ER Model (Revisited)

- Why ER model?
  - a very popular high-level conceptual data model
  - facilitates database design by specifying schema that represent the overall logical structure of the DB
  - entities and attributes: an attribute is a function which maps an entity set into a domain
    - Faculty (Name, Dept, SSN)
      - domain for attribute Dept = \{CS, EE, APMA, SYS\}
  - a particular entity is described by a set of values:
    - Faculty (Name: John Doe, Dept: CS, SSN: 123-45-6789)
  - entity type plays a particular role in a relationship: usually implicit but must be specified if not distinct
    - Parents (Person, Person), War (Country, Country)
Mapping Cardinality

- Relationships
  
  - 1:1, 1:N, N:M are distinguished by a directed line (→) and an undirected line (—)
  
  - a directed line represents "at most one", not requiring there must be one corresponding entity for every entity
  
  - a description of all possible associations in the real-world that is being modelled
  
  - 1:1 relationship is rather rare in databases, while N:M relationships are quite common (hard to represent)
  
  - naming relationships are sometimes tricky
  
  <ex> A relationship between Faculty and Students: Should it be advisee or advisor?
Mapping Cardinality

One-to-one relationship (one customer - one account)

One-to-many from customer to account
A customer can have several accounts, but no account can be shared

Many-to-one from customer to account
A customer can have only one account, but accounts can be shared

Many-to-many relationship
An ER diagram represents several assertions about the real-world. When attributes are added, more assertions are made. How can we ensure that it is "faithful"?

- A database is judged correct if it captures ER diagram correctly.
- There is no way of verifying that ER diagram is logically correct.
Key Attributes

- **Key and key attributes**
  - key: a unique value for an entity
  - key attributes: a group of one or more attributes that uniquely identify an entity in the entity set

- **Super key, candidate key, and primary key**
  - super key: a set of one or more attributes which allows to identify uniquely an entity in the entity set
  - candidate key: minimal super key
  - there can be many candidate keys

<ex> Employee (Name, Address, SSN, Salary, Project) (Name, Address) and (SSN) are candidate keys, but (Name, SSN) is not a candidate key

- primary key: a candidate key chosen by the DB designer
- denoted by underlining in ER diagram
Weak Entity Types

• Weak entity type
  - its existence depends on other entity (owner)
  - no key attributes of its own
  - cannot be identified without an owner entity
  - indicated in ER diagram by a doubly outlined boxes
  
  <ex> Transaction (T#, Type, Date, Amount) is a weak entity with Account (A#, Balance) as its owner entity

• Partial key
  - a set of attributes that can uniquely identify weak entities related to the same owner entity
  
  <ex> T# is a partial key in Transaction entity

• To use weak entity types or not?
  - basically the designer’s choice
  - preferable if it has many attributes and participates in relationships besides its owner entity types
Generalization

- Relationships among entity types
  - to emphasize the similarities among lower-level entity types and to hide their differences
  - attributes of higher-level entity sets are inherited by lower-level entity sets
Transforming ISA into Relations

1. Create a relation for the higher-level entity set, and for each lower-level entity set, create a relation with the primary key of the higher-level entity set

   Account (Number, Balance)
   Savings-acct (Number, Interest-rate)
   Checking-acct (Number, Overdraft-amount)

2. Do not create for higher-level entity set. For each lower-level entity set, create a relation with all the attributes of the higher-level entity set

   Savings-acct (Number, Balance, Interest-rate)
   Checking-acct (Number, Balance, Overdraft-amount)

- The second method is possible only when the generalization is
  - disjoint: no entity belongs to more than 2 subclass
  - complete: every member of superclass is a member of subclass