Ch 8: Enhanced Entity-Relationship (EE_R) Diagram

The ER modeling concepts discussed in Chapter 7 are sufficient for representing many database schemas for traditional database applications. Since the 1970's, need was felt for database schemas that reflect the data properties and constraints more precisely. This was particularly important for newer applications of database technology, such as databases for engineering design and manufacturing (CAD/CAM). Thus the notion of Enhanced Entity Relationship (EER) diagram was introduced.

Various semantic data models have been proposed in literature. Many of these concepts were also developed independently in related areas of computer science, such as the knowledge representation area of artificial intelligence and the object modeling/programming area in software engineering.

Subclasses, Superclasses, and Inheritance:
Subclass/Subtype: Employee is a Secretary, Employee is a Technician, Employee is an Engineer, etc.
Superclass or Supertype: Employee entity is the superclass of Secretary, technician, etc. subclasses. EMPLOYEE/SECRETARY and EMPLOYEE/TECHNICIAN are two class/subclass relationships (IS-A).
When we implement a superclass/subclass relationship in the database system, however, we may represent a member of the subclass as a distinct database object, say, a distinct record that is related via the key attribute to its superclass entity.
Type inheritance: member entity of a subclass inherits all the attributes of the entity of the superclass.

![EER diagram notation to represent subclasses and specialization.](image)

Three specializations of EMPLOYEE:
(SECRETARY, TECHNICIAN, ENGINEER)
(MANAGER)
(HOURLY_EMPLOYEE, SALARIED_EMPLOYEE)

\[\text{A class/subclass relationship is often called an IS-A (or IS-AN) relationship because of the way we refer to the concept. We say a SECRETARY is an EMPLOYEE, a TECHNICIAN is an EMPLOYEE, and so on.}\]
Specialization and Generalization:

Specialization

Specialization is the process of defining a set of subclasses of an entity type; this entity type is called the superclass of the specialization. The set of subclasses that forms a specialization is defined on the basis of some distinguishing characteristic of the entities in the superclass. For example, the set of subclasses (SECRETARY, ENGINEER, TECHNICIAN) is a specialization of the superclass EMPLOYEE that distinguishes among employee entities based on the job type of each employee entity.

Figure 8.2

Instances of a specialization.

Generalization

We can think of a reverse process of abstraction in which we suppress the differences among several entity types, identify their common features, and generalize them into a single superclass of which the original entity types are special subclasses. For example, consider the entity types CAR and TRUCK shown in Figure 8.3(a). Because they have several common attributes, they can be generalized into the entity type VEHICLE, as shown in Figure 8.3(b). Both CAR and TRUCK are now subclasses of the generalized superclass VEHICLE. We use the term generalization to refer to the process of defining a generalized entity type from the given entity types.

Another concept included in the EER model is that of a category or union type, which is used to represent a collection of objects (entities) that is the union of objects of different entity types.

Predicate-defined Subclass: Spuerclass attribute determines subclasses

User-defined: membership is specified individually for each entity by the user, not by any condition
Specific attributes (or local attributes): Subclass attributes

Constraints:

1. **Disjointness** (or Disjointedness) constraint means that an entity can be a member of at most one of the subclasses of the specialization. A member of Car entity set cannot belong to Truck entity set, too. 
   
   **Overlapping:** entity may be a member of more than one subclass of the specialization.

2. **Total/partial specialization** constraint: EMPLOYEE must be either an HOUURLY_EMPLOYEE or a SALARIED_EMPLOYEE vs. an Employee does not have to belong to Secretary, Technician, etc.

![Diagram](image)

**Figure 8.3**
Generalization. (a) Two entity types, CAR and TRUCK. (b) Generalizing CAR and TRUCK into the superclass VEHICLE.

Specialization process allows us to do the following:

- Define a set of subclasses of an entity type
- Establish additional specific attributes with each subclass
- Establish additional specific relationship types between each subclass and other entity types or other subclasses

**Constraints and Characteristics of Specialization and Generalization Hierarchies—Some Rules**

- Deleting an entity from a superclass implies that it is automatically deleted from all the subclasses to which it belongs.
- Inserting an entity in a superclass implies that the entity is mandatorily inserted in all predicate-defined (or attribute-defined, e.g., job_type) subclasses for which the entity satisfies the defining predicate.
- Inserting an entity in a superclass of a total specialization implies that the entity is mandatorily inserted in at least one of the subclasses of the specialization.
The second constraint on specialization is called the completeness (or totalness) constraint, which may be total or partial. A total specialization constraint specifies that every entity in the superclass must be a member of at least one subclass in the specialization. For example, if every EMPLOYEE must be either an HOURLY_EMPLOYEE or a SALARIED_EMPLOYEE, then the specialization \{HOURLY_EMPLOYEE, SALARIED_EMPLOYEE\} in Figure 8.1 is a total specialization of EMPLOYEE. This is shown in EER diagrams by using a double line to connect the superclass to the circle. A single line is used to display a partial specialization, which allows an entity not to belong to any of the subclasses. For example, if some EMPLOYEE entities do not belong to any of the subclasses \{SECRETARY, ENGINEER, TECHNICIAN\} in Figures 8.1 and 8.4, then that specialization is partial.

Notice that the disjointness and completeness constraints are independent. Hence, we have the following four possible constraints on specialization:

- Disjoint, total
- Disjoint, partial
- Overlapping, total
- Overlapping, partial
Specialization and Generalization Hierarchies and Lattices:

A specialization hierarchy has the constraint that every subclass participates as a subclass in only one class/subclass relationship; that is, each subclass has only one parent, which results in a tree structure or strict hierarchy.

In a specialization lattice, a subclass can be a subclass in more than one class/subclass relationship.

Figure 8.6
A specialization lattice with shared subclass ENGINEERING_MANAGER.
Specialization Lattice with Multiple Inheritance/Two Categories (Union types)

**Figure 8.7**
A specialization lattice with multiple inheritance for a UNIVERSITY database.

**Figure 8.8**
Two categories (union types): OWNER and REGISTERED VEHICLE.
A Sample UNIVERSITY EER Schema, Design Choices, and Formal Definitions:
For our sample database application, consider a UNIVERSITY database that keeps track of students and their majors, transcripts, and registration as well as of the university’s course offerings. The database also keeps track of the sponsored research projects of faculty and graduate students.

Requirements that led to this schema follows:
For each person, the database maintains information on the person’s Name [Name], Social Security number [Ssn], address [Address], sex [Sex], and birth date [Bdate]. Two subclasses of the PERSON entity type are identified: FACULTY and STUDENT. Specific attributes of FACULTY are rank [Rank] (assistant, associate, adjunct, research, visiting, and so on), office [Ooffice], office phone [Ophone], and salary [Salary]. All faculty members are related to the academic department(s) with which they are affiliated [BELONGS] (a faculty member can be associated with several departments, so the relationship is M:N). A specific attribute of STUDENT is [Class] (freshman=1, sophomore=2,..., graduate student=5).

Each STUDENT is also related to his or her major and minor departments (if known) [MAJOR] and [MINOR], to the course sections he or she is currently attending [REGISTERED], and to the courses completed [TRANSCRIPT]. Each TRANSCRIPT instance includes the grade the student received [Grade] in a section of a course. GRAD STUDENT is a subclass of STUDENT, with the defining predicate Class = 5. For each graduate student, we keep a list of previous degrees in a composite, multivalued attribute [Degrees]. We also relate the graduate student to a faculty advisor [ADVISOR] and to a thesis committee [COMMITTEE], if one exists. An academic department has the attributes name [Dname], telephone [Dphone], and office number [Office] and is related to the faculty member who is its chairperson [CHAIRS] and to the college to which it belongs [CD]. Each college has attributes college name [Cname], office number [Coffice], and the name of its dean [Dean].

A course has attributes course number [C#], course name [Cname], and course description [Cdesc]. Several sections of each course are offered, with each section having the attributes section number [Sec#] and the year and quarter in which the section was offered ([Year] and [Qtr]). Section numbers uniquely identify each section. The sections being offered during the current quarter are in a subclass CURRENT SECTION of SECTION, with the defining predicate Qtr = Current_qtr and Year = Current_year. Each section is related to the instructor who taught or is teaching it ([TEACH]), if that instructor is in the database.

The category INSTRUCTOR RESEARCHER is a subset of the union of FACULTY and GRAD STUDENT and includes all faculty, as well as graduate students who are supported by teaching or research. Finally, the entity type GRANT keeps track of research grants and contracts awarded to the university. Each grant has attributes grant title [Title], grant number [No], the awarding agency [Agency], and the starting date [St_date].
Figure 8.9
An EER conceptual schema for a UNIVERSITY database.
Design Choices for Specialization/Generalization

It is not always easy to choose the most appropriate conceptual design for a database application. In the previous chapter, some of the typical issues that confront a database designer were presented when choosing among the concepts of entity types, relationship types, and attributes to represent a particular miniworld situation as an ER schema. In this section, we discuss design guidelines and choices for the EER concepts of specialization/generalization and categories (union types). As we mentioned in earlier, conceptual database design should be considered as an iterative refinement process until the most suitable design is reached. The following guidelines can help to guide the design process for EER concepts:

- In general, many specializations and subclasses can be defined to make the conceptual model accurate. However, the drawback is that the design becomes quite cluttered. It is important to represent only those subclasses that are deemed necessary to avoid extreme cluttering of the conceptual schema.

- If a subclass has few specific (local) attributes and no specific relationships, it can be merged into the superclass. The specific attributes would hold NULL values for entities that are not members of the subclass. A type attribute could specify whether an entity is a member of the subclass.

- Similarly, if all the subclasses of a specialization/generalization have few specific attributes and no specific relationships, they can be merged into the superclass and replaced with one or more type attributes that specify the subclass or subclasses that each entity belongs to (see Section 9.2 for how this criterion applies to relational databases).

- Union types and categories should generally be avoided unless the situation definitely warrants this type of construct, which does occur in some practical situations. If possible, we try to model using specialization/generalization as discussed at the end of Section 8.4.

- The choice of disjoint/overlapping and total/partial constraints on specialization/generalization is driven by the rules in the miniworld being modeled. If the requirements do not indicate any particular constraints, the default would generally be overlapping and partial, since this does not specify any restrictions on subclass membership.
Example of Other Notation: Representing Specialization and Generalization:

UML notation:

In the following example, a blank triangle indicates a specialization/generalization with the disjoint constraint, and a filled triangle indicates an overlapping constraint.

Superclass (base class), and the subclass (leaf nodes) are called leaf classes. Abstract classes define attributes and operations but do not have objects corresponding to those classes. Concrete classes can have objects (entities) instantiated to belong to the class (used in DBMS). Template classes specify a template that can be further used to define other classes.

Figure 8.10
A UML class diagram corresponding to the EER diagram in Figure 8.3, illustrating UML notation for specialization/generalization.
Ch 9: Relational Database Design by ER- and EER-to-Relational Mapping:

In this chapter discusses how to design a relational database schema based on a conceptual schema design. Figure 7.1 presented a high-level view of the database design process, and in this chapter we focus on the logical database design or data model mapping step of database design.

Figure 9.1
The ER conceptual schema diagram for the COMPANY database.
7-Step process: ER-to-Relational Mapping Algorithm

Step 1: Mapping of Regular Entity Types:

Create the relations EMPLOYEE, DEPARTMENT, and PROJECT in Figure 9.2 to correspond to the regular entity types EMPLOYEE, DEPARTMENT, and PROJECT in Figure 9.1. The foreign key and relationship attributes, if any, are not included yet.

Entity relations are next, using the ER diagram:

(a) EMPLOYEE

(b) DEPARTMENT

(c) PROJECT

(d) DEPENDENT

(e) WORKS ON

(f) DEPT LOCATIONS

Figure 9.3
Illustration of some mapping steps.
(a) Entity relations after step 1.
(b) Additional weak entity relation after step 2.
(c) Relationship relation after step 3.
(d) Relation representing multivalued attribute after step 5.
Step 2: Mapping of Weak Entity Types

Create the weak entity

Step 3: Mapping of Binary 1:1 Relationship Types

Foreign key approach

Step 4: Mapping of Binary 1:N Relationship Types

For each regular binary 1:N relationship type R, identify the relation S that represents the participating entity type at the N-side of the relationship type.

Step 5: Mapping of Binary M:N Relationship Types

For each binary M:N relationship type R, create a new relation S to represent R. Include as foreign key attributes in S the primary keys of the relations that represent the participating entity types; their combination will form the primary key of S.

Step 6: Mapping of Multivalued Attributes

For each multivalued attribute A, create a new relation R. This relation R will include an attribute corresponding to A, plus the primary key attribute K—as a foreign key in R—of the relation that represents the entity type or relationship type that has A as a multivalued attribute.

Step 7: Mapping of N-ary Relationship Types

For each n-ary relationship type R, where n > 2, create a new relation S to represent R. Include as foreign key attributes in S the primary keys of the relations that represent the participating entity types. Also include any simple attributes of the n-ary relationship type (or simple components of composite attributes) as attributes of S. The primary key of S is usually a combination of all the foreign keys that reference the relations representing the participating entity types.

Table 9.1 Correspondence between ER and Relational Models

<table>
<thead>
<tr>
<th>ERMODEL</th>
<th>RELATIONAL MODEL</th>
</tr>
</thead>
<tbody>
<tr>
<td>Entity type</td>
<td>Entity relation</td>
</tr>
<tr>
<td>1:1 or 1:N relationship type</td>
<td>Foreign key (or relationship relation)</td>
</tr>
<tr>
<td>M:N relationship type</td>
<td>Relationship relation and two foreign keys</td>
</tr>
<tr>
<td>n-ary relationship type</td>
<td>Relationship relation and n foreign keys</td>
</tr>
<tr>
<td>Simple attribute</td>
<td>Attribute</td>
</tr>
<tr>
<td>Composite attribute</td>
<td>Set of simple component attributes</td>
</tr>
<tr>
<td>Multivalued attribute</td>
<td>Relation and foreign key</td>
</tr>
<tr>
<td>Value set</td>
<td>Domain</td>
</tr>
<tr>
<td>Key attribute</td>
<td>Primary (or secondary) key</td>
</tr>
</tbody>
</table>
The relationship type SUPPLY in Figure 7.17, can be mapped to the relation SUPPLY shown in Figure 9.4, whose primary key is the combination of the three foreign keys \{Sname, Part_no, Proj_name\}.

**Figure 9.4**
Mapping the n-ary relationship type SUPPLY from Figure 7.17(a).

<table>
<thead>
<tr>
<th>Sname</th>
<th>Proj_name</th>
<th>Part_no</th>
<th>Quantity</th>
</tr>
</thead>
</table>

**Figure 9.5**
Options for mapping specialization or generalization. (a) Mapping the EER schema in Figure 8.4 using option 8A. (b) Mapping the EER schema in Figure 8.8(b) using option 8B. (c) Mapping the EER schema in Figure 8.4 using option 8C. (d) Mapping Figure 8.5 using option 8D with Boolean type fields Mflag and Pflag.