Specifying Constraints in Supertype/Subtype Relationships

In this section, we introduce additional notation to represent constraints on supertype/subtype relationships. These constraints allow us to capture some of the important business rules that apply to these relationships. The two most important types of constraints that are described in this section are **completeness** and **disjointness** constraints

SPECIFYING COMPLETENESS CONSTRAINTS

A **completeness constraint** addresses the question of whether an instance of a supertype must also be a member of at least one subtype. The completeness constraint has two possible rules: total specialization and partial specialization.

The **total specialization rule** specifies that each entity instance of the supertype must be a member of some subtype in the relationship. The **partial specialization rule** specifies that an entity instance of the supertype is allowed not to belong to any subtype.

Total Specialization Rule

Figure 3-6a repeats the same example of PATIENT (Figure 3-3) and introduces the notation for total specialization. In this example, the business rule is the following: A patient must be either an outpatient or a resident patient. (There are no other types of patient in this hospital.) Total specialization is indicated by the **double line** extending from the PATIENT entity type to the circle.

In this example, every time a new instance of PATIENT is inserted into the supertype, a corresponding instance is inserted into either OUTPATIENT or RESIDENT PATIENT. If the instance is inserted into RESIDENT PATIENT, an instance of the relationship Is Assigned is created to assign the patient to a hospital bed.

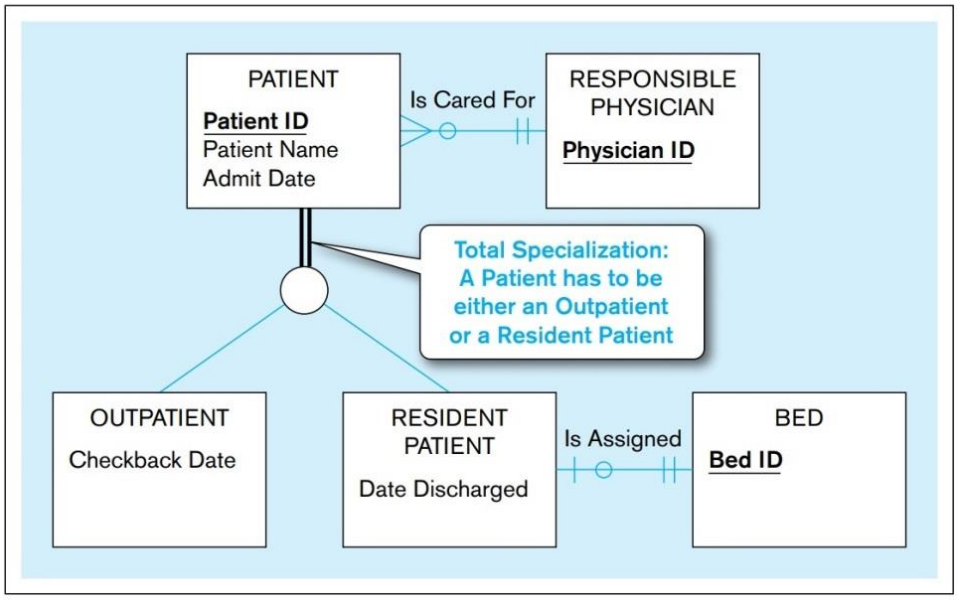


Figure 3-6-a Example of Total Specialization Rule

Partial Specialization Rule

In Figure 3-6b, three entity types have been defined: CAR, TRUCK, and MOTORCYCLE. At this stage, the data modeler intends to represent these separately on an E-R diagram. However, on closer examination, we see that the three entity types have a number of attributes in common: Vehicle ID (identifier), Vehicle Name (with components Make and Model), Price, and Engine Displacement. This fact suggests that each of the three entity types is really a version of a more general entity type. This more general entity type (named VEHICLE) together with the resulting supertype/subtype relationships is shown in Figure 3-6b. The entity CAR has the specific attribute No Of Passengers, whereas TRUCK has two specific attributes: Capacity and Cab Type. Thus, we group entity types along with their common attributes and at the same time preserve specific attributes that are peculiar to each subtype.

Notice that the entity type MOTORCYCLE is not included in the relationship because it does not satisfy the conditions for a subtype discussed earlier, because that the only attributes of MOTORCYCLE are those that are common to all vehicles; there are no attributes specific to motorcycles. Furthermore,

MOTORCYCLE does not have a relationship to another entity type. Thus, there is no need to create a MOTORCYCLE subtype. The fact that there is no MOTORCYCLE subtype suggests that it must be possible to have an instance of VEHICLE that is not a member of any of its subtypes.

Recall that in this example, motorcycle is a type of vehicle, but it is not represented as a subtype in the data model. Thus, if a vehicle is a car, it must appear as an instance of CAR, and if it is a truck, it must appear as an instance of TRUCK. However, if the vehicle is a motorcycle, it cannot appear as an instance of any subtype. This is an example of partial specialization, and it is specified by the **single line** from the VEHICLE supertype to the circle.

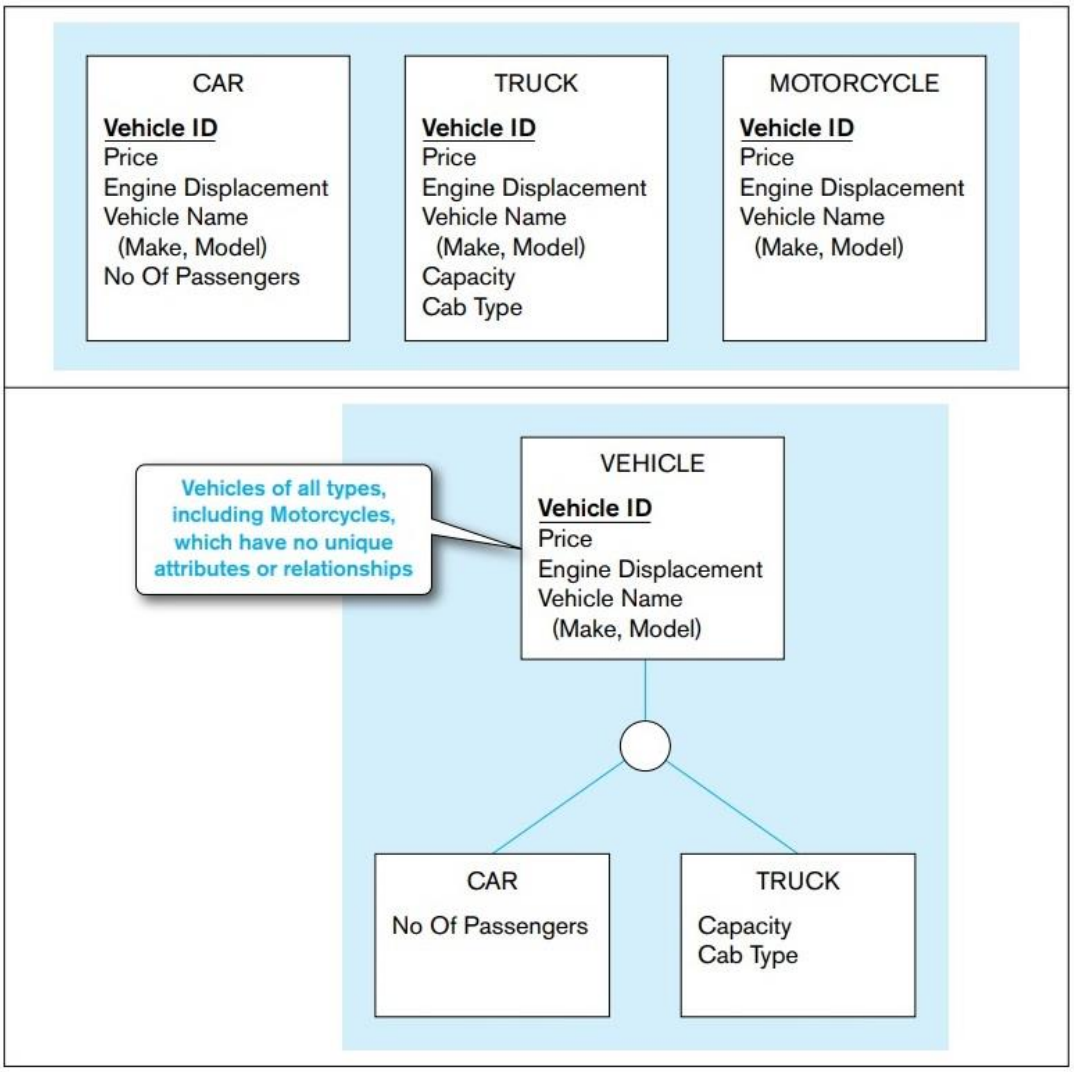


Figure 3-6 b Example of Partial Specialization Rule

Specifying Disjointness Constraints

A **disjointness constraint** addresses whether an instance of a supertype may simultaneously be a member of two (or more) subtypes. The disjointness constraint has two possible rules: the **disjoint rule** and the **overlap rule**.

The disjoint rule specifies that if an entity instance (of the supertype) is a member of one subtype, it cannot simultaneously be a member of any other subtype. **The overlap rule** specifies that an entity instance can simultaneously be a member of two (or more) subtypes. An example of each of these rules is shown in Figure 3-7.

Disjoint Rule

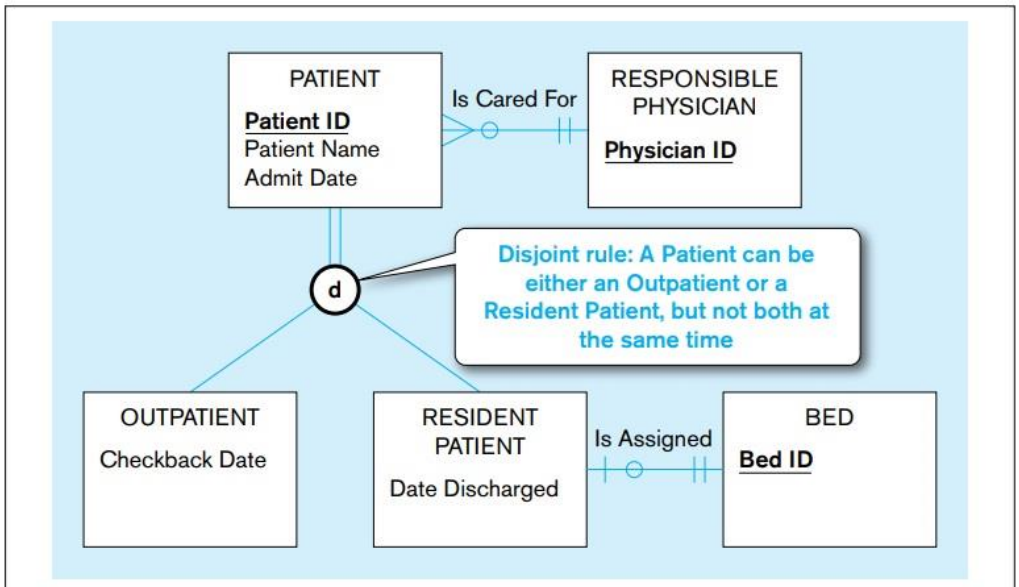
Figure 3-7a shows the PATIENT example from Figure 3-6a. The business rule in this case is the following: *At any given time*, a patient must be either an outpatient or a resident patient, but cannot be both. This is the **disjoint rule, as specified by the letter *d* in the circle joining the supertype and its subtypes**. Note in this figure, the subclass of a PATIENT may change over time, but at a given time, a PATIENT is of only one type.

Overlap Rule

Figure 3-7b shows the entity type PART with its two subtypes, MANUFACTURED PART and PURCHASED PART. In this example some parts are both manufactured and purchased. Some clarification of this statement is required. In this example, an instance of PART is a particular part number (i.e., a *type of part*), not an individual part (indicated by the identifier, which is Part No). For example, consider part number 4000. At a given time, the quantity on hand for this part might be 250, of which 100 are manufactured and the remaining 150 are purchased parts.

The **overlap rule** is specified by placing the letter *o* in the circle, as shown in Figure 3-7b. Notice in this figure that the total specialization rule is also specified, as indicated by the double line. Thus, any part must be either a purchased part or a manufactured part, or it may simultaneously be both of these.

FIGURE 3-7 Examples of disjointness constraints
(a) Disjoint rule



(b) Overlap rule