UT 6. XSD

1 QUÉ ES XSD

XSD (*XML Schema Definition*) es un lenguaje, también llamado simplemente ***XML Schema***, que sirve para definir la estructura de un documento XML, permitiendo su validación.

En los siguientes enlaces se puede consultar la *W3C Recommendation* de XSD, en la cual están basados estos apuntes de introducción a XML Schema:

- ***XML Schema Part 0: Primer*** www.w3.org/TR/xmlschema-0/
- ***XML Schema Part 1: Structures*** www.w3.org/TR/xmlschema-1/
- ***XML Schema Part 2: Datatypes*** www.w3.org/TR/xmlschema-2/

2 VALIDACIÓN DE UN DOCUMENTO XML CON XSD

EJEMPLO Se quiere almacenar una lista de marcadores de páginas web, guardando de cada uno de ellos su nombre, una descripción y su URL. Para ello, se ha escrito el siguiente documento XML (***"marcadores.xml"***) asociado al archivo ***"marcadores.xsd"***:

```
<?xml version="1.0" encoding="UTF-8"?>
<marcadores xsi:noNamespaceSchemaLocation="marcadores.xsd"
   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
      <pagina>
            <nombre>informatica</nombre>
            <descripcion>Tutoriales de informática.</descripcion>
            <url>http://www.informatica.com/</url>
      </pagina>
      <pagina>
            <nombre>Wikipedia</nombre>
            <descripcion>La enciclopedia libre.</descripcion>
            <url>http://www.wikipedia.org/</url>
      </pagina>
      <pagina>
            <nombre>W3C</nombre>
            <descripcion>World Wide Web Consortium.</descripcion>
            <url>http://www.w3.org/</url>
      </pagina>
</marcadores>
```

- Para vincular un esquema a un documento XML, es obligatorio que este último haga referencia al espacio de nombres http://www.w3.org/2001/XMLSchema-instance. Para ello, habitualmente se utiliza el prefijo xsi.
- El atributo noNameSchemaLocation permite referenciar a un archivo con la definición de un esquema que no tiene ningún espacio de nombres asociado. En este caso, dicho archivo es “marcadores.xsd”.

El esquema XML guardado en “marcadores.xsd” y que permita validar el documento XML “marcadores.xml” podría ser:

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
 <xs:element name="marcadores">
  <xs:complexType>
   <xs:sequence>
    <xs:element name="pagina" maxOccurs="unbounded">
     <xs:complexType>
      <xs:sequence>
       <xs:element name="nombre" type="xs:string"/>
       <xs:element name="descripcion" type="xs:string"/>
       <xs:element name="url" type="xs:string"/>
      </xs:sequence>
     </xs:complexType>
    </xs:element>
   </xs:sequence>
  </xs:complexType>
 </xs:element>
</xs:schema>
```

Para estar bien formado, un esquema XML tiene que cumplir las mismas reglas de sintaxis que cualquier otro documento XML.

Por otra parte, hay que tener en cuenta que, en todos los esquemas XML, el elemento raíz es “schema”. Ahora bien, para escribirlo, es muy común utilizar el prefijo xsd o xs.

Con xmlns:xs="http://www.w3.org/2001/XMLSchema" se ha indicado que:

- Los elementos y tipos de datos utilizados en el esquema pertenecen al espacio de nombres http://www.w3.org/2001/XMLSchema.
- Dichos elementos y tipos de datos deben llevar el prefijo xs (xs:schema, xs:element, xs:complexType, xs:string...).

Fíjese también que:

- Los elementos “marcadores” y “página” son de tipo complejo (complexType), ya que, contienen a otros elementos.
- sequence indica que los elementos hijo deben aparecer, en el documento XML, en el mismo orden en el que sean declarados en el esquema.
- Los elementos “nombre”, “descripción” y “url” son de tipo simple (string en este caso) y no pueden contener a otros elementos.
- Mediante maxOccurs="unbounded" se ha indicado que pueden aparecer ilimitados elementos “página” en el documento XML.

2.1 Definición de un espacio de nombres.

EJEMPLO En el siguiente documento XML se ha definido un espacio de nombres escribiendo xmlns:mar="http://www.informatica.com/marcadores":

```
<?xml version="1.0" encoding="UTF-8"?>
<mar:marcadores
xsi:schemaLocation="http://www.informatica.com/marcadores marcadores.xsd"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:mar="http://www.informatica.com/marcadores">
    <mar:pagina>
        <mar:nombre>informatica</mar:nombre>
        <mar:descripcion>Tutoriales de informática.</mar:descripcion>
        <mar:url>http://www.informatica.com/</mar:url>
    </mar:pagina>
    <mar:pagina>
        <mar:nombre>Wikipedia</mar:nombre>
        <mar:descripcion>La enciclopedia libre.</mar:descripcion>
        <mar:url>http://www.wikipedia.org/</mar:url>
    </mar:pagina>
    <mar:pagina>
        <mar:nombre>W3C</mar:nombre>
```

```
            <mar:descripcion>World Wide Web Consortium.</mar:descripcion>
            <mar:url>http://www.w3.org/</mar:url>
        </mar:pagina>
</mar:marcadores>
```

En el atributo schemaLocation se pueden escribir parejas de valores:

- En el primer valor de cada pareja, hay que hacer referencia a un espacio de nombres.
- En el segundo valor, se tiene que indicar la ubicación de un archivo donde hay un esquema de ese espacio de nombres.

En cuanto al archivo “marcadores.xsd”, ahora su código podría ser:

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema targetNamespace="http://www.informatica.com/marcadores"
elementFormDefault="qualified"
 xmlns:xs="http://www.w3.org/2001/XMLSchema"
  xmlns="http://www.informatica.com/marcadores">
  <xs:element name="marcadores">
   <xs:complexType>
     <xs:sequence>
       <xs:element name="pagina" maxOccurs="unbounded">
           <xs:complexType>
                <xs:sequence>
                        <xs:element name="nombre" type="xs:string"/>
                        <xs:element name="descripcion" type="xs:string"/>
                        <xs:element name="url" type="xs:string"/>
                </xs:sequence>
           </xs:complexType>
         </xs:element>
       </xs:sequence>
    </xs:complexType>
   </xs:element>
</xs:schema>
```

- En el atributo targetNamespace se está indicando que los elementos definidos en este esquema (“marcadores”, “página”, “nombre”, “descripción” y “url”), provienen del espacio de nombres http://www.informatica.com/marcadores.
- xmlns="http://www.informatica.com/marcadores" especifica que este es el espacio de nombres por defecto.
- El atributo elementFormDefault="qualified" indica que todos los elementos declarados localmente en el esquema tienen que estar calificados, es decir, tienen que pertenecer a un espacio de nombres. Por esta razón, en “marcadores.xml” se han escrito con el prefijo mar.

2.2 Elementos globales y locales en XSD

Los elementos pueden ser globales o locales:

- Los elementos globales son hijos directos del elemento raíz, **<xs:schema>** en este caso. En el ejemplo que estamos tratando, solamente existe un elemento global: **<xs:element name="marcadores">**.
- Los elementos locales son el resto de elementos.

Cuando se define un espacio de nombres, los elementos globales tienen que estar calificados, obligatoriamente.

2.3 elementFormDefault="unqualified"

EJEMPLO En el supuesto de que el valor del atributo **elementFormDefault** fuese **"unqualified"**, para que ***“marcadores.xml”*** fuese válido, se podría escribir algo similar a:

```
<?xml version="1.0" encoding="UTF-8"?>
<mar:marcadores xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:mar="http://www.informatica.com/marcadores"
 xsi:schemaLocation="http://www.informatica.com/marcadores marcadores.xsd">
  <pagina>
   <nombre>informatica</nombre>
   <descripcion>Tutoriales de informática.</descripcion>
   <url>http://www.informatica.com/</url>
  </pagina>
```

```
 <pagina>
  <nombre>Wikipedia</nombre>
  <descripcion>La enciclopedia libre.</descripcion>
  <url>http://www.wikipedia.org/</url>
 </pagina>
 <pagina>
  <nombre>W3C</nombre>
  <descripcion>World Wide Web Consortium.</descripcion>
  <url>http://www.w3.org/</url>
 </pagina>
</mar:marcadores>
```

- **"unqualified"** es el valor por defecto del atributo **elementFormDefault**.

2.3.1 elementFormDefault="qualified"

EJEMPLO Si el atributo **elementFormDefault** se definiese **"qualified"**, también sería válido el siguiente documento XML:

```
<?xml version="1.0" encoding="UTF-8"?>
<marcadores xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns="http://www.informatica.com/marcadores"
 xsi:schemaLocation="http://www.informatica.com/marcadores/marcadores.xsd">
 <pagina>
  <nombre>informatica</nombre>
  <descripcion>Tutoriales de informática.</descripcion>
  <url>http://www.informatica.com/</url>
 </pagina>
 <pagina>
  <nombre>Wikipedia</nombre>
  <descripcion>La enciclopedia libre.</descripcion>
  <url>http://www.wikipedia.org/</url>
 </pagina>
 <pagina>
  <nombre>W3C</nombre>
  <descripcion>World Wide Web Consortium.</descripcion>
  <url>http://www.w3.org/</url>
```

```
  </pagina>
</marcadores>
```

Ahora, ningún elemento lleva prefijo, al igual que el espacio de nombres al que pertenecen:

xmlns="http://www.informatica.com/marcadores"

Es qualified porque aunque no se escriba y se muestre todos los elementos pertenecen al sistema de nombres "**http://www.informatica.com/marcadores"**

3 Elementos simples

Los elementos simples solamente pueden contener texto (caracteres). Dicho de otro modo, los elementos simples no pueden contener a otro u otros elementos (hijos), ni tampoco pueden tener atributos. Ahora bien, el texto contenido en un elemento simple, puede ser de diferentes tipos de datos predefinidos en ***W3C XML Schema*** o definidos por el usuario (programador).

Los tipos de datos predefinidos pueden ser primitivos (string, boolean, decimal...) o derivados de estos (integer, ID, IDREF...). Véase en el siguiente enlace, una imagen donde se puede ver la relación que existe entre todos ellos:

- https://goo.gl/lK2l2I: este enlace lleva a una carpeta compartida de Google Drive –pública en la Web– donde se puede acceder a la imagen ***"w3c-xml-schema-type-hierarchy.gif"***.

Para definir un elemento simple se puede utilizar la siguiente sintaxis:

```
<xs:element name="nombre_del_elemento" type="tipo_de_dato"/>
```

EJEMPLO Para los siguientes elementos XML:

```
<nombre>Elsa</nombre>
<edad>23</edad>
```

Sus definiciones pueden ser:

```
<xs:element name="nombre" type="xs:string"/>
<xs:element name="edad" type="xs:integer"/>
```

3.1 Tipos de declaración de elementos simples (fixed, default)

Si se quiere indicar que un valor es fijo (fixed), se puede escribir, por ejemplo:

```
<xs:element name="mes" type="xs:string" fixed="agosto"/>
```

También, se puede especificar un valor por defecto (default), por ejemplo, tecleando:

```
<xs:element name="mes" type="xs:string" default="agosto"/>
```

4 Atributos

Para definir un atributo se puede emplear la siguiente sintaxis:
<xs:attribute name="nombre_del_atributo" type="tipo_de_dato"/>

EJEMPLO Para el elemento "curso" siguiente, donde aparece el atributo "grupo":

```
<curso grupo="B">2</curso>
```

Sus definiciones pueden ser:

```
<xs:element name="curso" type="xs:integer"/>
<xs:attribute name="grupo" type="xs:string"/>
```

- Todos los atributos pueden tomar por valor tipos simples.
- Por otra parte, cuando un elemento tiene al menos un atributo –como es el caso del elemento "curso" en este ejemplo– dicho elemento se dice que es complejo.

4.1 Tipos de declaración de atributos (fixed, default, optional, required)

Para indicar que el valor de un atributo es fijo (**fixed**), es posible escribir, por ejemplo:

```
<xs:attribute name="grupo" type="xs:string" fixed="B"/>
```

Para especificar el valor por defecto (**default**) de un atributo, se puede escribir:

```
<xs:attribute name="grupo" type="xs:string" default="B"/>
```

Para indicar que un atributo es obligatorio (**required**) escribirlo, se puede teclear:

```
<xs:attribute name="grupo" type="xs:string" use="required"/>
```

Por defecto, si no se indica nada, el atributo será opcional (**optional**).

5 Restricciones (facetas)

XML Schema permite definir restricciones a los posibles valores de los tipos de datos. Dichas restricciones se pueden establecer en diferentes aspectos, llamados facetas.

Dicho de otro modo, las facetas permiten definir restricciones sobre los posibles valores de atributos o elementos. Las facetas que pueden utilizarse son:

Facetas de XSD	
Faceta	***Descripción***
xs:length	Especifica **una longitud fija.**
xs:minLength	Especifica **una longitud mínima.**
xs:maxLength	Especifica **una longitud máxima.**
xs:pattern	Especifica **un** patrón **de** caracteres **admitidos.**
xs:enumeration	Especifica **una lista de valores admitidos.**
xs:whiteSpace	Especifica **cómo se debe** tratar **a los posibles espacios en blanco, las tabulaciones, los saltos de línea y los** retornos **de** carro **que puedan** aparecer.
xs:maxInclusive	Especifica **que el valor debe ser menor o igual que el indicado.**
xs:maxExclusive	Especifica **que el valor debe ser menor que el indicado.**
xs:minExclusive	Especifica **que el valor debe ser mayor que el indicado.**
xs:minInclusive	Especifica **que el valor debe ser mayor o igual que el indicado.**
xs:totalDigits	Especifica **el número máximo de dígitos que puede tener un número.**
xs:fractionDigits	Especifica **el número máximo de decimales que puede tener un número.**

Seguidamente, se muestran algunos ejemplos de restricciones definidas con una o más facetas:

5.1 xs:minExclusive y xs:maxInclusive

EJEMPLO En el siguiente código se define un elemento llamado “mes” con la restricción de que el valor que tome no pueda ser menor que 1 ni mayor que 12:

```
<xs:element name="mes">
    <xs:simpleType>
    <xs:restriction base="xs:integer">
        <xs:minInclusive value="1"/>
        <xs:maxInclusive value="12"/> </xs:restriction>
    </xs:simpleType>
</xs:element>
```
xs:simpleType permite definir un tipo simple y especificar sus restricciones.

- **xs:restriction** sirve para definir restricciones de un **xs:simpleType** (como se ha hecho en este ejemplo). También sirve para definir restricciones de un **xs:simpleContent** o de un **xs:complexContent.** Estos elementos se estudiarán más adelante.
- En el atributo **base** se indica el tipo de dato a partir del cual se define la restricción.
- **xs:minInclusive** sirve para especificar que el valor debe ser mayor o igual que el indicado en su atributo **value,** (en este caso, mayor o igual que 1).
- **xs:maxInclusive** sirve para especificar que el valor debe ser menor o igual que el indicado en su atributo **value,** (en este caso, menor o igual que 12).

También se podría haber escrito:

```
<xs:element name="mes" type="numeroMes"/>
    <xs:simpleType name="numeroMes">
        <xs:restriction base="xs:integer">
            <xs:minInclusive value="1"/>
            <xs:maxInclusive value="12"/>
        </xs:restriction>
    </xs:simpleType>
```

Haciendo esto, el tipo numeroMes definido, podría ser utilizado por otros elementos, ya que, no está contenido en el elemento “mes”.

5.2 xs:enumeration

EJEMPLO En el siguiente ejemplo se define un elemento llamado “color” con la restricción de que los únicos valores admitidos son: **"verde", "amarillo" y "rojo".**

```
<xs:element name="color">
	<xs:simpleType>
		<xs:restriction base="xs:string">
			<xs:enumeration value="verde"/>
			<xs:enumerationvalue="amarillo"/>
			<xs:enumeration value="rojo"/>
		</xs:restriction>
	</xs:simpleType>
</xs:element>
```

- **xs:enumeration** sirve para definir una lista de valores admitidos.

5.3 xs:pattern

EJEMPLO En el siguiente ejemplo se define un elemento llamado “letra” con la restricción de que el único valor admitido es una de las letras minúsculas de la "a" a la "z":

```
<xs:element name="letra">
	<xs:simpleType>
		<xs:restriction base="xs:string">
			<xs:pattern value="[a-z]"/>
		</xs:restriction>
	 </xs:simpleType>
</xs:element>
```

- **xs:pattern** sirve para definir un patrón de caracteres admitidos (en este caso se admite una única letra minúscula de la "a" a la "z"). El valor del patrón tiene que ser una expresión regular.

5.4 xs:length

EJEMPLO En el siguiente ejemplo se define un elemento llamado “clave” con la restricción de que su valor tiene que ser una cadena de, exactamente, doce caracteres:

```
<xs:element name="clave">
	<xs:simpleType>
		<xs:restriction base="xs:string">
			<xs:length value="12"/>
		</xs:restriction>
	</xs:simpleType>
</xs:element>
```

- xs:length sirve para especificar una longitud fija.

5.5 xs:whiteSpace

EJEMPLO En el siguiente ejemplo se define un elemento llamado “dirección” con la restricción de que los espacios en blanco, las tabulaciones, los saltos de línea y los retornos de carro que aparezcan en él, se deben mantener (**preserve**):

```
<xs:element name="direccion">
	<xs:simpleType>
		<xs:restriction base="xs:string">
			<xs:whiteSpace value="preserve"/>
		</xs:restriction>
	</xs:simpleType>
</xs:element>
```

xs:whiteSpace sirve para especificar cómo se debe tratar a los posibles espacios en blanco, las tabulaciones, los saltos de línea y los retornos de carro que puedan aparecer.

En vez de **preserve** también se puede utilizar:

- **replace** para sustituir todas las tabulaciones, los saltos de línea y los retornos de carro por espacios en blanco.
- **collapse** para, después de reemplazar todas las tabulaciones, los saltos de línea y los retornos de carro por espacios en blanco, eliminar todos los espacios en blanco únicos y sustituir varios espacios en blanco seguidos por un único espacio en blanco.

6 Extensiones

xs:extension sirve para extender un elemento simpleType o complexType.

6.1 xs:extension (complexContent)

EJEMPLO Dado el siguiente documento XML:

```
<?xml version="1.0" encoding="UTF-8"?>
<fichas xsi:noNamespaceSchemaLocation="fichas.xsd"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
	<ficha numero="1">
		<nombre>Eva</nombre>
		<edad>25</edad>
		<ciudad>París</ciudad>
		<pais>Francia</pais>
	</ficha>
	<ficha numero="2">
		<nombre>Giovanni</nombre>
		<edad>26</edad>
		<ciudad>Florencia</ciudad>
		<pais>Italia</pais>
	</ficha>
</fichas>
```

Y el archivo “fichas.xsd” que permite validarlo:

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
	<xs:element name="fichas">
		<xs:complexType>
			<xs:sequence>
				<xs:element name="ficha" type="infoPersonaAmpliada"
maxOccurs="unbounded"/>
			</xs:sequence>
		</xs:complexType>
	</xs:element>
	<xs:complexType name="infoPersonaAmpliada">
		<xs:complexContent>
			<xs:extension base="infoPersona">
				<xs:sequence>
					<xs:element name="ciudad" type="xs:string"/>
					<xs:element name="pais" type="xs:string"/>
				</xs:sequence>
			</xs:extension>
		</xs:complexContent>
	</xs:complexType>
	<xs:complexType name="infoPersona">
		<xs:sequence>
			<xs:element name="nombre" type="xs:string"/>
			<xs:element name="edad" type="edadPersona"/>
		</xs:sequence>
		<xs:attribute name="numero" type="xs:integer"/>
	</xs:complexType>
	<xs:simpleType name="edadPersona">
		<xs:restriction base="xs:integer">
```

```
                    <xs:minExclusive value="-1"/>
                    <xs:maxExclusive value="131"/>
                </xs:restriction>
        </xs:simpleType>
</xs:schema>
```

- Obsérvese que, infoPersonaAmpliada se basa en infoPersona, añadiéndole dos elementos: “ciudad” y “país”.
- En cuanto a xs:complexContent, sirve para definir restricciones o extensiones a un tipo complejo (complexType).

6.2 xs:extension (simpleContent)

xs:simpleContent permite definir restricciones o extensiones a elementos que solo contienen datos, es decir, no contienen a otros elementos.

EJEMPLO El siguiente archivo **“precios.xsd”:**

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
  <xs:element name="precios">
      <xs:complexType>
        <xs:sequence>
          <xs:element name="precio" maxOccurs="unbounded">
              <xs:complexType>
                <xs:simpleContent>
                  <xs:extension base="xs:decimal">
                    <xs:attribute name="moneda" type="xs:string">
                   </xs:attribute>
                  </xs:extension>
                </xs:simpleContent>
             </xs:complexType>
```

```
            </xs:element>
          </xs:sequence>
        </xs:complexType>
    </xs:element>
</xs:schema>
```

Permite validar el siguiente documento XML:

```
<?xml version="1.0" encoding="UTF-8"?>
    <precios xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="precios.xsd">
        <precio moneda="Euro">5</precio>
        <precio moneda="Dólar">6.2</precio>
        <precio moneda="Libra esterlina">4.3</precio>
</precios>
```

Nótese que, utilizando xs:extension, al elemento “precio” se le ha incorporado el atributo moneda.

7 . Elementos complejos

Un elemento es complejo (complexType) cuando contiene uno o más elementos y/o atributos. De entre las posibles combinaciones de elementos y/o atributos que puede contener un elemento complejo (1 elemento y 0 atributos, 1 elemento y 1 atributo, 1 elemento y varios atributos, 0 elementos y 1 atributo...) cabe destacar las siguientes:

- Un elemento complejo puede estar vacío, es decir, no contener elementos ni texto, pero sí tener al menos un atributo.
- Un elemento complejo puede contener contenido mixto, es decir, contener uno o más elementos, además de texto. Por otra parte, podría tener atributos, o no.

7.1 Elemento vacío

EJEMPLO En el siguiente código se ha definido vacío el elemento “bola”, no pudiendo contener ni otros elementos ni texto. Ahora bien, véase que sí tiene un atributo, llamado numero:

```
<xs:element name="bola">
	<xs:complexType>
		<xs:attribute name="numero" type="numeroDeBola"/>
	</xs:complexType>
</xs:element>

<xs:simpleType name="numeroDeBola">
	<xs:restriction base="xs:positiveInteger">
		<xs:minInclusive value="1"/>
		<xs:maxExclusive value="90"/>
	</xs:restriction>
</xs:simpleType>
```

- **xs:positiveInteger** indica que el valor del atributo numero debe ser un número entero mayor que cero.

7.2 Contenido mixto

Fíjese que, en el siguiente código se ha definido el elemento “persona” de tipo complejo mixto (mixed="true"):

```
<xs:element name="persona">
	<xs:complexType mixed="true">
		<xs:sequence>
			<xs:element name="nombre" type="xs:string"/>
			<xs:element name="ciudad" type="xs:string"/>
			<xs:element name="edad" type="xs:positiveInteger"/>
		</xs:sequence>
	</xs:complexType>
</xs:element>
```

8 Indicadores

Los indicadores permiten establecer cómo se van a escribir –o utilizar– los elementos en un documento XML. Hay siete tipos de indicadores que se pueden clasificar en:

- **Indicadores de orden**: secuencia (sequence), todo (all) y elección (choice).
- **Indicadores de ocurrencia:** maxOccurs y minOccurs.
- **Indicadores de grupo:** de elementos (group) y de atributos (attributeGroup).

8.1 Indicadores de orden (xs:sequence, xs:all, xs:choice).

Mientras que **xs:sequence** sirve para especificar el orden en el que obligatoriamente deben aparecer los elementos hijo de un elemento, **xs:all** sirve para indicar que dichos elementos pueden aparecer en cualquier orden.

EJEMPLO El siguiente archivo "**lugar.xsd**":

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
  <xs:element name="lugar">
    <xs:complexType>
      <xs:sequence>
          <xs:element name="ciudad">
              <xs:complexType>
                <xs:all>
                        <xs:element name="nombre" type="xs:string"/>
                        <xs:element name="pais" type="xs:string"/>
              </xs:all>
           </xs:complexType>
        </xs:element>
     </xs:sequence>
   </xs:complexType>
  </xs:element>
</xs:schema>
```

Permite validar el siguiente documento XML:

```
<?xml version="1.0" encoding="UTF-8"?>
<lugar xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="lugar.xsd">
 <ciudad>
  <pais>Italia</pais>
  <nombre>Florencia</nombre>
 </ciudad>
</lugar>
```

Por otra parte, xs:choice sirve para especificar que solamente se permite escribir uno de los elementos hijo. Por ejemplo, en este caso, se podría utilizar para indicar que habría que elegir entre escribir el "nombre" o escribir el "país" de la "ciudad", pero no ambos.

8.2 Indicadores de ocurrencia (maxOccurs, minOccurs)

maxOccurs y **minOccurs** permiten establecer, respectivamente, el número máximo y mínimo de veces que puede aparecer un determinado elemento. El valor por defecto para **maxOccurs** y **minOccurs** es 1.

EJEMPLO Dado el siguiente documento XML ***"paises.xml"***:

```
<?xml version="1.0" encoding="UTF-8"?>
<paises xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="paises.xsd">
 <pais>
  <nombre>Argentina</nombre>
  <ciudad>Buenos Aires</ciudad>
  <ciudad>Rosario</ciudad>
 </pais>
 <pais>
  <nombre>México</nombre>
  <ciudad>Guadalajara</ciudad>
```

```
   <ciudad>Monterrey</ciudad>
   <ciudad>Cancún</ciudad>
   <ciudad>Mérida</ciudad>
   <ciudad>Ciudad de México</ciudad>
  </pais>
  <pais>
   <nombre>Colombia</nombre>
  </pais>
</paises>
```

Considerando que se quiere especificar que:

- “país” pueda aparecer una o ilimitadas veces.
- “nombre” tenga que escribirse obligatoriamente, y solo una vez, dentro de “país”.
- De cada “país” puedan escribirse de cero a cinco “ciudades”.

El código del archivo ***“paises.xsd”*** que permita validar ***“paises.xml”***, podría ser:

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
 <xs:element name="paises">
  <xs:complexType>
   <xs:sequence>
    <xs:element name="pais" maxOccurs="unbounded">
     <xs:complexType>
      <xs:sequence>
       <xs:element name="nombre" type="xs:string"/>
       <xs:element name="ciudad" type="xs:string" minOccurs="0" maxOccurs="5"/>
      </xs:sequence>
     </xs:complexType>
    </xs:element>
   </xs:sequence>
  </xs:complexType>
 </xs:element>
</xs:schema>
```

8.3 Indicadores de grupo (xs:group, xs:attributeGroup)

xs:group sirve para agrupar un conjunto de declaraciones de elementos relacionados.

EJEMPLO Dado el siguiente documento XML “personas.xml”:

```
<?xml version="1.0" encoding="UTF-8"?>
<personas xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="personas.xsd">
 <persona>
  <nombre>Eva</nombre>
  <edad>25</edad>
  <pais>Francia</pais>
  <telefono>999888777</telefono>
 </persona>
 <persona>
  <nombre>Giovanni</nombre>
  <edad>26</edad>
  <pais>Italia</pais>
  <telefono>111222333</telefono>
 </persona>
</personas>
```

Y el archivo “personas.xsd” que permite validarlo:

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">

 <xs:element name="personas">
  <xs:complexType>
   <xs:sequence>
    <xs:element name="persona" type="datosDePersona" maxOccurs="unbounded"/>
   </xs:sequence>
```

```
  </xs:complexType>
 </xs:element>

 <xs:complexType name="datosDePersona">
  <xs:sequence>
   <xs:group ref="datosBasicos"/>
   <xs:element name="telefono" type="xs:string"/>
  </xs:sequence>
 </xs:complexType>

 <xs:group name="datosBasicos">
  <xs:sequence>
   <xs:element name="nombre" type="xs:string"/>
   <xs:element name="edad" type="xs:positiveInteger"/>
   <xs:element name="pais" type="xs:string"/>
  </xs:sequence>
 </xs:group>

</xs:schema>
```

Obsérvese que, se ha definido el grupo **datosBasicos**, el cual ha sido incorporado a la definición del tipo complejo **datosDePersona**.

Del mismo modo, **attributeGroup** sirve para definir un grupo de atributos. Por ejemplo, para validar el siguiente documento XML:

```
<?xml version="1.0" encoding="UTF-8"?>
<personas xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="personas.xsd">
 <persona nombre="Eva" edad="25" pais="Francia"/>
 <persona nombre="Giovanni" edad="26" pais="Italia"/>
</personas>
```

Se puede escribir el siguiente código en el archivo ***"personas.xsd"***:

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">

 <xs:element name="personas">
  <xs:complexType>
   <xs:sequence>
    <xs:element name="persona" maxOccurs="unbounded">
     <xs:complexType>
      <xs:attributeGroup ref="datosDePersona"/>
     </xs:complexType>
    </xs:element>
   </xs:sequence>
  </xs:complexType>
 </xs:element>

 <xs:attributeGroup name="datosDePersona">
  <xs:attribute name="nombre" type="xs:string"/>
  <xs:attribute name="edad" type="xs:positiveInteger"/>
  <xs:attribute name="pais" type="xs:string"/>
 </xs:attributeGroup>
</xs:schema>
```

En este caso, se ha definido el grupo de atributos datosDePersona, el cual ha sido incorporado a la definición del elemento persona.