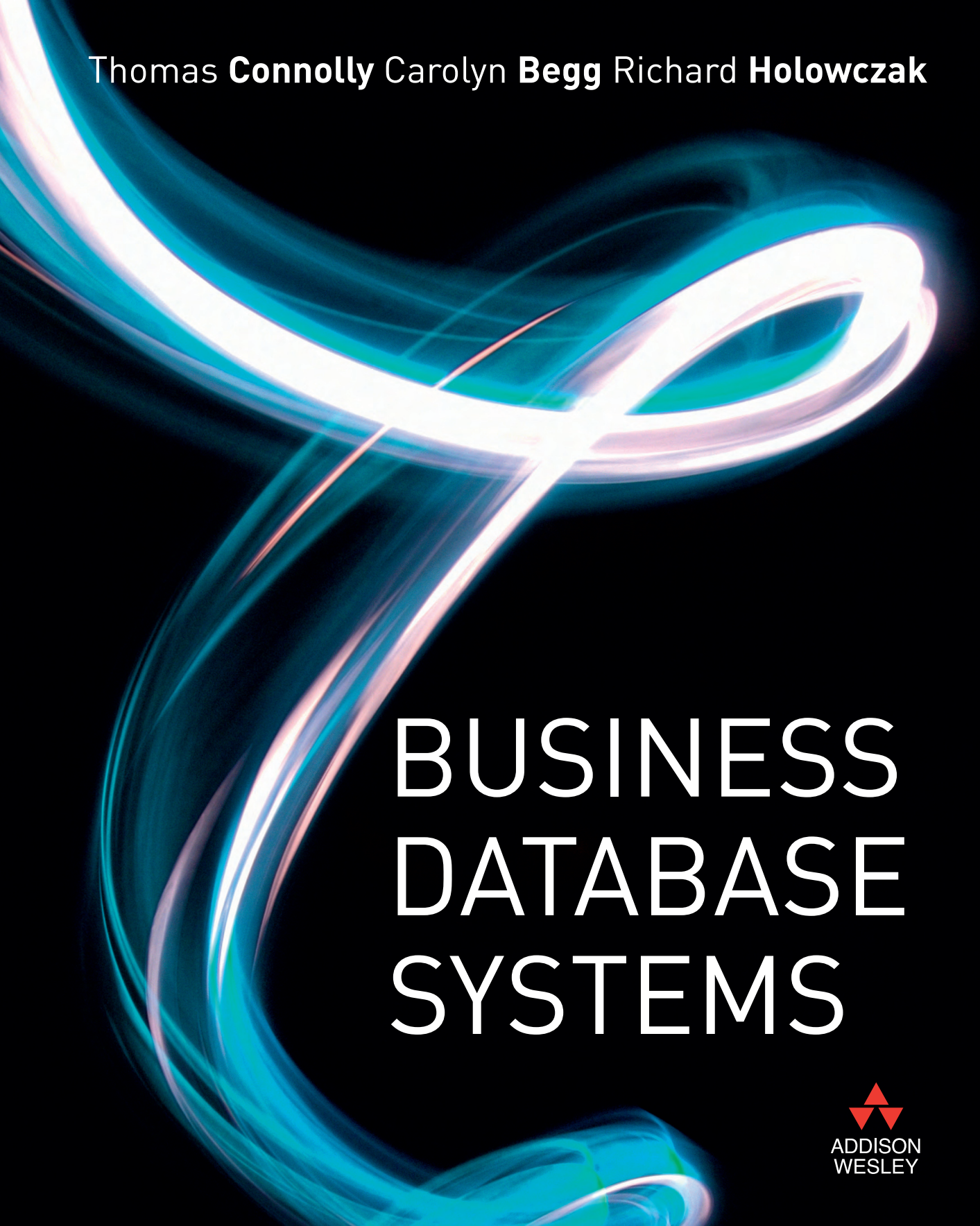
Thomas **Connolly** Carolyn **Begg** Richard **Holowczak**



BUSINESS DATABASE SYSTEMS

ADDISON WESLEY

BUSINESS DATABASE SYSTEMS

Visit the *Business Database Systems* Companion Website at **www.pearsoned.co.uk/connolly** to find valuable **student** learning material including:

- Lecture slides
- An implementation of the *StayHome Online Rentals* database system in Microsoft Access®
- An SQL script for each common data model described in Appendix I to create the corresponding set of base tables for the database system
- An SQL script to create an implementation of the *Perfect Pets* database system

Exercises

6.10 Create an ER model for each of the following descriptions:

(a) Each company operates four departments, and each department belongs to one company. Each company has a unique name, and each department has a unique number and name.

(b) Each department in part (a) employs one or more employees, and each employee works for one department. Each employee has a number, name (including first and last name), date of birth, and age.

(c) Each of the employees in part (b) may or may not have one or more dependants, and each dependant belongs to one employee. Each dependant has a name, relationship to employee, and contact telephone numbers up to a maximum of three.

(d) Each employee in part (c) may or may not have an employment history. Each employment history has the name of the organization that the employee worked for and in what capacity, the start date and finish date for each employment.

(e) Represent all the ER models described in (a), (b), (c), and (d) as a single ER model. Provide any assumptions necessary to support your model.

The final answer to this exercise is shown as Figure 7.5 of Exercise 7.8.

6.11 Create an ER model for each of the following descriptions:

(a) A large organization has several car parks, which are used by staff.

(b) Each car park has a unique name, location, capacity, and number of floors (where appropriate).

(c) Each car park has car parking spaces, which are uniquely identified using a space number.

(d) Members of staff can request the use of a car parking space. Each member of staff has a unique number, name, telephone extension number, and vehicle license number.

(e) Represent all the ER models described in (a), (b), (c), and (d) as a single ER model. Provide any assumptions necessary to support your model.

The final answer to this exercise is shown as Figure 7.6 of Exercise 7.9.

6.12 Create an ER model for each of the following descriptions:

(a) A sailing club has members. Each member has a unique member number, name, date of birth, and gender.

(b) Most members own at least one dinghy. However, a dinghy can be owned by more than one member. Each dinghy has a boat name, sail number, and boat class (such as Topper, Mirror, Contender). (Hint: The sail number uniquely identifies each boat in the same class.)

(c) The sailing club offers a range of membership types including adult, child, and a non-sailing social membership. Each membership type has an annual subscription rate, which runs for a year from 1st March. The date that each member pays his or her annual membership is recorded.

(d) The sailing club also owns dinghies, which can be borrowed by members for a daily fee.

(e) Represent the ER models described in (a), (b), (c), and (d) as a single ER model. Provide any assumptions necessary to support your model.

CHAPTER

7

Enhanced ER modeling

Preview

We covered the basic concepts associated with entity–relationship (ER) modeling in Chapter 6. These basic concepts are often perfectly adequate for the representation of the data requirements for many different database systems. However, the basic ER concepts can be limiting for more complex database systems that require modeling with either a large amount of data and/or data with complex interrelationships. This stimulated the need to develop additional 'semantic' modeling concepts. The original ER model with additional semantic concepts is referred to as the **enhanced entity–relationship (EER)** model. In this chapter, we describe some of the more useful concepts associated with the EER model called specialization/generalization and show how these concepts can be applied.

The database design methodology presented in Appendix D provides an option to use the enhanced concepts of the EER model in Step 1.6. The choice of whether to include this step is largely dependent on whether the designer considers that using these enhanced modeling concepts facilitates or hinders the process of database design.

Throughout this chapter we use, adapt, and or extend examples taken from the *StayHome Online Rentals* case study described in Section 5.4. We also use examples not included in this case study to allow discussion on a particular topic.

Learning objectives

In this chapter you will learn:

- The limitations of the basic ER modeling concepts and the requirements to model more complex applications using enhanced data modeling concepts.
- The main concepts associated with the enhanced entity–relationship (EER) model called specialization/generalization.
- A notation for displaying specialization/generalization in an EER diagram using UML.

7.1 Specialization/generalization

The concept of specialization/generalization is associated with special types of entities known as superclasses and subclasses, and the process of **attribute inheritance**. We begin this section by defining what superclasses and subclasses are and by examining superclass/subclass relationships. We describe the process of attribute inheritance and contrast the process of specialization with generalization. We also show how to represent specialization/generalization in a diagram using the UML (Unified Modeling Language) notation.

7.1.1 Superclasses and subclasses

Superclass

An entity that includes one or more distinct groupings of its occurrences, which require to be represented in a data model.

A general entity called a **superclass** includes groupings of more specific kinds of entities called **subclasses**. For example, an entity that may have many distinct subclasses is `Staff`. The entities that are members of the `Staff` entity may be classified as `Manager` and `Assistant`. In other words, the `Staff` entity is the superclass of the `Manager` and `Assistant` subclasses.

7.1.2 Superclass/subclass relationships

Subclass

A distinct grouping of occurrences of an entity, which require to be represented in a data model.

The relationship between a superclass and any one of its subclasses is one-to-one (1:1) and is called a superclass/subclass relationship. For example, `Staff/Manager` forms a superclass/subclass relationship. Each member of a subclass is also a member of the superclass but has a distinct role.

Superclasses and subclasses can be used to avoid describing different types of entities with possibly different attributes within a single entity. For example, `Manager` may have special attributes such as `mgrStartDate` and `bonus`, and so on. If all staff attributes and those specific to particular jobs are represented by a single `Staff` entity, this may result in a lot of nulls for the job specific attributes. Clearly, managers have common attributes with other staff, such as `staffNo`, `name`, `position`, and `salary`, but it is the unshared attributes that cause problems when trying to represent all members of staff within a single entity. Defining superclasses/subclasses can also show relationships that are associated only with particular subclasses of staff and not with staff in general. For example, managers may have distinct relationships that are not appropriate for all staff, such as `Manager` *`Requires`* `Car`.

To illustrate some of the points being made above, consider the table called `AllStaff` in Figure 7.1. The table holds the details of all members of staff no matter what position they hold. A consequence of holding the details of all members of staff in one table is that while the columns appropriate to all staff are filled (namely, `staffNo`, `name`, `position`, `salary`, and `dCenterNo`), those that are only applicable to particular job roles will be only partially filled. (Note that a column called `eMail`, which is associated with all staff of the *StayHome Online Rental* case study is not included in this example.) For example, the columns associated with the `Manager` subclass (namely `mgrStartDate` and `bonus`) have no values for those members of staff holding the position of Assistant.

Columns appropriate for ALL staff

Columns appropriate for Managers

Column appropriate for Assistants

AllStaff

staffNo	name	position	salary	dCenterNo	mgrStartDate	mgrBonus	shift
S1500	Tom Daniels	Manager	48000	D001	1/11/06	2500	
S0003	Sally Adams	Assistant	30000	D001			Early
S0010	Mary Martinez	Manager	51000	D002	1/01/06	5000	
S3250	Robert Chin	Assistant	33000	D002			Late
S2250	Sally Stern	Manager	48000	D004	01/08/06	2650	
S0415	Art Peters	Manager	42000	D003	01/08/06	3250	

Figure 7.1 The `AllStaff` table holding details of all members of staff

Note There are two important reasons for introducing the concepts of superclasses and subclasses into an ER model. The first reason is that it avoids describing similar concepts more than once, thereby saving time and making the ER model more readable. The second reason is that it adds more semantic information to the design in a form that is familiar to many people. For example, the assertions that 'Manager IS-A member of staff' and 'van IS-A type of vehicle' communicate significant semantic content in an easy-to-follow form.

7.1.3 Attribute inheritance

As mentioned above, an entity occurrence in a subclass represents the same 'real world' object as in the superclass. Hence, a member of a subclass inherits those attributes associated with the superclass, but may also have subclass-specific attributes. For example, a member of the `Manager` subclass has subclass-specific attributes, `mgrStartDate`, and `bonus`, and all the attributes of the `Staff` superclass, namely `staffNo`, `name`, `position`, `salary`, and `dCenterNo`.

A subclass is an entity in its own right and so it may also have one or more subclasses. A subclass with more than one superclass is called a *shared subclass*. In other words, a member of a shared subclass must be a member of the associated superclasses. As a consequence, the attributes of the superclasses are inherited by the shared subclass, which may also have its own additional attributes. This process is referred to as *multiple inheritance*.

Note An entity and its subclasses and their subclasses, and so on, is called a **type hierarchy**. Type hierarchies are known by a variety of names including **specialization hierarchy** (for example, `Manager` is a specialization of `Staff`), **generalization hierarchy** (for example, `Staff` is a generalization of `Manager`), and **IS-A hierarchy** (for example, `Manager` IS-A [member of] `Staff`). We describe the process of specialization and generalization in the following sections.

7.1.4 Specialization process

Specialization

The process of maximizing the differences between members of an entity by identifying their distinguishing characteristics.

Specialization is a top-down approach to defining a set of superclasses and their related subclasses. The set of subclasses is defined on the basis of some distinguishing characteristics of the entities in the superclass. When we identify a subclass of an entity, we then associate attributes specific to the subclass (where necessary), and also identify any relationships between the subclass and other entities or subclasses (where necessary).

7.1.5 Generalization process

Generalization

The process of minimizing the differences between entities by identifying their common features.

The process of **generalization** is a bottom-up approach, which results in the identification of a generalized superclass from the original subclasses. The process of generalization can be viewed as the reverse of the specialization process. For example, consider a model where `Manager` and `Assistant` are represented as distinct entities. If we apply the process of generalization on these entities, we attempt to identify any similarities between them such as common attributes and relationships. As stated earlier, these entities share attributes common to all staff, and therefore we would identify `Manager` and `Assistant` as subclasses of a generalized `Staff` superclass.

Diagrammatic representation

UML has a special notation for representing subclasses and superclasses. For example, consider the specialization/generalization of the `Staff` entity into subclasses that represent job roles. The `Staff` superclass and the `Manager` and `Assistant` subclasses can be represented in the EER diagram illustrated in Figure 7.2. Note that the `Staff` superclass and the subclasses, being entities, are represented as rectangles. Specialization/generalization subclasses are attached by lines to a triangle that points towards the superclass. The label below the triangle, shown as {Optional, And}, describes the constraints on the specialization/generalization relationship. These constraints are discussed in more detail in the following section.

Attributes that are specific to a given subclass are listed in the lower section of the rectangle representing that subclass. For example, the `mgrStartDate` and `bonus` attributes are associated only with the `Manager` subclass, and are not applicable to the `Assistant` subclass. Similarly, we also show the attribute specific to the `Assistant` subclass, namely `shift`.

Figure 7.2 also shows relationships that are applicable to specific subclasses or to just the superclass. For example, the `Manager` subclass is related to the `DistributionCenter` entity through the *`Manages`* relationship, whereas the `Staff` entity is related to the `DistributionCenter` entity through the *`Has`* relationship.

Note The multiplicity of `Manager` in the *`Manages`* relationship is 1..1, whereas previously, in Figure 6.7, the multiplicity of `Staff` in the *p* relationship was 0..1 (in other words, `Manager` has mandatory participation whereas `Staff` had optional participation).

In Figure 7.3, the `Staff` specialization/generalization has been expanded to show a shared subclass called `AssistantManager` and the `Assistant` subclass

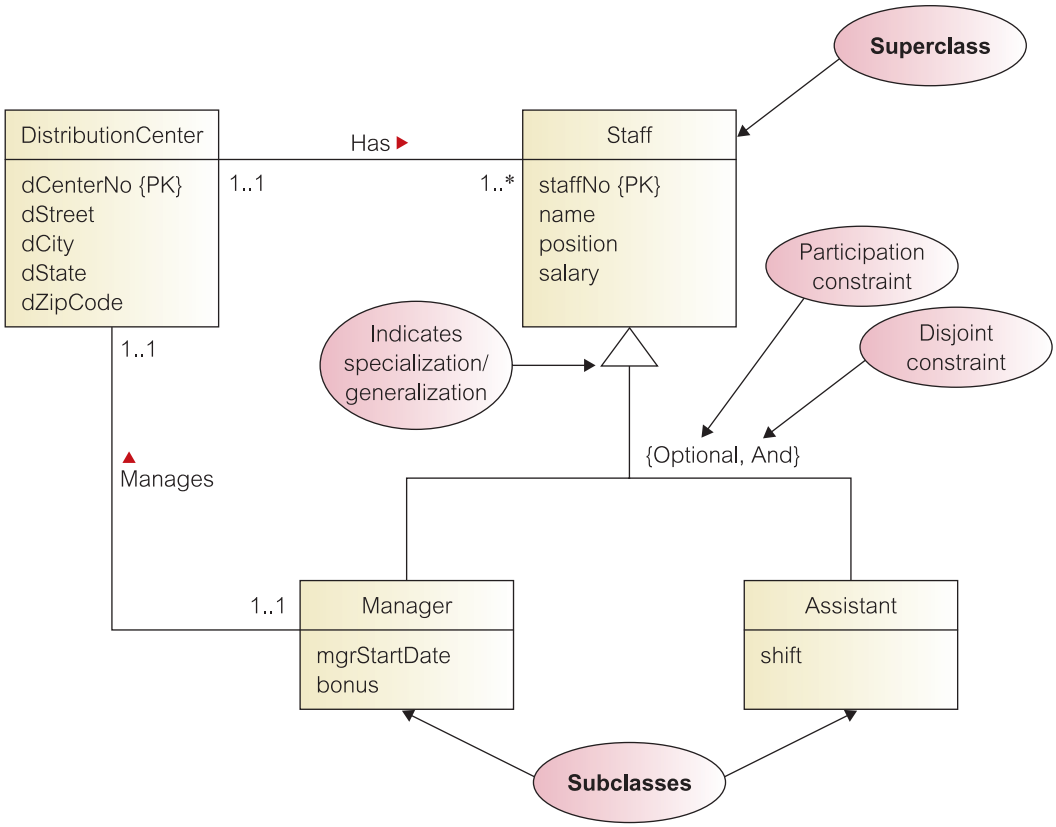


Figure 7.2 Specialization/generalization of the `Staff` entity into subclasses representing job roles

with its own subclass called `TraineeAssistant`. In other words, a member of the `AssistantManager` shared subclass must be a member of the `Manager` and `Assistant` subclasses and `Staff` superclass. As a consequence, the attributes of the `Staff` superclass (`staffNo`, `name`, `position`, `salary`), and the attributes of the subclasses `Manager` (`mgrStartDate` and `bonus`) and `Assistant` (`shift`) are inherited by the `AssistantManager` subclass.

`TraineeAssistant` is a subclass of `Assistant`, which is a subclass of `Staff`. This means that a member of the `TraineeAssistant` subclass must be a member of the `Assistant` subclass and the `Staff` superclass. As a consequence, the attributes of the `Staff` superclass (`staffNo`, `name`, `position`, `salary`) and the attribute of the `Assistant` subclass (`shift`) are inherited by the `TraineeAssistant` subclass, which also has its own additional attribute called `startDate`.

Note The *StayHome Online Rental* case study in Section 5.4 described only Manager and Assistant staff roles in any detail. However, to allow discussions of a superclass/subclasses relationship with {Optional, And} constraints, we have allowed for other staff roles and for staff to exist that are not associated with any subclasses.