A Sample UNIVERSITY EER Schema, Design Choices, and Formal Definitions

A Sample UNIVERSITY EER Schema, Design Choices, and Formal Definitions

In this section, we first give an example of a database schema in the EER model to illustrate the use of the various concepts discussed here and in Chapter 7. Then, we discuss design choices for conceptual schemas, and finally we summarize the EER model concepts and define them formally in the same manner in which we formally defined the concepts of the basic ER model in Chapter 7.

1. The UNIVERSITY Database Example

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. This schema is shown in Figure 8.9. A discussion of the 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 [Foffice], office phone [Fphone], 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, multi-valued attribute [Degrees]. We also relate the graduate student to a faculty advisor [ADVISOR] and to a thesis committee [COMMITTEE], if one exists.

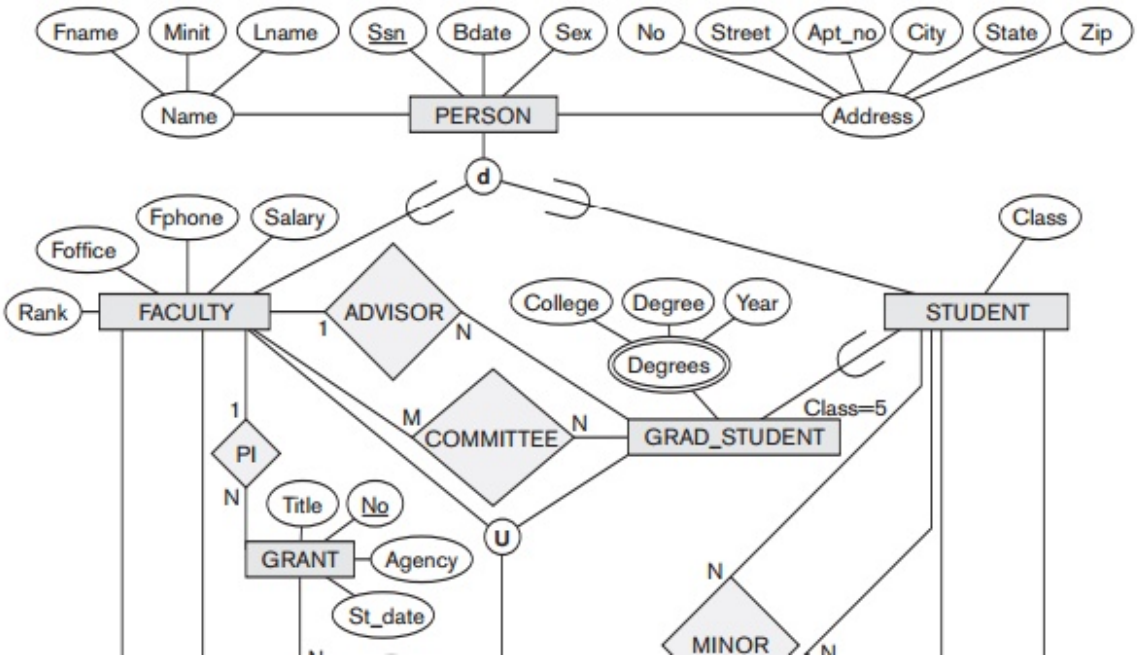
An academic department has the attributes name [Dname], telephone [Dphone], and office number [Office] and is related to the faculty member who is its chairperson [CHAIRS] and to the college to which it belongs [CD]. Each college has attributes college name [Cname], office number [Coffice], and the name 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]).10 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 teach-ing 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 sup-ported 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



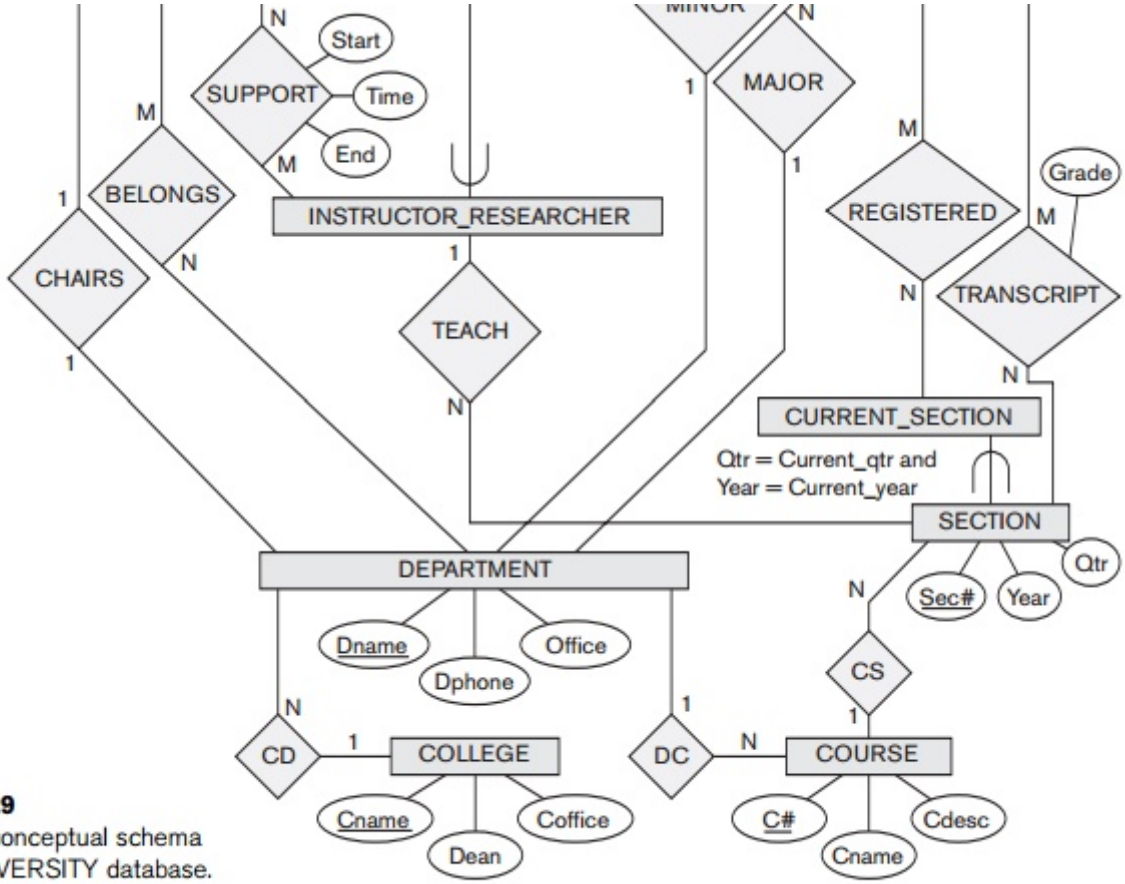


Figure 8.9
An EER conceptual schema for a UNIVERSITY database.

date [St_date]. A grant is related to one principal investigator [PI] and to all researchers it supports [SUPPORT]. Each instance of support has as attributes the starting date of support [Start], the ending date of the support (if known) [End], and the percentage of time being spent on the project [Time] by the researcher being supported.

2. Design Choices for Specialization/Generalization

It is not always easy to choose the most appropriate conceptual design for a database application. In Section 7.7.3, we presented some of the typical issues that confront a database designer 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 Section 7.7.3, conceptual database design should be considered as an iterative refinement process until the most suitable design is reached. The fol-lowing 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 sub-class 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.

As an example of applying these guidelines, consider Figure 8.6, where no specific (local) attributes are shown. We could merge all the subclasses into the EMPLOYEE entity type, and add the following attributes to EMPLOYEE:

An attribute Job_type whose value set {‘Secretary’, ‘Engineer’, ‘Technician’} would indicate which subclass in the first specialization each employee belongs to.

An attribute Pay_method whose value set {‘Salaried’, ‘Hourly’} would indicate which subclass in the second specialization each employee belongs to.

An attribute Is_a_manager whose value set {‘Yes’, ‘No’} would indicate whether an individual employee entity is a manager or not.

3. Formal Definitions for the EER Model Concepts

We now summarize the EER model concepts and give formal definitions. A **class** is a set or collection of entities; this includes any of the EER schema constructs of group entities, such as entity types, subclasses, superclasses, and categories. A **subclass** *S* is a class whose entities must always be a subset of the entities in another class, called the **superclass** *C* of the **superclass/subclass** (or **IS-A**) **relationship**. We denote such a relationship by *C/S.* For such a superclass/subclass relationship, we must always have

$S \subseteq C$

A **specialization** $Z = \{S_1, S_2, ..., S_n\}$ is a set of subclasses that have the same super-class *G;* that is, G/S_i is a superclass/subclass relationship for $i = 1, 2, ..., n$. *G* is called a **generalized entity type** (or the **superclass** of the

specialization, or a **generalization** of the subclasses $\{S_1, S_2, ..., S_n\}$). Z is said to be **total** if we always (at any point in time) have

$$\bigcup_{i=1}^{n} S_i = G$$

Otherwise, Z is said to be **partial**. Z is said to be **disjoint** if we always have

$S_i \cap S_j = \varnothing$ (empty set) for $i \neq j$

Otherwise, Z is said to be **overlapping**.

A subclass S of C is said to be **predicate-defined** if a predicate p on the attributes of C is used to specify which entities in C are members of S; that is, $S = C[p]$, where $C[p]$ is the set of entities in C that satisfy p. A subclass that is not defined by a predicate is called **user-defined**.

A specialization Z (or generalization G) is said to be **attribute-defined** if a predicate $(A = c_i)$, where A is an attribute of G and c_i is a constant value from the domain of A, is used to specify membership in each subclass Si in Z. Notice that if ci ≠ cj for i ≠ j, and A is a single-valued attribute, then the specialization will be disjoint.

A **category** T is a class that is a subset of the union of n defining superclasses $D_1, D_2, ..., D_n$, $n > 1$, and is formally specified as follows:

$T \subseteq (D_1 \cup D2 \ ... \cup D_n)$

A predicate p_i on the attributes of D_i can be used to specify the members of each D_i that are members of T. If a predicate is specified on every D_i, we get

$T = (D_1[p_1] \cup D2[p_2] \ ... \cup D_n[p_n])$

We should now extend the definition of **relationship type** given in Chapter 7 by allowing any class—not only any entity type—to participate in a relationship. Hence, we should replace the words *entity type* with

class in that definition. The graphical notation of EER is consistent with ER because all classes are represented by rectangles.

Prev Page (https://www.brainkart.com/article/Modeling-of-UNION-Types-Using-Categories_11441/)

e/Example-of-Other-Notation--Representing-Specialization-and-Generalization-in-UML-Class-Diagrams_11443/)

Prev Page (https://www.brainkart.com/article/Modeling-of-UNION-Types-Using-Categories_11441/)

e/Example-of-Other-Notation--Representing-Specialization-and-Generalization-in-UML-Class-Diagrams_11443/)

Related Topics

Database Management Systems (https://www.brainkart.com/subject/Database-Management-Systems_181/)

Anna University - All Subjects (https://www.brainkart.com/menu/anna-university/)

Anna University CSE - All Subjects (https://www.brainkart.com/menu/anna-university-cse/)

Anna University IT - All Subjects (https://www.brainkart.com/menu/anna-university-it/)

Engineering - All Subjects (https://www.brainkart.com/menu/engineering/)

Computer Science Engineering - All Subjects (https://www.brainkart.com/menu/computer-science-engineering/)

Fundamentals of Database Systems : Conceptual Modeling and Database Design : The Enhanced Entity-Relationship (EER) Model

The Enhanced Entity-Relationship (EER) Model (https://www.brainkart.com/article/The-Enhanced-Entity-Relationship-(EER)-Model_11437/)

Subclasses, Superclasses, and Inheritance (https://www.brainkart.com/article/Subclasses,-Superclasses,-and-Inheritance_11438/)

Specialization and Generalization (https://www.brainkart.com/article/Specialization-and-Generalization_11439/)

Constraints and Characteristics of Specialization and Generalization Hierarchies (https://www.brainkart.com/article/Constraints-and-Characteristics-of-Specialization-and-Generalization-Hierarchies_11440/)

Modeling of UNION Types Using Categories (https://www.brainkart.com/article/Modeling-of-UNION-Types-Using-Categories_11441/)

A Sample UNIVERSITY EER Schema, Design Choices, and Formal Definitions (https://www.brainkart.com/article/A-Sample-UNIVERSITY-EER-Schema,-Design-Choices,-and-Formal-Definitions_11442/)

Example of Other Notation: Representing Specialization and Generalization in UML Class Diagrams (https://www.brainkart.com/article/Example-of-Other-Notation--Representing-Specialization-and-Generalization-in-UML-Class-Diagrams_11443/)

Data Abstraction, Knowledge Representation, and Ontology Concepts (https://www.brainkart.com/article/Data-Abstraction,-Knowledge-Representation,-and-Ontology-Concepts_11444/)

(https://www.gamezop.com/?id=4953)

(https://www.gamezop.com/?id=4953)


(https://www.gamezop.com/?id=4953)

(https://4954.play.quizzop.com/)

(https://4954.play.quizzop.com/)

(https://4954.play.quizzop.com/)

(https://4954.play.quizzop.com/)

(https://4954.play.quizzop.com/)

(https://4954.play.quizzop.com/)