2 The Entity-Relationship Model

This chapter defines all the major entity-relationship (ER) concepts that can be applied to the conceptual data modeling phase of the database life cycle. In Section 2.1, we will look at the simple level of ER modeling described in the original work by Chen and extended by others. The simple form of the ER model is used as the basis for effective communication with the end user about the conceptual database. Section 2.2 presents the more advanced concepts; although they are less generally accepted, they are useful to describe certain semantics that cannot be constructed with the simple model.

2.1 Fundamental ER Constructs

2.1.1 Basic Objects: Entities, Relationships, Attributes

The basic ER model consists of three classes of objects: entities, relationships, and attributes.

Entities

Entities are the principal data objects about which information is to be collected; they usually denote a person, place, thing, or event of informational interest. A particular occurrence of an entity is called an *entity instance* or sometimes an *entity occurrence*. In our example, employee,

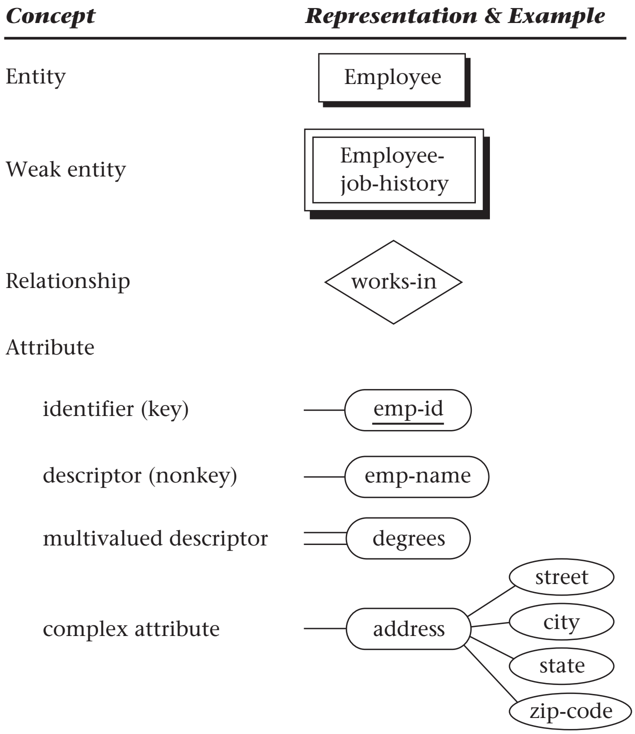

Figure 2.1 The basic ER model

department, division, project, skill, and location are all examples of entities. For easy reference, entity names will henceforth be capitalized throughout this text (e.g., Employee, Department, and so forth). The entity construct is a rectangle as depicted in Figure 2.1. The entity name is written inside the rectangle.

Relationships

Relationships represent real-world associations among one or more entities, and, as such, have no physical or conceptual existence other than that which depends upon their entity associations. Relationships are described in terms of degree, connectivity, and existence. These terms are defined in the sections that follow. The most common meaning associated with the term *relationship* is indicated by the connectivity between entity occurrences: one-to-one, one-to-many, and many-to-many. The relationship construct is a diamond that connects the associated entities, as shown in Figure 2.1. The relationship name can be written inside or just outside the diamond.

A *role* is the name of one end of a relationship when each end needs a distinct name for clarity of the relationship. In most of the examples given in Figure 2.2, role names are not required because the entity names combined with the relationship name clearly define the individual roles of each entity in the relationship. However, in some cases role names should be used to clarify ambiguities. For example, in the first case in Figure 2.2, the recursive binary relationship "manages" uses two roles, "manager" and "subordinate," to associate the proper connectivities with the two different roles of the single entity. Role names are typically nouns. In this diagram, one employee's role is to be the "manager" of up to *n* other employees. The other role is for particular "subordinates" to be managed by exactly one other employee.

Attributes

Attributes are characteristics of entities that provide descriptive detail about them. A particular occurrence of an attribute within an entity or relationship is called an attribute value. Attributes of an entity such as Employee may include emp-id, emp-name, emp-address, phone-no, fax-no, job-title, and so on. The attribute construct is an ellipse with the attribute name inside (or an oblong, as shown in Figure 2.1). The attribute is connected to the entity it characterizes.

There are two types of attributes: identifiers and descriptors. An identifier (or key) is used to uniquely determine an instance of an entity; a descriptor (or nonkey attribute) is used to specify a nonunique characteristic of a particular entity instance. Both identifiers and descriptors may consist of either a single attribute or some composite of attributes. For example, an identifier or key of Employee is emp-id, and a descriptor of Employee is emp-name or job-title. Key attributes are underlined in the ER diagram, as shown in Figure 2.1. We note, briefly, that you can have more than one identifier (key) for an entity, or you can have a set of attributes that compose a key (see Section 6.1.2).

Some attributes, such as specialty-area, may be multivalued. The notation for multivalued attributes is a double attachment line, as shown in Figure 2.1. Other attributes may be complex, such as an address that further subdivides into street, city, state, and zip code. Complex attributes are constructed to have attributes of their own; sometimes, however, the individual parts of a complex attribute are specified as individual attributes in the first place. Either form is reasonable in ER notation.

Entities have internal identifiers that uniquely determine the existence of entity instances, but weak entities derive their identity from the identifying attributes of one or more "parent" entities. Weak entities are often depicted with a double-bordered rectangle (see Figure 2.1), which denotes that all occurrences of that entity depend on an associated (strong) entity for their existence in the database. For example, in Figure 2.1, the weak entity Employee-job-history is related to the entity Employee and dependent upon Employee for its own existence.

2.1.2 Degree of a Relationship

The degree of a relationship is the number of entities associated in the relationship. Binary and ternary relationships are special cases where the degrees are 2 and 3, respectively. An *n*-ary relationship is the general form for any degree *n*. The notation for degree is illustrated in Figure 2.2. The binary relationship, an association between two entities, is by far the most common type in the natural world. In fact, many modeling systems use only this type. In Figure 2.2, we see many examples of the association of two entities in different ways: Department and Division, Department and Employee, Employee and Project, and so on. A binary recursive relationship (for example, "manages" in Figure 2.2) relates a particular Employee to another Employee by management. It is called recursive because the entity relates only to another instance of its own type. The binary recursive relationship construct is a diamond with both connections to the same entity.

A ternary relationship is an association among three entities. This type of relationship is required when binary relationships are not sufficient to accurately describe the semantics of the association. The ternary relationship construct is a single diamond connected to three entities, as shown in Figure 2.2. Sometimes a relationship is mistakenly modeled as ternary when it could be decomposed into two or three equivalent binary relationships. When this occurs, the ternary relationship should be eliminated to achieve both simplicity and semantic purity. Ternary relationships are discussed in greater detail in Section 2.2.3 and Chapter 6.

An entity may be involved in any number of relationships, and each relationship may be of any degree. Furthermore, two entities may have any number of binary relationships between them, and so on for any *n* entities (see *n*-ary relationships defined in Section 2.2.4).

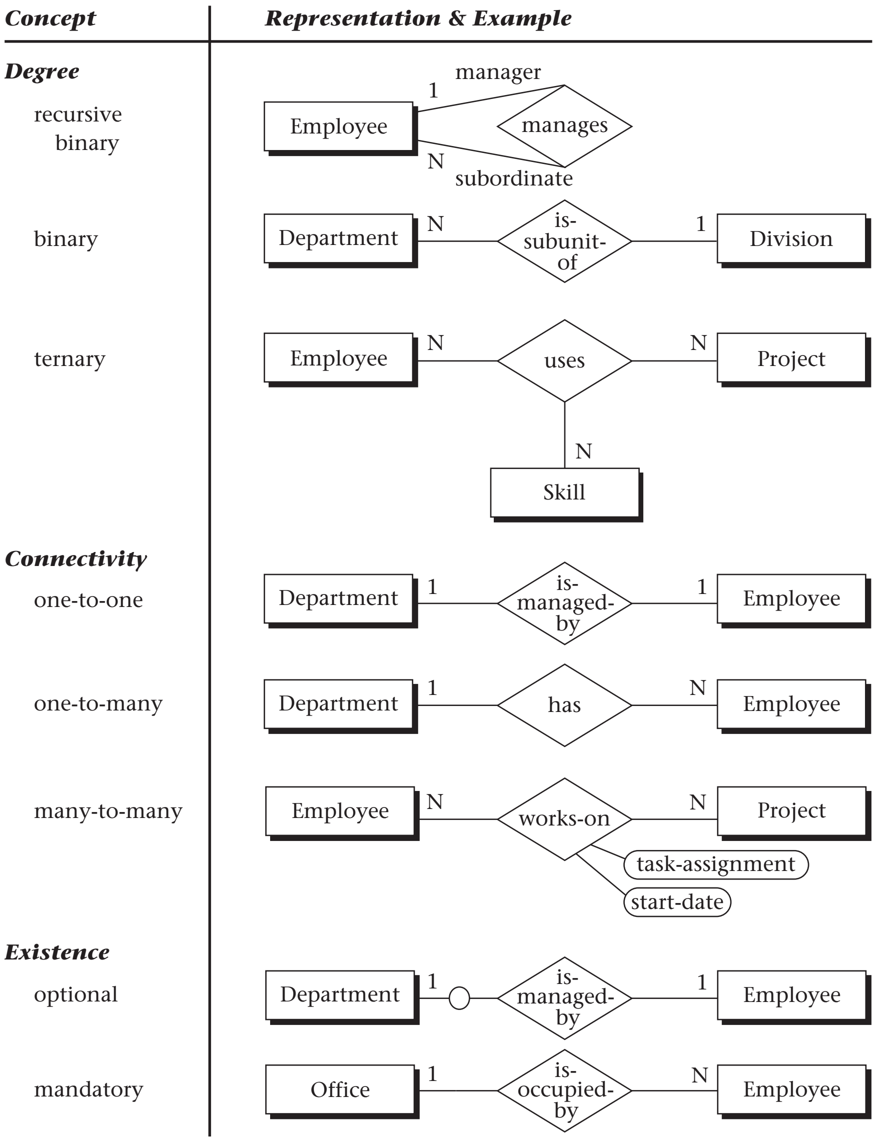

Figure 2.2 Degrees, connectivity, and attributes of a relationship

2.1.3 Connectivity of a Relationship

The *connectivity* of a relationship describes a constraint on the connection of the associated entity occurrences in the relationship. Values for connectivity are either "one" or "many." For a relationship between the entities Department and Employee, a connectivity of one for Department and many for Employee means that there is at most one entity occurrence of Department associated with many occurrences of Employee. The actual count of elements associated with the connectivity is called the *cardinality* of the relationship connectivity; it is used much less frequently than the connectivity constraint because the actual values are usually variable across instances of relationships. Note that there are no standard terms for the connectivity concept, so the reader is admonished to consider the definition of these terms carefully when using a particular database design methodology.

Figure 2.2 shows the basic constructs for connectivity for binary relationships: one-to-one, one-to-many, and many-to-many. On the "one" side, the number one is shown on the connection between the relationship and one of the entities, and on the "many" side, the letter *N* is used on the connection between the relationship and the entity to designate the concept of many.

In the one-to-one case, the entity Department is managed by exactly one Employee, and each Employee manages exactly one Department. Therefore, the minimum and maximum connectivities on the "is-managed-by" relationship are exactly one for both Department and Employee.

In the one-to-many case, the entity Department is associated with ("has") many Employees. The maximum connectivity is given on the Employee (many) side as the unknown value *N*, but the minimum connectivity is known as one. On the Department side the minimum and maximum connectivities are both one, that is, each Employee works within exactly one Department.

In the many-to-many case, a particular Employee may work on many Projects and each Project may have many Employees. We see that the maximum connectivity for Employee and Project is *N* in both directions, and the minimum connectivities are each defined (implied) as one.

Some situations, though rare, are such that the actual maximum connectivity is known. For example, a professional basketball team may be limited by conference rules to 12 players. In such a case, the number

12 could be placed next to an entity called "team members" on the many side of a relationship with an entity "team." Most situations, however, have variable connectivity on the many side, as shown in all the examples of Figure 2.2.

2.1.4 Attributes of a Relationship

Attributes can be assigned to certain types of relationships as well as to entities. An attribute of a many-to-many relationship, such as the "works-on" relationship between the entities Employee and Project (Figure 2.2), could be "task-assignment" or "start-date." In this case, a given task assignment or start date only has meaning when it is common to an instance of the assignment of a particular Employee to a particular Project via the relationship "works-on."

Attributes of relationships are typically assigned only to binary many-to-many relationships and to ternary relationships. They are not normally assigned to one-to-one or one-to-many relationships, because of potential ambiguities. For example, in the one-to-one binary relationship "is-managed-by" between Department and Employee, an attribute "start-date" could be applied to Department to designate the start date for that department. Alternatively, it could be applied to Employee as an attribute for each Employee instance, to designate the employee's start date as the manager of that department. If, instead, the relationship is many-to-many, so that an employee can manage many departments over time, then the attribute "start-date" must shift to the relationship, so each instance of the relationship that matches one employee with one department can have a unique start date for that employee as manager of that department.

2.1.5 Existence of an Entity in a Relationship

Existence of an entity occurrence in a relationship is defined as either mandatory or optional. If an occurrence of either the "one" or "many" side entity must always exist for the entity to be included in the relationship, then it is mandatory. When an occurrence of that entity need not always exist, it is considered optional. For example, in Figure 2.2 the entity Employee may or may not be the manager of any Department, thus making the entity Department in the "is-managed-by" relationship between Employee and Department optional.

Optional existence, defined by a zero on the connection line between an entity and a relationship, defines a minimum connectivity of zero. *Mandatory existence* defines a minimum connectivity of one. When existence is unknown, we assume the minimum connectivity is one (that is, mandatory).

Maximum connectivities are defined explicitly on the ER diagram as a constant (if a number is shown on the ER diagram next to an entity) or a variable (by default if no number is shown on the ER diagram next to an entity). For example, in Figure 2.2, the relationship "is-occupied-by" between the entity Office and Employee implies that an Office may house from zero to some variable maximum (*N*) number of Employees, but an Employee must be housed in exactly one Office, that is, mandatory.

Existence is often implicit in the real world. For example, an entity Employee associated with a dependent (weak) entity, Dependent, cannot be optional, but the weak entity is usually optional. Using the concept of optional existence, an entity instance may be able to exist in other relationships even though it is not participating in this particular relationship.

The term existence is also associated with identification of a data object. Many DBMSs provide unique identifiers for rows (Oracle ROWIDs, for example). Identifying an object such as a row can be done in an existence-based way. It can also be done in a value-based way by identifying the object (row) with the values of one or more attributes or columns of the table.

2.1.6 Alternative Conceptual Data Modeling Notations

At this point we need to digress briefly to look at other conceptual data modeling notations that are commonly used today and compare them with the Chen approach. A popular alternative form for one-to-many and many-to-many relationships uses "crow's-foot" notation for the "many" side (see Figure 2.3a). This form is used by some CASE tools, such as Knowledgeware's Information Engineering Workbench (IEW). Relationships have no explicit construct but are implied by the connection line between entities and a relationship name on the connection line. Minimum connectivity is specified by either a 0 (for zero) or perpendicular line (for one) on the connection lines between entities. The term *intersection entity* is used to designate a weak entity, especially an entity that is equivalent to a many-to-many relationship. Another popular form used today is the IDEFIX notation [IDEF1X, 2005], conceived by Robert G. Brown [Bruce, 1992]. The similarities with the Chen notation are obvious in Figure 2.3b. Fortunately, any of these forms is reasonably easy to learn and read, and their equivalence with the basic ER concepts is obvious from the diagrams. Without a clear standard for the ER model, however, many other constructs are being used today in addition to the three types shown here.

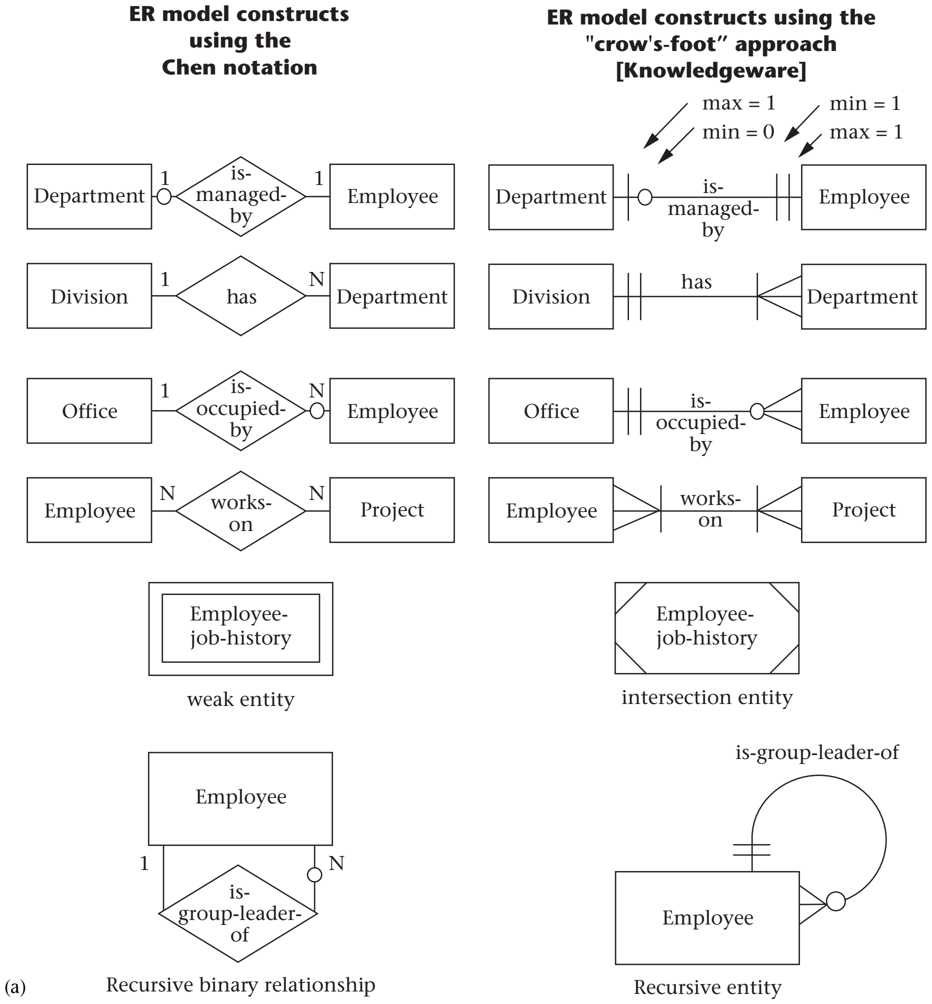

Figure 2.3 Conceptual data modeling notations

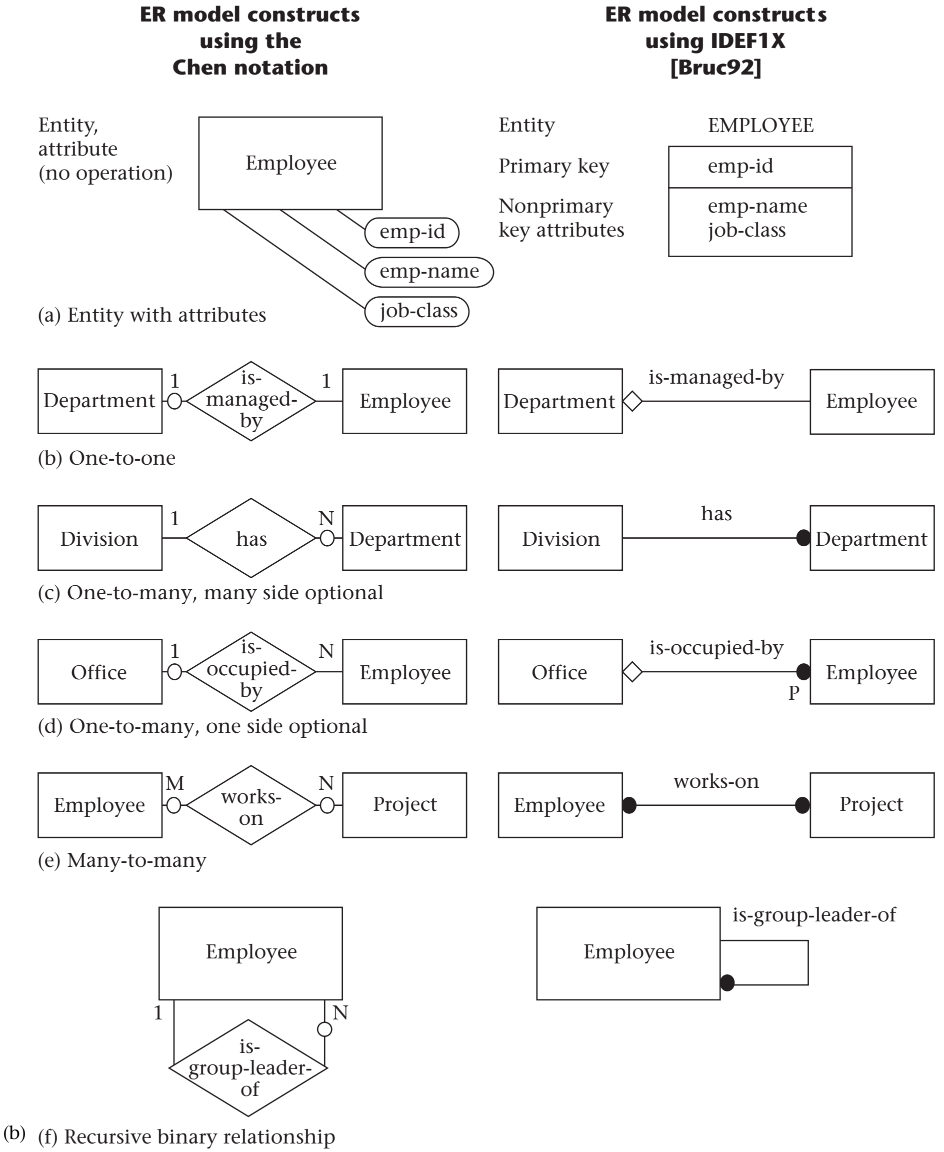

Figure 2.3 *(continued)*