
DBMS
data
database
distributed
shared
parallel
system
processing
network
sites
multiple
architecture
DBMSs
Distributed
DDBMS
memory
processors
centralized
access
performance
computer
disk
stored
systems
Parallel
processor
fragments
may
require
site
users
applications
computers
nothing
Shared
technology
Figure
bottleneck
interconnection
administered
organizational
transparent
management
Database
Management
provide
make
local
single
one
split
sometimes
decentralized
large
disks
islands
resources
remote
defined
controlled
based
logically
global
associated
consists
partitioned
used
least
must
System
capable
mode
number
user
dis
application
Parallel
Distributed
Shared
databases
logical
nodes
definition
development
control
scalability
approach
improve
distinction
typically
across
major
fact
eliminates
reliability
physically
within
machines
like
share
transparent
makes
per
often
Date
directly
operational
referred
computing
scalable
architectures
order
structure
collection
provides
way
incompatible
consider
controlled
remote
defined
resources
islands
large
disks
based
stored
disk
computer
access
processors
centralized
bottleneck
interconnection
Figure
technology
capable
must
System
used
least
partitioned
consists
associated
global
logically
way
incompatible
structure
collection
provides
controlled
based
remote
defined
resources
islands
large
disks
scalable
architectures
order
split
one
single
sometimes
computing
referred
operational
local
network
decentralized
directly
Date
often
provide
make
sites
multiple
application
management
transparent
number
dis
user
per
makes
Management
bottleneck
interconnection
Figure
administered
organizational
mode
centralized
processors
access
computer
disk
stored
systems
logically
global
associated
consists
partitioned
used
least
must
System
capable
technology
nodes
definition
development
memory
DBMSs
Distributed
logical
distinction
control
scalability
approach
improve
Database
applications
computers
shared
parallel
system
nothing
Shared
databases
share
DDBMS
Parallel
processor
fragments
may
require
site
users
physically
within
machines
like
typically
across
major
fact
eliminates
reliability
communication
distribution
data
database
distributed
DBMS

Database System Concept (CSE 3103)

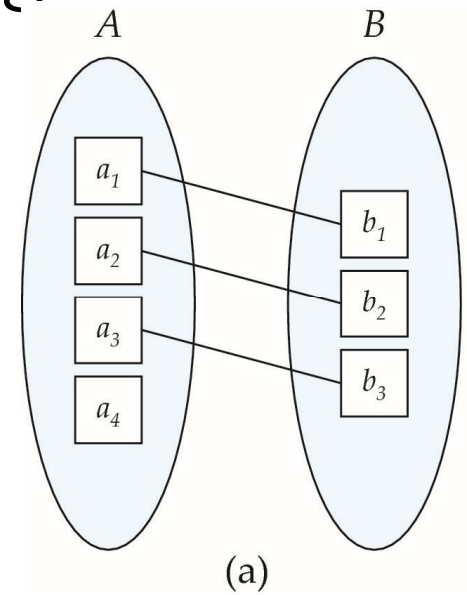
Lecture 04-Day 02

Nazmus Sakib, Assistant Professor, Dept. of CSE, AUST

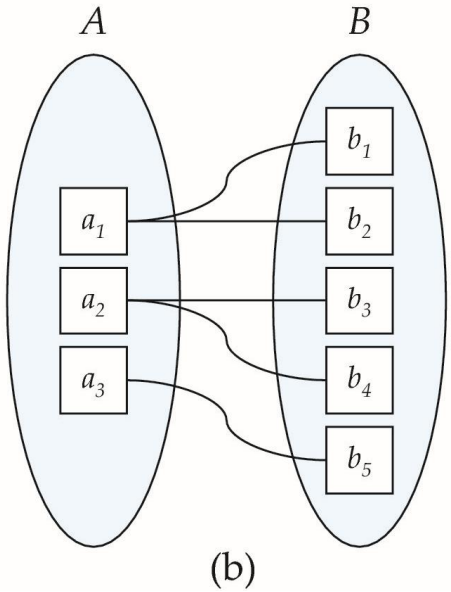
Mapping Cardinality Constraints

- 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



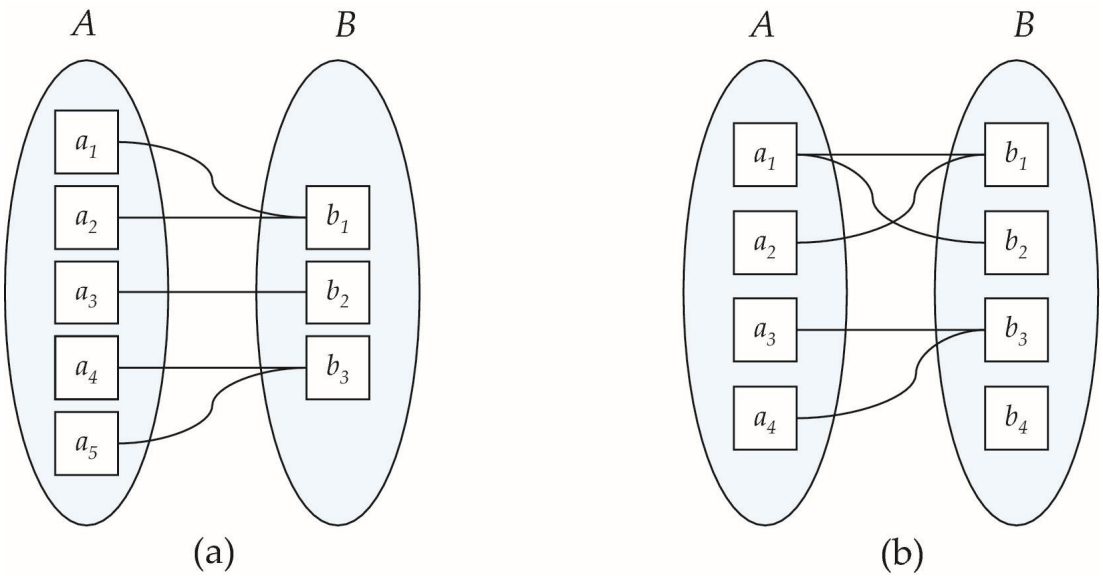
One to one



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

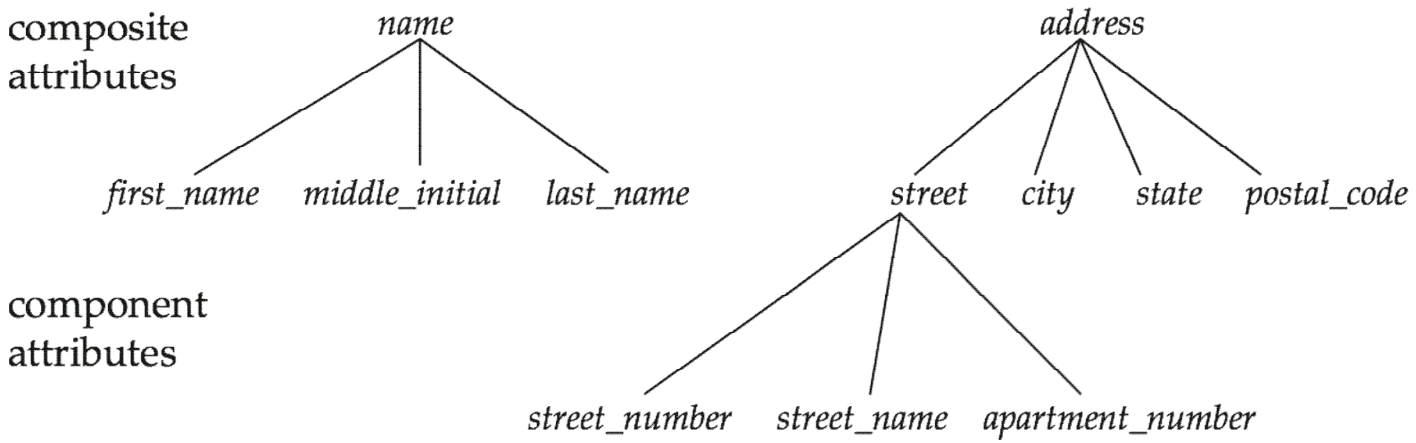
Many to many

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
    - 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



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.
- 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.*