

# Fundamentals of DATABASE SYSTEMS

FOURTH EDITION

ELMASRI  NAVATHE

## Chapter 4 - Part II

### Enhanced Entity-Relationship and UML Modeling

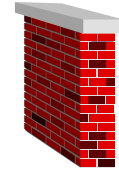
This material may be used at the beginning of the transparencies for Chapter 4. It sets the stage for looking at modeling from a general standpoint. The details of abstraction mentioned here are covered in Section 4.8, but the constraints are previously discussed in Section 3.4 already.



Copyright © 2004 Elmasri and Navathe.

# THE BASICS

- Fundamental Principle of Modeling:
  - Data Abstraction
- Basic Process of Modeling
  - Define building blocks for holding groups of data
  - Use rules of a data model to establish relationships among blocks
- Add constraints - structural/ semantic



## Part 1: Fundamentals of Data Modeling

- 1 Inputs to Data Modeling
- 2 The Process of Modeling
- 3 Data Modeling Abstractions
- 4 Classification
- 5 Aggregation
- 6 Identification
- 7 Generalization
- 8 Coverage Constraints in Generalization
- 9 Cardinality and Participation Constraints

## Inputs to Data Modeling

- Using the products of requirements analysis
- Verbal and written communication among users and designers
- Knowledge of meaning of data
  - Existing Programs
  - Existing Files
  - Existing Documents
  - Existing Reports
- Application Planning / Documentation and Design

## Overall Process of Modeling

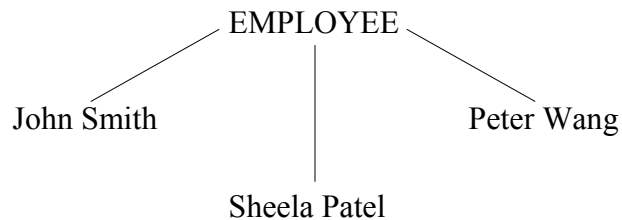
- Abstraction
- Use of some modeling discipline (Data Model)
- Use of a representation technique
  - Language
  - Diagramming
  - Tools
- Analysis of business rules/semantic constraints (these are typically beyond the capability of the data model)

## Types of Abstractions

Classification	A is a <u>member of</u> class B
Aggregation	B,C,D are aggregated into A A is <u>made of/composed of</u> B,C,D
Generalization	B,C,D can be generalized into A, <b>B</b> <u>is-an</u> A, C <u>is- an</u> A, D <u>is-an</u> A
Specialization	A can be specialized into B,C,D B,C,D are special cases of A

## Classification Abstraction

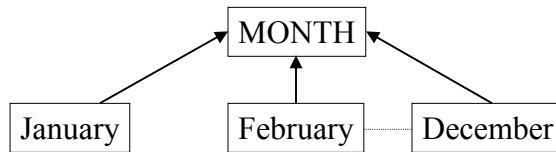
- Relationship between a class and its members  
John Smith, Sheela Patel, and Peter Wang are all employees.  
They are all members of a class: EMPLOYEE class



Each individual employee is a member of the class  
EMPLOYEE

## Classification Abstraction (contd.)

Exhaustive enumeration of members:



January, February etc. are members of the class “MONTH”

↑ Represents “member-of” relationship

In object-oriented modeling :

MONTH : an Object type or class

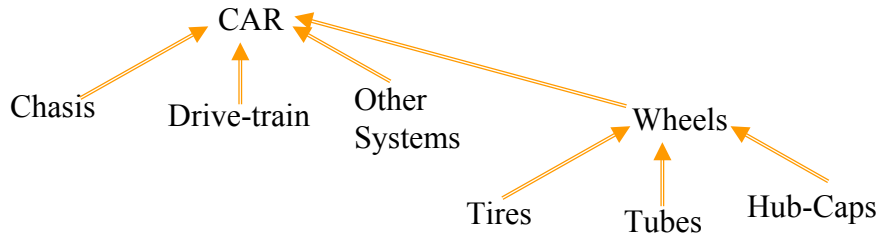
January ... December : objects that belong to class MONTH

## Classification - Class Properties

- Collection of similar entities or concepts into a higher level concept
- EMPLOYEE class collects all employees into one class
- A class has properties called “class properties”
- EMPLOYEE class has **class properties** - e.g., average salary, total number of employees
- Each member has values for own properties (e.g. name, address, salary): called **member properties**

## Aggregation Abstraction

Defines a new class from a set of classes which are identified as components of the root class



—> represents IS-PART-OF (component) relationship

Root class: CAR

Component Classes: Chassis, Drive-Train, Other Systems, Wheels

Root class: Wheels

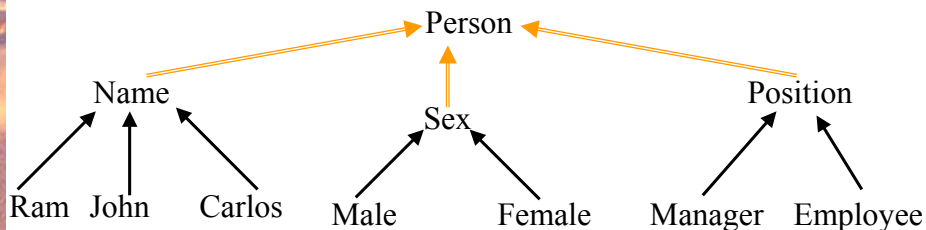
Component Classes: Tires, Tubes, Hub-Caps

## Classification and Aggregation

Classification and Aggregation are used to build schemas

Example: class Person

Representation:



Name, Sex, and Position aggregate into Person. They are classes themselves.

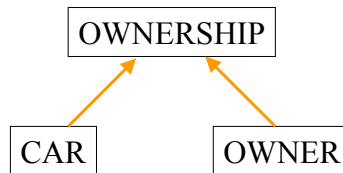
Ram, John, Carlos are classified into Name or Name is a classification of Ram, John, Carlos

## Two Contexts for Aggregation

Aggregate two or more classes into a higher level concept. It may be considered a relationship or association between them.

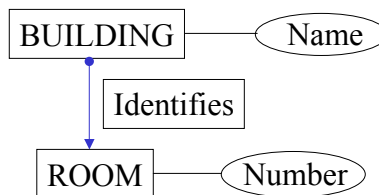
Context1: CAR is an aggregate (composition) of Chassis, Drive-train, Other Systems, Wheels.

Context 2: OWNERSHIP is an aggregate (relationship) of CAR and OWNER



## Identification

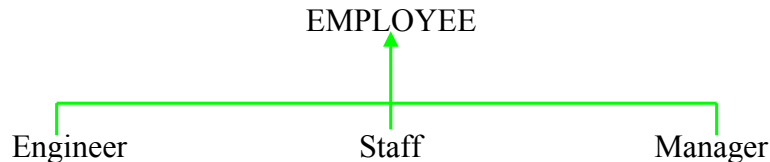
Identifies one concept (an instance of it) from another concept.



## Generalization Abstraction

Defines a set-subset relationship between a class and a set of member classes.

Establishes a mapping (or a relationship) from the **generic class** to the **member class** (or subclass, or subset class).



GENERIC CLASS: EMPLOYEE

MEMBER CLASS: Engineer, Staff, Manager

Implies that all properties associated with the Employee class are inherited by the three leaf classes.

## Data Abstraction (contd.)

Process of hiding (suppressing) unnecessary details so that the high level concept can be made more visible.

This enables programmers, designers, etc., To communicate easily and to understand the application's data and functional requirements easily.

### TYPES OF ABSTRACTION

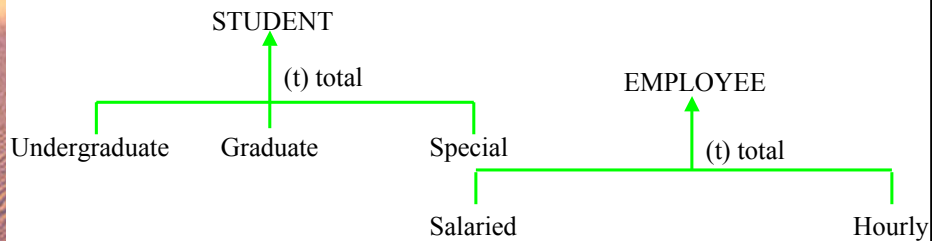
Classification:	IS-A-MEMBER-OF
Aggregation:	IS-MADE-OF, IS-ASSOCIATED-WITH
Composition:	IS-MADE-OF (similar to aggregation) (A COMPRISES B,C,D)
Identification:	IS-IDENTIFIED-BY
Generalization:	IS-A      IS-LIKE      IS-KIND-OF

## Coverage Constraints for Generalization Abstraction

### TYPE 1 : Total vs. Partial coverage

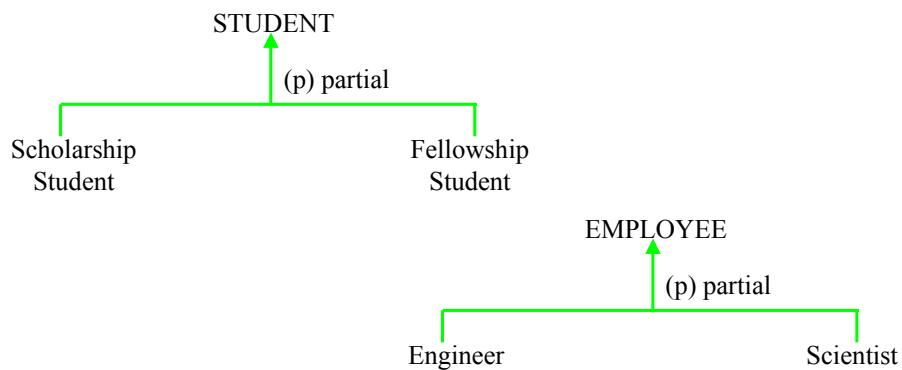
**Total:** The coverage is total if each member of the generic class is mapped to at least one member among the member classes

**Partial:** The coverage is partial if there are some member(s) of the generic class that cannot be mapped to any member among the member classes



## Coverage Constraints for Generalization Abstraction (contd.)

### Partial Coverage Constraint examples:

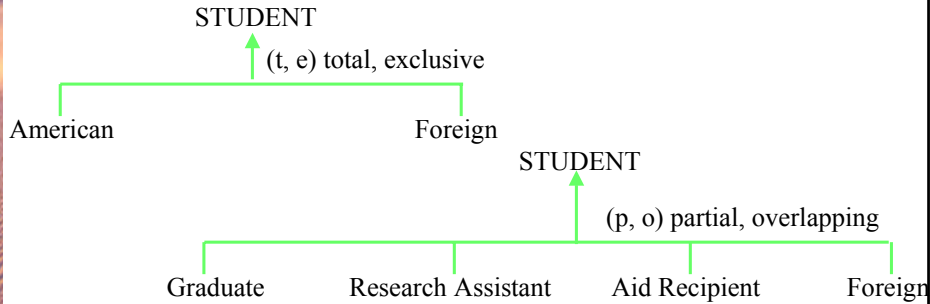


## Coverage Constraints for Generalization Abstraction

### TYPE 2: EXCLUSIVE VS. OVERLAPPING (Disjointedness Constraint)

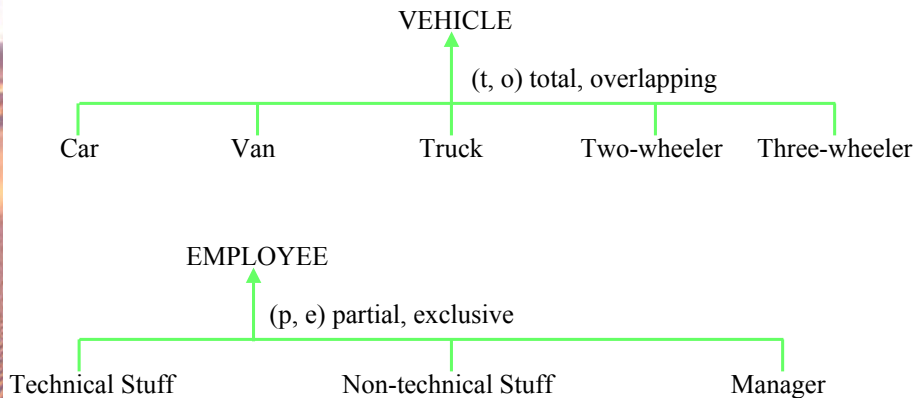
**EXCLUSIVE** constraint: A member of the generic class is mapped to one element of at most one subset class.

**OVERLAPPING** constraint: There exists some member of the generic class that can be mapped to two or more of the subset classes.



## Coverage Constraints for Generalization Abstraction (contd.)

More examples of different combinations:



## Cardinality Constraints

**Cardinality Constraint:** Quantification of the relationship between two concepts or classes (a constraint on aggregation)

**MINIMUM (A,B) = n**

At a minimum, one instance of A is related to at least n instances of B.

n = 0	MIN(A,B) = 0	MIN(Person, Car) = 0
n = 1	MIN(A,B) = 1	MIN(Cust, Ship-address) = 1
n = inf.	MIN(A,B) = inf.	NOT POSSIBLE
n = x (fixed)	MIN(A,B) = x	MIN(Car, Wheels) = 4

## Cardinality Constraints (contd.)

**MAXIMUM (A,B) = n**

At a maximum, one instance of A is related to at least n instances of B.

n = 0	MAX(A,B) = 0	DOES NOT ARISE
n = 1	MAX(A,B) = 1	MAX(Cust, Ship-address) = 1
n = inf.	MAX(A,B) = inf.	MAX(Cust, Orders) = inf.
n = x (fixed)	MAX(A,B) = x	MAX(Stud, Course) = 6

## Participation Constraints

<b>MIN (A,B) = 0</b>	<b>Optional Participation</b>
<b>MIN (A,B) = 1</b>	<b>Mandatory Participation</b>
<b>MAX (A,B) = 0</b>	<b>No Participation</b>
<b>MIN (A,B) = x, MAX (A,B) = y</b>	<b>Range Constrained Participation</b>

## Summary of Modeling Concepts

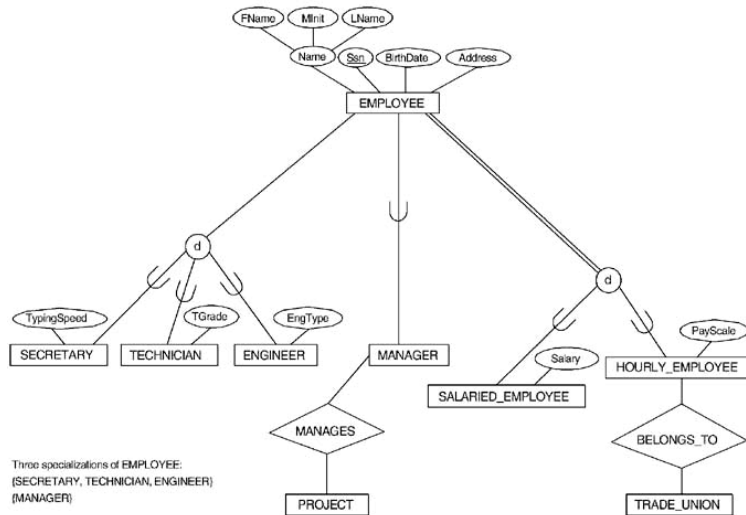
### ABSTRACTIONS

- CLASSIFICATION
- AGGREGATION (COMPOSITION AND ASSOCIATION)
- IDENTIFICATION
- GENERALIZATION AND SPECIALIZATION

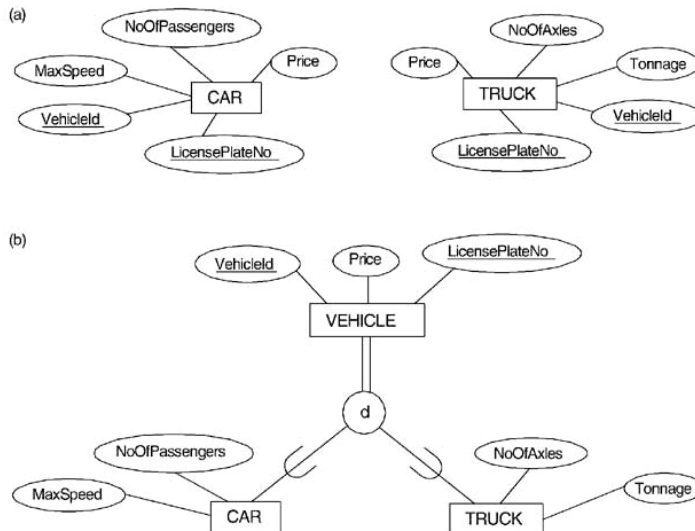
### CONSTRAINTS

- CARDINALITY (Min. and Max)
- PARTICIPATION
- COVERAGE (Total vs. Partial, Exclusive vs. Overlapping)

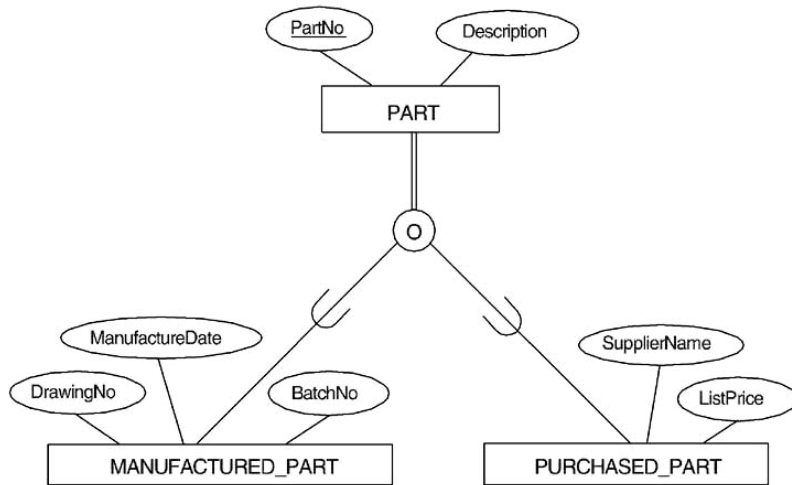
**Figure 4.1** EER diagram notation for representing specialization and subclasses.



**Figure 4.3** Examples of generalization. (a) Two entity types CAR and TRUCK. (b) Generalizing car and TRUCK into VEHICLE.



**Figure 4.5** Notation for specialization with overlapping (nondisjoint) subclasses.



**Figure 4.6** A specialization lattice with the shared subclass ENGINEERING\_MANAGER.

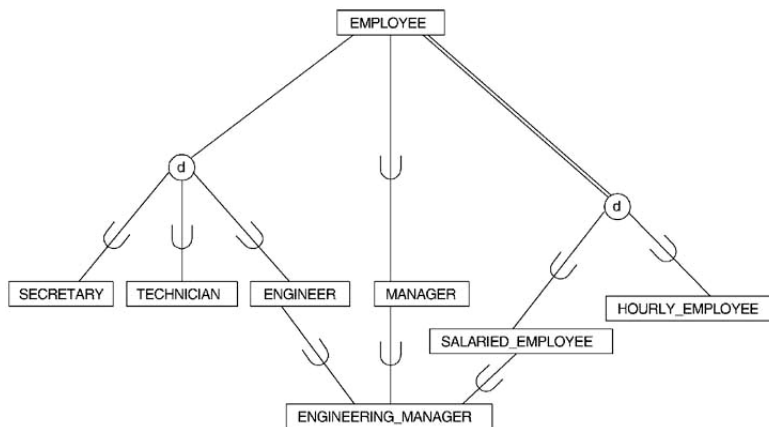


Figure 4.7 A specialization lattice (with multiple inheritance) for a UNIVERSITY database.

