Chapter 7: Entity-Relationship Model

Edited by Radhika Sukapuram

Database System Concepts, 7th Ed.

Covered so far

- Assumed a database schema
- Designed queries, constraints and updates
- Understood security schemes

But how to arrive at a database schema ?

For complex real world applications, one cannot directly arrive at a database schema

Design Phases

- Understand the data needs of the prospective database users by talking to users and domain experts
  - Output: textual descriptions of user requirements
- Choose a data model
  - Apply the concepts of the chosen data model, translate requirements into a conceptual schema of the database.
  - Entity-Relationship model is typically used
  - What are the entities ? What are there relationships between them ?
  - Output: Entity-Relationship diagram and functional requirements
  - Review:
    - to find out if requirements are met
    - No conflicts
    - No redundant features

Design Phases cont.

- Functional requirements describe
  - The kinds of operations – transactions – that will be performed on the data
- Implementation
  - Logical design
    - map E-R diagram to a relational model
    - 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 – file organisation, index structures
- Changes after implementation
  - Changes to logical schema are hard – affect application code
  - Do logical design with care!

Design Approaches

- Entity Relationship Model
  - 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 (later)
  - Formalize what designs are bad, and test for them

Outline of the ER Model

ER model -- Database Modeling

- The ER data model - to facilitate database design
  - by allowing specification of an enterprise schema
  - The enterprise schema represents the overall logical structure of a database.
  - Useful in mapping the meanings and interactions of real-world enterprises onto a conceptual schema.
- The ER data model employs three basic concepts:
  - entity sets,
  - relationship sets,
  - attributes.
- ER diagram is a graphical representation

Entity Sets

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

    instructor = (*ID, name, street, city, salary*)
    course= (*course_id, title, credits*)
- A minimal 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

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

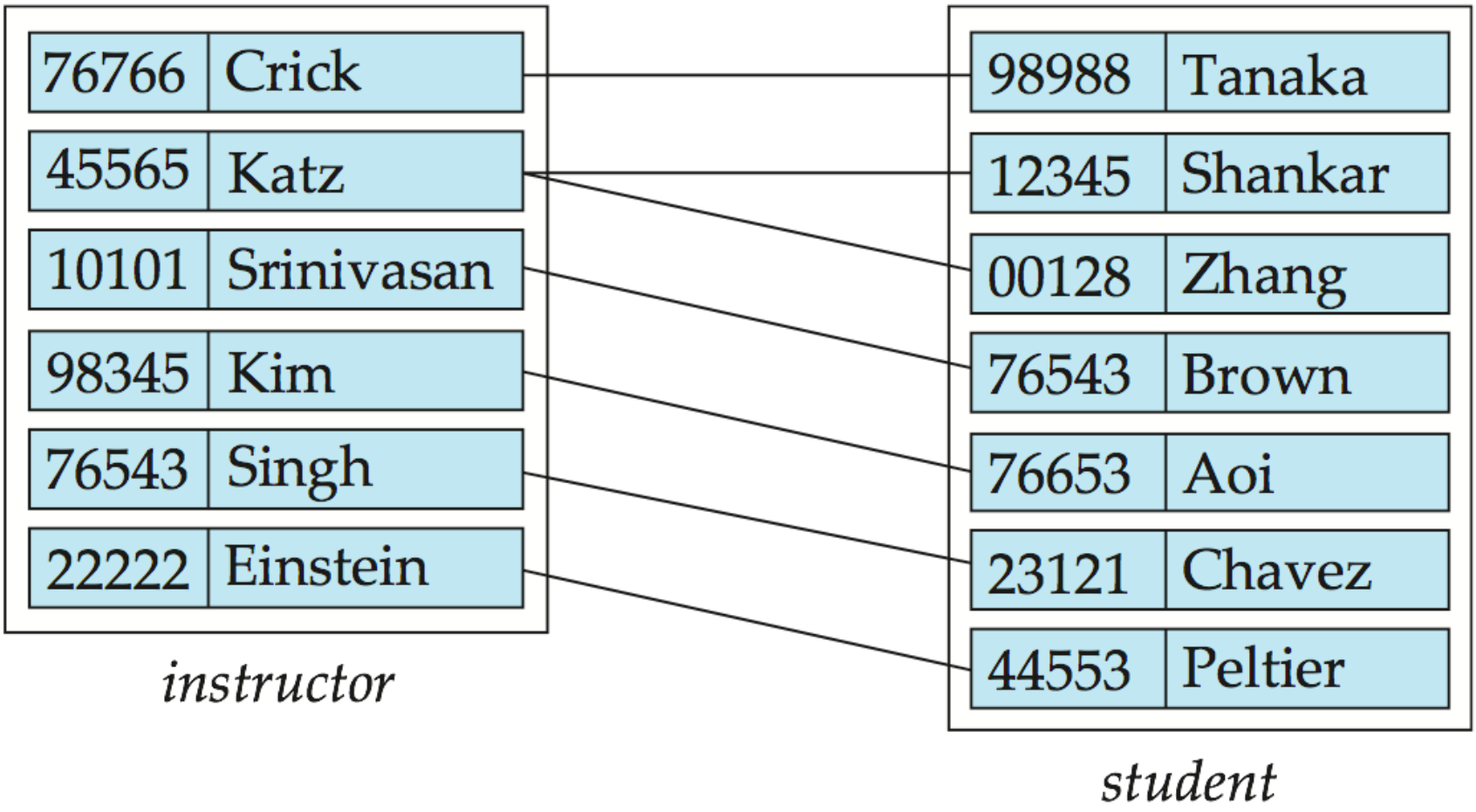
$$\{(e_1, e_2, \ldots e_n) \mid e_1 \in E_1, e_2 \in E_2, \ldots, e_n \in E_n\}$$

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

  - Example:

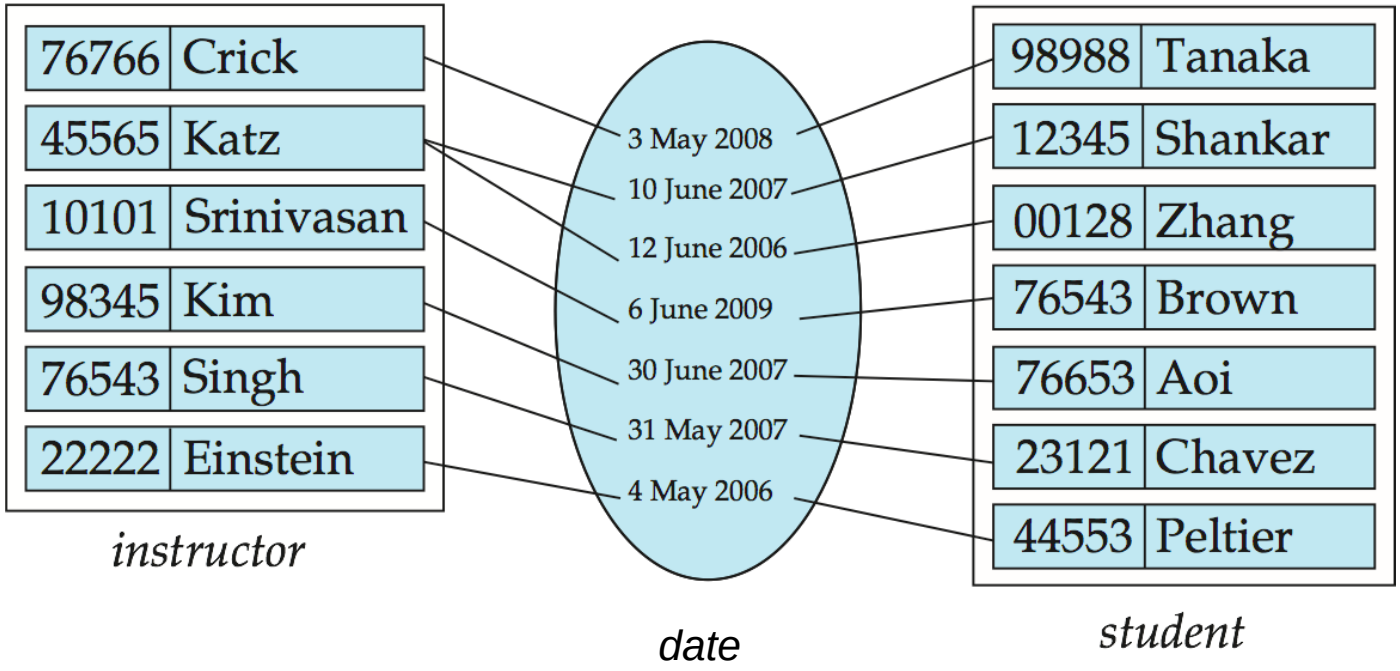
$$(44553,22222) \in advisor$$

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*
- *date* tracks when the student started being associated with the advisor



Degree of a Relationship Set

- binary relationship
  - involves two entity sets (or degree two).
- 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*

Roles

- Entity sets of a relationship need not be distinct
  - If distinct, the function an entity set plays is implicit
  - Otherwise each occurrence of an entity set plays a “role” in the relationship
  - The same entity set participates in a relationship set more than once, in different roles
  - Called recursive relationship set
- One *course* is a pre-requisite for another *course*
  - Use a relationship set called *prereq*
    - Modelled by ordered pairs of *course* entities
    - (*course_role*, *prereq_role*) – *prereq_role* is the pre-requisite
    - (*prereq_role, course_role*) pairs are excluded
    - *prereq* is a relationship between *course_role* and *prereq_role*

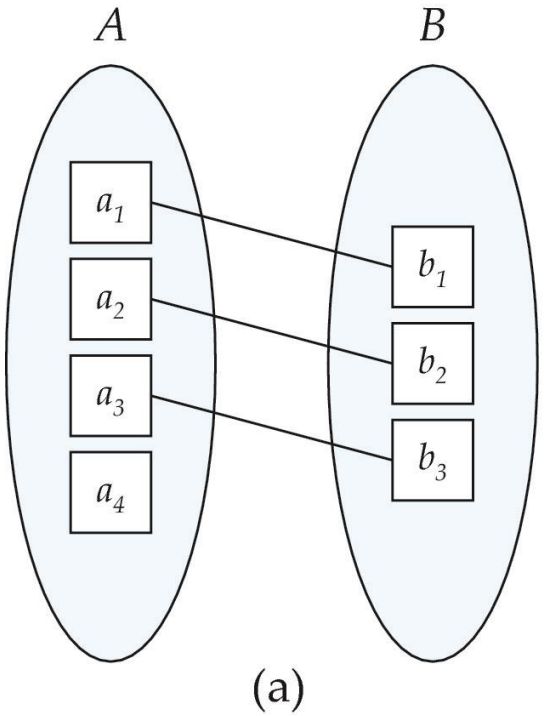
Mapping Cardinality Constraints

- Mapping cardinalities or cardinality ratios:
  - 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

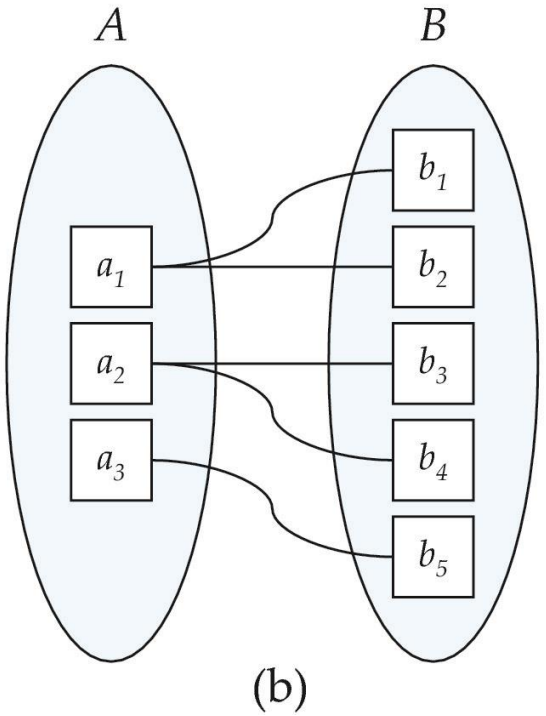
Mapping cardinalities

- One-to-one
  - An entity in A is associated with at most one entity in B and an entity in B is associated with at most one entity in A
- One-to-many
  - An entity in A is associated with zero or more entities in B. An entity in B is associated with at most one entity in A

Mapping Cardinalities



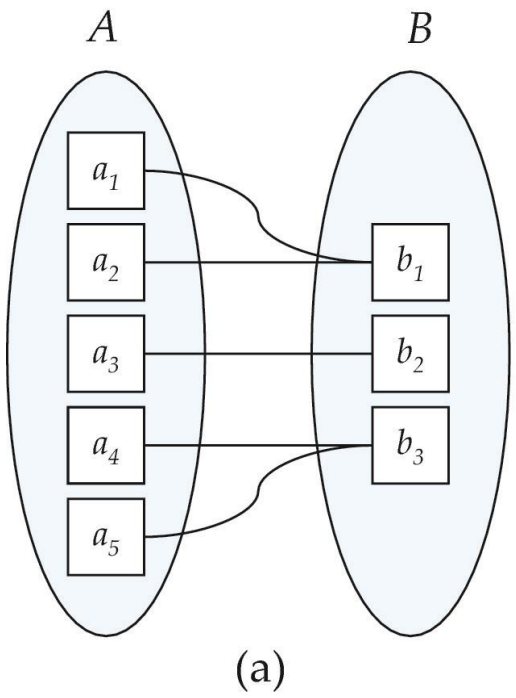
One to one



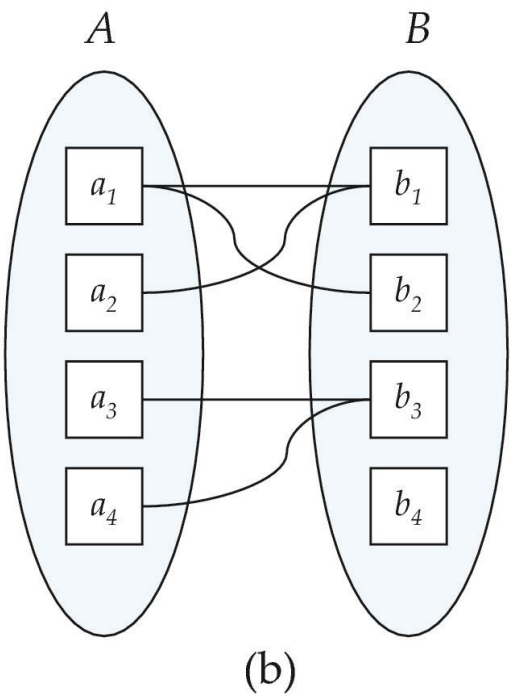
One to many

Note: Some entities in *A* and *B* may not be mapped to any entities 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

Participation Constraints

- Participation of an entity set E in a relationship R is total if every entity in E participates in at least one relationship in R
  - Every *student* entity must be related to at least one *instructor* through the *advisor* relationship
- If only some entities of an entity set E participate in a relationship R, the participation is partial
  - Only some of the *instructor* entities are related to *student* entities through the relationship *advisor*

Keys for relationship sets

- Superkeys, candidate keys and primary key apply to entity sets
- R is a relationship set involving entities E_1, E_2, …, E_n
- *primarykey*(E_1) denotes the set of attributes that form the primary key of E_1
- Assume that attribute names of all primary keys are unique
  - Suppose R has no attributes associated with it
    - The set of attributes

    primarykey(E_1) ∪ *primarykey*(E_2) ∪ …. ∪ *primarykey*(E_n) describes a relationship in R
  - Suppose R has attributes a_1, a_2, …, a_m associated with it
    - The set of attributes

    primarykey(E_1) ∪ *primarykey*(E_2) ∪ …. ∪ *primarykey*(E_n) ∪ {a_1, a_2, …, a_m } describes a relationship in R

Keys for relationship sets cont.

- Whether R has attributes or not,
  - *primarykey*(E_1) ∪ *primarykey*(E_2) ∪ …. ∪ *primarykey*(E_n) form a superkey for R
- Attribute names of primary keys across entities are not unique
  - Attributes are renamed : <name of entity set> . <name of attribute>
- An entity set participates more than once in a relationship
  - Attributes are renamed : <role name> . <name of attribute>
  - Example: courses and pre-requisites

Primary keys for a relationship set

- Structure of primary key of a relationship set depends on mapping cardinality
- Suppose *advisor* is a relationship set between *instructor* and *student*
  - many-to-many : union of primary keys of *instructor* and *student*
  - many-to-one from *student* to *instructor* : each *student* can be advised by at most one *instructor* : primary key of *student*
  - many-to-one from *instructor* to *student* : each *instructor* can advise at most one *student* : primary key of *instructor*
  - one-to-one: either candidate key can be used as the primary key

Primary keys: non-binary relationship sets

- No cardinality constraints:

primarykey(E_1) ∪ *primarykey*(E_2) ∪ …. ∪ *primarykey*(E_n) form a superkey for R and is the only candidate key

- With cardinality constraints: specified later