Chapter 7: Entity-Relationship Model

The original presentation is infused with more information and slides
by Verena Kantere

Database System Concepts, 7th Ed.

Chapter 7: Entity-Relationship Model

- Design Process
- Modeling
- Constraints
- E-R Diagram
- Design Issues
- Weak Entity Sets
- Extended E-R Features
- Database Design

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

Outline of the ER Model

ER model -- Database Modeling

- The ER data mode was developed to facilitate database design by allowing specification of an enterprise schema that represents the overall logical structure of a database.
- The ER model is very useful in mapping the meanings and interactions of real-world enterprises onto a conceptual schema. Because of this usefulness, many database-design tools draw on concepts from the ER model.
- The ER data model employs three basic concepts:
  - entity sets,
  - relationship sets,
  - attributes.
- The ER model also has an associated diagrammatic representation, the ER diagram, which can express the overall logical structure of a database graphically.

Entity Sets

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

    instructor = (*ID, name, street, city, salary*)
    course= (*course_id, title, credits*)
- A subset of the attributes form a **primary key** of the entity set; i.e., uniquely identifiying each member of the set.

Entity Sets -- *instructor* and *student*

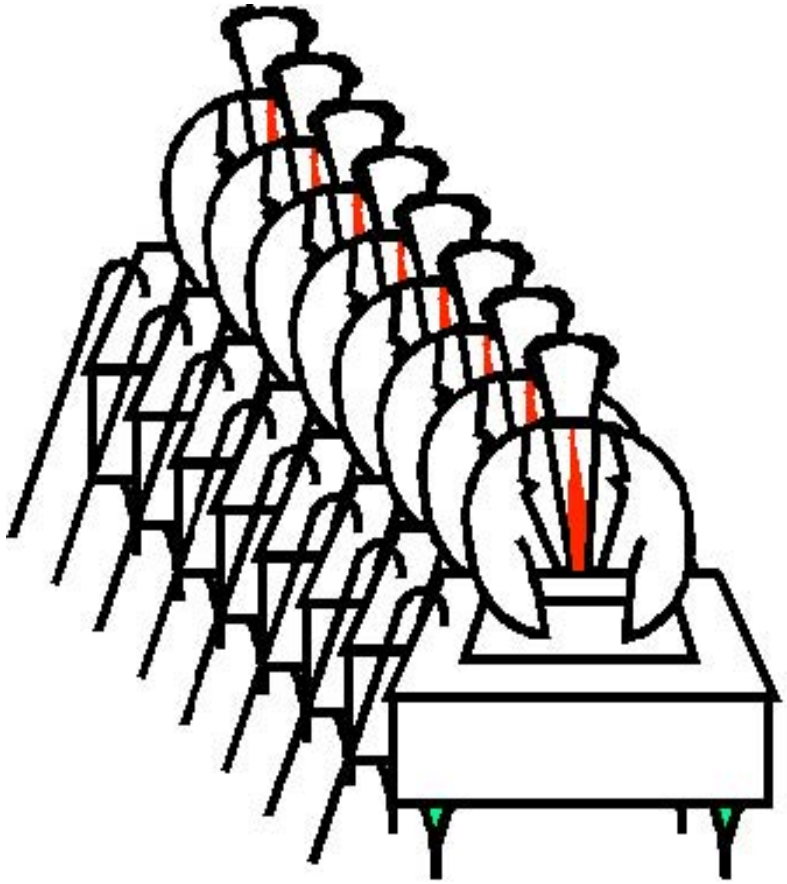
instructor_ID	instructor_name
76766	Crick
45565	Katz
10101	Srinivasan
98345	Kim
76543	Singh
22222	Einstein

instructor

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

student

Difference between entity and entity set



Relationship Sets

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

  Example:

  44553 (Peltier) — *advisor* — 22222 (Einstein)

  student entity — relationship set — *instructor* entity

- A **relationship set** is a mathematical relation among $n \geq 2$ entities, each taken from entity sets

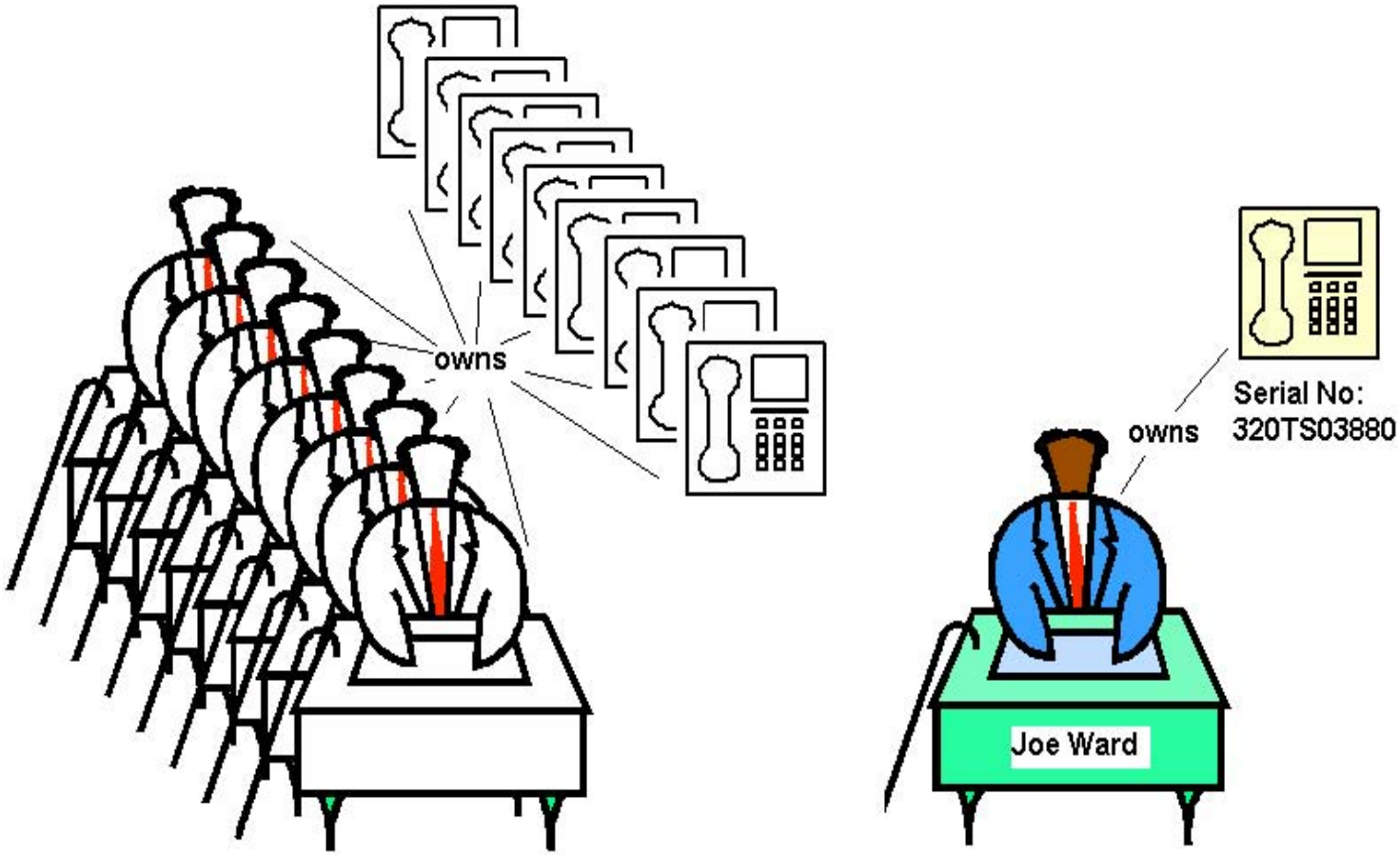
$$\{(e_1, e_2, \ldots e_n) \mid e_1 \in E_1, e_2 \in E_2, \ldots, e_n \in E_n\}$$

  where $(e_1, e_2, \ldots, e_n)$ is a relationship

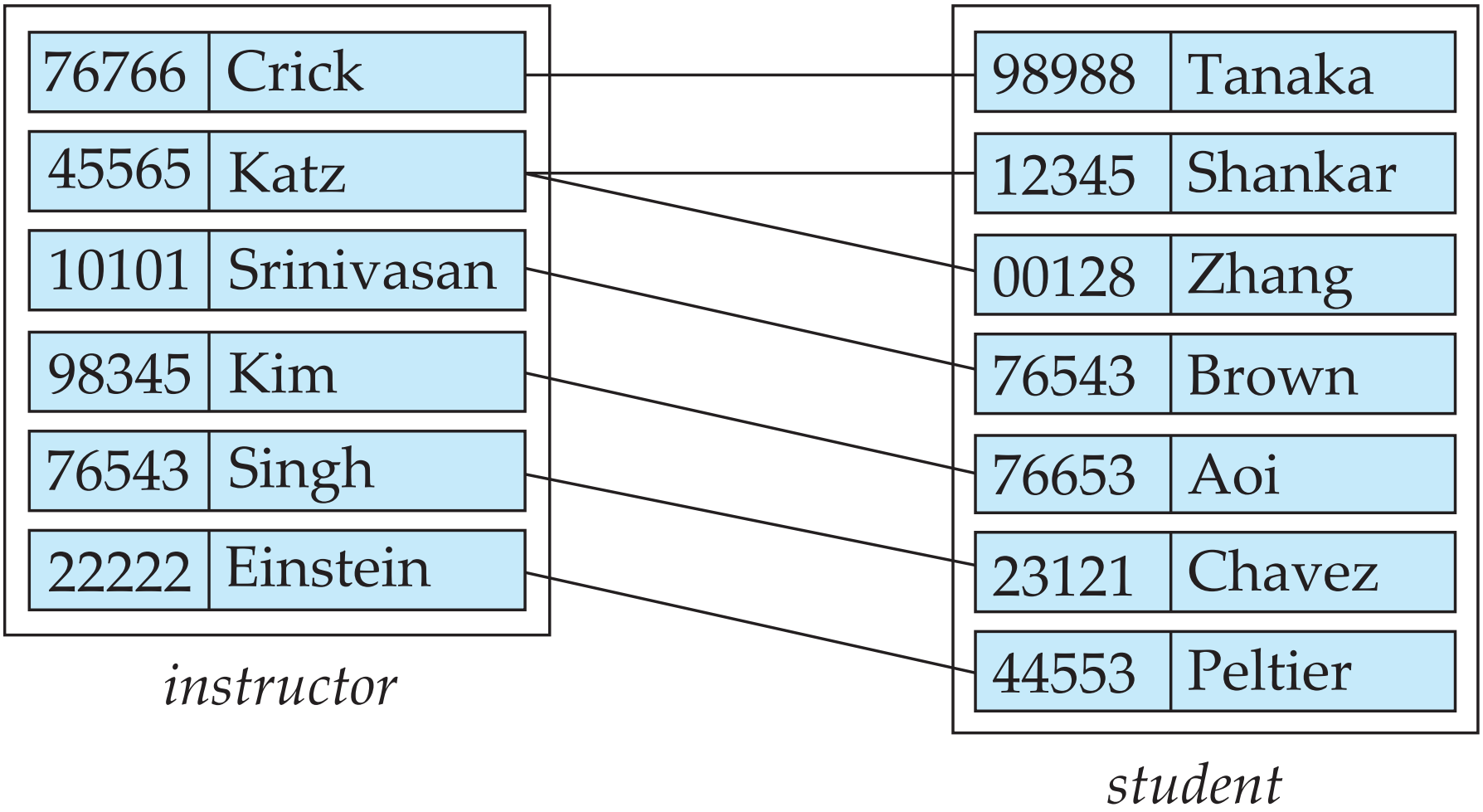
  - Example:

$$(44553,22222) \in advisor$$

Difference between relationship and relationship set

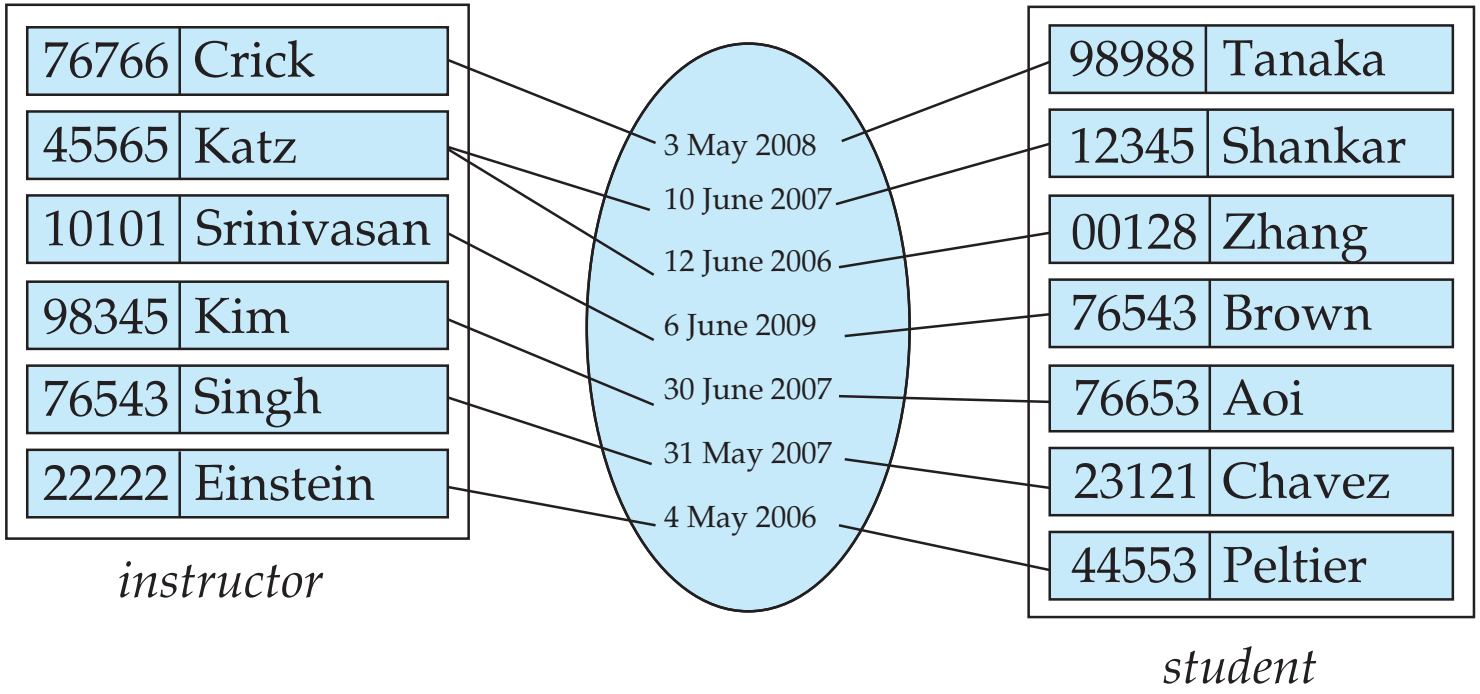


Relationship Set *advisor*



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



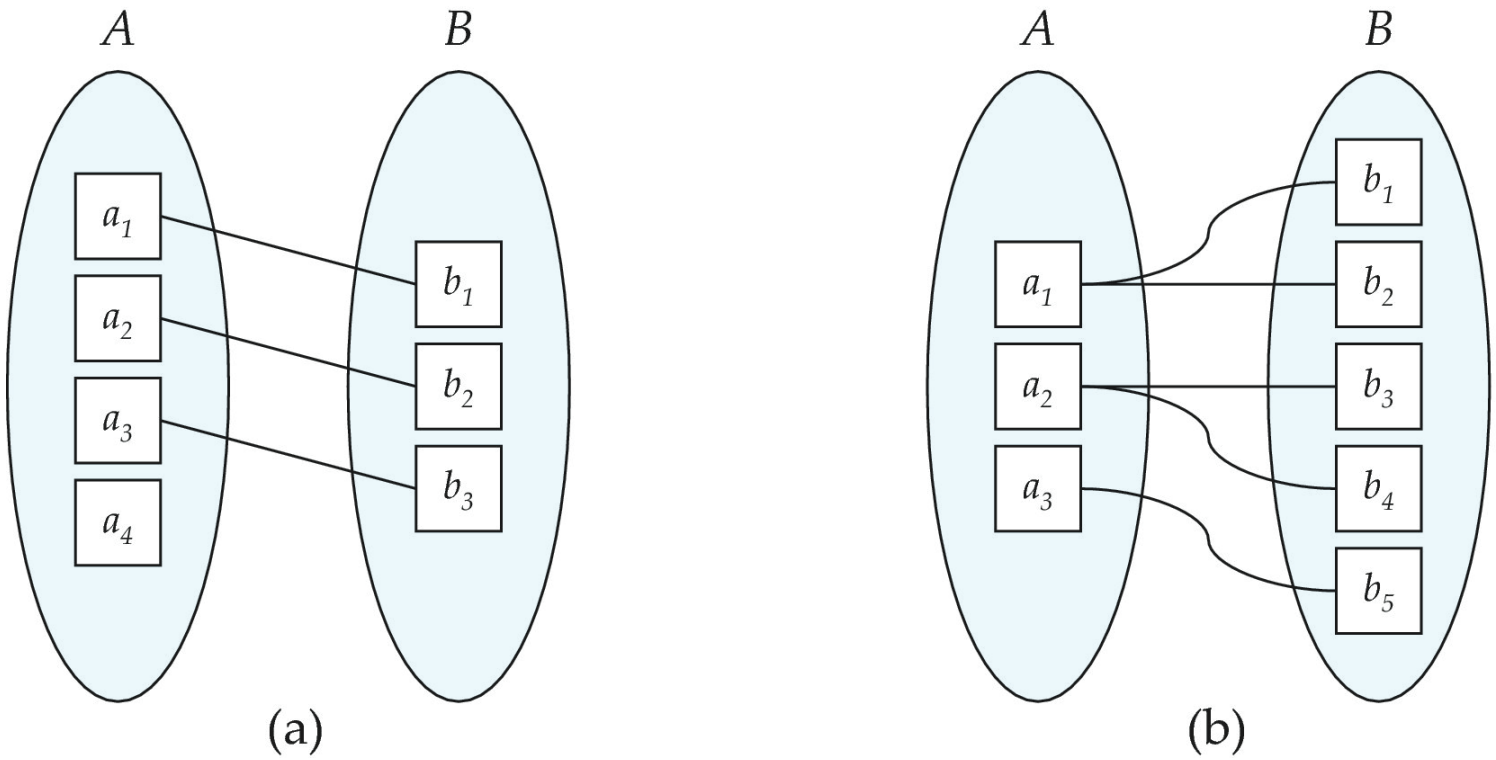
Degree of a Relationship Set

- 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.)
  - Example: *students* work on research *projects* under the guidance of an *instructor*.
  - 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

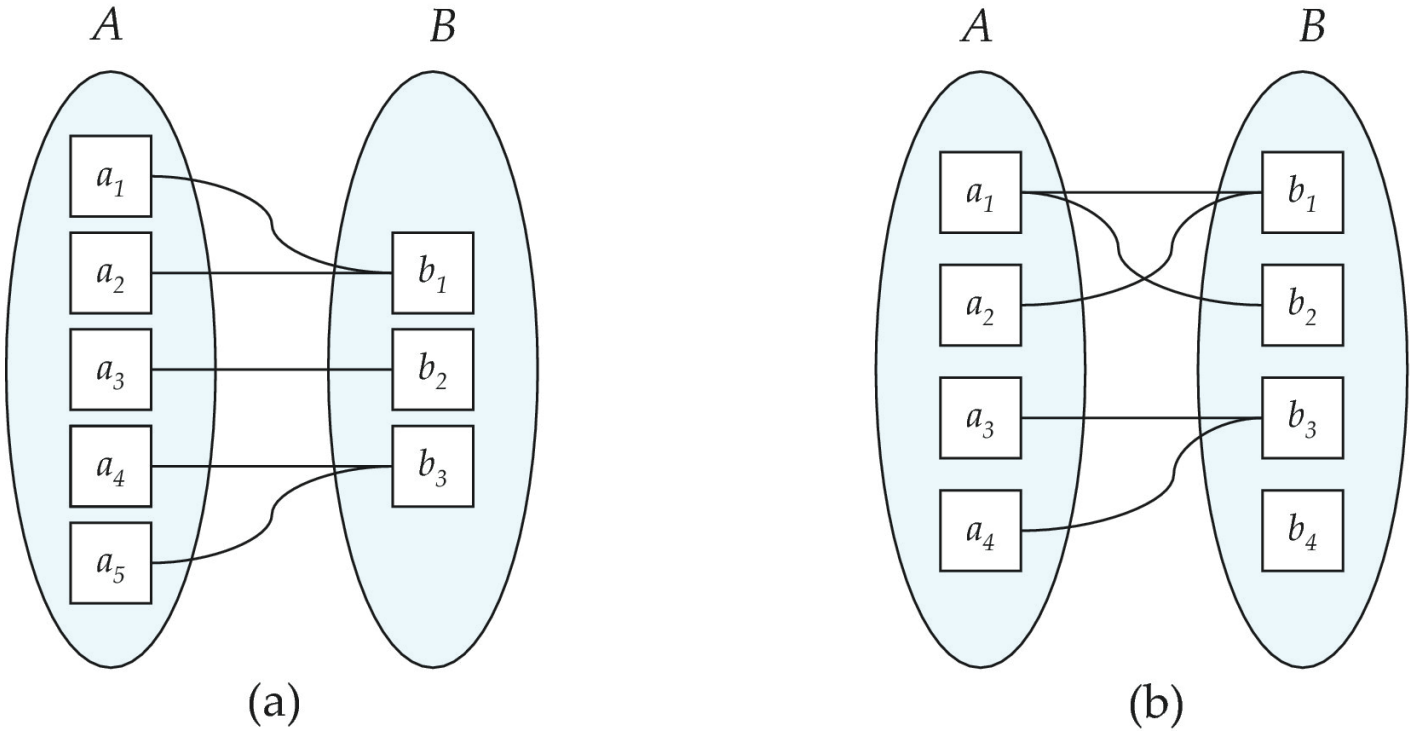


One to one

One to many

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

Mapping Cardinalities



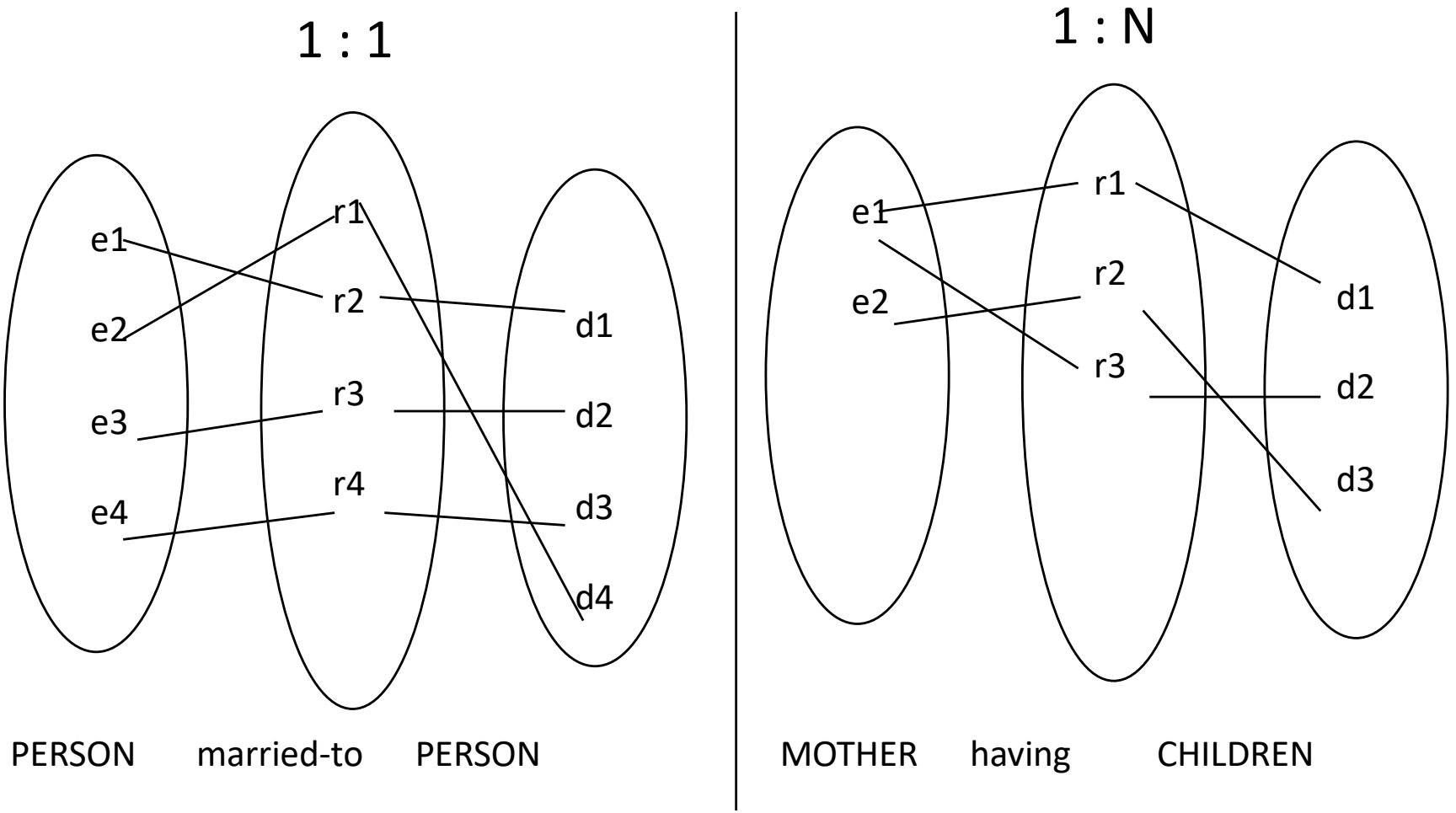
Many to one

Many to many

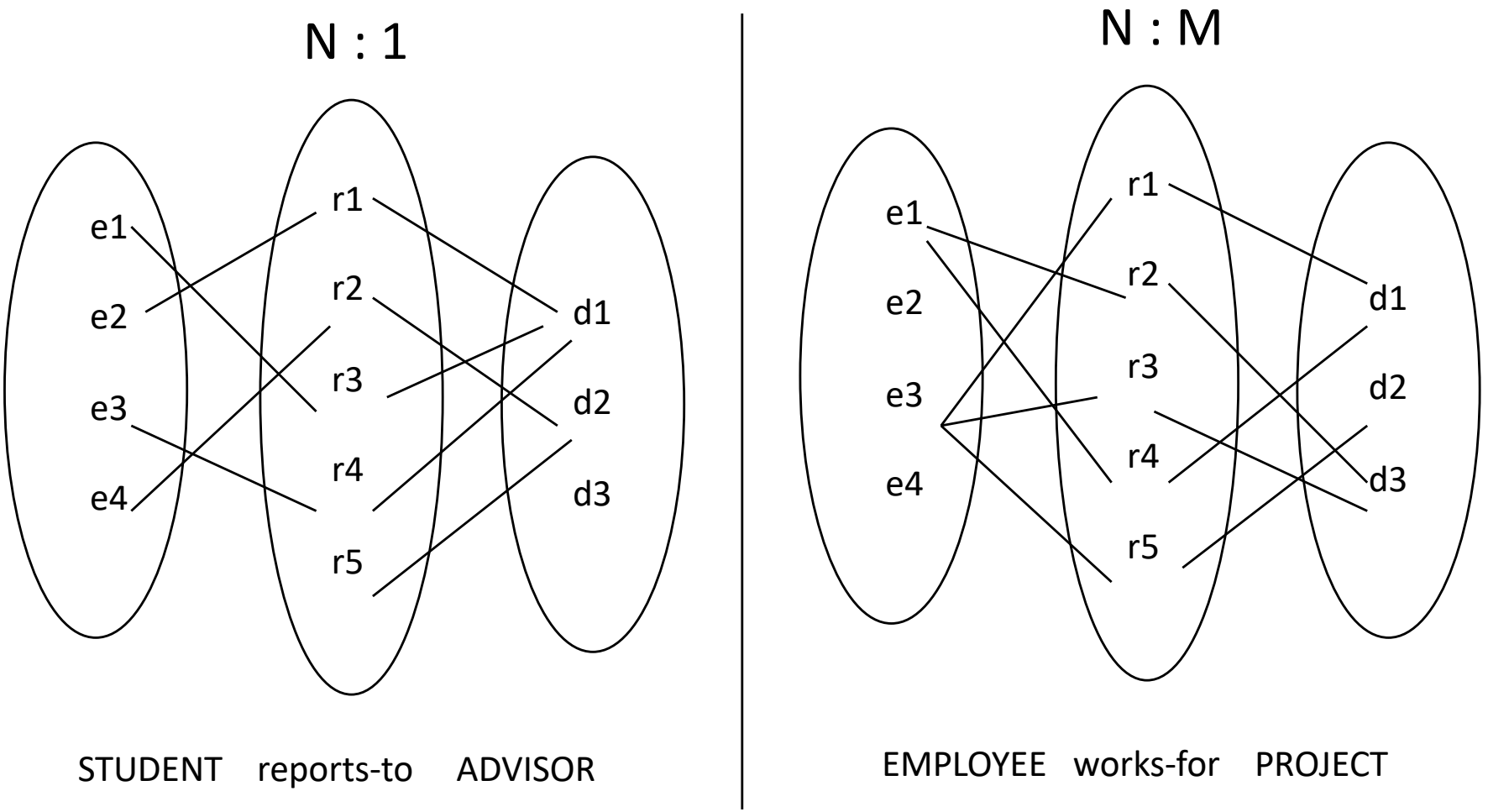
Note: Some elements in A and B may not be mapped to any elements in the other set

Mapping Cardinalities

.



Mapping Cardinalities



Structural constraints: Relationships

- **Multiple relationships**: More than one relationship set can exist between two entity sets

  e.g. WORKS-FOR and MANAGES between EMPLOYEE and DEPARTMENT.

- **Recursive relationship set** A relationship that connects two entities of the same entity set

  e.g. the relationship set SUPERVISION connects EMPLOYEE (with the role of supervised) with another EMPLOYEE (with the role of supervisor)

Structural constraints: Relationships

- **Existence dependency** defines if the participation of an entity in a relationship set is *total* or *partial*

  e.g., all EMPLOYEES participate in the WORKS-FOR **total**

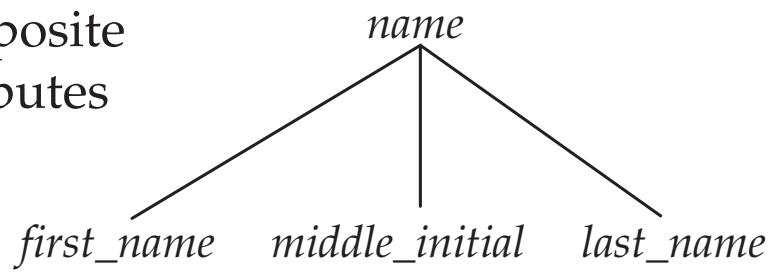
  but, in the relationship set MANAGES the EMPLOYEES that are not managers do not participate – **partial**

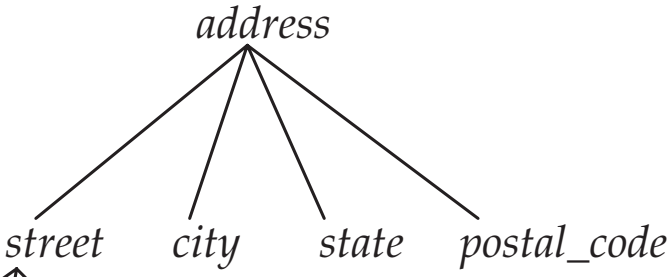
Complex Attributes

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

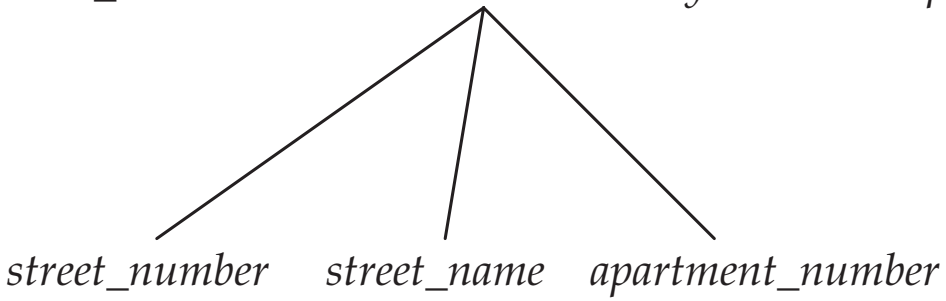
Composite Attributes

composite attributes





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.

Structural constraints: Attributes

- An attribute or a set of attributes of an entity set or a relationship set for which every entity or relationship has to have a unique value, is the key or superkey.

  E.g. the SIN of the EMPLOYEE, the NAME and ADRRESS of the EMPLOYEE, the SIN and ADDRESS of the EMPLOYEE etc.

- A candidate key is a *minimal* key (i.e. no subset of its attributes can be a key)

  E.g. the SIN is a key for EMPLOYEE, but the SIN and NAME is not.

Structural constraints: Attributes

- The **primary key** is one of the candidate keys that we decide for it to be the *identifier* for the entity set or the relationship set.

  E.g. The SIN is a good choice for the primary key of the entity set EMPLOYEE

- A **foreign key** is a set of one or more attributes of an entity set (or a relationship set) that corresponds to the primary key of another entity set or a relationship set).

  E.g., for the relationship set WORKS-FOR, the attribute SIN can be a foreign key (since it is a primary key for EMPLOYEE).

Weak Entity Sets

- Consider a *section* entity, which is uniquely identified by a *course_id*, *semester, year*, and *sec_id*.
- Clearly, section entities are related to course entities. Suppose we create a relationship set *sec_course* between entity sets *section* and *course*.
- Note that the information in *sec_course* is redundant, since *section* already has an attribute *course_id*, which identifies the course with which the section is related.
- One option to deal with this redundancy is to get rid of the relationship s*ec_course*; however, by doing so the relationship between *section* and *course* becomes implicit in an attribute, which is not desirable.

Weak Entity Sets (Cont.)

- An alternative way to deal with this redundancy is to not store the attribute *course_id* in the *section* entity and to only store the remaining attributes *section_id*, *year*, and *semester.* However, the entity set *section* then does not have enough attributes to identify a particular *section* entity uniquely; although each *section* entity is distinct, sections for different courses may share the same s*ection_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.*

Weak entities - keys

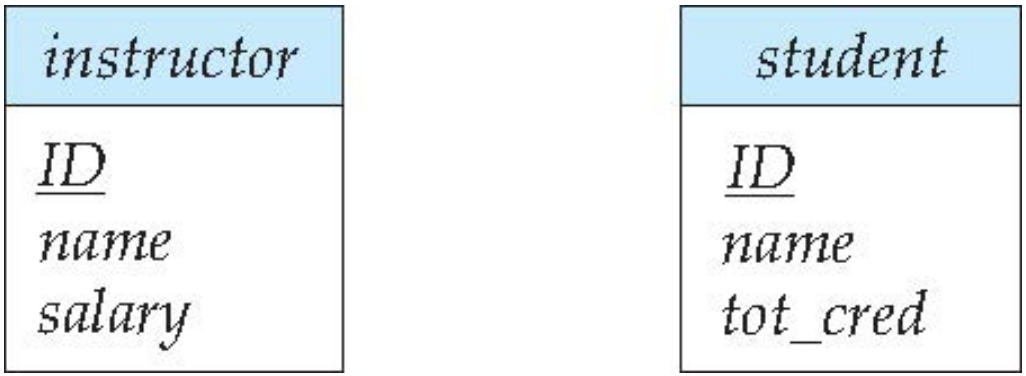
- **The weak entities:**
  - Do not have their own key
  - The depend on a strong entity
    - Total one-to-many relationship from the strong to the weak entity.
  - Partial key is the set of attributes that distinguishes a weak entity from the others that are connected with the same strong entity with the same relationship set.
  - The primary key of the weak entity is the primary key of the strong entity with the partial key of the wean entity.

  E.g. the entity set DEPENDENT, i.e. the dependent members of an EMPLOYEE..

E-R Diagrams

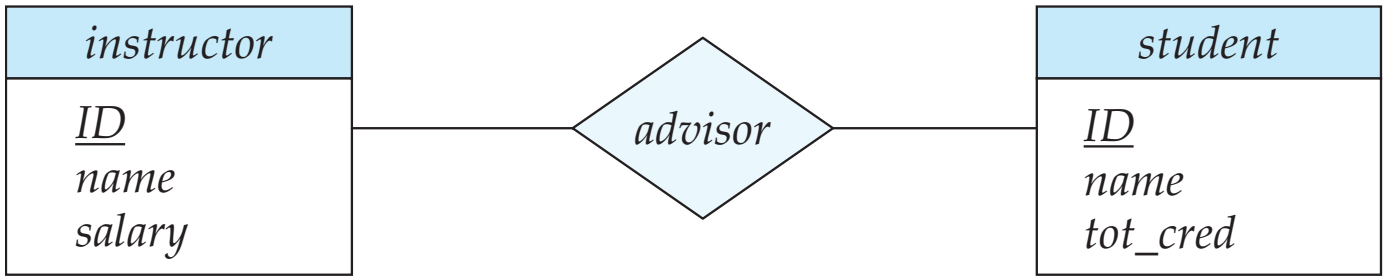
Entity Sets

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

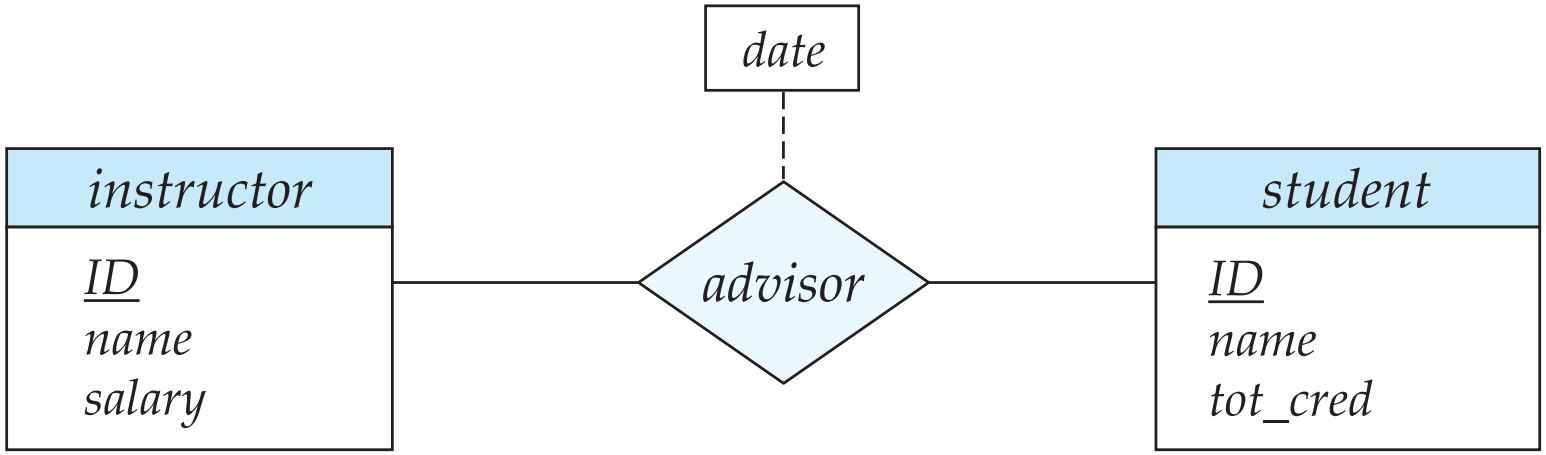


Relationship Sets

- Diamonds represent relationship sets.

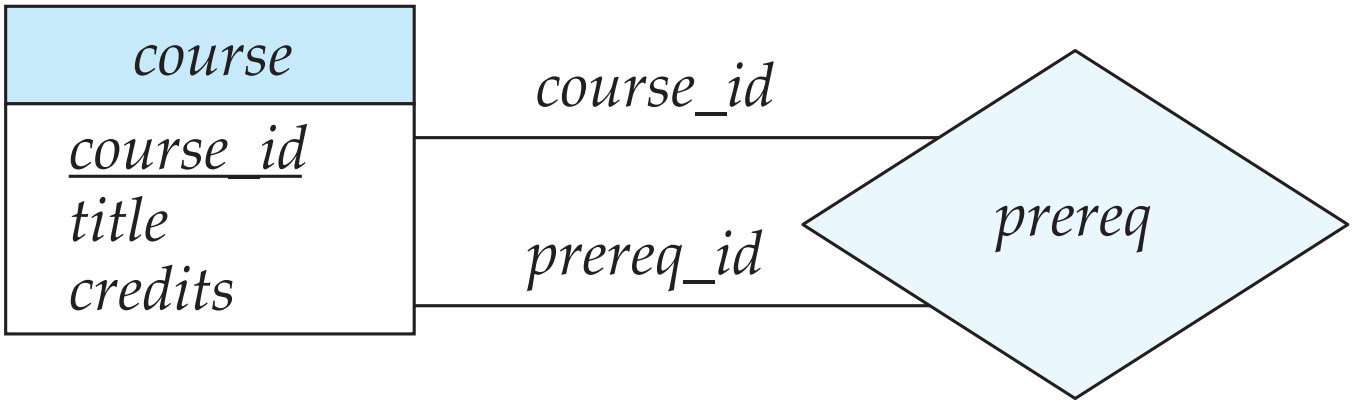


Relationship Sets with Attributes



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 (→), signifying "one," or an undirected line (—), signifying "many," between the relationship set and the entity set.

- One-to-one relationship between an *instructor* and a *student* :
  - A student is associated with at most one *instructor* via the relationship *advisor*
  - A *student* is associated with at most one *department* via *stud_dept*

One-to-Many Relationship

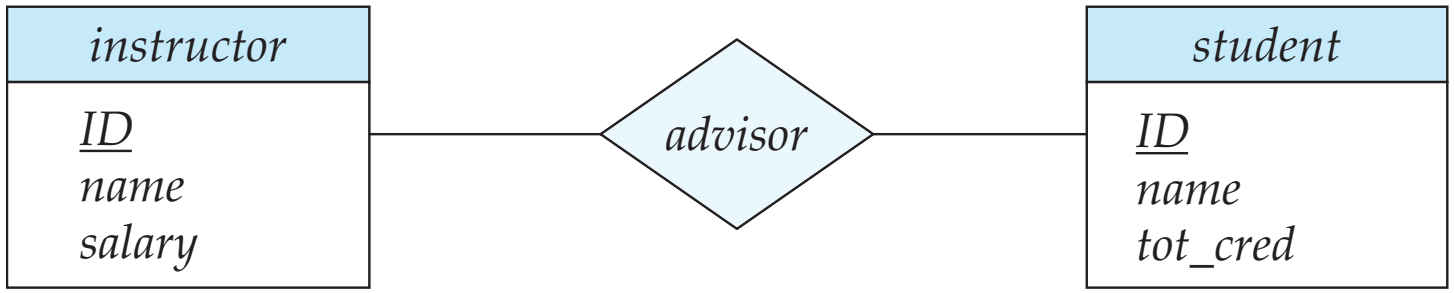
- one-to-many relationship between an *instructor* and a *student*
  - an instructor is associated with several (including 0) students via *advisor*
  - a student is associated with at most one instructor via advisor,

Many-to-One Relationships

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

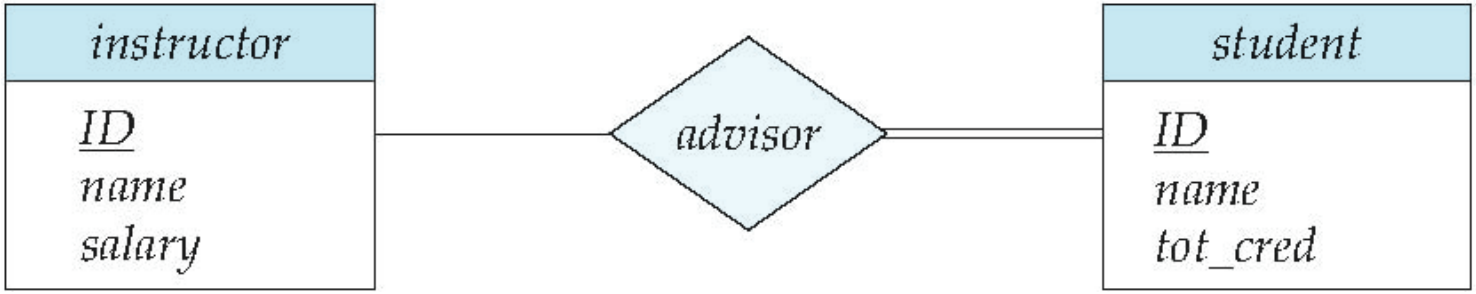
Many-to-Many Relationship

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



Total and Partial Participation

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



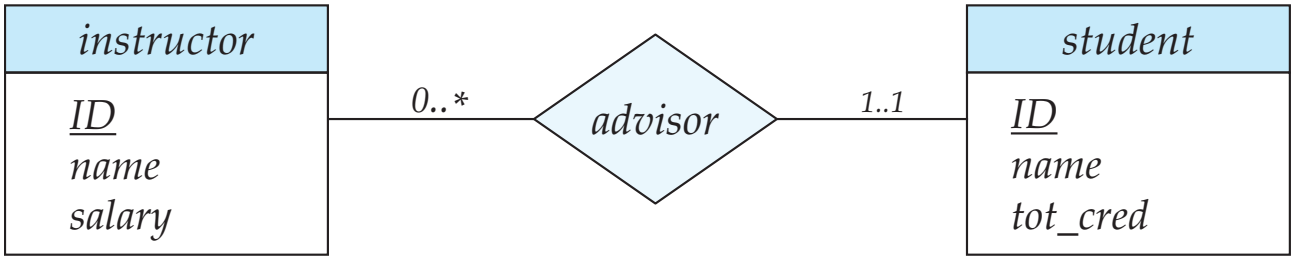
participation of *student* in *advisor* relation is total

  - every *student* must have an associated instructor

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

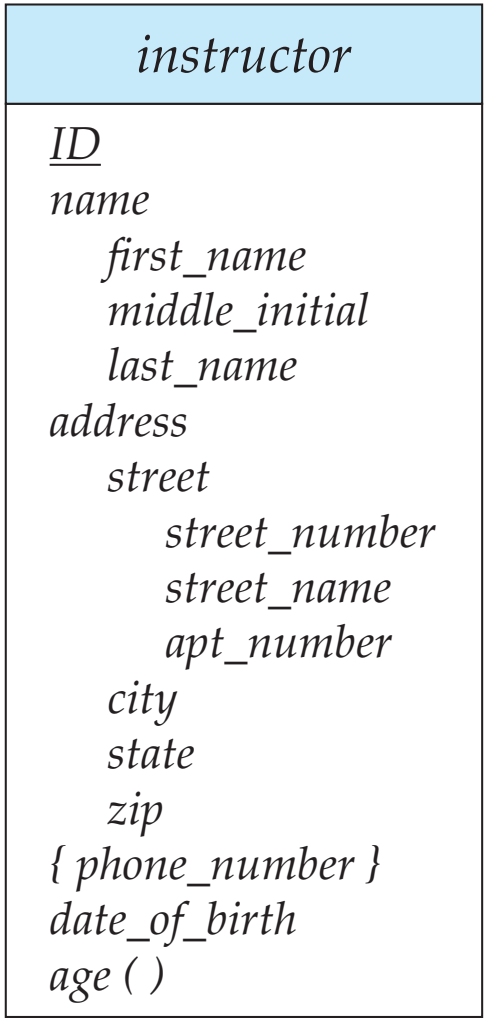
Notation for Expressing More Complex Constraints

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



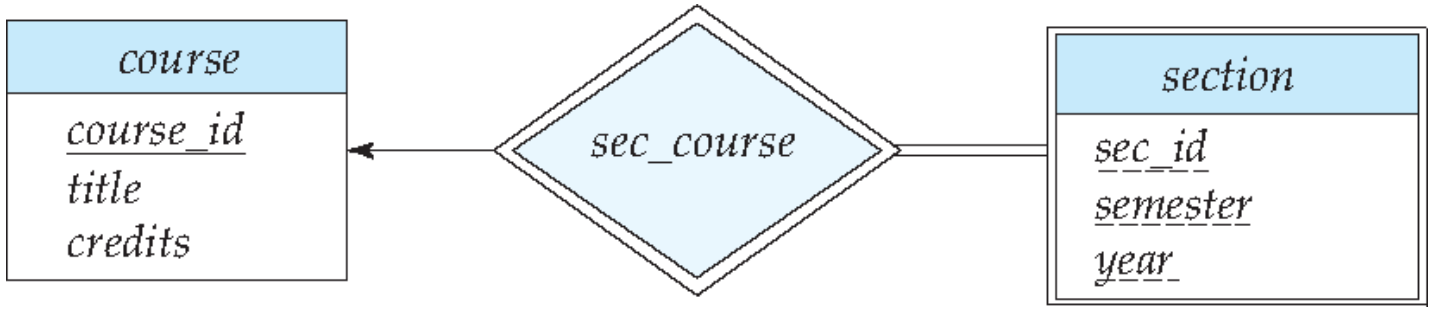
Instructor can advise 0 or more students. A student must have 1 advisor; cannot have multiple advisors

Notation to Express Entity with Complex Attributes

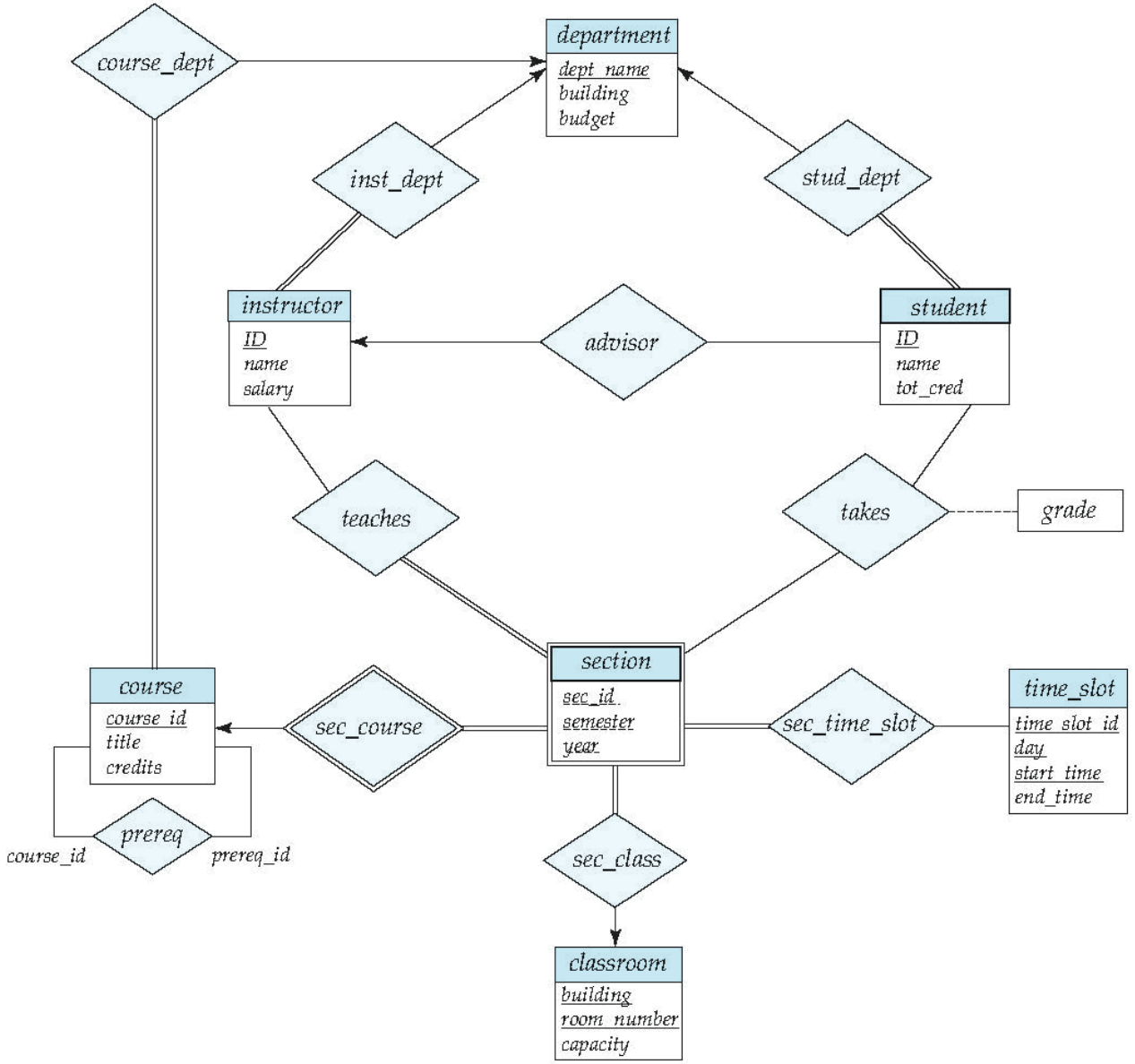


Expressing Weak Entity Sets

- In E-R diagrams, a weak entity set is depicted via a double rectangle.
- We underline the discriminator (or else partial key) 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*)

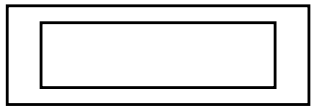


E-R Diagram for a University Enterprise

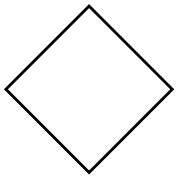


ER model: other graphical symbols

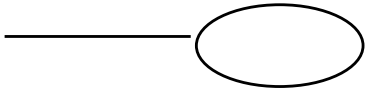
ENTITY SET



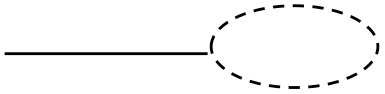
WEAK ENTITY SET



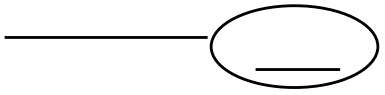
RELATIONSHIP SET



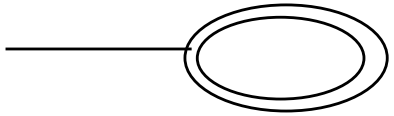
ATTRIBUTE



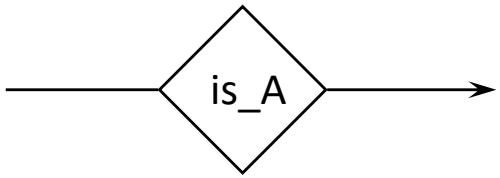
DERIVED Attribute



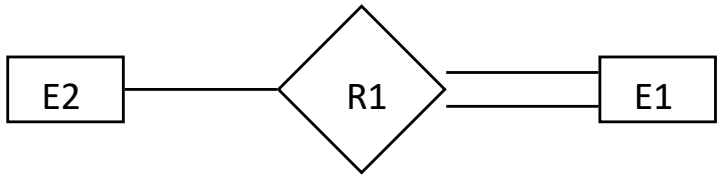
PRIMARY KEY



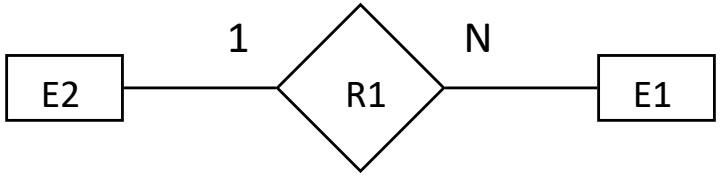
MULTIVALUED ATTRIBUTE



GENERALIZATION

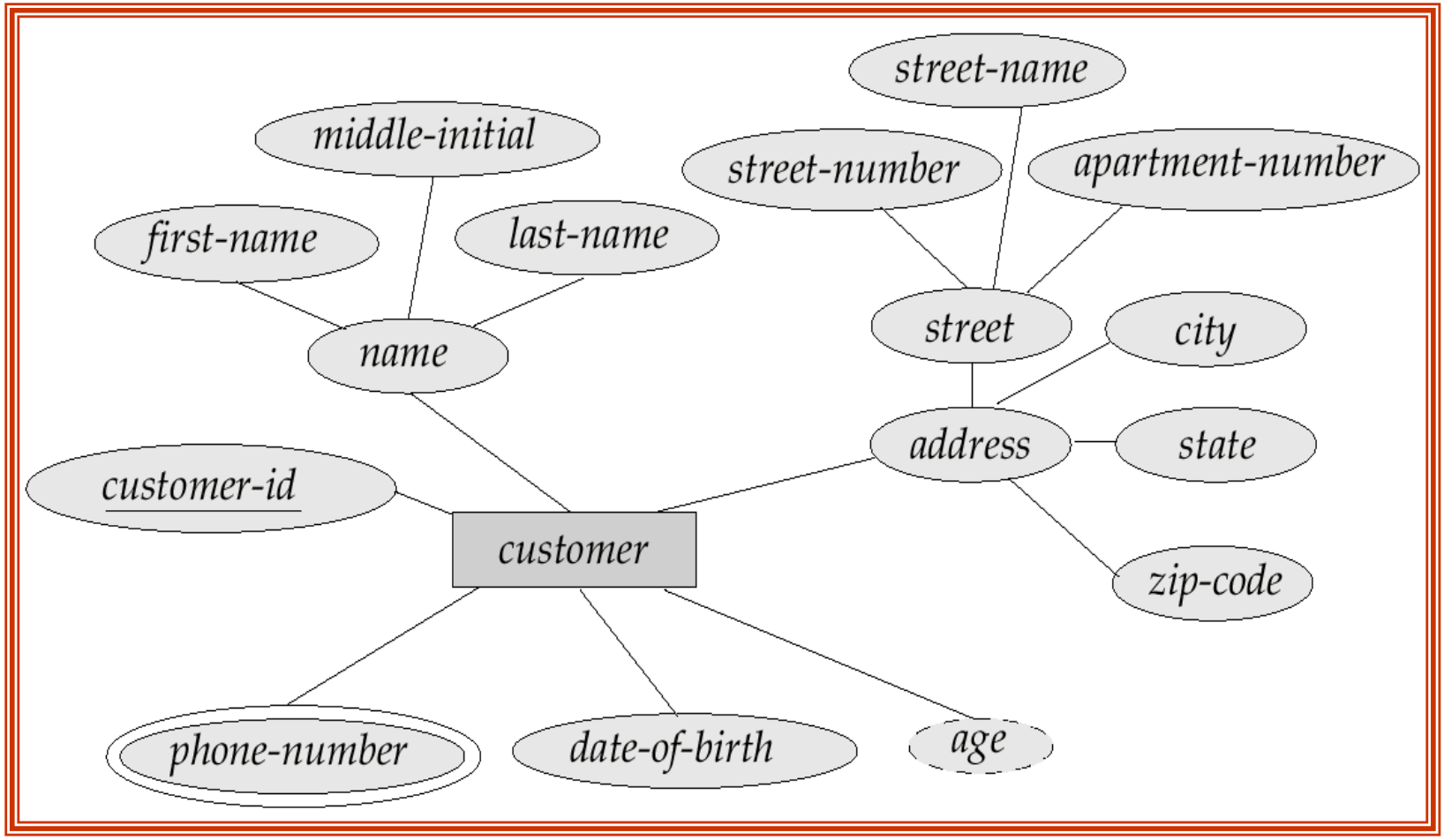


Total PARTICIPATION of E1 in R1

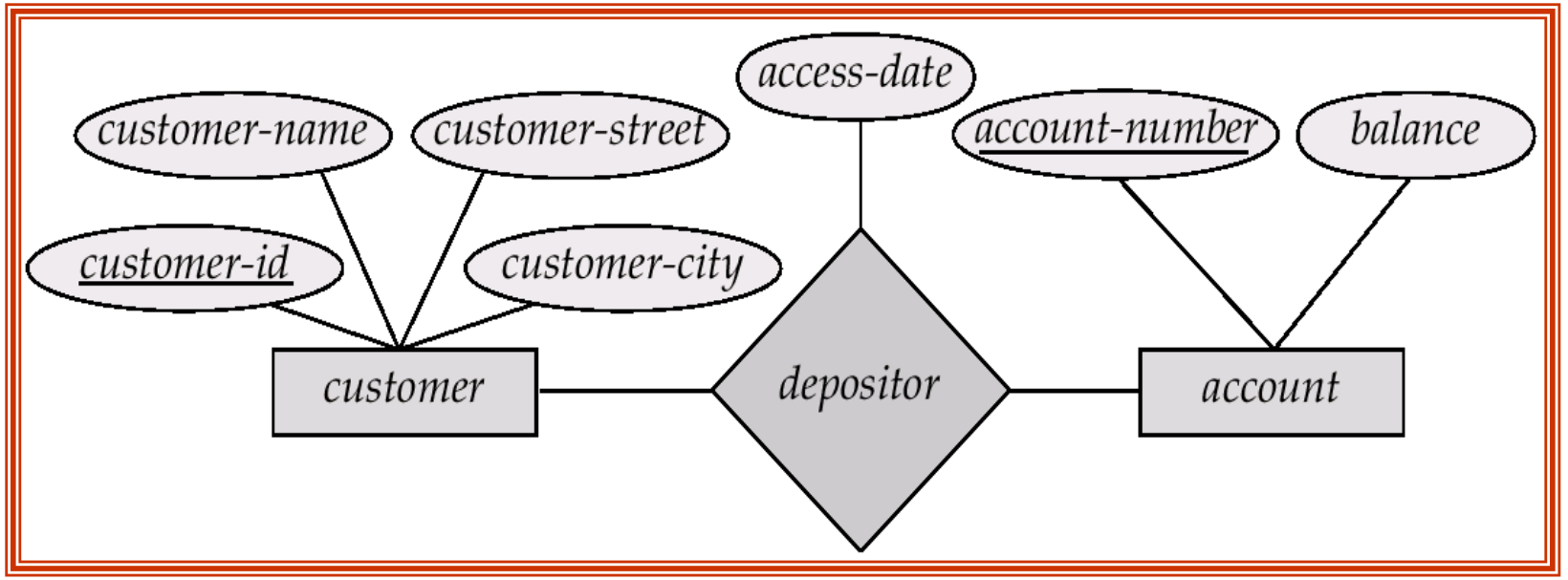


Cardinality ratio 1:N for E2:E1 in R1

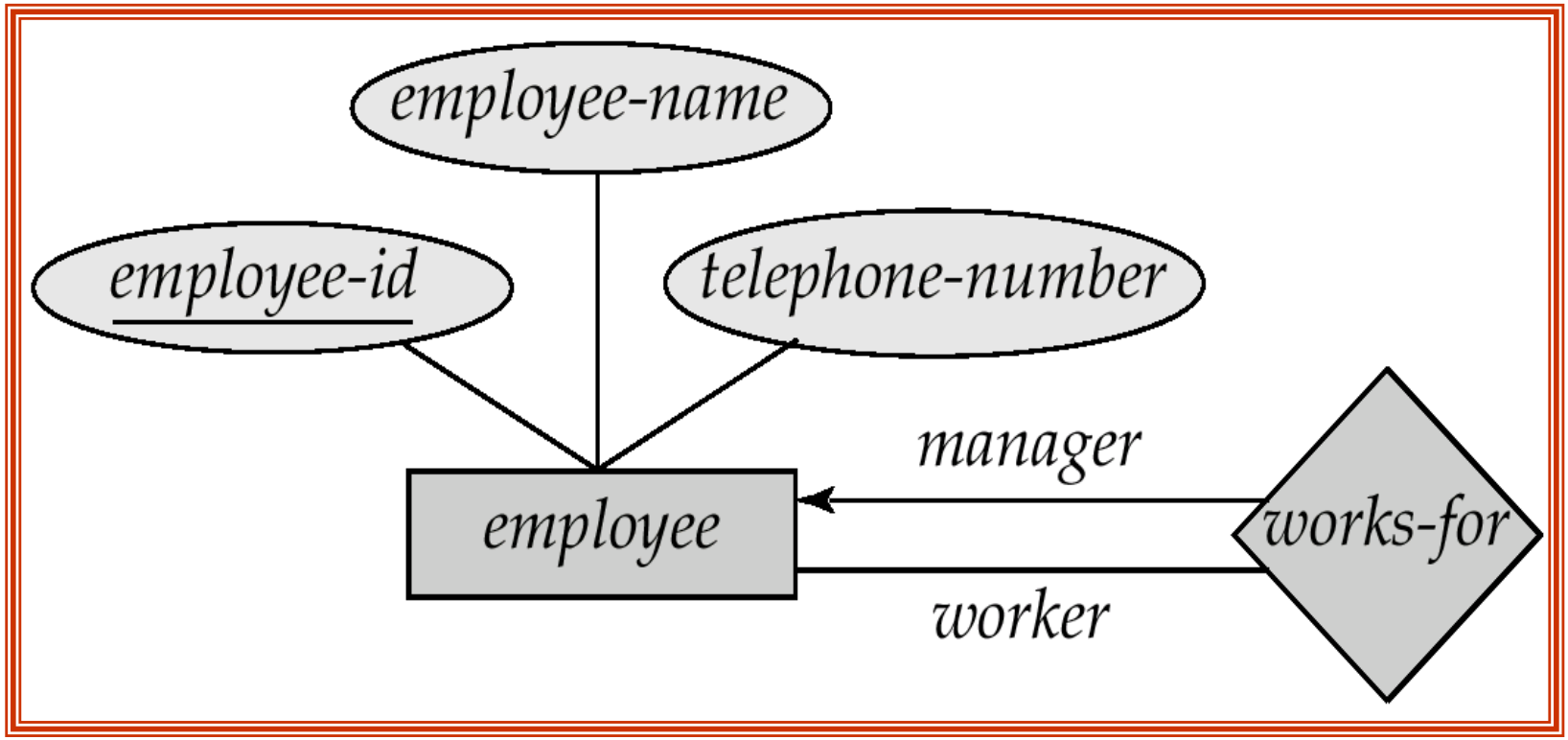
Example of an entity set



Example of a relationship set with attributes

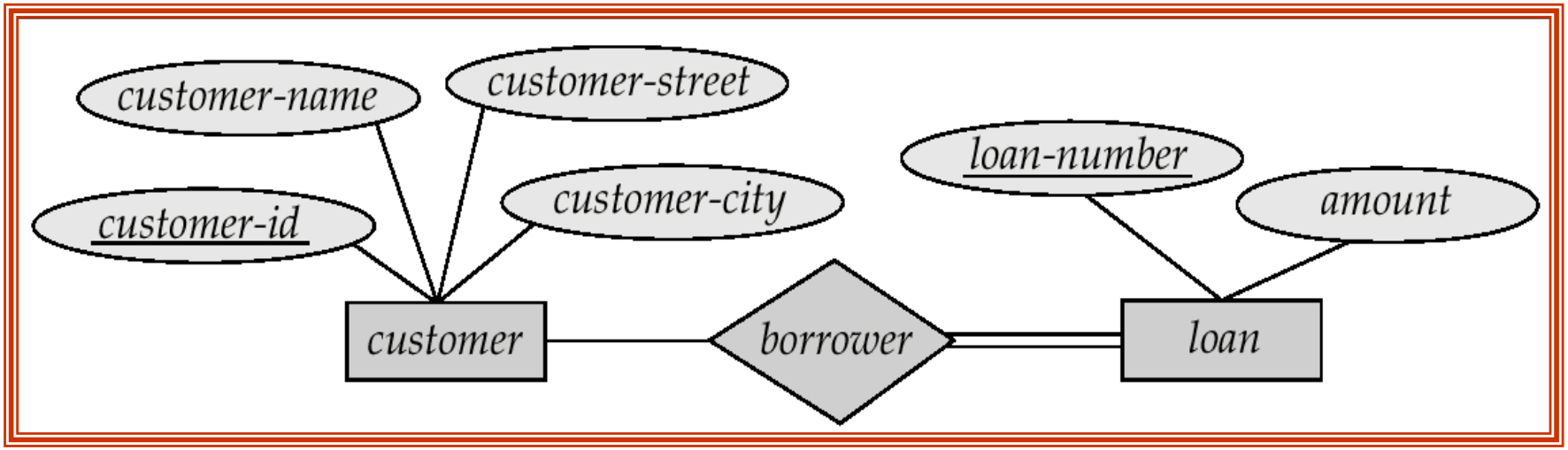


Roles


employee-name
employee-id
telephone-number
manager
employee
works-for
worker

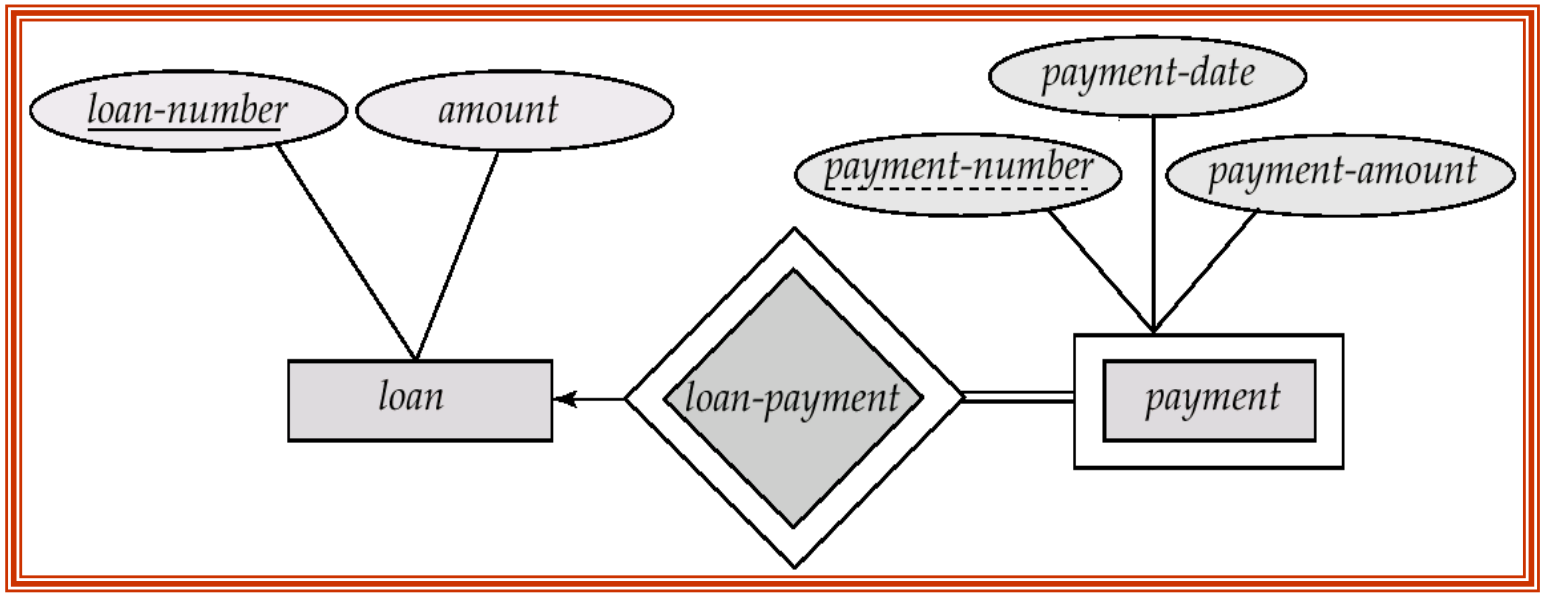
Participation

- participation of loan in borrower is total
- every loan must have a customer associated to it via borrower
- participation of customer in borrower is partial



Weak entity sets

- The partial key is underlined with a stitched line
  - payment-number – partial key of payment entity set
- Primary key for payment – (loan-number, payment-number)



Example: DB for a company

- DB requirements for a company
  - A company is organized in DEPARTMENTS. Every department has a name, number, and an employee that MANAGES the department. We are interested in the start day of the manager.
  - A department is distributed in several *locations. Every department controls some* PROJECTS, and every project has a *name, number and is located in a specific location.*

Example: DB for a company

- Concerning the EMPLOYEEs, we need to keep information about their *social security number, their address, their salary, their sex and their birth date*

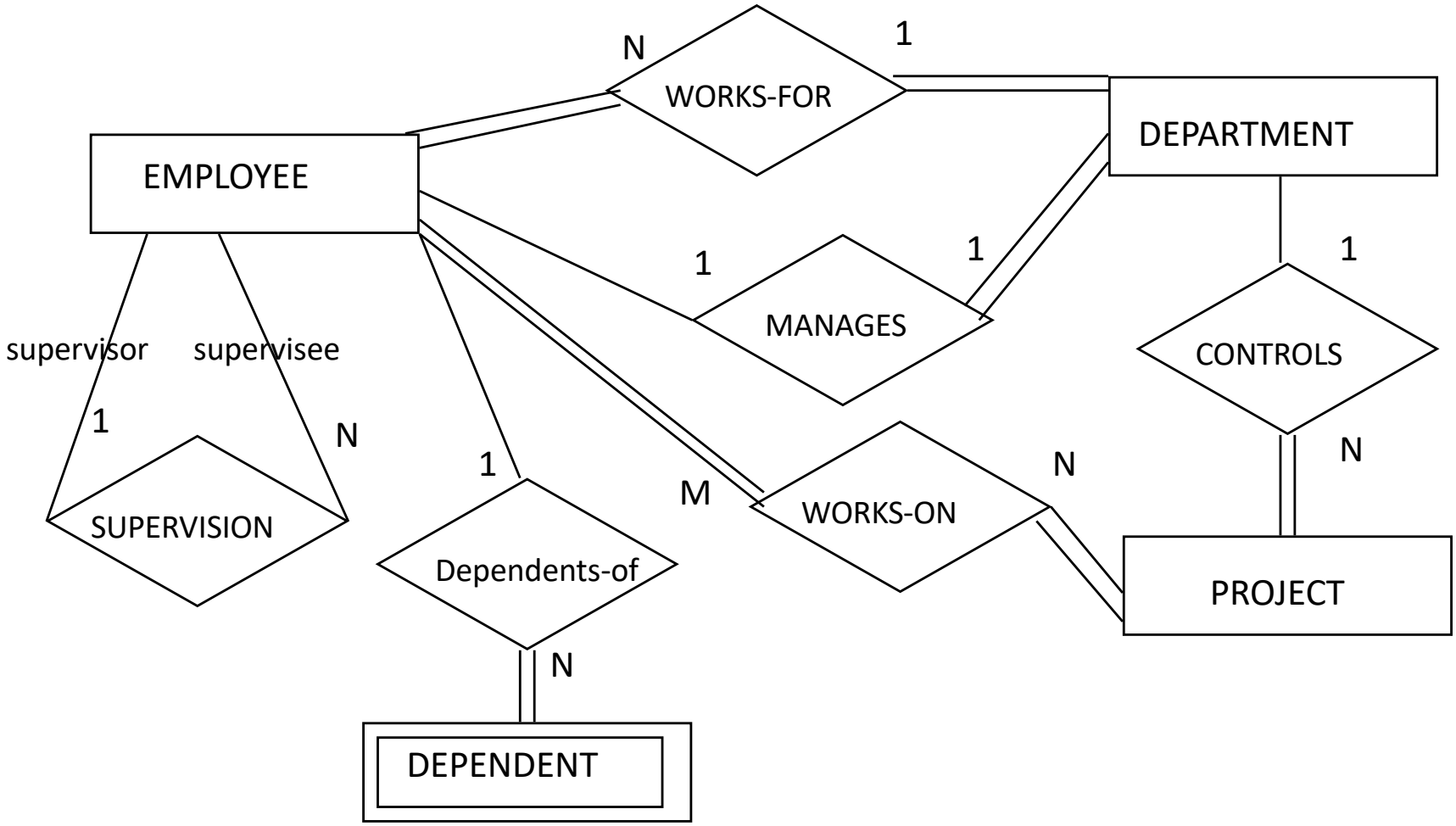
- Every employee WORKS FOR a department, but can WORK ON several projects. Also, we keep information about the *number of hours that an employee works on a projects, as well as his/her direct supervisor*

- Every employee can have DEPENDENTS. For the dependents we need to know their *name, their date of birth, their sex and and their relationship with the employee*

ER Description of the Company

- EMPLOYEE -- SSN, Name, BirthDate, Sex, Address, Salary
- DEPARTMENT -- Number, Name,Locations, NoOfEmployees
- PROJECT -- Number, Name, Location
- DEPENDENT -- Name, Sex, BirthDate, Relationship
- WORKS-ON -- HoursPerWeek
- MANAGES -- StartDate

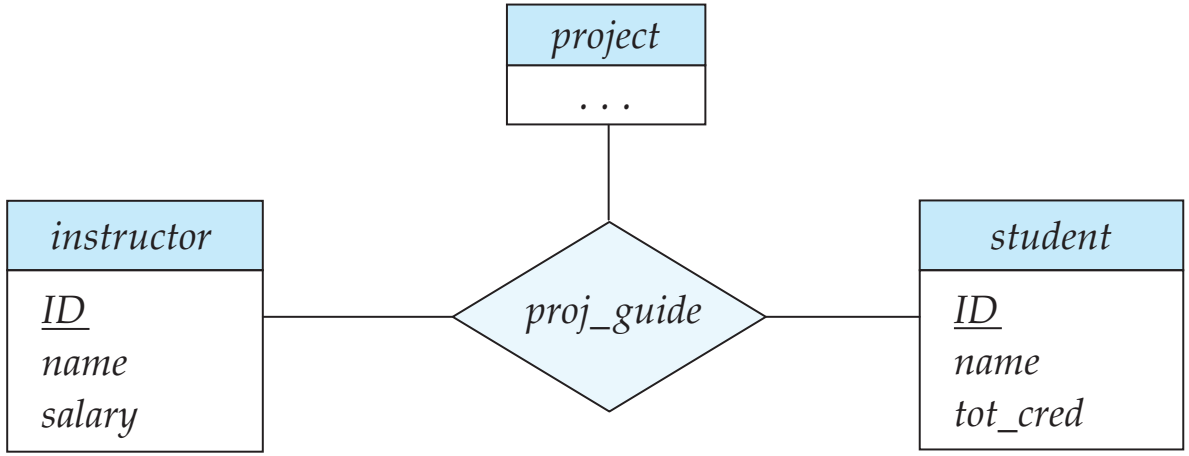
ER Description of the Company



Advanced Topics

Non-binary Relationship Sets

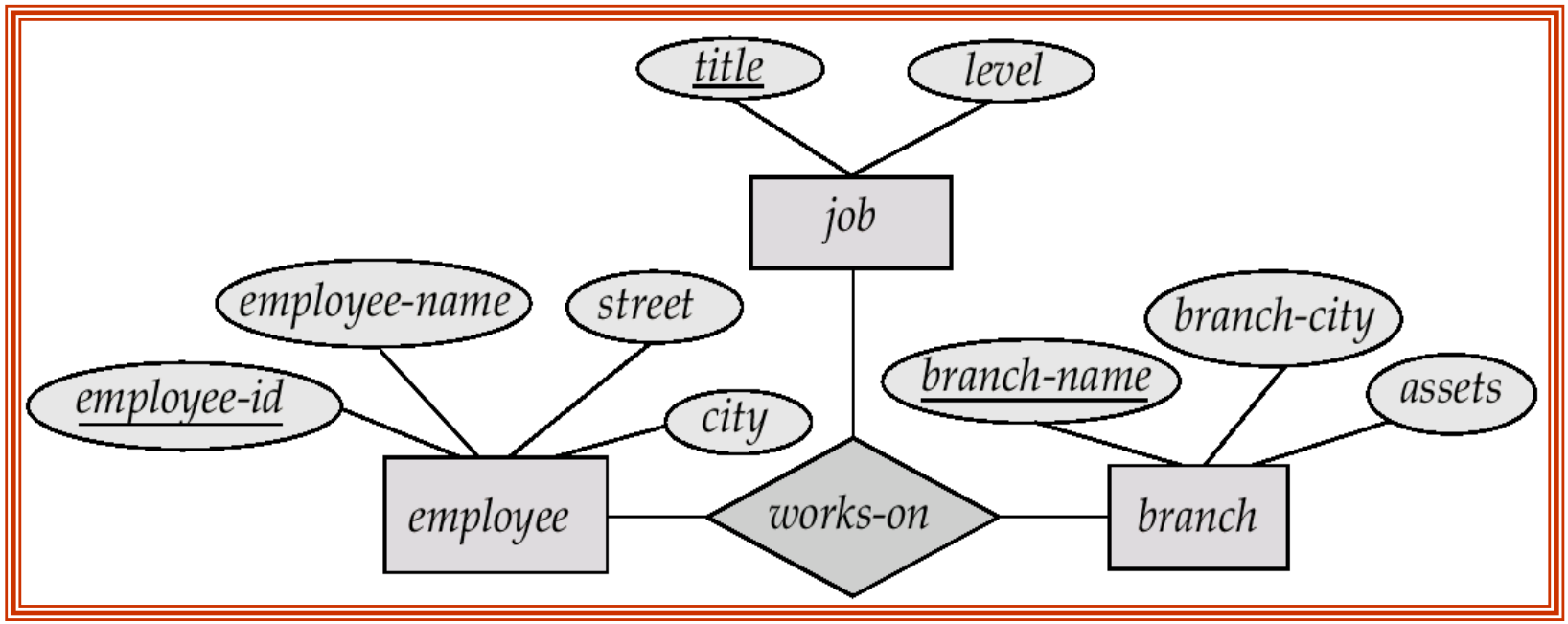
- Most relationship sets are binary
- There are occasions when it is more convenient to represent relationships as non-binary.
- E-R Diagram with a Ternary Relationship



Cardinality Constraints on Ternary Relationship

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

Ternary Relationship



End of Chapter 7