



Chapter 7: Entity-Relationship Model

Database System Concepts, 7th 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



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.



Design Phases (Cont.)

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. Database design requires that we find a “good” collection of relation schemas.
 - Business decision – What attributes should we record in the database?
 - Computer Science decision – What relation schemas should we have and how should the attributes be distributed among the various relation schemas?
- Physical Design – Deciding on the physical layout of the database



Design Approaches

- Entity Relationship Model (covered in this chapter)
 - Models an enterprise as a collection of *entities* and *relationships*
 - 4 Entity: a “thing” or “object” in the enterprise that is distinguishable from other objects
 - Described by a set of *attributes*
 - 4 Relationship: an association among several entities
 - Represented diagrammatically by an *entity-relationship diagram*:
- Normalization Theory (Chapter 8)
 - Formalize what designs are bad, and test for them



Outline of the ER Model



ER model -- Database Modeling

- The ER data model was developed to facilitate database design by allowing specification of an **enterprise schema** that represents the overall logical structure of a database.
- The ER 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 an object that exists and is distinguishable from other objects.
 - Example: specific person, company, event, plant
- An **entity set** is a set of entities of the same type that share the same properties.
 - Example: set of all persons, companies, trees, holidays
- An entity is represented by a set of attributes; i.e., descriptive properties possessed by all members of an entity set.
 - Example:
$$\text{instructor} = (ID, name, street, city, salary)$$
$$\text{course} = (course_id, title, credits)$$
- A subset of the attributes form a **primary key** of the entity set; i.e., uniquely identifying each member of the set.



Entity Sets -- *instructor* and *student*

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



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

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

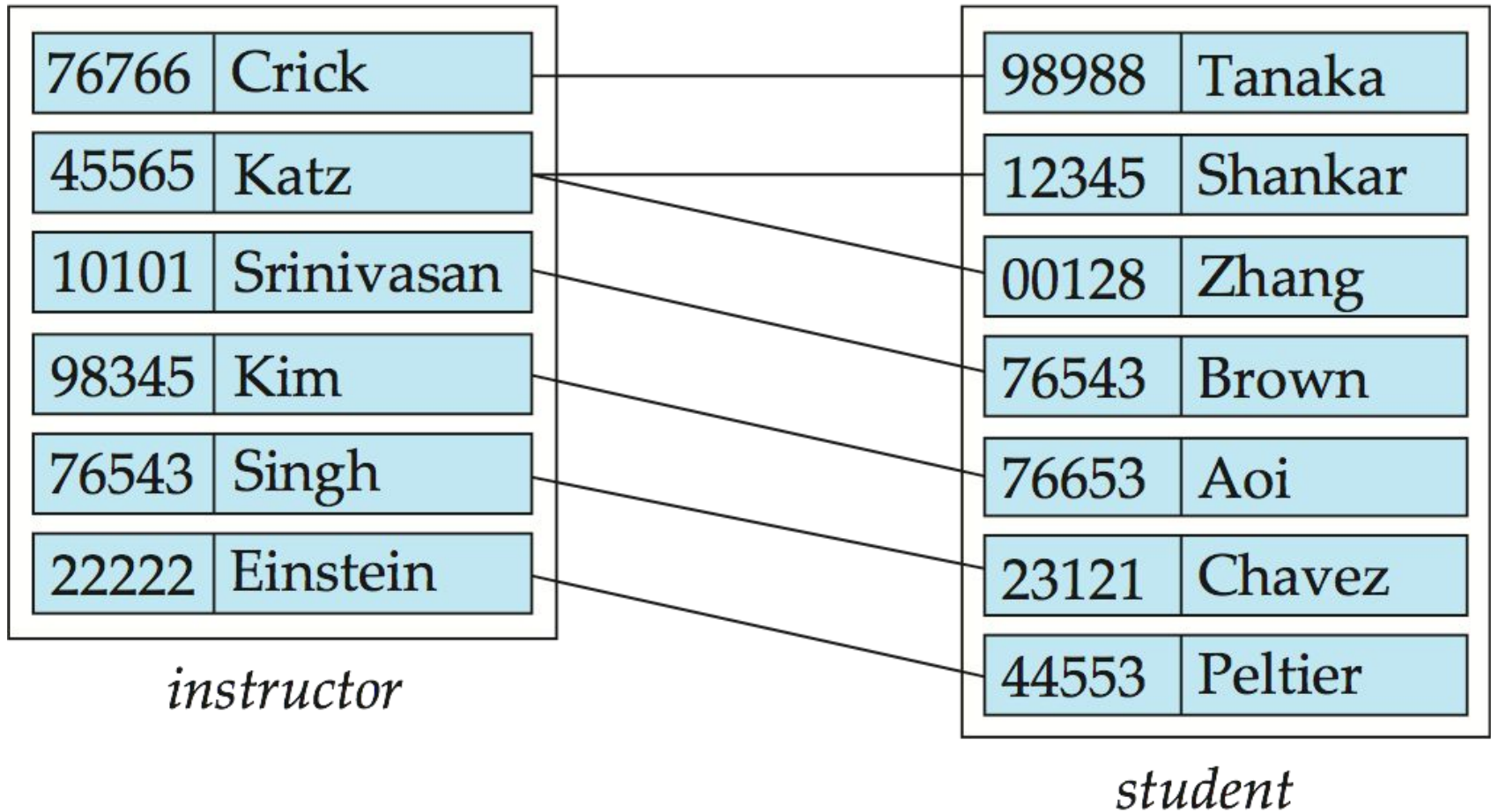
where (e_1, e_2, \dots, e_n) is a relationship

- Example:

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



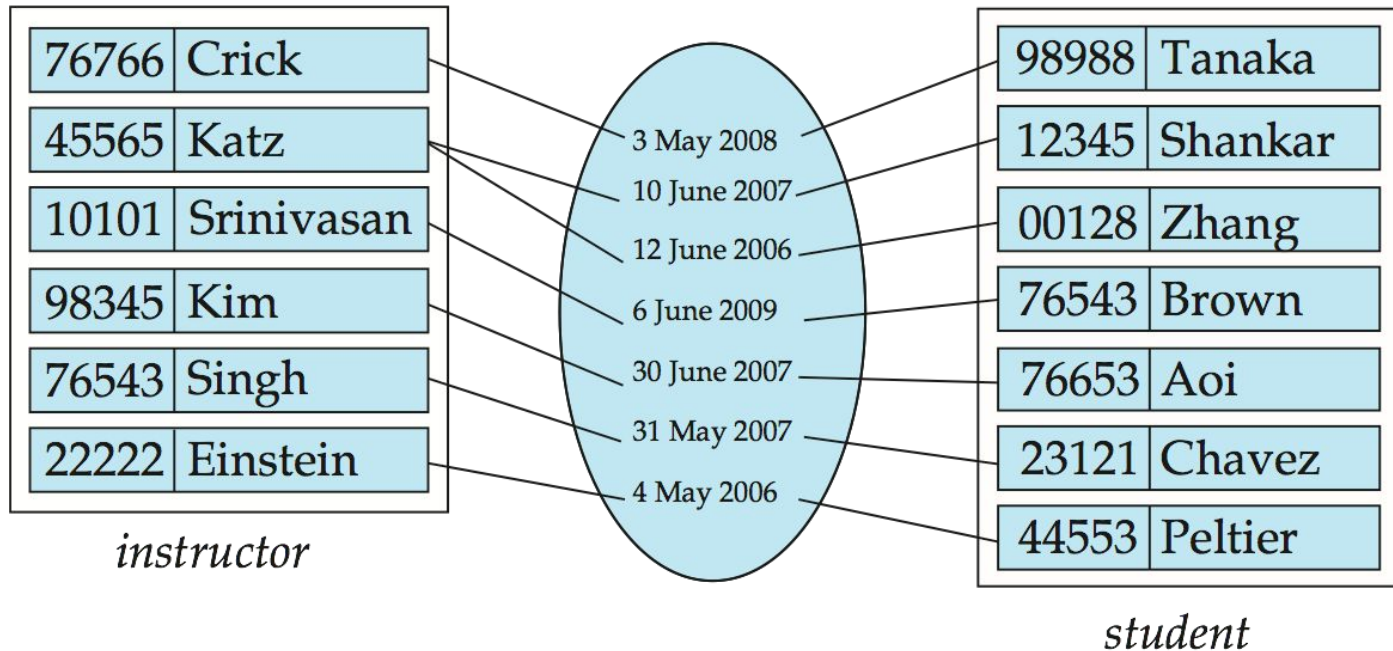
Relationship Set *advisor*





Relationship Sets (Cont.)

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





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

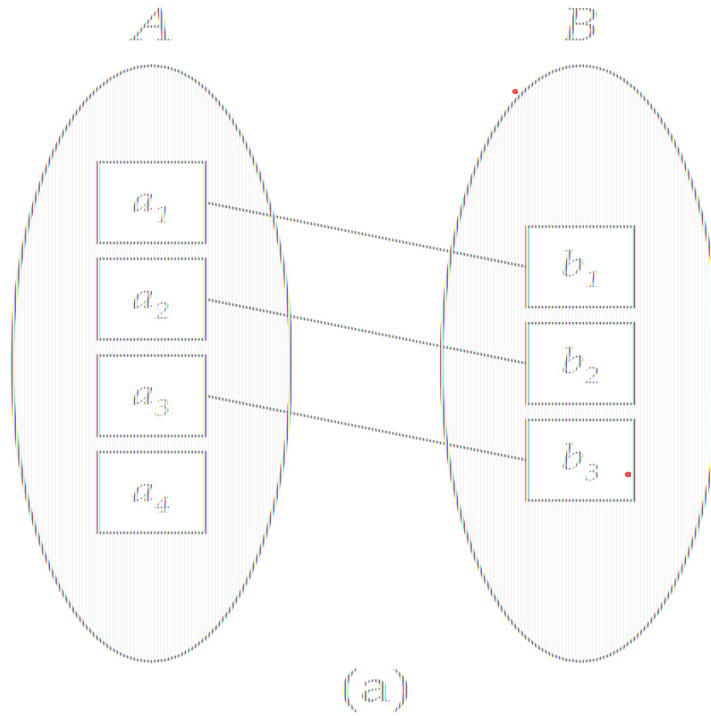


Mapping Cardinality Constraints

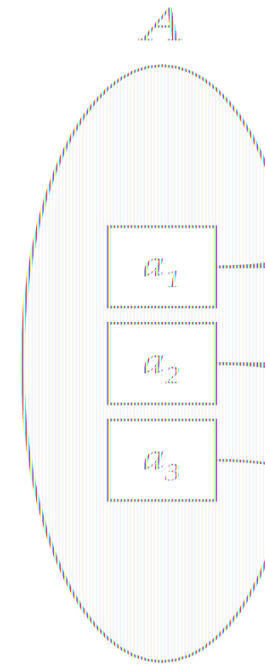
- Express the number of entities to which another entity can be associated via a relationship set.
- Most useful in describing binary relationship sets.
- For a binary relationship set the mapping cardinality must be one of the following types:
 - One to one
 - One to many
 - Many to one
 - Many to many



Mapping Cardinalities



One to one

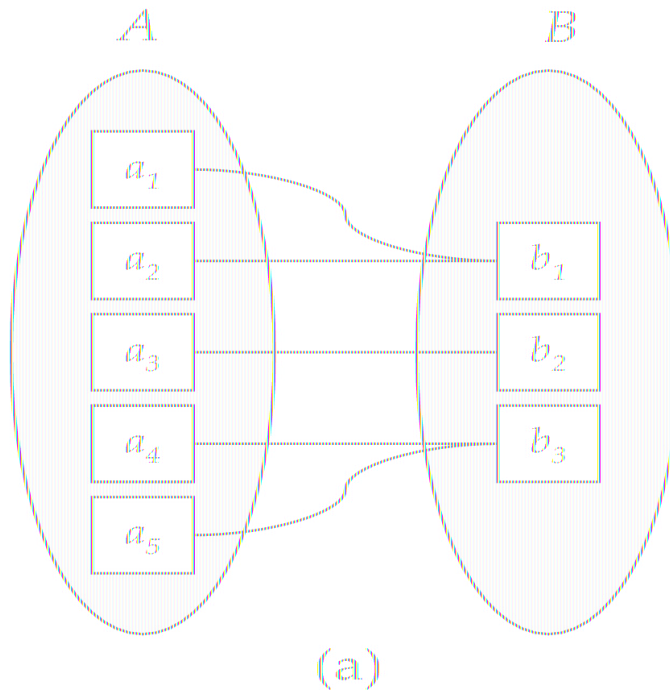


One to many

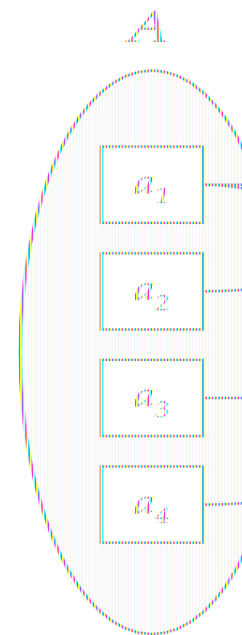
Note: Some elements in A and B may not be mapped to any elements in the other set



Mapping Cardinalities



Many to one



Many to many

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



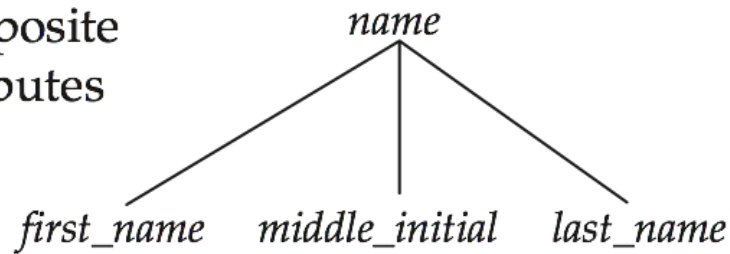
Complex Attributes

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

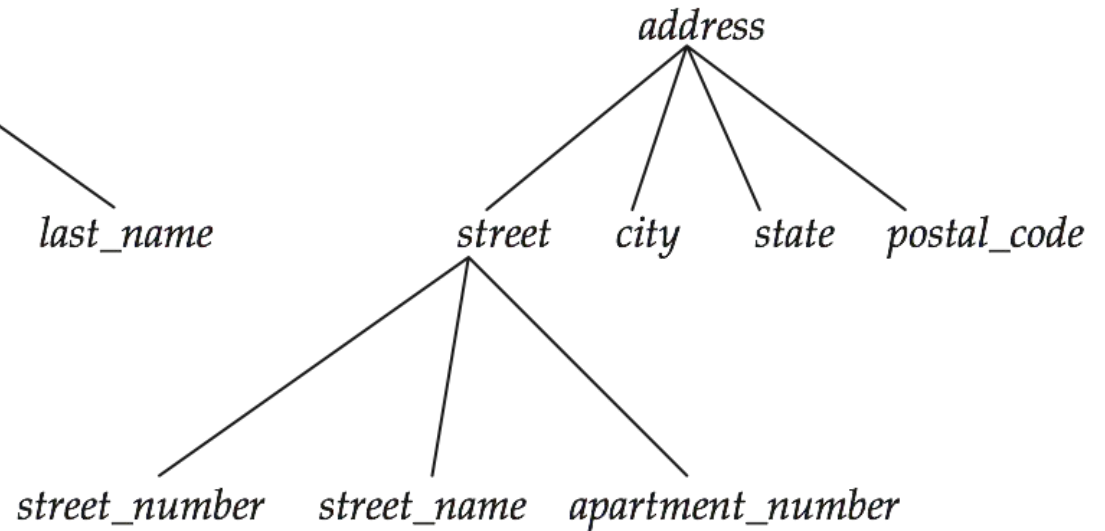


Composite Attributes

composite
attributes



component
attributes





Redundant Attributes

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



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.



Weak Entity Sets (Cont.)

- A 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*.

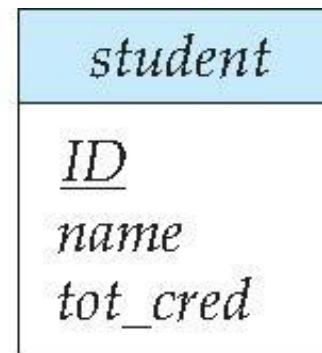
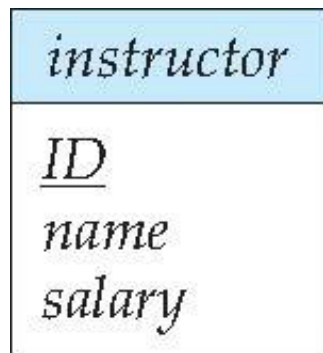


E-R Diagrams



Entity Sets

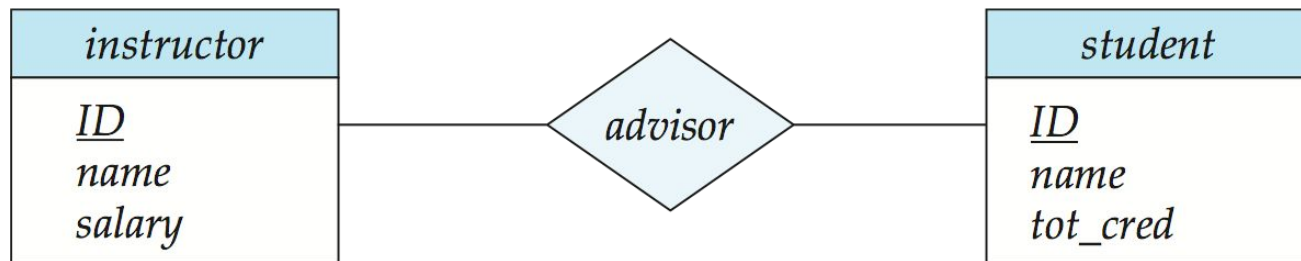
- Entities can be represented graphically as follows:
 - Rectangles represent entity sets.
 - Attributes listed inside entity rectangle
 - Underline indicates primary key attributes





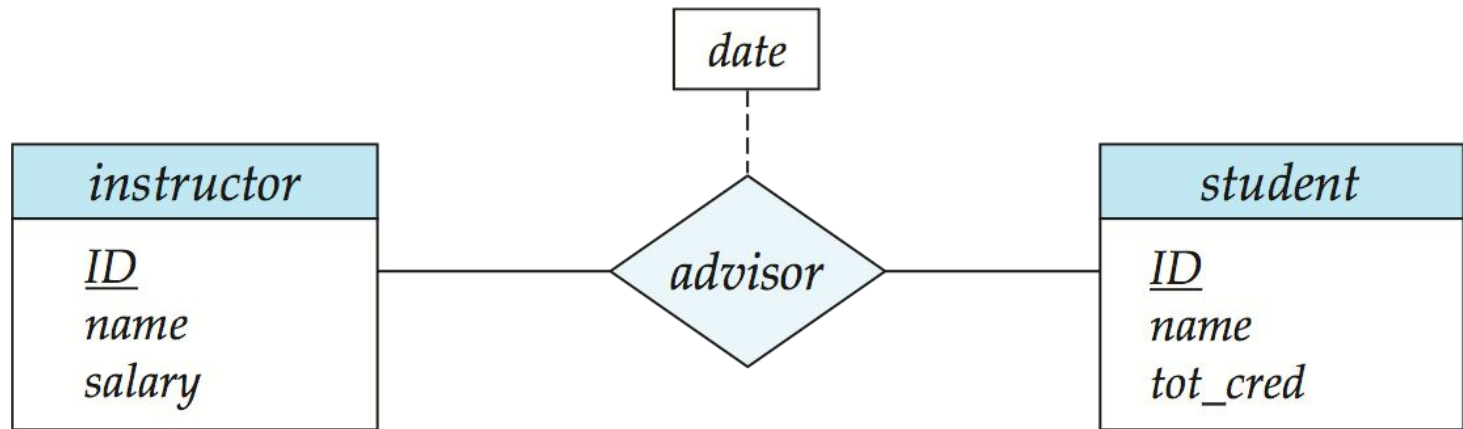
Relationship Sets

- Diamonds represent relationship sets.





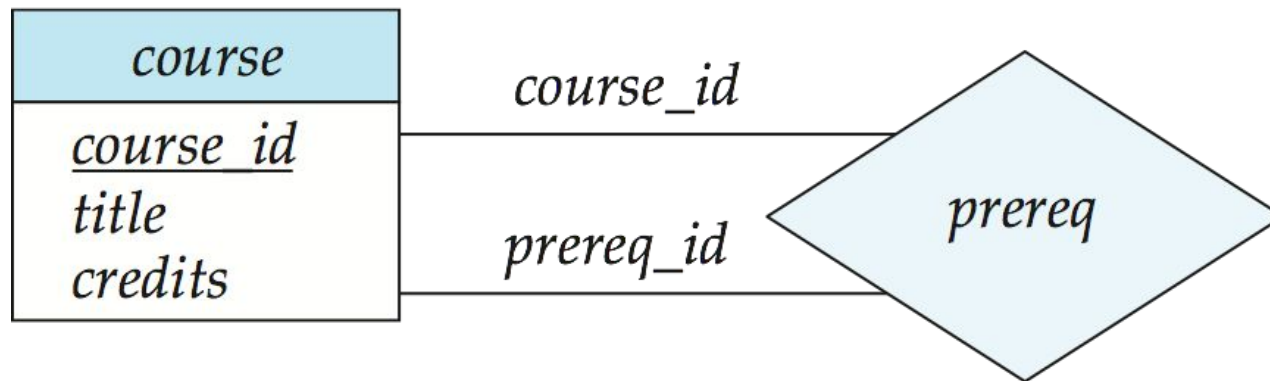
Relationship Sets with Attributes





Roles

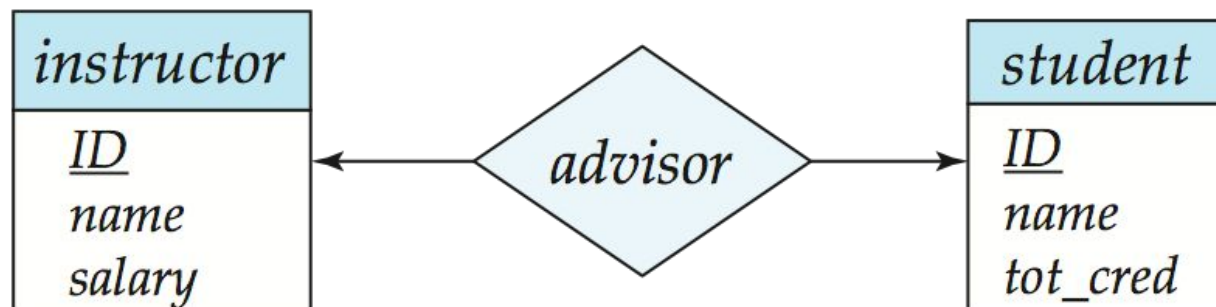
- Entity sets of a relationship need not be distinct
 - Each occurrence of an entity set plays a “role” in the relationship
- The labels “*course_id*” and “*prereq_id*” are called **roles**.





Cardinality Constraints

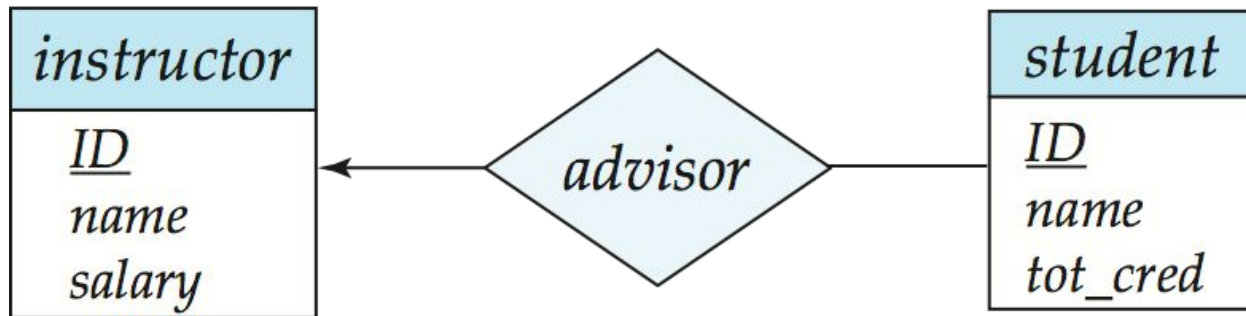
- We express cardinality constraints by drawing either a directed line (\rightarrow), signifying “one,” or an undirected line (—), signifying “many,” between the relationship set and the entity set.
- One-to-one relationship between an *instructor* and a *student* :
 - A student is associated with at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





One-to-Many Relationship

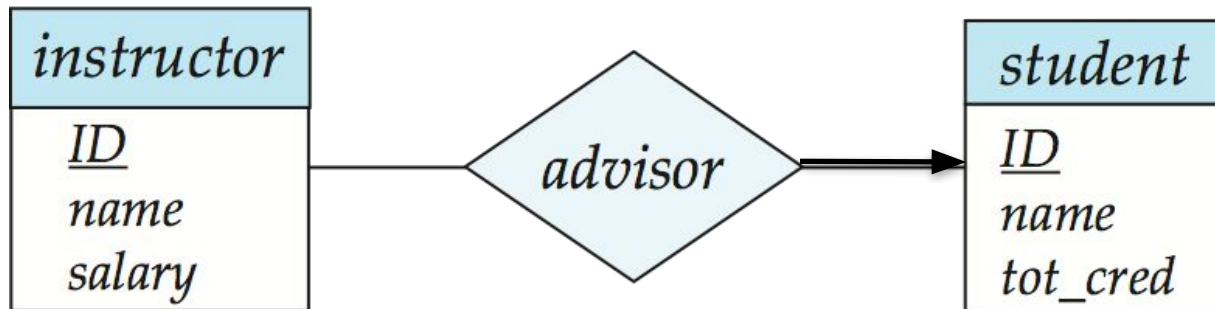
- one-to-many relationship between an *instructor* and a *student*
 - an instructor is associated with several (including 0) students via *advisor*
 - a student is associated with at most one instructor via *advisor*,





Many-to-One Relationships

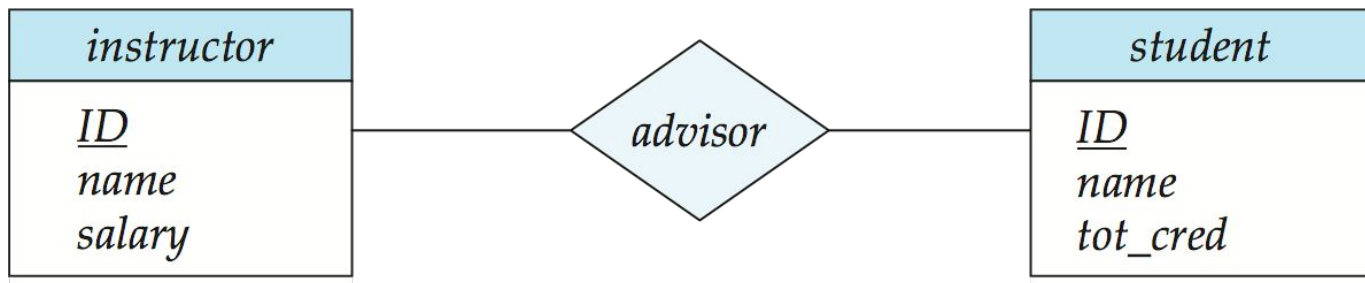
- In a many-to-one relationship between an *instructor* and a *student*,
 - an instructor is associated with at most one student via *advisor*,
 - and a student is associated with several (including 0) instructors via *advisor*





Many-to-Many Relationship

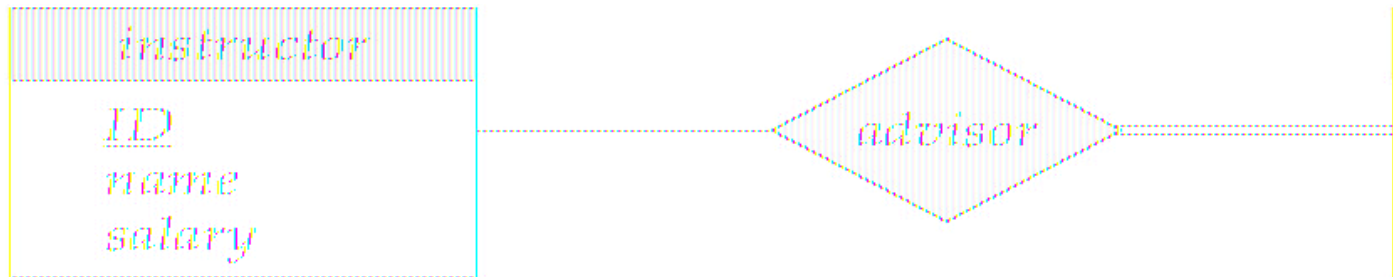
- An instructor is associated with several (possibly 0) students via *advisor*
- A student is associated with several (possibly 0) instructors via *advisor*





Total and Partial Participation

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



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

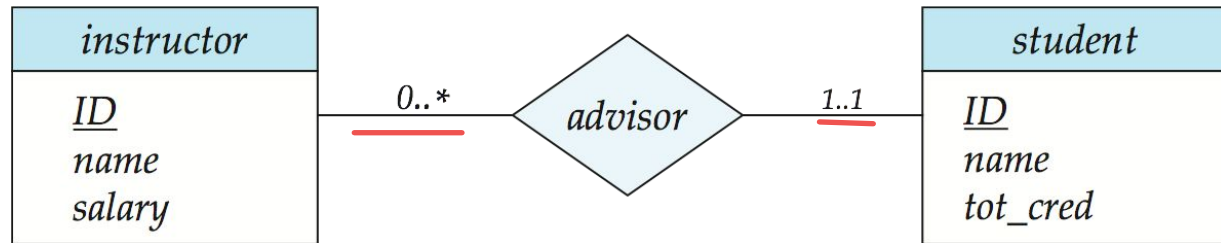
4 every *student* must have an associated instructor

- 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

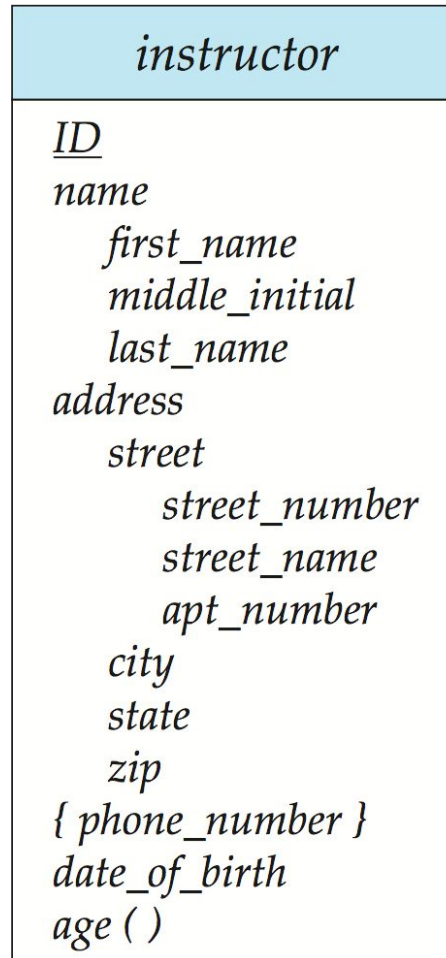
- A line may have an associated minimum and maximum cardinality, shown in the form $l..h$, where l is the minimum and h the maximum cardinality
 - A minimum value of 1 indicates total participation.
 - A maximum value of 1 indicates that the entity participates in at most one relationship
 - A maximum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



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



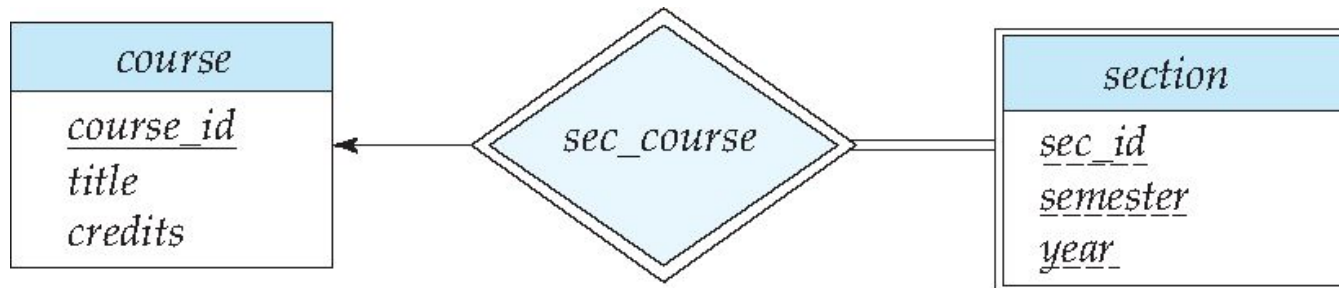
Notation to Express Entity with Complex Attributes





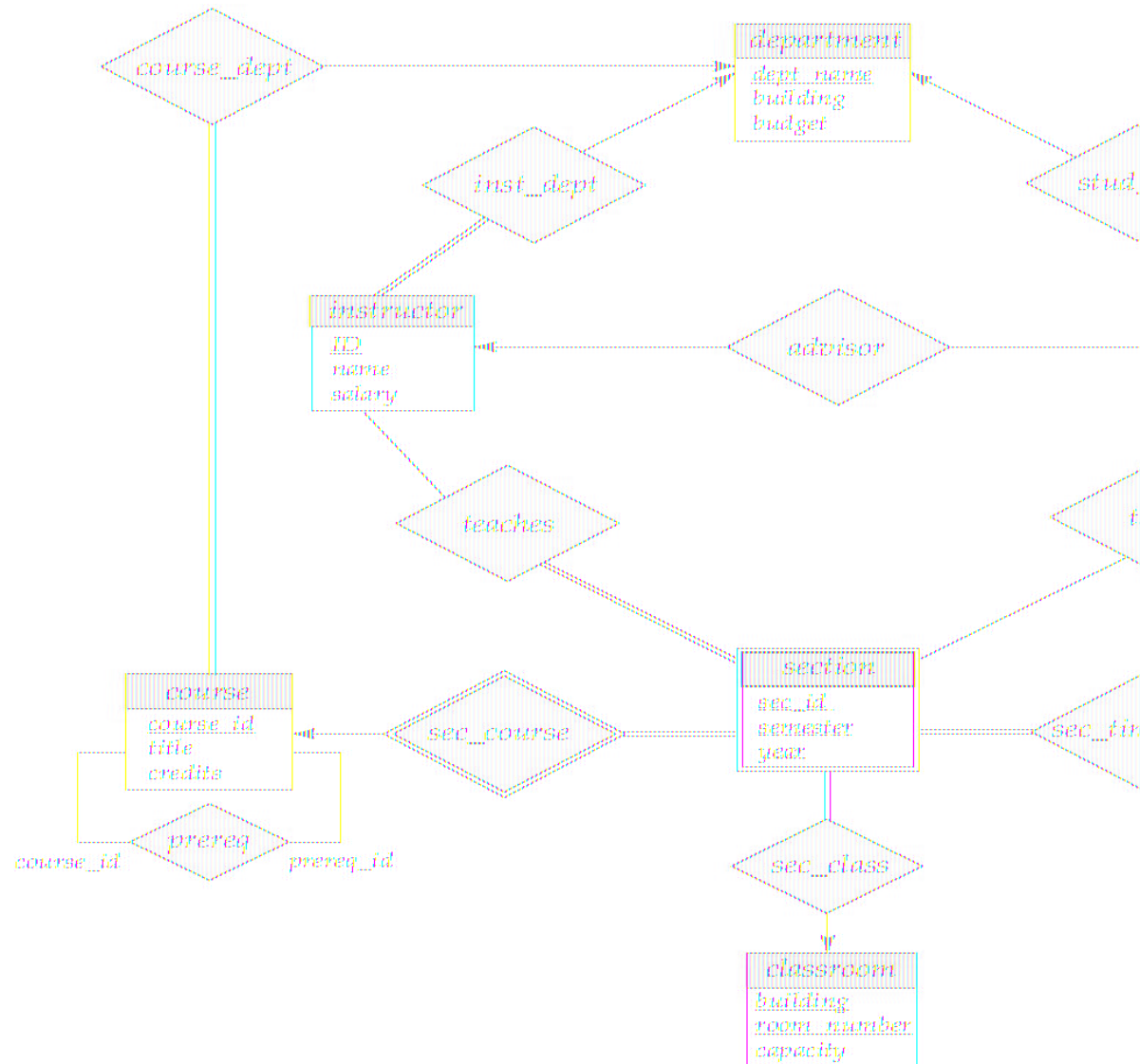
Expressing Weak Entity Sets

- 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



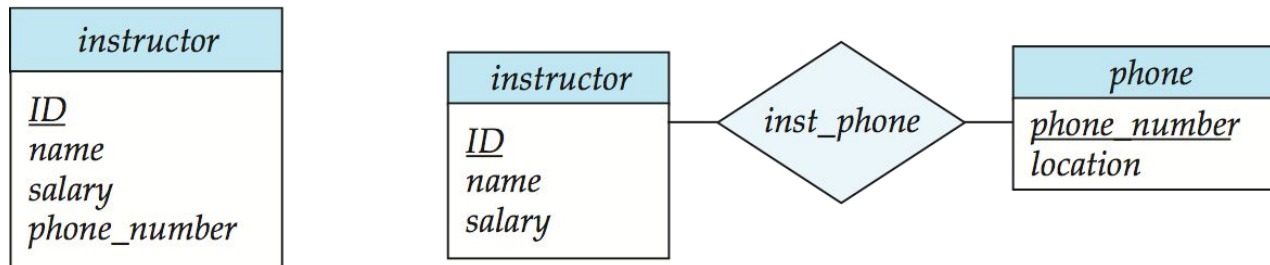


Design Issues



Entities vs. Attributes

- Use of entity sets vs. attributes



- Use of phone as an entity allows extra information about phone numbers (plus multiple phone numbers)
- The distinctions mainly depend on the structure of the real-world enterprise being modeled, and on the semantics associated with the attribute in question.

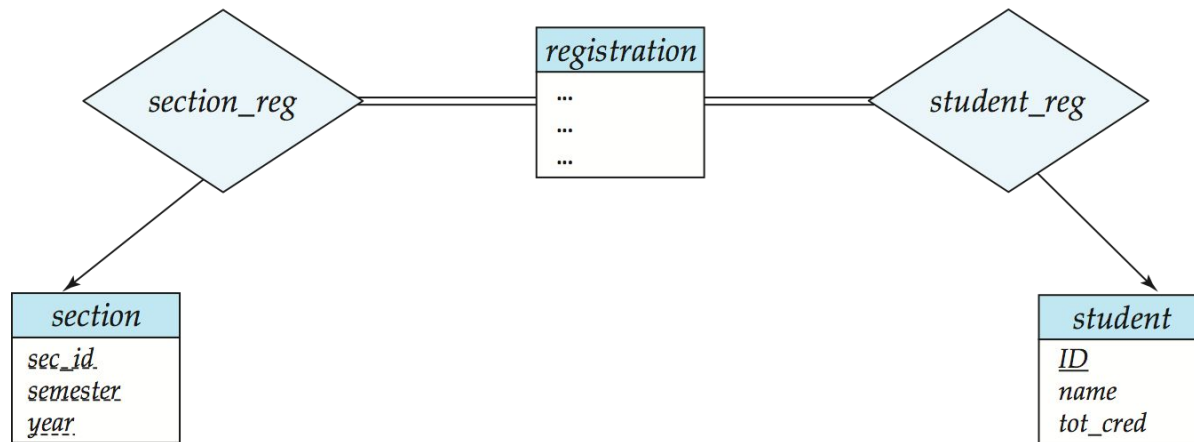


Entity sets vs. Relationship sets

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

It is not always clear whether an object is best expressed by an entity set or a relationship set.

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





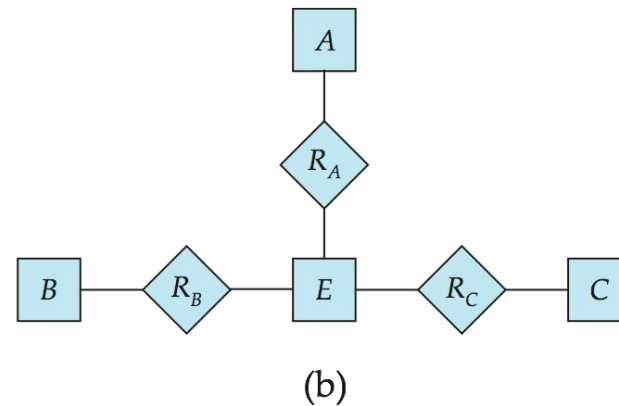
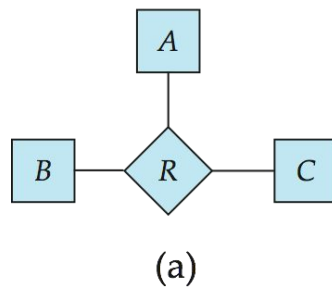
Binary Vs. Non-Binary Relationships

- Although it is possible to replace any non-binary (n -ary, for $n > 2$) relationship set by a number of distinct binary relationship sets, 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
 - For example, a ternary relationship *parents*, relating a child to his/her father and mother, is best replaced by two binary relationships, *father* and *mother*
 - But there are some relationships that are naturally non-binary.
Example: *proj_guide*



Converting Non-Binary Relationships to Binary Form

- In general, any non-binary relationship can be represented using binary relationships by creating an artificial entity set.
 - Replace R between entity sets A , B and C by an entity set E , and three relationship sets:
 1. R_A , relating E and A
 2. R_B , relating E and B
 3. R_C , relating E and C





Converting Non-Binary Relationships to Binary Form

- If the relationship set R had any attributes, these are assigned to entity set E
- Create an identifying attribute for E and add any attributes of R to E
- For each relationship (a_i, b_i, c_i) in R , create
 1. a new entity e_i in the entity set E
 2. add (e_i, a_i) to R_A
 3. add (e_i, b_i) to R_B
 4. add (e_i, c_i) to R_C



Converting Non-Binary Relationships (Cont.)

- Also need to translate constraints
 - Translating all constraints may not be possible
 - There may be instances in the translated schema that cannot correspond to any instance of R
 - We can avoid creating an identifying attribute by making E a weak entity set identified by the three relationship sets

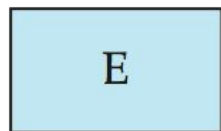


E-R Design Decisions

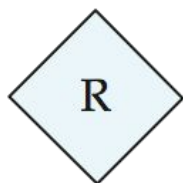
- The use of an attribute or entity set to represent an object.
- Whether a real-world concept is best expressed by an entity set or a relationship set.
- The use of a ternary relationship versus a pair of binary relationships.
- The use of a strong or weak entity set.



Summary of Symbols Used in E-R Notation



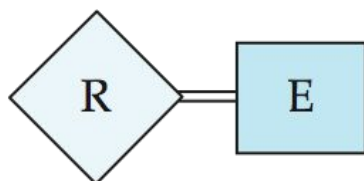
entity set



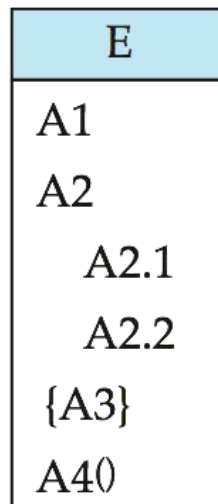
relationship set



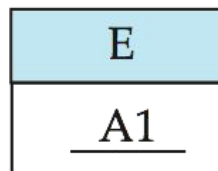
identifying
relationship set
for weak entity set



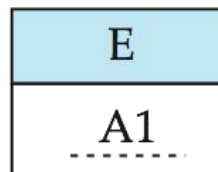
total participation
of entity set in
relationship



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



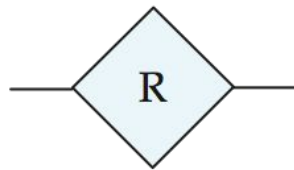
primary key



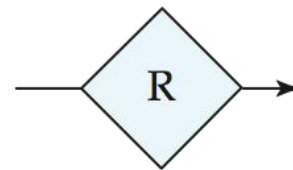
discriminating
attribute of
weak entity set



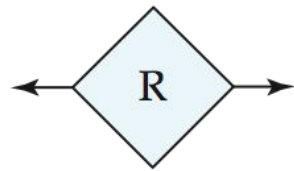
Symbols Used in E-R Notation (Cont.)



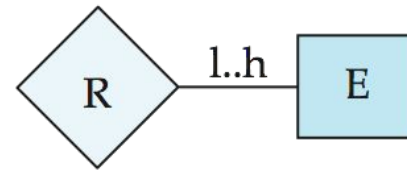
many-to-many
relationship



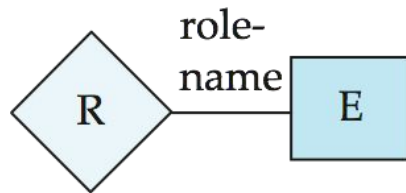
many-to-one
relationship



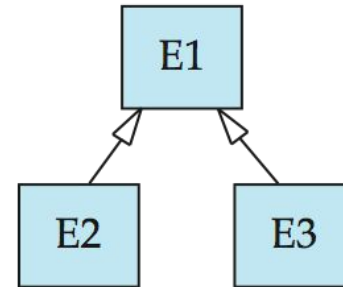
one-to-one
relationship



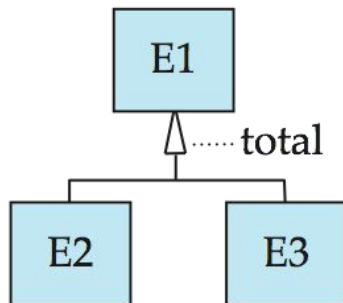
cardinality
limits



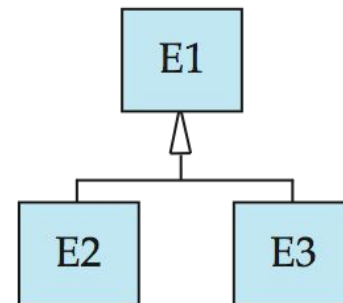
role indicator



ISA: generalization
or specialization



total (disjoint)
generalization



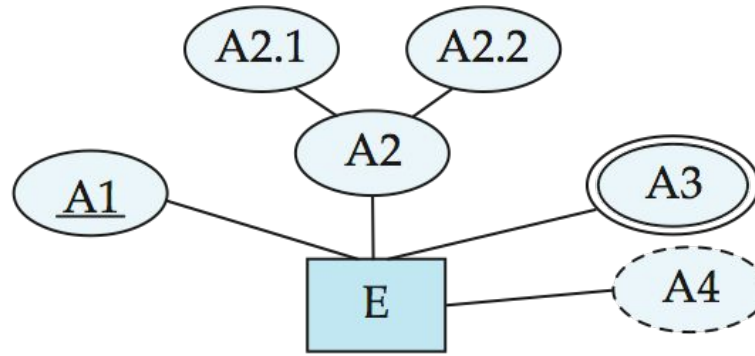
disjoint
generalization



Alternative ER Notations

- Chen, IDE1FX, ...

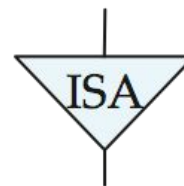
entity set E with
simple attribute A1,
composite attribute A2,
multivalued attribute A3,
derived attribute A4,
and primary key A1



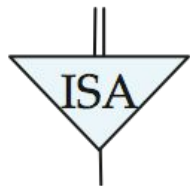
weak entity set



generalization



total
generalization



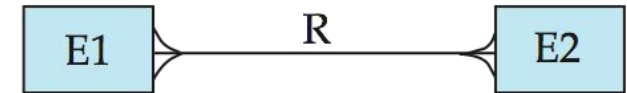
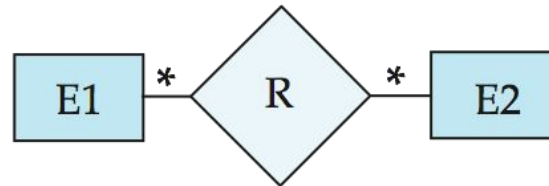


Alternative ER Notations

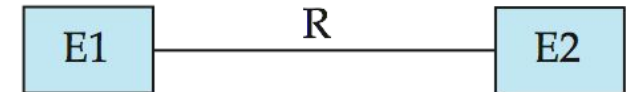
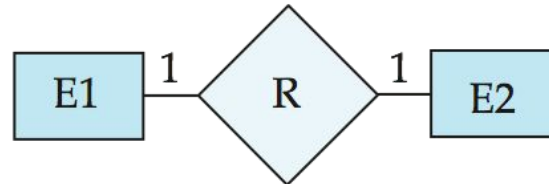
Chen

IDE1FX (Crows feet notation)

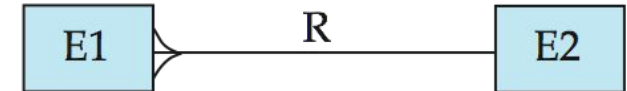
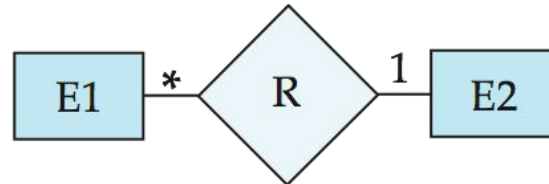
many-to-many
relationship



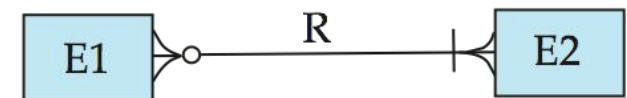
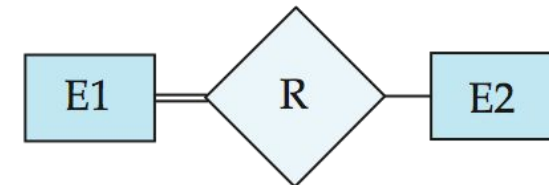
one-to-one
relationship



many-to-one
relationship



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





UML (Unified Modeling Language)

- UML has many components to graphically model different aspects of an entire software system. Other components include models of user interactions with the system, specification of functional modules of the system and their interaction, etc.
- **Class diagram-** A class diagram is similar to an E-R diagram.
- **Use case diagram-** Use case diagrams show the interaction between users and the system, in particular the steps of tasks that users perform (such as withdrawing money or registering for a course).
- **Activity diagram-** Activity diagrams depict the flow of tasks between various components of a system.
- **Implementation diagram-** It show the system components and their interconnections, both at the software component level and the hardware component level.



End of Chapter 7

Database System Concepts, 7th Ed.

©Silberschatz, Korth and Sudarshan
See www.db-book.com for conditions on re-use