Chapter 7: Entity-Relationship Model

CSE 225 Database Management System

Sohrab Hossain
CSE, EDU

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 – The designer maps the high-level conceptual schema onto the implementation data model of the database system that will be used. The implementation data model is typically the relational data model.

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?

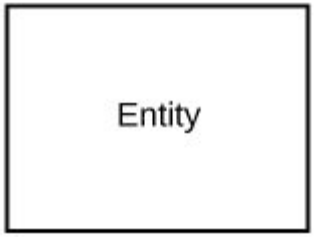
- Physical Design – Deciding on the physical layout of the database like RAID and hashing

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.

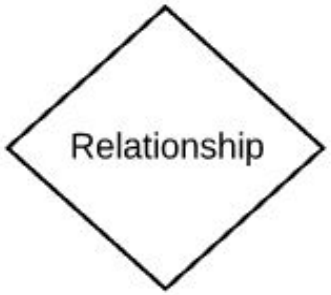
ER diagram symbols

Rectangle



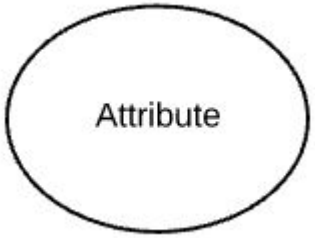
a thing with distinct and independent existence.

Diamond



Relationships are associations between or among entities

Ellipse



Attributes are characteristics of an entity, a many-to-many relationship, or a one-to-one relationship.

Entity Sets

- An **entity** is an object that exists and is distinguishable from other objects.
  - Example: student, instructor, course
- An **entity set** is a set of entities of the same type that share the same properties.
  - Example: set of all student, instructor, course
  - 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*)

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

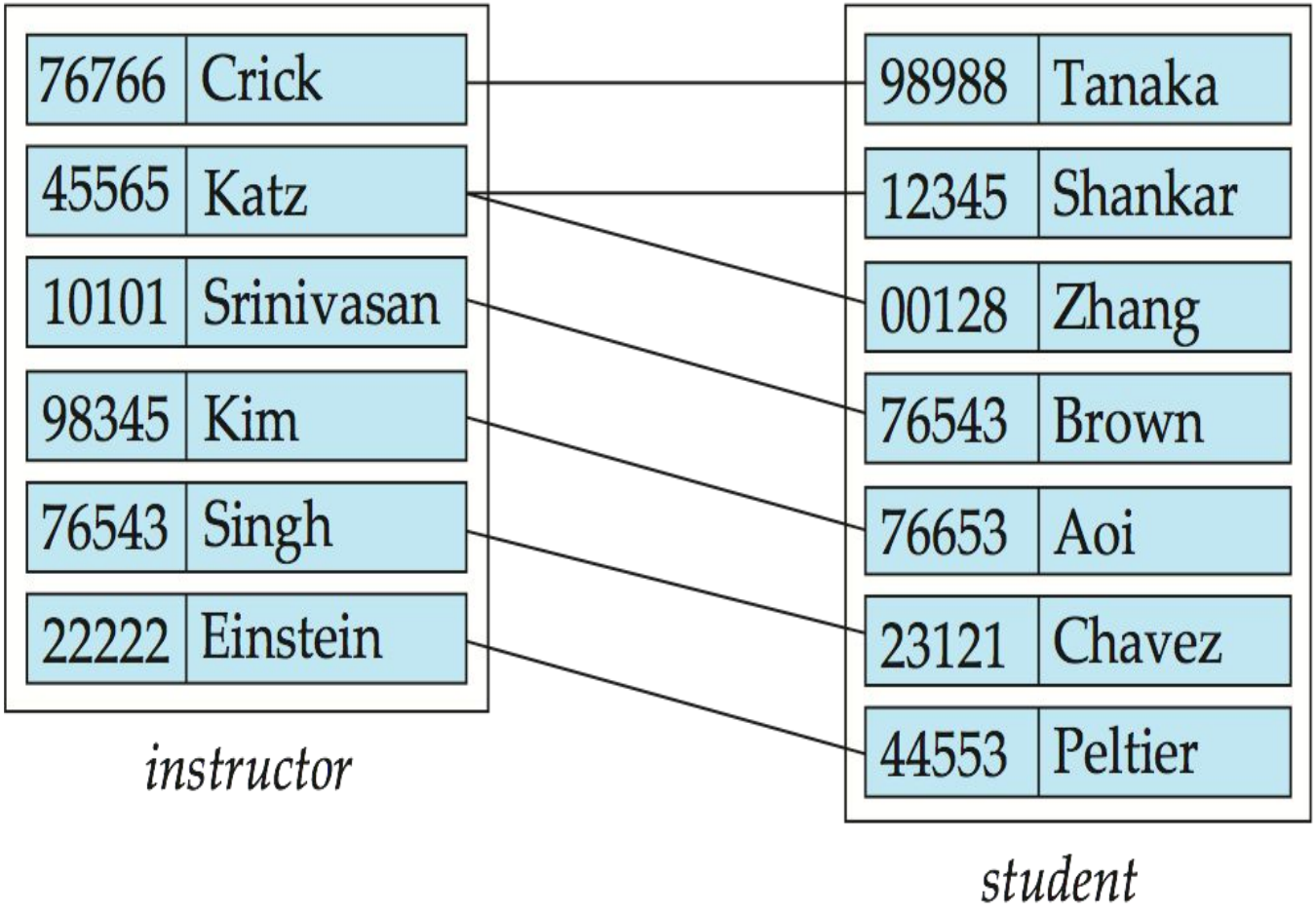
$$\{(e_1, e_2, \ldots e_n) \mid e_1 \in E_1, e_2 \in E_2, \ldots, e_n \in E_n\}$$

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

  – Example:

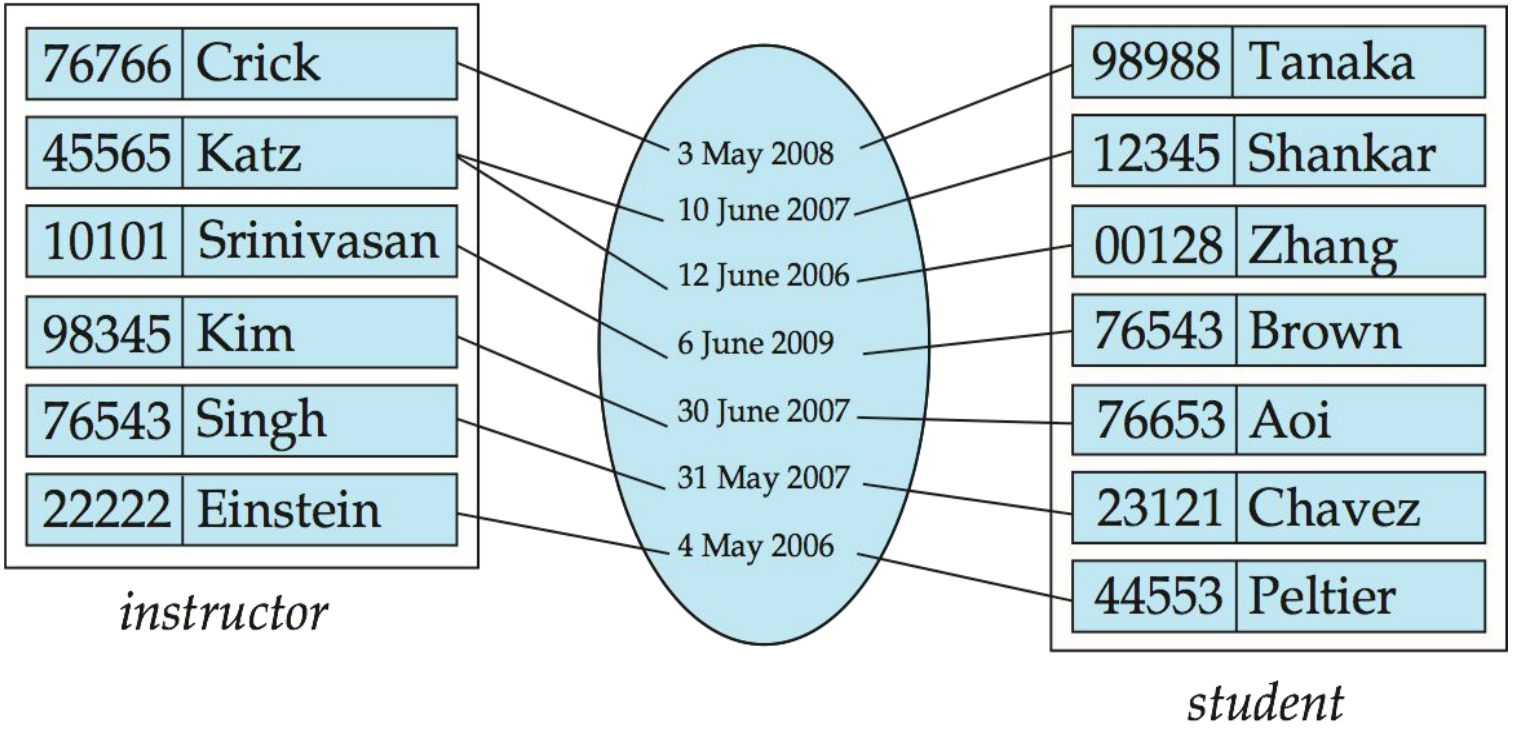
$$(44553,22222) \in advisor$$

Relationship Set *advisor*



Relationship Sets (Cont.)

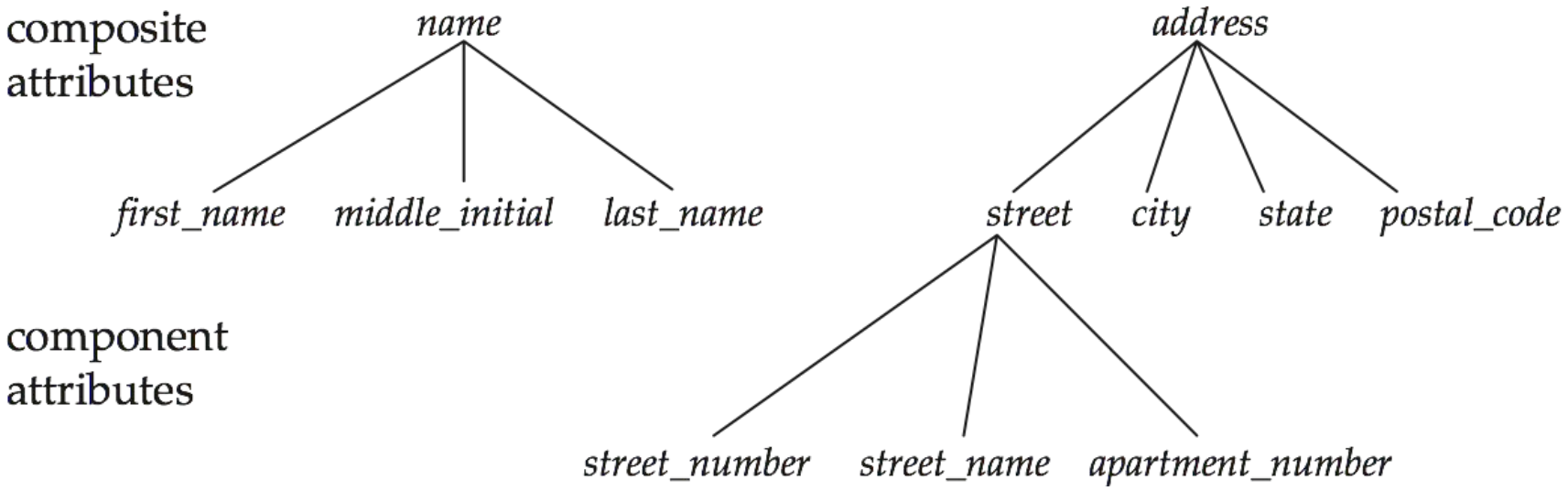
- An attribute can also be associated with a relationship set.
- For instance, the *advisor* relationship set between entity sets *instructor* and *student* may have the attribute *date* which tracks when the student started being associated with the advisor



Attributes

- Attribute types:
  - **Simple** they are not divided into subparts
  - **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
  - **Null Value**
- **Domain** – the set of permitted values for each attribute

Composite Attributes



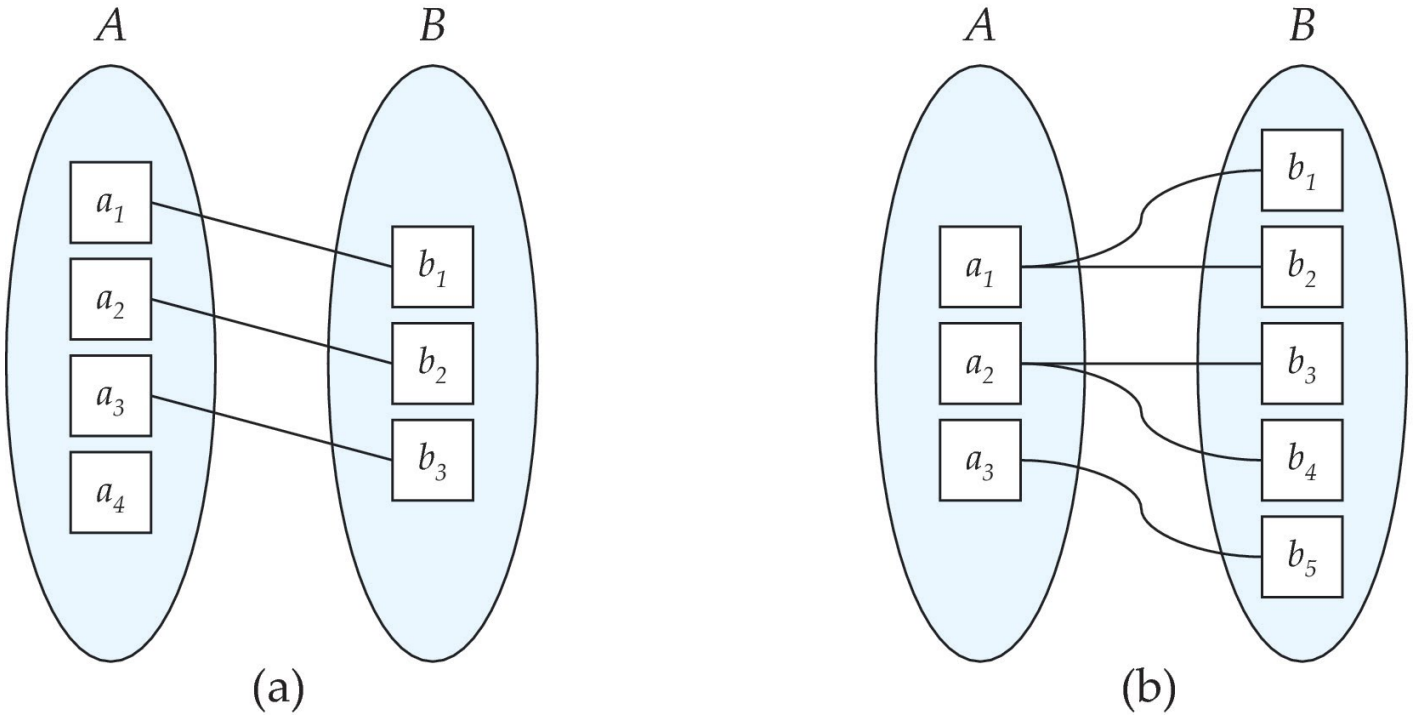
Redundant Attributes

- 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.

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

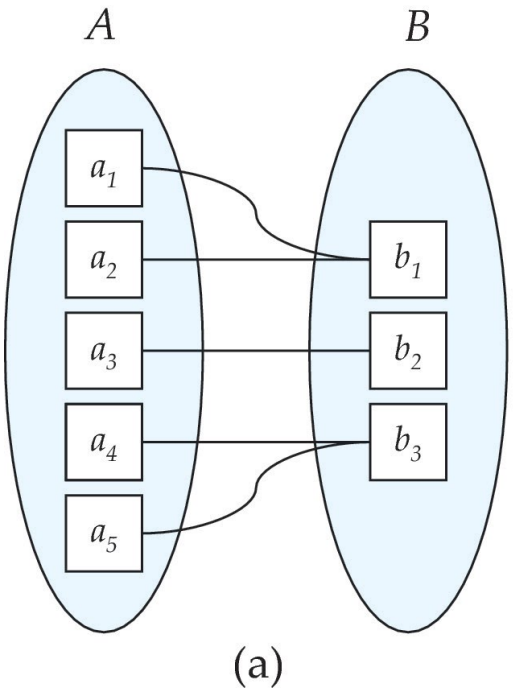


One to one

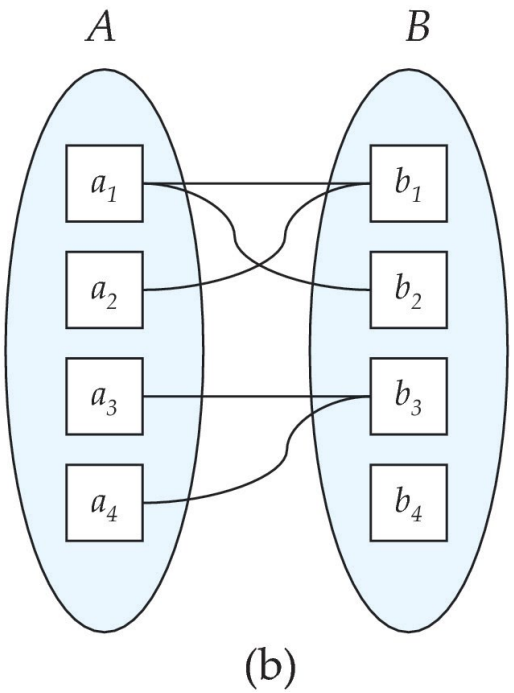
One to many

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