Database Design using ER Model –Part 1

Design Phases

- **Initial Phase**

    -characterize fully the data needs of the prospective database users.

    -output- User requirements specification.

- **Conceptual Design Phase**
  - Chooses a data model
  - Applying the concepts of the chosen data model for translating these requirements into a conceptual schema of the database.
  - A fully developed conceptual schema indicates the functional requirements of the enterprise.
    - Describe the kinds of operations (or transactions) that will be performed on the data.

Design Process

- Final Phase -- Moving from an abstract data model to the implementation of the database

  - Logical Design – Deciding on the database schema.
    - The designer maps the high level conceptual schema into relational schema.

  - Physical Design – Deciding on the physical layout of the database
    - Form of file organization, choice of index structures etc.

Design Alternatives

- In designing a database schema, we must ensure that we avoid two major pitfalls:
  - Redundancy: a bad design may result in repeat information.
    - Redundant representation of information may lead to data inconsistency among the various copies of information
  - Incompleteness: a bad design may make certain aspects of the enterprise difficult or impossible to model.

ER Model

- Widely used conceptual level data model
  - proposed by Peter P Chen in 1970s
- Data model to describe the database system at the requirements collection stage.
- The ER data model employs three basic concepts:

  entity ,

  relationship ,

  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

- 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

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

instructor

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

student

Entity Sets -- *instructor* and *student*

Attributes

- Each entity is described by a set of attributes; **i.e., descriptive properties possessed by all members of an entity set.**
  - •Example:
  - *instructor = (ID, name, salary)*
    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.

Types of Attributes

- Simple Attributes

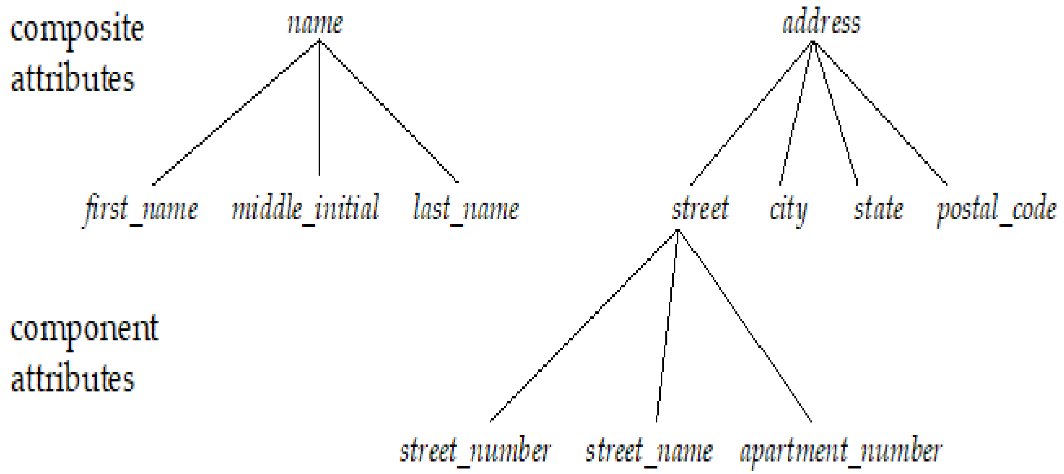
  having atomic or indivisible values.

  example: Dept–a string, PhoneNumber–an eight digit number

- Composite Attributes

  having several components in the value.

  example:



Types of Attributes

- Single-valued

  having only one value rather than a set of values.

  for instance, PlaceOfBirth–single string value.

- Multi-valued

  having a set of values rather than a single value.

  for instance, CoursesEnrolledattribute for student

  EmailAddress attribute for student

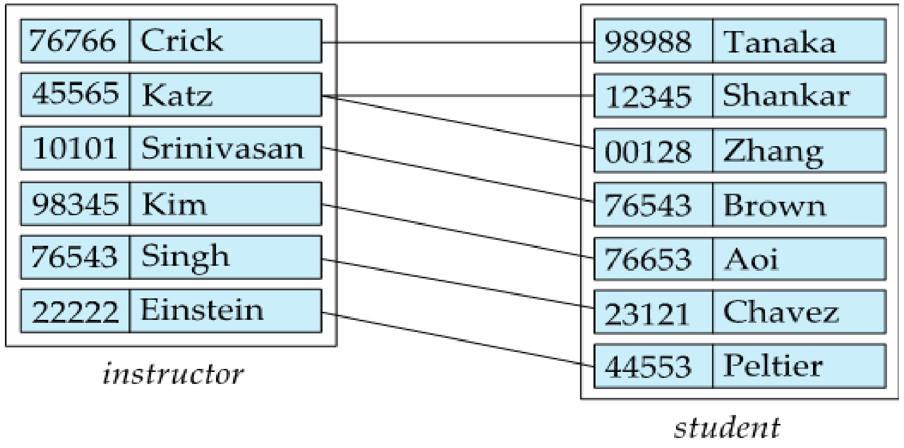
- Derived Attributes
  - Attribute value is dependent on some other attribute.
  - example:Agedepends on DateOfBirth.So age is a derived attribute.

Relationships

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

  44553 (Peltier) *advisor* 22222 (Einstein)

  student entity relationship set *instructor* entity
- we define the relationship set *advisor* to denote the associations between students and the instructors who act as their advisors.



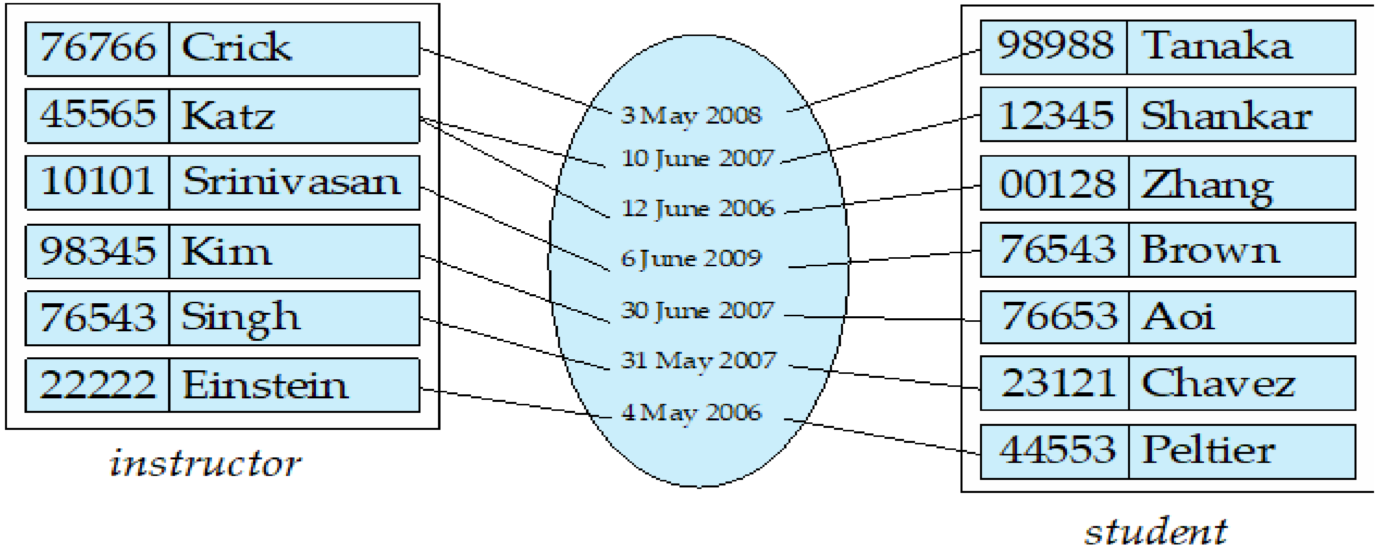
Relationships

- A **relationship set** is a mathematical relation among more than two or more entities, each taken from entity sets.

- **Degree of a relationship**
  - the number of participating entities.
    - Degree 2: binary
    - Degree 3: ternary
    - Degree n: n-ary
  - Binary relationships are very common and widely used.

Relationship Sets

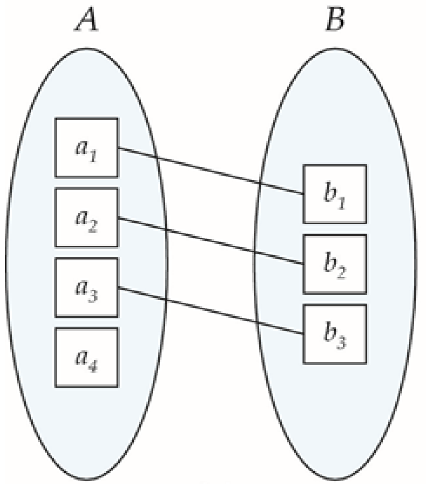
- An attribute can also be associated with a relationship set.
- For instance, the *advisor* relationship set between entity sets *instructor* and *student* may have the attribute *date* which tracks when the student started being associated with the advisor



Mapping Cardinality Constraints (Cardinality ratio)

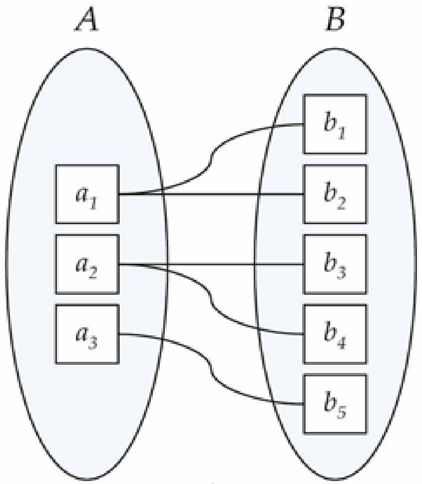
- Express the number of entities to which another entity can be associated via a relationship set.
- Most useful in describing binary relationship sets.
- For a binary relationship set the mapping cardinality must be one of the following types:
  - One to one
  - One to many
  - Many to one
  - Many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

Mapping Cardinalities



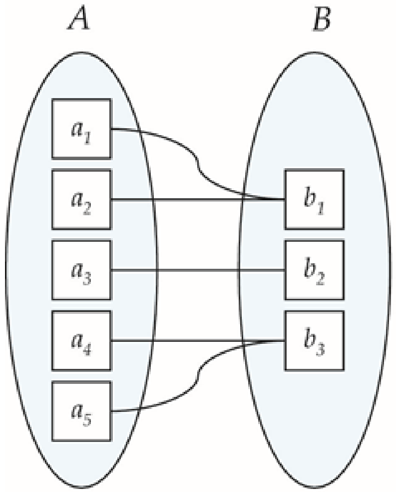
(a)

One to one



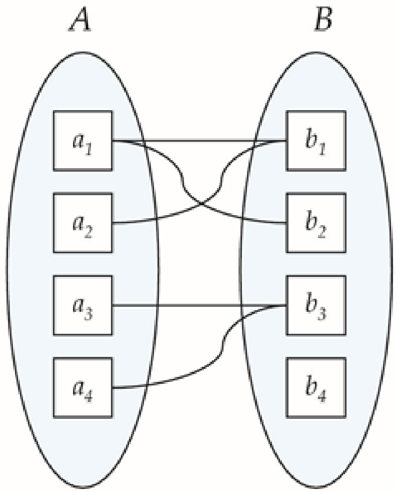
(b)

One to many



(a)

Many to one



(b)

Many to many

Participation Constraints

- **Total participation** : every entity in the entity set participates in at least one relationship in the relationship set.
  - participation of *student* in *advisor r*elation 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

Weak Entity Sets

- 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.**
- 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**.

Weak Entity- Example