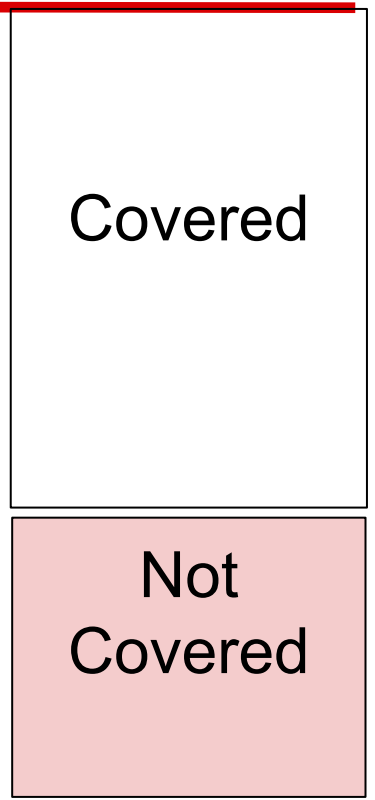
# Conceptual Database Design

Adapted from Chapter 16 (Connolly & Begg)

# Conceptual database design

1. Identify the entity types
2. Identify the relationship types
3. Identify the attributes
4. Identify the attribute domains
5. Identify the candidate keys and primary key
6. Apply generalization (is-a), aggregation (has-a), composition (part-of)
7. Check model for dependency
8. Validate conceptual model against user transactions
9. Review model with user



# Identify entity types

1. Identify the nouns in the user requirement specification
2. Entities should be major objects NOT properties of other objects.
3. Objects that have existence in their own right
4. Look for entity types that may be synonyms of each other
   a. Document the synonyms
5. All entity names should be well descriptive

# Identify relationship types

1. Identify the verbs in the user requirement specification
2. Classify relationships as complex, binary or recursive.
3. Determine the multiplicity of each relationship
4. Check for fan and chasm traps
5. Document and assign meaningful names to the relationships

# Identify entity and relationship attributes

1. Identify the properties or the qualities of the entity types
2. Classify each attribute as:
   a. Simple versus composite attribute
   b. Single versus multi-valued attribute
   c. Derived attribute (ensure attribute can be derived from given attributes)

# Determine candidate, primary keys

1. Identify the candidate keys
2. Choose the primary key from the candidate keys that are:
   a. Candidate key with the minimal set of attributes
   b. Candidate key that is least likely to be updated
   c. Candidate key with the fewest number of bytes
   d. Candidate key with the lowest maximum value
   e. Candidate key that is easiest to manipulate for a user.
3. All other candidate keys are designated as alternate keys
4. Be willing to add new attributes that provide uniqueness if the current candidate keys are composite
5. Make sure that keys are properly identified for weak entity

# EER to represent hierarchical relationships

1. Generalization (is-a) allows us to represent super and subclasses for an entity type.
   a. Participation - all members of the superclass must fall into a subclass {Mandatory | Optional}
   b. Disjoint - subclasses do not share members {And | Or}
2. Composition (Part-of) allows us to represent an entity type that composes another entity type (strong ownership).
3. Aggregation (Has-a) allows us to represent an entity type that has a collection of another entity type

# Check model for redundancy

1. Review 1-1 relationships to ensure the entity types are really different entity types and not synonyms.
2. Remove redundant relationships: relationships that provide the same information as another relationship.
   a. Multiple paths between entity types are a potential source for redundancy
3. Consider time and its effect on each relationship
   a. Some relationships may seem redundant but really are necessary due to changes in relationships due to time

# Validate conceptual model with transactions

1. The conceptual data model must provide a response for all user defined transactions.
2. If model cannot provide an answer, the conceptual model is not complete.
3. Two methods that use two different representations of data model:
   a. Textual description of the user transaction
   b. Transaction pathway through the conceptual model to retrieve response for the transaction

# Review conceptual model with user

1. Must get sign-off from the user that the model capture all necessary data.
2. Implies user has verified all transactions can be answered