The Three Phases of Database Design

Adapted from Chapter 16, 17, 18 (Connolly & Begg)
Step 1: Conceptual database design

Entity relationship modeling (ER) creates a diagrammatic representation of a conceptual model

Conceptual database design: build a conceptual representation of the data domain.

Process must identify the necessary entities, relationships and attributes
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

Covered
Not Covered
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 the 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
Summary (Conceptual Design)

Creating a conceptual design is an iterative process, where your goal is to produce an unambiguous representation of the data domain and its processes.

The goal is communication between a technical team and a nontechnical team to ensure the proposed technical solution fulfills the needs of the enterprise.
Step 2: Logical database design

Once the conceptual model is created, we need to target a specific data model.

We will be targeting the relational data model.

Logical database design: translate the conceptual model to the logical structure of the database. For the relational model we must design the relations for the schema.
Steps to the logical database design

1. Derive relations for logical data model
2. Validate relations using normalization
3. Validate relations against user transactions
4. Check integrity constraints
5. Review logical data model with user

Step 2 will be described in a separate lecture.
Derive relations for ...

1. Strong entity types
2. Weak entity types
3. 1 to many (1:*) binary relationship types
4. 1 to 1 (1:1) binary relationship types
5. 1 to 1 (1:1) recursive relationship types
6. Superclass /subclass relationship types
7. Many to Many (*:* ) binary relationship types
8. Complex relationship types
9. Multi-values attributes
## Maps to a relation

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Strong entity</td>
<td>Create a relation that contains all simple attributes</td>
</tr>
<tr>
<td>Weak entity</td>
<td>Create a relation that contains all simple attributes - primary key must take into account the owner entity’s key</td>
</tr>
<tr>
<td><em>=:</em> binary relationship</td>
<td>Create a relation for the relationship, including all relationship attributes. Each entity in the relation is a foreign key in the relationship’s relation.</td>
</tr>
<tr>
<td>1:1 binary relationship</td>
<td></td>
</tr>
<tr>
<td>Mandatory participation</td>
<td>Combine entities into 1 relation</td>
</tr>
<tr>
<td>Optional participation</td>
<td>Define a foreign key for relation associated with mandatory participation</td>
</tr>
<tr>
<td>Both entities optional</td>
<td>Your choice for representation (either can have FK)</td>
</tr>
<tr>
<td>Multi-valued attributes</td>
<td>Define a relation for the multi-valued attribute and create a foreign key to the relation representing the containing entity</td>
</tr>
<tr>
<td>Complex relationship</td>
<td>Create a relation for the relationship, including all relationship attributes. Each entity in the relation is a foreign key in the relationship’s relation.</td>
</tr>
</tbody>
</table>
## Maps to a foreign key

<table>
<thead>
<tr>
<th>Entity/Relationship</th>
<th>Mapping to logical design</th>
</tr>
</thead>
<tbody>
<tr>
<td>1:* binary relationship</td>
<td>Define a foreign key on the “many” side. It points to a candidate key on the “1” side”. All relationship attributes are stored in the “many” relationship. No relation necessary for relationship.</td>
</tr>
<tr>
<td><em>:</em> binary relationship</td>
<td>Create a relation for the relationship, including all relationship attributes. Each entity in the relation is a foreign key in the relationship’s relation.</td>
</tr>
<tr>
<td>1:1 binary relationship</td>
<td></td>
</tr>
<tr>
<td>Optional participation</td>
<td>Define a foreign key for relation associated with mandatory participation</td>
</tr>
<tr>
<td>Both entities optional</td>
<td>Your choice for representation (either can have FK)</td>
</tr>
<tr>
<td>Multi-valued attributes</td>
<td>Define a relation for the multi-valued attribute and create a foreign key to the relation representing the containing entity</td>
</tr>
</tbody>
</table>
## Superclass/Subclass conversion

<table>
<thead>
<tr>
<th>Participation</th>
<th>Disjoint constraint</th>
<th>Mapping to logical design</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mandatory</td>
<td>Nondisjoint (AND)</td>
<td>Single relation with 1 or more attributes acting as a discriminator for the subclasses</td>
</tr>
<tr>
<td>Mandatory</td>
<td>Disjoint (OR)</td>
<td>Many relations one for each subclass/superclass combination</td>
</tr>
<tr>
<td>Optional</td>
<td>Nondisjoint (AND)</td>
<td>Two relations, 1 relation for the superclass and 1 relation for all of the subclasses, subclass needs a discriminating attribute to differentiate type of subclass</td>
</tr>
<tr>
<td>Optional</td>
<td>Disjoint (OR)</td>
<td>Many relations, one relation for the superclass, one relation for each subclass</td>
</tr>
</tbody>
</table>
Classwork: create relations for UML
Normalization is covered

In a separate presentation.
Check integrity constraints

Types of integrity constraints

1. **Identifying attributes that are required**
   - a. For each column decide if it needs to have a value

2. **Attribute domain constraints**
   - a. List or describe the legal values for each attribute (NULL allowed?)

3. **Multiplicity**
   - a. Ensure the relationship constraints are properly represented

4. **Entity integrity**
   - a. Primary key attributes cannot hold a NULL value

5. **Referential integrity**
   - a. Foreign key created in the child tuple linking to existing parent tuple

6. **General constraints**
Referential integrity defines DB behavior

Define the desired database behavior to ensure that a child relation NEVER references a parent relation instance that does not exist.

Review changes to the child relation.

1. CREATE a new record in the child relation
   a. If all foreign key attributes are NULL (no check by DB).
   b. If not NULL ensure parent tuple exists

2. UPDATE a foreign key attribute in the child relation
   a. Same as above

3. DELETE a record from the child relation
   a. Operation cannot violate referential integrity (no check by DB).
Referential integrity defines DB behavior

Review changes to the parent relation.

1. UPDATE a primary key attribute in the relation
   a. Identify the child tuples in the other table referencing this instance
   b. May choose to not allow update (ON UPDATE RESTRICT)
   c. May choose to allow UPDATE to parent relation to propagate to child (ON UPDATE CASCADE)
   d. May choose to remove the link between the 2 entities (ON UPDATE SET NULL or ON UPDATE SET DEFAULT)

2. DELETE a record from the parent relation
   a. Same as above except DELETE as oppose to UPDATE

3. CREATE a record in the parent relation
   a. No check to be done
Classwork: convert to a logical db design
Summary (Logical Design)

The logical design must represent all entities and relationships within the conceptual design.

For the relational model, we have relations and foreign keys to represent the logical design. All entities, attributes and relationships must be represented by these relations and foreign keys.
Step 3: Physical database design

Once the logical model is created, we need to choose the targeted database management system and determine the best method for physically implementing the logical model.

Describes the base relations, file organizations, and indexes and any associated integrity constraints and security measures.

Each vendor provides different data storage mechanisms, constraint representation, etc...
Steps to the physical database design

1. Define base relations for the chosen DBMS
2. Design representation of derived data
3. Design general constraints for the table
   a. Not enforced in MySQL but may be provided
4. Design file organizations and indexes
   a. MySQL uses the InnoDB structure a version of B trees
   b. Indexes: storage mechanism used to speed up data retrieval
   c. We will study indexes and B trees later in the semester
   d. Estimate disk space requirements
5. Design user views
6. Design security mechanism
Design base relations

1. Specify unique name for each relation
2. Specify list of simple attributes and domains, default values, NULLs permitted
3. Specify primary key and foreign keys
4. Specify referential integrity constraints
Design representation of derived data

1. Ensure data needed to derive field is present
2. If data is difficult to derive, some designers may choose to store the derived field as a simple field and define rules for it to be updated
Summary (Physical Design)

The database schema is created, including support for the derived fields.

The file organizations, and indexes and any associated integrity constraints and security measures are chosen.

User views and security constraints are chosen.