Structure of an XML Document

Let’s get started by reviewing the structure of our example breaking this document down and see at how it really works.

The XML Declaration

The first line in the example document is a *processing instruction* that identifies the document as an XML document type. Processing instructions are special types of instructions, and we will go into detail on how they are used later. Let’s break the line down into parts.

```
<?xml version=“1.0”?>
```

The first part is the processing instruction code. This is the <? and ?> characters. The ? is the identifier that is used to specify this markup type. The second point of note is the “xml” statement. The third is the “version” attribute, which defines the version of XML that this document complies with. This instruction also specifies whether the document is stand-alone (as is the case with this example), or requires a separate DTD in order to make sense of the data contained therein.

Tip: While the XML declaration is optional, it is a very good idea to always include it for maximum portability

The Root Element

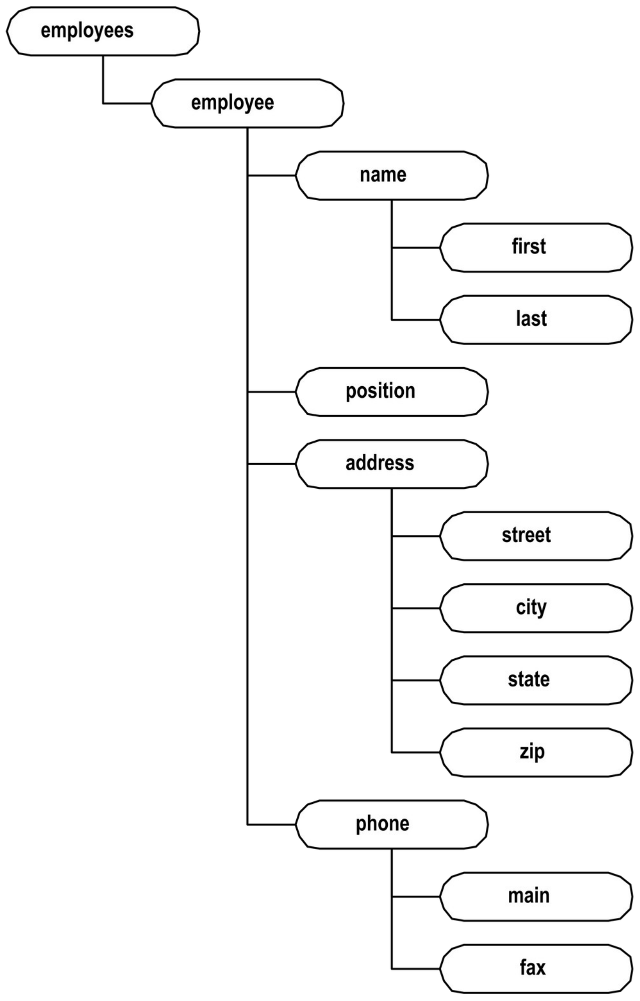
Each XML document must have a root element, and there can be only one. A root element is the element that encapsulates all other elements in the XML document.
In our example, <employees> is the root element. Notice that all other elements are located with the <employees> and </employees> tag.

```
<?xml version=”1.0”?>
<employees>
  . . . (rest of document omitted)
</employees>
```

The Logical Structure

In theory, there are two types of structure in XML. The first is the *logical* structure, the second is the *physical* structure. The logical structure has nothing to do with the physical entities associated in an XML document, but instead has to do with the order of the elements that it contains. The logical structure is independent of the physical structure because the logical structure includes all external entities that may be referenced by an XML document.

The following diagram illustrates the logical structure of our employee example.



File: Chapter 2\Employee.xml Logical Diagram

A key concept in XML is the idea of *element relationship*. Elements are said to be related to each other in one of three ways: *parent*, *child, or sibling*. This is not an either/or model as you will see. Relationship is relative to the way that you are referring to an element.

Parents

Parent elements are elements that contain other elements. An example of a parent is the <name> element. <name> has two children: <first> and <last>. An element is a parent to the elements that it contains.

```
<name>
  <first>John</first>
  <last>Doe</last>
</name>
```

Child

Child elements are elements that are contained within a parent element. Using our previous example, both <first> and <last> are child elements to <name>.

```
<name>
  <first>John</first>
  <last>Doe</last>
</name>
```

Siblings

Siblings are elements that share a parent are called *siblings*. While <first> and <last> are children of <name>, they are also siblings to each other. Another example of siblings include <name>, <position>, <address> and <phone>.

XML and Databases

XML is *hierarchical* in nature. The concepts of parent, child, and sibling elements are not new to data storage and databases in general. Those who have used hierarchical databases like IMS and CICS will easily understand the logical structure of an XML document. For those who are familiar with newer databases that utilize the relational model, XML may seem a little odd.

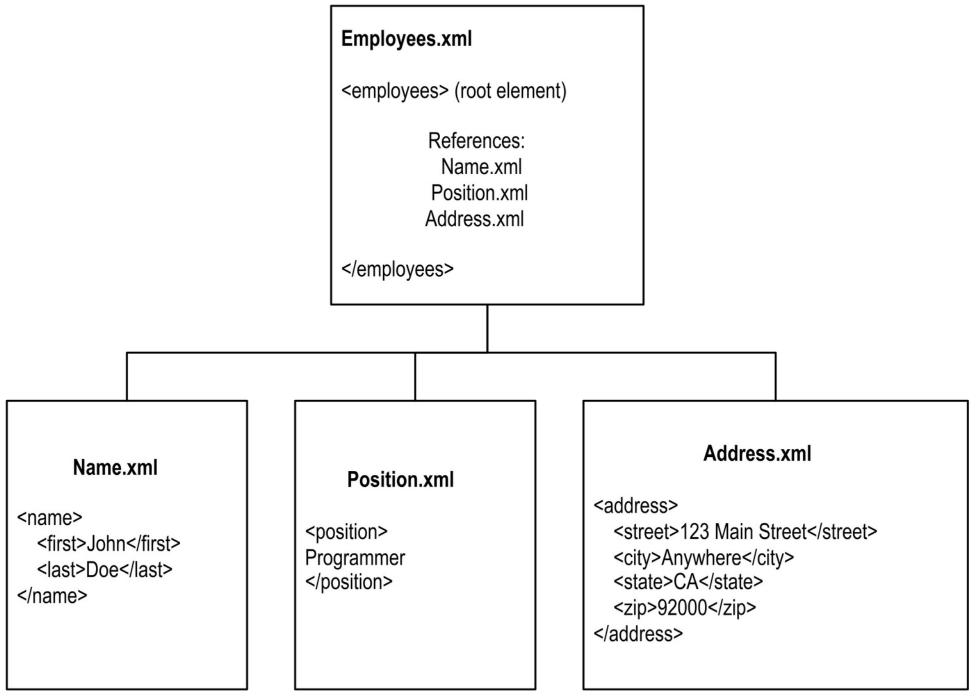
When we get into *content modeling*, you will see that it is easy to model both types of databases using XML.

The Physical Structure

Understanding the logical structure is fundamental to using XML within your software applications. Understanding how to manipulate the physical structure is fundamental to creating and maintaining XML documents.

If you will recall, the physical storage of an XML document is called an entity. An entity can reference another entity (i.e. one XML document references another XML document) and the result is a single logical structure of elements. This means that we could physically breakup our employee example into separate XML documents, make reference to them from a primary document, and treat them as a single logical entity.

The following is an example of how we could do this.



Being able to break a document's logical structure into multiple physical parts and reference them individually or from other documents is extremely powerful. It can also get you into a lot trouble. Let's say for arguments sake that Position.xml contained a </employees> tag. This would ruin the logical structure and cause the XML processor to be unable to process the document. The lesson here is that you need to be very careful in defining the contents of your physical structure so that your logical structure can remain intact.

Tip: A good rule of thumb is to not have partial element definitions in your documents. In other words, do not begin an element in one document, and end it in another.

This is a good place to recall the need for clearly commenting your XML documents!

Synchronous Structures

It is absolutely critical to keep structures synchronous. In other words, your tags have to match up in correct order. Let's say that you are putting some HTML inside of an XML document and the HTML has some basic tags in it.

```
<text>This is <emph>italicized</emph>
  text.</text>
```

This is perfectly valid XML. However, if the <emph> tag was omitted like the following statement, the XML processor would not know how to handle it.

```
<text>This is italicized</emph> text<text>
```

Since there is no beginning <emph> tag, the XML processor encounters the closing tag and does not know how to handle it. The rule of XML is to always make sure that your tags match up so that the logical flow of the XML document is synchronous.