ER Notation

- Entity
- Weak Entity
- Relationship
- Identifying Relationship

- Attribute
- Key Attribute
- Multivalued Attribute
- Composite Attribute
- Derived Attribute

Total Participation of $E_2$ in $R$
Cardinality Ratio $1:N$ for $E_1:E_2$ in $R$
Structural Constraint $(\text{min, max})$ on Participation of $E$ in $R$
Notes:
A LEG (segment) is a nonstop portion of a flight.
A LEG_INSTANCE is a particular occurrence of a LEG on a particular date.
Enhanced ER Model (EER)

- aka Extended Entity-Relationship Model
- adds Inheritance
  - indicates that one entity type is an extension of another entity type
  - often referred to as an IS-A relationship
Inheritance

http://www.agilemodeling.com/artifacts/classDiagram.htm
Inheritance defines a subclass relationship
  - A subclass inherits all properties (members) of the superclass
- This is the perspective of most modern programming languages
Set Theoretic View of Inheritance

- B ⊂ A, C ⊂ A
- Every B is also an A
- Every C is also an A

Everything true about a member of a set is also true about any member of its subsets.
EER IsA Notation

PERSON

Name

Phone

ID

IsA

STUDENT

Major

Class

PROFESSOR

Dept

Office

(not used in our textbook)
EER Subset Notation

preferred notation: shows directionality of inheritance
EER Subset Notation (variant)

The circle may be omitted when not needed.
Inheritance of Properties

- Student and professor entities have all attributes defined for a person, plus additional attributes.
- Keys are also inherited. Subtypes should not define new keys.
Inheritance of Relationships

- Relationships are also inherited by subtypes

Every mouse must be sold with a computer? **TRUE**

Every computer must be sold with a mouse? **FALSE**
Constraints on Inheritance

- **Disjointness**: an entity can be a member of at most one subtype
  - a person *may be* a student or *may be* a professor, but *not both*

- **Covering**: every entity of the supertype must also be a member of at least one subtype
  - every person *must be* a professor or a student
Disjoint Subsets

- $B \subseteq A, C \subseteq A$
- $B \cap C = \emptyset$

no entity is in both $B$ and $C$
Overlapping Subsets

- $B \subseteq A$, $C \subseteq A$
- $B \cap C = \emptyset$

an entity may be in both $B$ and $C$

overlapping = non-disjoint
Covering Subsets

- $B \subseteq A$, $C \subseteq A$
- $B \cup C = A$

every entity in $A$ is also in $B$ or $C$
Non-Covering Subsets

- $B \subseteq A$, $C \subseteq A$
- $B \cup C = A$  

some entities in $A$ are not in $B$ or $C$
Inheritance Constraint Notation

• IsA (triangle) notation:
  write “disjoint” and/or “covering” next to the triangle

• Subset notation,
  ‘d’ in the circle \(\rightarrow\) disjointness
  ‘o’ in the circle \(\rightarrow\) no disjointness (overlap)

required participation from supertype indicates a covering constraint
Non-disjoint, Non-covering

- Every A can also be a B or a C, or both, or neither
Disjoint, Non-covering

- Every A can also be a B or a C or but not both
Non-disjoint, Covering

- Every A must be a B or a C or both
Every A must be a B or a C, but not both

Disjoint, Covering
disjoint, covering inheritance:
every car is a vehicle
every truck is a vehicle
every vehicle is either a car or a truck
no vehicle is both a car and a truck
disjoint, non-covering inheritance:
employees may be secretaries, technicians or engineers, but not more than one of these
non-disjoint, covering inheritance:
every part is a manufactured part, or a purchased part, or both (a purchased, manufactured part)

** How many attributes does a purchased, manufactured part have?**
** How would we model this in UML (C++ or Java)?**
** Interpret this schema.

** Can you find a real-world example?
Multiple inheritance gives us a lattice, rather than a hierarchy

** Could we have engineering managers without defining the E_M class? (compare to previous example)
Unions

- Union defines a type as the union of other types
- OWNER = PERSON U COMPANY
- OWNER is called a union type or category
  - OWNER is the subtype of the union of PERSON and COMPANY
- Not multiple inheritance
  - an OWNER does not need all the attributes from both PERSON and COMPANY
Comparison

B and C are subtypes of A
B ⊂ A
B ⊂ A
BUC ⊂ A

A is a subtype of BUC
A ⊂ BUC