Redundant Attributes in Entity Sets
- 7.5 Entity-Relationship Diagrams
- 7.6 Reduction to Relational Schemas
- 7.7 Entity-Relationship Design Issues
- 7.8 Extended E-R Features
- 7.9 Alternative Notation for Modeling Data
- 7.10 Other Aspects of Database Design



Other Aspects of Database Design

- Data Constraints and Relational Database Design
- Usage Requirements: Queries, Performance
 - Throughput
 - Response time
- Authorization Requirements
 - Creating views
 - Control user's accessibility to views and relations
- Data Flow, Work Flow
 - Specification of a series of queries and updates for accomplishing tasks
- Schema updates rarely happen, but are very hectic!
- Data Conversion is necessary in interacting multiple databases
- Other Issues
 - Enterprise evolution
 - Interaction with other enterprises
 - End user vs. Database designer