Attributes

Attributes are the characteristics that describe entities and relationships. For example, a Student entity may be described by attributes including: student number, name, address, date of birth, degree major. An Invoice entity may be described by attributes including: invoice number, invoice date, invoice total. In an ERD using the Chen notation, we illustrate attributes using ovals as shown below.

Suppose we have entity types Student and Course that are associated via an enroll relationship. An attribute that helps to describe a student enrolled in a course is grade:

A common convention for naming attributes is to use singular nouns. A naming convention may require one of:

- All characters are in upper case.
- All characters are in lower case.
- Only the first character is in upper case.
- Each part of a multipart name has the first character capitalized.

A typical convention is for attribute names to have a prefix that indicates the entity the attribute describes. Subsequent characters are sufficiently descriptive to identify the attribute. Some examples of attribute names:

- empLname = employee last name
- stuGpa = student grade point average
- prodCode = product code
- invNum = invoice number

In practice a naming convention is important, and you should expect the organization you are working for to have a standard approach for naming things appearing in a model. A substantial data model will have tens, if not into the hundreds, of entity sets, many more attributes and relationships. It becomes important to easily understand the concept underlying a specific name; a naming convention can be helpful.
Exercise

Based on information you have supplied on registration and application forms, what attributes could describe the Student entity type at your institution?

**Atomic Attributes**

An attribute is considered atomic (or simple) if it does not contain any meaningful smaller components.

For example, suppose "Gender" is an attribute in our design. The Gender attribute has a small set of possible values, for example M or F. It is not meaningful to decompose Gender into smaller units, and so we say Gender is a Simple attribute.

As another example consider an attribute for product price, prodPrice. A sample value for prodPrice is $21.03. Of course, one could decompose prodPrice into two attributes where one attribute represents the dollar component (21), and the other attribute represents the cents component (03), but our assumption here is that such a decomposition is not meaningful to the intended application or system that will make use of it. So we would consider prodPrice to be atomic because it cannot be usefully decomposed into meaningful components.

**Exercise:**

Consider an attribute for the employee's last name, such as empLname. Can this be decomposed into smaller meaningful attributes?

**Composite Attributes**

An attribute is considered composite if it comprises two or more other attributes.

Consider an attribute such as name that comprises first and last names. For example, suppose an employee's name is John McKenzie. The first name is John and the last name is McKenzie. It is easy to appreciate that one application may only want the last name, another may display the first name followed by the last name, and yet another application may display the last name, a comma, and then the first name.

Since it is meaningful to decompose empName into two attributes for first name, firstName, and last name, lastName, we consider the name attribute to be a composite attribute (firstName and lastName are non-composite; they are atomic attributes). A composite attribute is an attribute that is shown as comprising two or more simpler attributes; we show a composite attribute below.
Derived Attributes

If an attribute's value can be determined from the values of other attributes, then the attribute is derivable, and is said to be a derived attribute.

If an attribute's value can be derived from other attributes, you should consider dropping the attribute from the model. Perhaps you would keep it, if keeping it helped understandability (you can still decide to drop the attribute from the physical model). Derived attributes are shown with a dotted lined oval, see the figure below.

Sometimes an attribute of one entity is derived from attributes in other entities. Consider the attribute for the total of an Invoice, InvTotal. A value of InvTotal is derivable; it can be computed by knowing the value of InvLineAmount from each related invoice line. Keeping an attribute such as InvTotal in the model can easily promote communication with other interested parties (other business users). This example also indicates a situation where deriving a value (on-the-fly) could be an expensive operation to perform (to determine the value of an invoice would mean accessing all relevant invoice line data in the database). Such an attribute may find its way into the eventual physical model for that reason.
Descriptive Attributes

Most attributes simply serve a purpose to describe some characteristic of entities. For example, in a University database the Student entity type would have several attributes such as: Student Number, First Name, Last Name, Gender, Local Phone Number, Home Phone Number, .... Most of these attributes will be descriptive. An attribute such as Gender cannot be a Key (nor a Partial Key) for students.

Non-key attributes are attributes that are not part of a key. Generally, most attributes are simply descriptive, and fall into this category. Consider attributes for first name, last name, birth date; usually these attributes are non-key attributes. Consider a name attribute. People may join your organization and arrive with a name; we expect many people in a large organization to have the same first name, same last name, and even the same combination of first and last name.

Names cannot usually be used as a key. Names that are chosen for entities such as departments in an organisation could be keys because of the way the company would choose department names. They wouldn't give two different departments the same name.

Consider a birth date attribute. Many people could have the same birth date, and so different people could have the same birth date; birth date would not be a key attribute for a person entity type; it would be a non-key attribute of person.

Determining key attributes is an important exercise, and one that requires careful consideration. Most attributes are non-key attributes.

Attribute Domains

A Domain specifies the valid set of values for an attribute.

For each attribute in our ERD, we must assign a domain. One of the tasks for a database designer is to decide on the domains that are pertinent for the database. For instance, for a college that assigns each student a student number the database designer could decide the domain for student numbers is the set of positive 7-digit integers.

As we analyze the requirements for a system, we must define domains and we must associate attributes to domains. A single domain may be related to many attributes. For instance, the total for an invoice, invTotal, and the price of a product, prodPrice, are both associated with the domain of non-negative currency values. The quantity on an invoice line, invLineQty, is associated with the domain of non-negative integers.

Knowing the underlying domains in your model is important. They help to complete your analysis, they are indispensable for coding specifications, and they are useful for defining meaningful error messages.
Key Attributes

An attribute is a Key if values of the attribute uniquely identify instances of a corresponding entity set.

For example, suppose "Student Number" is an attribute for the Student entity type in our design for a University database. Suppose Student Number is just one of several attributes; others are: First Name, Last Name, and Gender. Suppose further that the University assigns each student a unique student number, and assuming that we intend to have just one entity instance per student, Student Number is a Key. In an ERD, keys are shown underlined:

For each entity type, we prefer to have an attribute, or a combination of attributes that identify individual entities.

We define a key to be an attribute, or a minimal set of attributes, that uniquely identify entities in an entity set. By minimal we mean that all of the attributes are required (none can be omitted). For instance, a typical key for an invoice line entity type would be the combination of invoice number and invoice line number. Both attributes are required to identify a particular invoice line. However, such a key is composite (see the section on composite attributes).

Multi-Valued Attributes

An attribute is considered multi-valued if there can be many values associated with it at any one point in time.

Now, suppose someone proposes to track the university degrees earned by employee's with an attribute named empDegree. When an attribute could have none, one, or several values, we say the attribute is multi-valued. For a given employee and point in time, empDegree could have multiple values, and so we say it is multi-valued. If you need to keep track of all the degrees that each person has obtained, then there would be multiple values to store.

The following shows sample data for three employees. We assume that each employee has just a single employee number, and that we are keeping only one phone number per employee; both empNum and empPhone are considered to be single-valued.

empNum empPhone empDegree
Exercise:

Consider the addresses that a University might keep track of for a student: mailing address, home address, etc.

Can the attribute for address be modelled as a multi-valued attribute? Can it also be composite?

Partial-Key Attributes

An attribute is a Partial Key if a Key from a related entity type must be used in conjunction with the attribute in question to uniquely identify instances of a corresponding entity set.

For example, suppose "Course Number" is an attribute of the Course entity type in our design for a University database. Suppose Course Number alone cannot uniquely identify courses. Rather, to identify a course we must include the Department Number attribute as well.

At the U of W, keys for a Course entity type include the Key of the Department entity set: ACS-2914, ACS-3902, BIO-1914.
Course Number is a partial key. In Universities this situation is quite common.

Exercise:

Sometimes we have attributes that distinguish entities of an entity set from other entities in the same set, but only relative to some other related entity. This situation arises naturally when we model things like invoices and invoice lines. If invoice lines are assigned line numbers (1, 2, 3, etc.), these line numbers distinguish lines on an invoice from one another. However, for any given line number value, there could be many invoice lines (from separate invoices) with that particular line number. A partial key (or discriminator) is an attribute that distinguishes instances of a weak entity set relative to a strong entity. Invoice line number is a partial key for invoice lines. Illustrate the attributes, entity types and relationships.

**Single-Valued Attributes**

An attribute is considered single-valued if there is at most one value associated with it at any one point in time.

For example, suppose "gender" is an attribute in our design. For most applications we would say that gender is single-valued; at any given point in time, there is just one value (male, female) recorded for gender for a person.

We characterize an attribute as being single-valued if, for any instance of the pertinent entity set, there is only one value at a given time for the attribute.

Single-valued attributes are shown with a simple oval, one with a single line for a border (as opposed to a double-lined border). In all of our examples so far, we have assumed that each attribute was single-valued.
Exercise

Consider the title and description attributes that describe a course. Are these single-valued attributes?

Stored Attributes

If an attribute's value cannot be determined from the values of other attributes, then the attribute's value for an entity must be kept as part of the entity and we refer to such an attribute as a Stored attribute.

When an attribute is specified for an entity or relationship type, and when it is depicted with a solid line (as opposed to a dashed line), it is considered a stored attribute. The designers intent is that the attributes value will be part of an instance of the entity or relationship. This is in contrast to a derived attribute. In the example below where we have attributes for an employee, the birth date is shown as a stored attribute and current age is shown as derived.