# MODELING ENTITIES AND ATTRIBUTES

The basic constructs of the E-R model are entities, relationships, and attributes. The richness of the E-R model allows designers to model real-world situations accurately and expressively.

## ENTITIES

An **entity** is a person, a place, an object, an event, or a concept in the user environment about which the organization wishes to maintain data. Thus, an entity has a noun name. Some examples of each of these *kinds* of entities follow:

| | |
|---|---|
| *Person:* | EMPLOYEE, STUDENT, PATIENT |
| *Place:* | STORE, WAREHOUSE, STATE |
| *Object:* | MACHINE, BUILDING, AUTOMOBILE |
| *Event:* | SALE, REGISTRATION, RENEWAL |
| *Concept:* | ACCOUNT, COURSE, WORK CENTER |

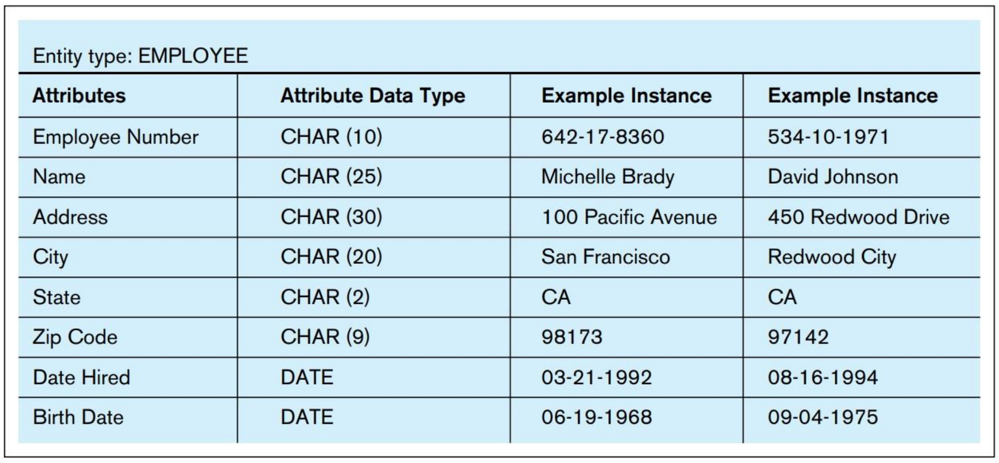
## ENTITY TYPE VERSUS ENTITY INSTANCE

There is an important distinction between entity types and entity instances. An **entity type** is a collection of entities that share common properties or characteristics. Each entity type in an E-R model is given a singular name.

We use capital letters for names of entity type(s). In an E-R diagram, the entity name is placed inside the box representing the entity type.

An **entity instance** is a single occurrence of an entity type. Figure 2-3 illustrates the distinction between an entity type and two of its instances. An entity type is described just once (using metadata) in a database, whereas many instances of that entity type may be represented by data stored in the database.

For example, there is one EMPLOYEE entity type in most organizations, but there may be hundreds (or even thousands) of instances of this entity type stored in the database.



Entity type: EMPLOYEE

| Attributes | Attribute Data Type | Example Instance | Example Instance |
|---|---|---|---|
| Employee Number | CHAR (10) | 642-17-8360 | 534-10-1971 |
| Name | CHAR (25) | Michelle Brady | David Johnson |
| Address | CHAR (30) | 100 Pacific Avenue | 450 Redwood Drive |
| City | CHAR (20) | San Francisco | Redwood City |
| State | CHAR (2) | CA | CA |
| Zip Code | CHAR (9) | 98173 | 97142 |
| Date Hired | DATE | 03-21-1992 | 08-16-1994 |
| Birth Date | DATE | 06-19-1968 | 09-04-1975 |

**FIGURE 2-3** **Entity type EMPLOYEE with two instances**

## ENTITY TYPE VERSUS SYSTEM INPUT, OUTPUT, OR USER

A common mistake people make when they are learning to draw E-R diagrams is to confuse data entities with other elements of an overall information systems model. A simple rule to avoid such confusion is that *a true data entity will have many possible instances, each with a distinguishing characteristic, as well as one or more other descriptive pieces of data*.

## Strong versus Weak Entity Types

Most of the basic entity types to identify in an organization are classified as strong entity types. A **strong entity type** is one that exists independently of other entity types. (Some data modeling software, in fact, use the term *independent entity*.) Examples include STUDENT, EMPLOYEE, AUTOMOBILE, and COURSE. Instances of a strong entity type always have a unique characteristic (called an *identifier*) - that is, an attribute or a combination of attributes that uniquely distinguish each occurrence of that entity. In contrast, a **weak entity type** is an entity type whose existence depends on some other entity type. (Some data modeling software, in fact, use the term *dependent entity*.) A weak entity type has no business meaning in an E-R diagram without the entity on which it depends. The entity type on which the weak entity type depends is called the **identifying owner** (or simply *owner* for short). A weak entity type does not typically have its own identifier. Generally, on an E-R diagram, a weak entity type has an attribute that serves as a *partial* identifier. During a later design stage, a full identifier will be formed for the weak entity by combining the partial identifier with the identifier of its owner or by creating a surrogate identifier attribute.

An example of a weak entity type with an identifying relationship is shown in Figure 2-5. EMPLOYEE is a strong entity type with identifier Employee ID (we note the identifier attribute by underlining it). DEPENDENT is a weak entity type, as indicated by the double-lined rectangle. The relationship between a weak entity type and its owner is called an **identifying relationship**. In Figure 2-5, Carries is the identifying relationship (indicated by the double line). The attribute Dependent Name serves as a *partial* identifier. (Dependent Name is a composite attribute that can be broken into component parts, as we describe later.) We use a double underline to indicate a partial identifier. During a later design stage, Dependent Name will be combined with Employee ID (the identifier of the owner) to form a full identifier for DEPENDENT. Some additional examples of strong and weak entity pairs are: BOOK–BOOK COPY and PRODUCT–SERIAL PRODUCT.

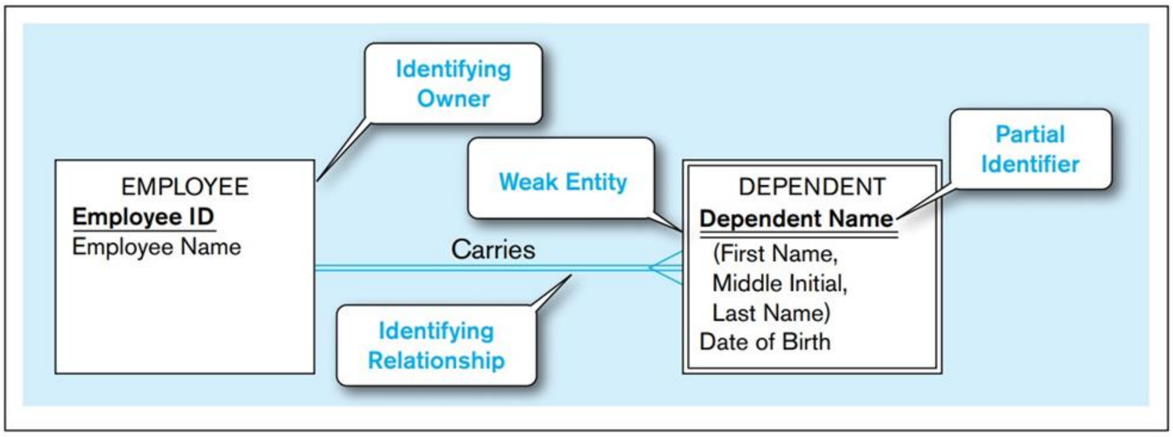


**Figure 2-5 Example of a weak entity and its identifying relationship**

## Attributes

Each entity type has a set of attributes associated with it. An **attribute** is a property or characteristic of an entity type that is of interest to the organization. (Later, we will see that some types of relationships may also have attributes.) Thus, an attribute has a noun name. Following are some typical entity types and their associated attributes:

| | |
|---|---|
| STUDENT | Student ID, Student Name, Home Address, Phone Number |
| AUTOMOBILE | Vehicle ID, Color, Weight, Horsepower |
| EMPLOYEE | Employee ID, Employee Name, Payroll Address, Skill |

In naming attributes, we use an initial capital letter followed by lowercase letters. If an attribute name consists of more than one word, we use a space between the words and we start each word with a capital letter, for example, Employee Name or Student Home Address. In E-R diagrams, we represent an attribute by placing its name in the entity it describes. Attributes may also be associated with relationships, as described later. Note that an attribute is associated with exactly one entity or relationship.

### Required Versus Optional Attributes

Each entity (or instance of an entity type) potentially has a value associated with each of the attributes of that entity type. An attribute that must be present for each entity instance is called a **required attribute**, whereas an attribute that may not have a value is called an **optional attribute**. In various E-R diagramming notations, a symbol might appear in front of each attribute to indicate whether it is required (e.g., *) or optional (e.g., o), or required attributes will be in **boldface**, whereas optional attributes will be in normal font (the format we use in this text); an attribute without a value is said to be null.

## Simple versus Composite Attributes

Some attributes can be broken down into meaningful component parts (detailed attributes). A common example is Name, which we saw in Figure 2-5; another is Address, which can usually be broken down into the following component attributes: Street Address, City, State, and Postal Code. A **composite attribute** is an attribute, such as Address, that has meaningful component parts, which are more detailed attributes. Figure 2-7 shows the notation that we use for composite attributes applied to this example.

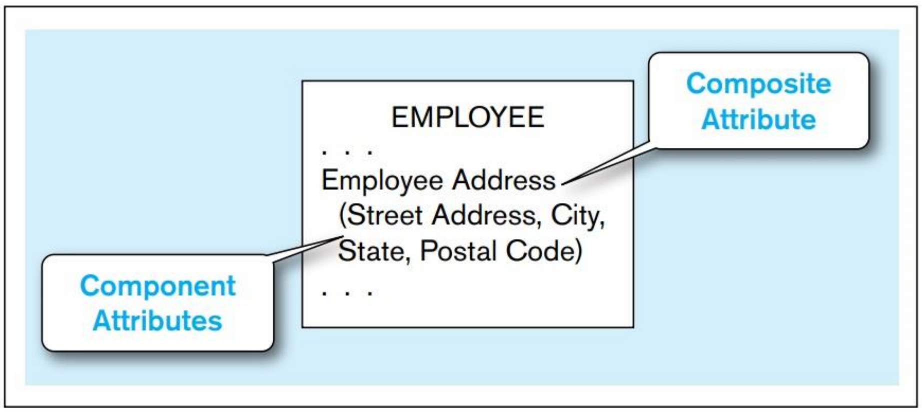


**Figure 2-7 a composite attribute**

A **simple (or atomic) attribute** is an attribute that cannot be broken down into smaller components that are meaningful for the organization. For example, all the attributes associated with AUTOMOBILE are simple: Vehicle ID, Color, Weight, and Horsepower.

## Single-Valued versus Multivalued Attributes

It frequently happens that there is an attribute that may have more than one value for a given instance. For example, the EMPLOYEE entity type in Figure 2-8 has an attribute named Skill, whose values record the skill (or skills) for that employee. Of course, some employees may have more than one skill, such as PHP Programmer and C++ Programmer. A **multivalued attribute** is an attribute that may take on more than one value for a given entity (or relationship) instance. In this text, we indicate a multivalued attribute with curly brackets around the attribute name, as shown for the Skill attribute in the EMPLOYEE example in Figure 2-8.

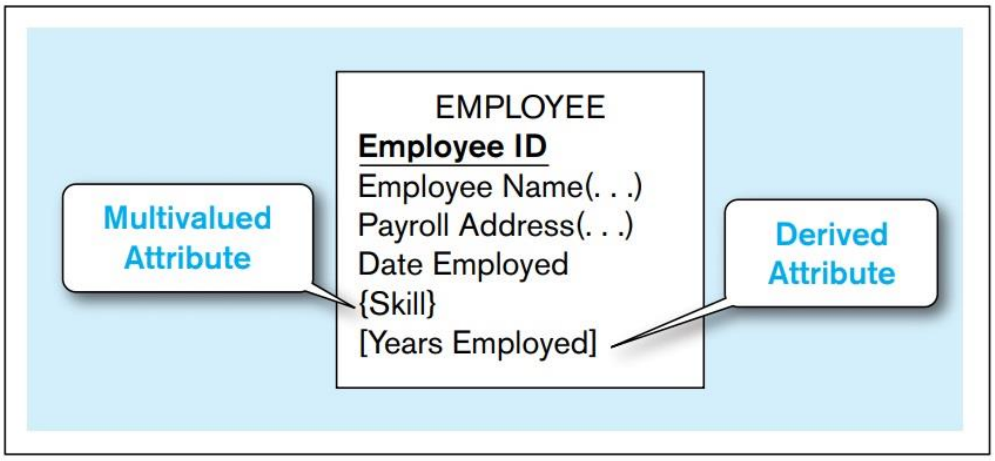


**Figure 2-8** **Entity with multivalued attribute (Skill) and derived attribute (Years Employed)**

Multivalued and composite are different concepts, although beginner data modelers often confuse these terms. Skill, a multivalued attribute, may occur multiple times for each employee; Employee Name and Payroll Address are both likely composite attributes, each of which occurs once for each employee, but which have component, more atomic attributes, which are not shown in Figure 2-8 for simplicity.

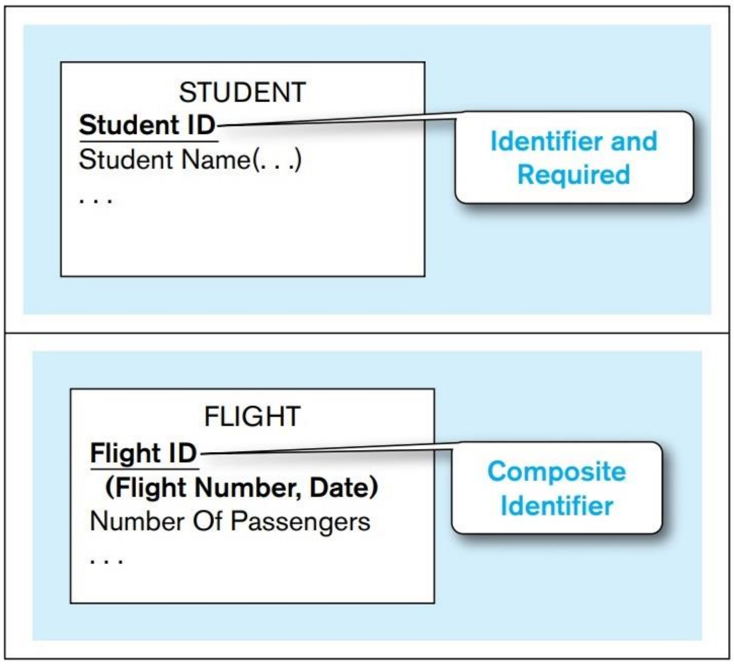
## Stored Versus Derived Attributes

Some attribute values that are of interest to users can be calculated or derived from other related attribute values that are stored in the database. For example, suppose that for an organization, the EMPLOYEE entity type has a Date Employed attribute. If users need to know how many years a person has been employed, that value can be calculated using Date Employed and today's date. A **derived attribute** is an attribute whose values can be calculated from related attribute values (plus possibly data not in the database, such as today's date, the current time). We indicate a derived attribute in an E-R diagram by using square brackets around the attribute name, as shown in Figure 2-8 for the Years Employed attribute. In some situations, the value of an attribute can be derived from attributes in related entities.

## IDENTIFIER ATTRIBUTE

An **identifier** is an attribute (or combination of attributes) whose value distinguishes individual instances of an entity type. That is, no two instances of the entity type may have the same value for the identifier attribute. The identifier for the STUDENT entity type introduced earlier is Student ID, whereas the identifier for AUTOMOBILE is Vehicle ID. Notice that an attribute such as Student Name is not a candidate identifier, because many students may potentially have the same name, and students, like all people, can change their names. We underline identifier names on the E-R diagram, as shown in the STUDENT entity type example in Figure 2-9a. To be an identifier, the attribute is also required (so the distinguishing value must exist), so an identifier is also in bold.

**FIGURE 2-9 Simple and composite identifier attributes**
**(a) Simple identifier attribute**
**(b) Composite identifier attribute**



For some entity types, there is no single (or atomic) attribute that can serve as the identifier (i.e., that will ensure uniqueness). However, two (or more) attributes used in combination may serve as the identifier. A **composite identifier** is an identifier that consists of a composite attribute. Figure 2-9b shows the entity FLIGHT with the composite identifier Flight ID. Flight ID in turn has component attributes Flight Number and Date. This combination is required to identify uniquely individual occurrences of FLIGHT. We use the convention that the composite attribute (Flight ID) is underlined to indicate it is the identifier, whereas the component attributes are not underlined.

Some entities may have more than one candidate identifier. If there is more than one candidate identifier, the designer must choose one of them as the identifier by using the following criteria:

**1.** Choose an identifier that will not change its value over the life of each instance of the entity type.

**2.** Choose an identifier such that for each instance of the entity, the attribute is guaranteed to have valid values and not be null (or unknown).

## NAMING AND DEFINING ATTRIBUTES

In addition to the general guidelines for naming data objects, there are a few special guidelines for naming attributes, which follow:

• An attribute name is a *singular noun or noun phrase* (such as Customer ID, Age, Product Minimum Price, or Major).

• An attribute name should be *unique*. No two attributes of the same entity type may have the same name, and it is desirable, for clarity purposes, that no two attributes across all entity types have the same name.

• For clarity purposes, *each attribute name should follow a standard format*.