Database Design with Entity Relationship Model

Vijay Kumar
SICE, Computer Networking
University of Missouri-Kansas City
Kansas City, MO
kumarv@umkc.edu
Database Design Process

Database design process integrates relevant data in such a manner that it can be processed through a mechanism for recording the facts. A database of an organization is an information repository and represents facts about the organization. It is manipulated by some software to incorporate the changes that take place in the organization.

The database design is a complex process. The complexity arises mainly because the identification of relationships among individual components and their representation for maintaining correct functionality are highly involved. The degree of complexity increases if there are many to many relationships among individual components. The process of identification usually requires a number of steps, which can be presented as follows:

![Diagram of database design process]

The presentation of the facts involves two items (a) data (b) operations on the data. This defines two approaches for designing a database for an organization (a) data driven and (b) function driven.

**Data Driven approach**

Relevant data is analyzed and necessary schemas are constructed. Applications for processing the data are then developed. The following diagram illustrates the data driven approach.

![Data driven diagram]

**Function Driven approach**
Necessary functions are first designed and programmed for manipulating information. These applications are then implemented. The following diagram illustrates the function driven approach.

![Diagram]

Function driven approach

Both approaches are complementary to each other and in reality both are utilized.

**Schema Design**

A database schema of an organization is a description of the structure of the database in terms of *relationship types* and *entity types*. A database schema can be created (described) in any natural language. Such description is useful to understand the data processing requirements of the organization. An organization is regarded from database viewpoint as a set of functions and objects on to which these functions are applied. For example, the following description of an organization (university) explains its data processing requirements of a specific portion. It is also a schema and it is commonly referred to as *External schema*.

**University.**

*University is an organization. One of its activities is to grant different categories of degrees to its students. To achieve this the university stores relevant data about its student, staff and faculty members. For students it stores, data about their last name, age, sex, place and state of birth, city and state of residence of their family, places and states where they lived before, courses that they have passed, with name, code, professor, grade and date. It also stores information about courses they are presently taking, and for each day, places and hours where classes are held (e.g., the database) course meets at most once every Monday). For graduate students it stores the name of the person who advises and the total number of credits in the last year. For Ph. D. students it stores the title and the research area of their thesis. For teachers it stores the last name, age, place and state of birth, name of the department they work in, telephone number, title, status, and topic of their research are stored.*

The above external schema describes one of the activities of the university in an informal manner. It can be further refined (made more precise and less ambiguous) by doing the following.
Step1: Choose appropriate terms.
   Ex: For places use cities. For period use number of years, etc.
Step2: Avoid using instances instead of types.
   Ex: For Monday use once in a weekday.
Step3: Avoid roundabout expression.
   Ex: For name of the person who advises use advisor.
Step4: Check synonyms and homonyms.
   Ex: Synonyms teacher, professors, etc. Ex: Homonyms places used with different meaning.
Step5: Make explicit references among terms.
   Ex: It is not clear if telephone number is a property of the professor or of the department.
Step6: Prepare and use a glossary of terms.

The description of the University after refinement

In a university database, data about students and professors is represented. For students, we represent, last name, age, sex, city and state of birth, city and state of residence of their family, course where they lived before, courses that they have passed, with name, code, professor, grade and date. We also represent courses they are attending in the current year, and for each day of the week, rooms and hours where classes are held (each course meets at most once in one day). For graduate students, we represent the name of the advisor and the total number of credits in the last year. For Ph. D. students, we represent the title and the research area of their thesis. For professors we represent, last name, age, place and state of birth, name of the department they work in, telephone number of the department, title, marital status, and research area.

Partitioning of sentences into homogeneous groups

General sentence: In a university database, data about students and professors is represented.

Sentences on students: For students, we represent, last name, age, sex, city and state of birth, city and state of residence of their family, City and States where they lived before, courses that they have passed, with name, code, professor, grade and date

Sentences on courses: We also represent courses they are attending in the current year, and for each day Of the week, Rooms and hours where classes are held (each course meets at most once in One day).

Sentences on specific types of students: For graduate students, we represent the name of Advisor and the total number of credits in the last year. For Ph. D. students, we represent the title and the research area of their thesis

Sentences on professors: For Professors we represent, last name, age, place and state of birth, name of the department they work in, telephone number Of the department, title, Marital status, and research Area.
The refined *external schema*, which is also referred to as “*External view*” is less ambiguous, thus more precise.

<table>
<thead>
<tr>
<th>Input</th>
<th>Analysis and Refinement</th>
<th>Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>- Informal description of application</td>
<td></td>
<td></td>
</tr>
<tr>
<td>- Narratives, memo, etc.</td>
<td>Analysis and Refinement</td>
<td></td>
</tr>
<tr>
<td>- Verbal description</td>
<td></td>
<td>External schema</td>
</tr>
</tbody>
</table>

An external schema presents a very high level picture of the organization. It just indicates but does not illustrate the relationship among different components, entities, etc. It is not easy to understand from the external schema, how two or more entities interact with each other and what is the degree of their relationship, i.e., two or more entities are related together with a relationship type. It is essential to have this information incorporated in the database design if it has to be processed with any application. Thus, the external schema must be enhanced and converted to the next detailed schema, which is referred to as *conceptual schema*.

A conceptual schema exhibits hidden semantics of the database. It uses a semantic data model called *Entity-Relationship Model (E-R model)* to achieve this. First we discuss essential terms and concepts of E-R model.

**Database modeling concepts**

**Abstraction**: A mental process for extracting a subset of relevant characteristics and suppressing irrelevant characteristics of a set of objects. There are three types of abstraction (a) *Classification*, (b) *Aggregation*, and (c) *Generalization*

**Classification**

Classification abstraction defines a class of objects with a common set of properties. It collects similar things into a class. Example: class month = {January, February, . . .}. January, February, etc., are members of class *month*. In other words, January, February, etc., are classified as *month*. The arc represents the relationship: IS_MEMBER_OF. Thus, we say *January IS_MEMBER_OF class month*.

**Aggregation**

Aggregation abstraction defines a new class from a set of classes, which are identified as the component of the root class. Example: root class Bicycle and the components of the root class are {Wheel, Pedal, . . .}. The arc represents: IS_PART_OF. Thus, we say *Wheel IS_PART_OF Bicycle*. 

Classification and Aggregation

Classification and Aggregation can be combined to model a number of situations. We illustrate this with the following example. Example: root class Person. Name, Sex, and Position are aggregated into Person. Smith, Place and Prasad are classified into the class Name. Similarly Male and Female are classified into Sex, and Manager and Director is classified into Position. The following figure illustrates the construct.

Generalization

- Defines a subset relationship between two or more classes.
- Establishes a mapping from the generic class to the subset classes.

Example: A class Person is a generalization of two subclasses Man and Woman. Note that it is not a classification as discussed above. We aggregated Name, Sex, and Position to Person. We identified it with IS-MADE-OF. In generalization on the other hand, we say Man IS-A Person. The generalization of Man and Woman is represented as follows.

Types of Abstraction

We have the following types of abstraction
- Aggregation: IS-MADE-OF
- Generalization: IS-A, IS-LIKE
- Classification: IS-A-MEMBER-OF
- Composition (similar to aggregation): IS-MADE-OF
- Identification: IS-IDENTIFIED-BY

Coverage Constraints: Total Coverage, Partial Coverage, Exclusive Coverage and Overlapping Coverage.

Total coverage: The coverage of generalization is total (t) if each element of the generic class is
mapped to at least one element of a subclass. Example: A manager is an employee.

**Partial coverage:** There exists one element of a generic class that cannot be mapped to any element of a subclass. In the following figure an element of Person cannot be mapped to Man or Employee if Women are not allowed to work in a department.

**Exclusive coverage:** If an element of the generic class is mapped to at most one element of a subset class. In the following example a vehicle can be mapped to at most one subclass (i.e., either car or bicycle).

**Overlapping coverage:** At least one element of the generic class, which can be mapped to two or more subset classes.

**Generalization hierarchy for the entity Person**

**Database modeling concepts**

**Binary Aggregation:** It is a mapping established between two classes.
Uses establishes relationship between Person and Building. Similarly Owns establishes relationship between Person and Building.

**Constraints**

**Cardinality:** It is quantification of participation of an object in a relationship.

**Minimal cardinality (min-card):** Let A be the aggregation or relationship in which classes C1 and C2 participate. Then min-card is a constraint of this aggregation (relationship). Thus min-card (C1, A) and min-card (C2, A) are the minimum number of mappings in which each element of C1 and C2 must participate respectively. Consider aggregation USES (PERSON, BUILDING). If each person uses at least one building, then min-card (person, uses) = 1. If some building is not inhabited, then min-card (building, uses) = 0.

Now consider another aggregation OWNS (PERSON, BUILDING). If some person does not own a building, then min-card (person, owns) = 0. If a building must be owned by a person, then min-card (building, owns) = 1.

If min-card (C1, A) = 0, then C1 has an optional participation in the aggregation. If min-card (C1, A) = 1, then C1 has a mandatory participation in the aggregation. Thus, min-card constraint is also known as participation constraint.

**Maximal cardinality (max-card):** Let A be an aggregation among classes C1 and C2. max-card (C1, A) and max-card (C2, A) are the maximum number of mappings in which each element of C1 and C2 may participate respectively.

**For USES aggregation**
- If each person uses many buildings, then max-card (person, uses) = many
- If a building is inhabited by many, then max-card (building, uses) = many

**For OWNS aggregation**
- If a person own many buildings, then max-card (person, owns) = many
- If a building is owned exactly by one person, then max (building, owns) = 1, and min-card (building, owns) = 1
Aggregations are classified into four types based on max cardinality:

max-card (C1, A) = 1 and max-card (C2, A) = 1, then aggregation C1 to C2 is **one-to-one**
max-card (C1, A) = n and max-card (C2, A) = 1, then aggregation C1 to C2 is **one-to-many**
max-card (C1, A) = 1 and max-card (C2, A) = n, then aggregation C1 to C2 is **many-to-one**
max-card (C1, A) = n and max-card (C2, A) = n, then aggregation C1 to C2 is **many-to-many**

### The Entity-Relationship Model

**Basic Elements**

- **Entity (strong entity)**
- **Weak Entity**
- **Relationship type**
- **Identifying Relationship type**
- **Attribute**
- **Key attribute**
- **Multivalue attribute**
- **Composit attribute**

**Cardinality ratio** 1:n for E1:E2 in R

**Structural constraint (min, max) on participation of E in R**

<table>
<thead>
<tr>
<th>Entity:</th>
<th>A real or an abstract object. Ex: Person, house, etc.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Relation:</td>
<td>Aggregation of two or more entities.</td>
</tr>
<tr>
<td>Recursive relationship:</td>
<td>Binary relationship connecting an entity to itself.</td>
</tr>
<tr>
<td>Attribute</td>
<td>Identifies elementary properties of an entity or of a relationship.</td>
</tr>
<tr>
<td>Composit Attribute</td>
<td>A set of smaller atomic attributes, which are combined together to form a composite attribute. Example: color. The color of the same car can take different values (i.e., red, blue, etc.).</td>
</tr>
<tr>
<td>Single valued attribute:</td>
<td>Takes only one value. Example: name, account number, etc.</td>
</tr>
</tbody>
</table>
**Multivalued attribute:** May take more than one value. Example: degree, job, title, etc.

Attributes are characterized by minimal and maximal cardinality. Suppose $A$ is an attribute of an entity $E$. $\text{min-card}(A, E) = 0$ means the value of $A$ of $E$ can be ignored or may be null in processing instances of $E$. On the other hand, if $\text{min-card}(A, E) = 1$, then the value of $A$ must be specified in processing instance of $E$. If $\text{max-card}(A, E) = 1$, then $A$ is a single value attribute and if $\text{max-card}(A, E) > 1$, then $A$ is a multivalue attribute.

**Binary relationship**

**Ternary relationship**

**Recursive relationship**

**Example E-R Schema**

**Schema transformation:** A schema can go through a number of transformations before the final scheme is reached. These transformations are achieved through a number of primitives. We only discuss top-down primitives.
Top-down primitives: A simple structure. The starting schema is a single concept, and the resulting schema is a set of concepts. All names are refined into new names, and the logical connection is inherited by a single concept of the resulting schema.

Top-down schema transformation primitives

<table>
<thead>
<tr>
<th>Primitive</th>
<th>Starting schema</th>
<th>Resulting schema</th>
</tr>
</thead>
<tbody>
<tr>
<td>T1: Entity → Related entities</td>
<td></td>
<td></td>
</tr>
<tr>
<td>T2: Entity → Generalization</td>
<td></td>
<td></td>
</tr>
<tr>
<td>T3: Entity → Uncorelated entities</td>
<td></td>
<td></td>
</tr>
<tr>
<td>T4: Relationship → Parallel</td>
<td></td>
<td></td>
</tr>
<tr>
<td>T5: Relationship → Entity with</td>
<td></td>
<td></td>
</tr>
<tr>
<td>T6: Attribute development</td>
<td></td>
<td></td>
</tr>
<tr>
<td>T7: Composit attribute development</td>
<td></td>
<td></td>
</tr>
<tr>
<td>T8: Attribute refinement</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Top-down schema transformation primitives description

T1: Refines an entity into a relationship between two or more entities. Example: Place into City and State.

T2: Refines an entity into a generalization hierarchy or a subset. Example: Person is refined into Male and Female.
T3: Splits an entity type into a set of independent entities. The effect of this primitive is to introduce new entities and not to establish relationship or generalization among them. Example: Award is split into two entities Nobel_prize and Oscar.

T4: Refines a relationship into two (or more) relationships among the same entities. Example: Relationship related to between Person and City is refined into Lives_in and Born_in.

T5: Refines a relationship into entities and relationships. Applying this primitive corresponds to recognizing that a relationship between two concepts should be expressed via a third concept, which was hidden in the previous representation. Example: Relationship Works_in between Employee and Department is refined into a more complex aggregation that includes the entity Manager and two new relationships.

T6: Refines an entity (or a relationship) by introducing its attributes. Example: Attributes Name and Age are generated for entity Person.

T7: Refines an entity (or a relationship) by introducing composite attributes. Example: Composite attribute Address is generated for Person.
T8: Refines a simple attribute into either a composite or into a group of attributes. Example: Attribute *Date* is refined into a composite attribute *Day, Month, and Year*. The attribute *Health Data* is refined in terms of the attributes *Health State* and *Date_Last_Vaccination*.

**Criteria for choosing the right modeling constructs**

**First example**
Q: *Should a real world object be modeled as an entity or as an attribute?*

A: Object should be an entity if a number of attributes could be associated with it for proper identification and description, either now or later. Object should be an attribute, if it has an atomic nature. For example, *Color* should be an attribute, unless we identify Color either as a process (e.g., painting) where a number of attributes codes are to be recorded (e.g., type, shade, gray-scale, manufacturer, or as an object with properties (e.g., car-color with details).

Q: *Should generalization be used or an attribute?*

A: Generalization should be used if later in the design some attributes can be associated with lower level entities but not with the generic entity. For example: in generalization of person, attribute 'maiden name' can be associated with entity Woman. If no attribute is likely to be associated with any of the lower level entity then an attribute should be chosen.

Q: *Should a composite attribute or a set of attributes be chosen?*

A: Composite attribute is chosen when a meaningful name can be assigned to the set of attributes, e.g., data, address. Otherwise a set of simple attributes should be chosen.