Lenguajes de Marcas y Sistemas de Información

UT6. XML: ESQUEMAS Y VOCABULARIOS

Mª JESÚS BRAVO 2024-2025

XML SCHEMA (XSD): INTRODUCCIÓN

Problemas de los DTD:

- Fueron concebidos para sistemas de publicación (SGML): contenidos textuales.
- No tienen tipado de elementos (declaraciones globales).
- Las cardinalidades son: 0, 1, infinito.
- Sintaxis especial y poco clara (no es XML).
- XML, por su orientación Web, tiene necesidades nuevas respecto a SGML.
- No permiten la reutilización sencilla de código.
- No pueden validar espacios de nombres.

XML SCHEMA (XSD): INTRODUCCIÓN

Un XML Schema:

- Define los elementos que pueden aparecer en un documento
- Define los atributos que pueden aparecer en un documento
- Define qué elementos son elementos hijos en la jerarquía
- Define el orden de los elementos hijos
- Define la cantidad de elementos hijos
- Define si un elemento es vacío (empty) o puede incluir texto
- Define los tipos de datos para los elementos y los atributos
- Define los valores por default y fijos para los elementos y los atributos

XML SCHEMA (XSD): INTRODUCCIÓN

Ventajas de XSD sobre DTD:

- Dan mayor control sobre la creación de documentos XML
- Utilizan XML como sintaxis
- Son extensibles a futuras adiciones
- Son más ricos y poderosos que los DTDs
- Soportan “Tipado de Datos”
- Soportan namespaces

XML SCHEMA (XSD): INTRODUCCIÓN

La documentación consta de tres partes:

- XML Schema Part 0: Primer (es un documento introductorio (no muy teórico) y con múltiples ejemplos)

  www.w3.org/TR/xmlschema-0/

- XML Schema Part 1: Structures (estructura XML de la especificación)

  www.w3.org/TR/xmlschema-1/

- XML Schema Part 2: Datatypes (otros recursos y herramientas)

  www.w3.org/TR/xmlschema-2/

CONCEPTOS BÁSICOS: ESTRUCTURA

- Un XML-Schema es un documento XML.
  - Por convenio llevan extensión ".xs" o ".**xsd**"
- Comienza con la declaración de documento XML.
- Utiliza la declaración del espacio de nombres de XML-Schema para sus meta-elementos. Se usan los prefijos xsd: o **xs:**
- La base de un archivo XSD es la siguiente:

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
<xs:element name="root-element">
  <!-- Aquí escribes tu código -->
</xs:element>
</xs:schema>
```

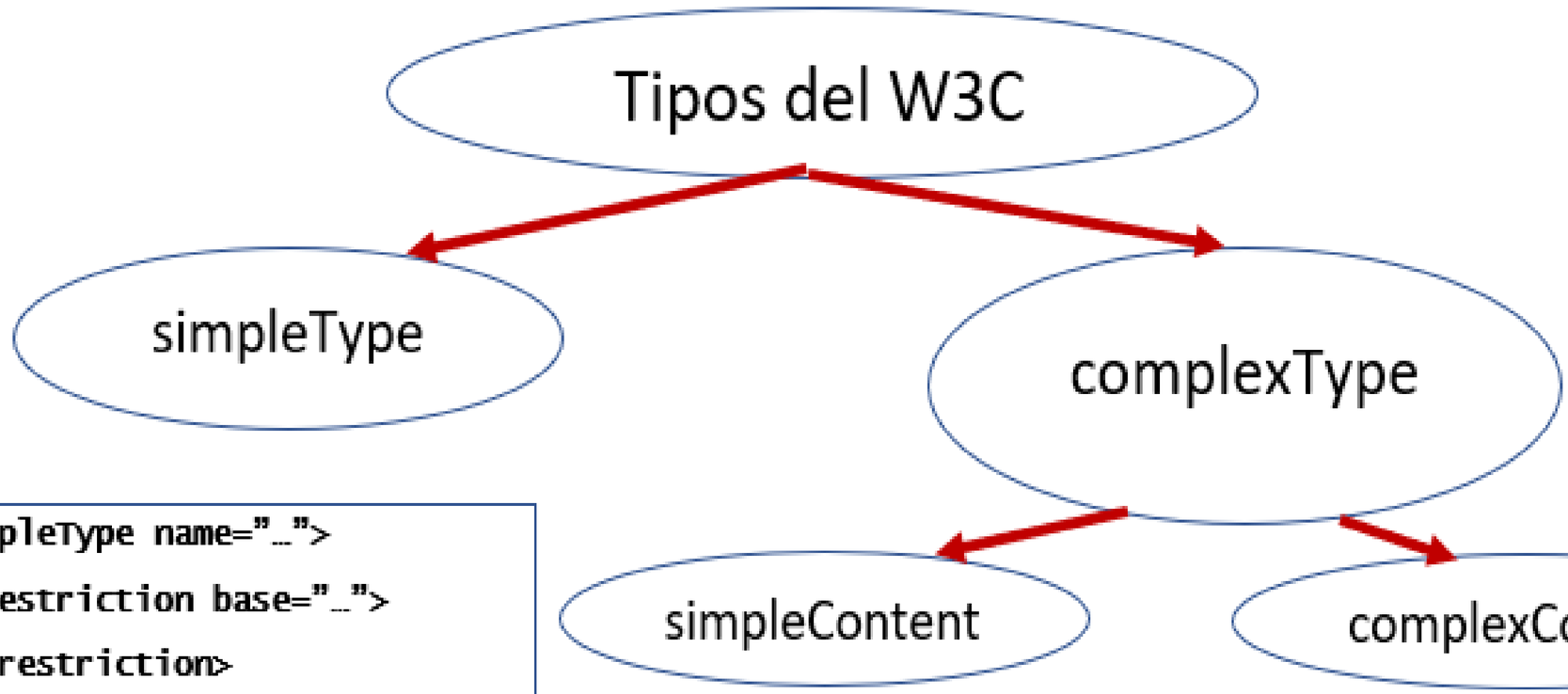
CÓMO VINCULAR UN ESQUEMA XSD A UN DOCUMENTO XML - EJEMPLO

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

CÓMO VINCULAR UN ESQUEMA XSD A UN DOCUMENTO XML - EJEMPLO

```
<?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>
```

TIPOS DE ELEMENTOS




```
<xsd:simpleType name="…">
  <xsd:restriction base="…">
  </xsd:restriction>
</xsd:simpleType>
```

```
<xsd:complexType name="…">
  <xsd:simpleContent>
   <xsd:extensión base="…">
   </xsd:extension>
  </xsd:simpleContent>
</xsd:complexType>
```

```
<xsd:complexType name="…">
  <xsd:complexContent>
     <xsd:restriction
           base="xsd:anyType">
      </xsd:restriction>
  </xsd:complexContent>
</xsd:complexType>
```

DEFINIR ELEMENTOS

Para definir un elemento, debemos tener claro que existen dos tipos de elementos **simples** y **complejos**.

- Un **elemento simple** no contiene atributos y no contiene otros elementos (como hijos).

```
<nombre>Pedro Pérez</nombre >
<edad>36</edad >
<fechanaci>1990-03-27</fechanaci >
```

- Para escribir su código XSD se pondrá: (especificamos el nombre del elemento y el tipo de dato)

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

DEFINIR ELEMENTOS

- Un **elemento complejo** contiene atributos y/o otros elementos.

```
<empleado rol="admin">
    <nombre>John</nombre>
    <apellido>Smith</apellido>
</empleado>
```

- El código XSD para validar este elemento será:

```
<xs:element name=“empleado">
  <xs:complexType>
    <xs:sequence>
     <xs:element name=“nombre" type="xs:string"/>
        <xs:element name=“apellido" type="xs:string"/>
    </xs:sequence>
    <xs:attribute name="rol" type="xs:string"/>
  </xs:complexType>
</xs:element>
```

ELEMENTO SIMPLE

```
<xs:element name="nombre_elemento" type="tipo_dato" [default="valor“]/>
```

- Tipos **predefinidos**: primitivos (string, boolean, decimal...) o derivados de éstos (integer, ID, IDREF...) Diagrama W3C

```
<xs:element name="cantidad" type="xs:integer"/>
```

- Tipos **personalizados**: extendiendo para tal fin los tipos de datos predefinidos.

Los nuevos tipos deben ser “derivados” desde un tipo base o bien predefinido u otro tipo simple existente. El tipo base se **restringe** utilizando un número de **facetas** para obtener el nuevo tipo.

```
<xs:element name="edad" type="tipoEdad"/>
 <xs:simpleType name="tipoEdad">
    <xs:restriction base="xs:integer">
       <xs:minInclusive value="16"/>
       <xs:maxInclusive value="65"/>
    </xs:restriction>
 </xs:simpleType>
```

DERIVACIÓN DE TIPOS SIMPLES

Existen tres mecanismos de derivación:

- Por **restricción**: se restringe el rango de valores del tipo de datos.
- Por **enumeración** (lista): el rango se describe explícitamente como la lista de valores válidos.
- Por **unión** : mediante unión de dos o más tipos de datos.

NOTA: El mecanismo de **extensión** sólo se aplica a los tipos de datos complejos.

TIPOS SIMPLES – RESTRICCIONES.

Restricción	Descripción
enumeration	Define una lista de valores aceptables
fractionDigits	Especifica el número máximo de decimales permitidos. Debe ser igual o mayor que cero
Length	Especifica el número exacto de caracteres o elementos de listas permitidas. Debe ser igual o mayor que cero
maxExclusive	Especifica los límites superiores para los valores numéricos (el valor debe ser inferior a este valor)
maxInclusive	Especifica los límites superiores para valores numéricos (el valor debe ser inferior o igual a este valor)
maxLength	Especifica el número máximo de caracteres permitidos o elementos de lista. Debe ser igual o mayor que cero
minExclusive	Especifica los límites más bajos para los valores numéricos (el valor debe ser mayor que este valor)
minInclusive	Especifica los límites más bajos para los valores numéricos (el valor debe ser mayor que o igual a este valor)
minLength	Especifica el número mínimo de caracteres o elementos de listas permitidas. Debe ser igual o mayor que cero
pattern	Define la secuencia exacta de caracteres que son aceptables
totalDigits	Especifica el número exacto de dígitos permitidos. Debe ser mayor que cero
whiteSpace	Especifica el espacio en blanco (saltos de línea, tabuladores, espacios y retornos de carro) se maneja

LIMITACIÓN DE VALORES NUMÉRICOS

Imposición de valores máximos y mínimos:

```
<xs:simpleType name="Tipo1">
 <xs:restriction base="xs:integer">
        <xs:maxExclusive value="2000" />
        <xs:minExclusive value ="500"  />
 </xs:restriction>
</xs:simpleType>
```

Entero entre 1 y 31:

```
<xs:simpleType name="diames">
    <xs:restriction base="xs:integer">
       <xs:minInclusive value="1" />
       <xs:maxInclusive value="31" />
    </xs:restriction>
</xs:simpleType>
```

LIMITACIÓN DE LA LONGITUD DE LAS CADENAS

Usando el elemento xs:lenght:

```
<xs:simpleType name="TipoEstado">
   <xs:restriction base="xs:string">
      <xs:length value="2" />
   </xs:restriction>
</xs:simpleType>
```

Podemos establecer límites superiores e inferiores en una cadena de texto:

```
<xs:simpleType name ="TipoEstado2">
   <xs:restriction base="xs:string">
      <xs:minLength value="2" />
      <xs:maxLength value="13" />
   </xs:restriction>
</xs:simpleType>
```

UTILIZANDO ENUMERACIONES

Para limitar los posibles valores que puede tomar una etiqueta, indicando en cada una el valor posible:

```
<xs:simpleType  name="cod_USA">
  <xs:restriction base="xs:string">
     <xs:enumeration value="AK"/>
     <xs:enumeration value="AL"/>
     <xs:enumeration value="AR"/>
  </xs:restriction>
</xs:simpleType>
```

FORZAR TEXTO PARA QUE SE ADAPTE A UN PATRÓN

Expresiones regulares

- **()** agrupación subexpresión
- **|** uno u otro
- **[ ]** intervalo
- **[^]** no considera intervalo
- **.** Cualquier carácter exceptuando salto de línea
- ***** cero o más ocurrencia Ej: (ab)*: cualquier cosa entre paréntesis puede aparecer cero o más veces.
- **+** 1 o más ocurrencias Ej:(ab)+: cualquier cosa entre paréntesis puede aparecer una o más veces.
- **?** aparece 0 ó 1 vez. Ej: a?
- **a{n}**: "a" puede aparecer en una cadena n veces. Puede ponerse {2,6} -> 2,3,4,5,6
- **\s** un carácter de espacio en blanco (espacio o tabulación)
- **\S** cualquier carácter no espacio
- **\d** cualquier carácter numérico
- **\D** cualquier carácter que no sea dígito
- **\w** cualquier carácter alfanumérico (letra número o carácter de subrayado)

EJEMPLOS EXPRESIONES REGULARES

a*x	x, ax, aax, aaax ....
a?x	ax, x
a+x	ax, aax, aaax ....
(a|b)+x	ax, bx, aax, abx, bax, bbx, aaax, aabx, abax, abbx, baax, babx, bbax, bbbx, aaaax ....
[abcde]x	ax, bx, cx, dx, ex
[a-e]x	ax, bx, cx, dx, ex
[-ae]x	-x, ax, ex
[ae-]x	ax, ex, -x
[^0-9]x	cualquier carácter no-dígito seguido del carácter x
\Dx	cualquier carácter no-dígito seguido del carácter x
.x	cualquier carácter seguido del carácter x
.*abc.*	1x2abc, abc1x2, z3456abchooray ....
ab{2}x	abbx
ab{2,4}x	abbx, abbbx, abbbbx
ab{2,}x	abbx, abbbx, abbbbx ....
(ab){2}x	ababx

EJEMPLOS EXPRESIONES REGULARES

Sólo permite 3 dígitos seguidos por un guión y dos letras mayúsculas

```
<xs:simpleType name="sku">
   <xs:restriction  base="xs:string">
      <xs:pattern value="\d{3}-[A-Z]{2}"/>
   </xs:restriction>
</xs:simpleType>
```

Qué permite?:

```
<xs:pattern value="(\d{3})\s\d{3} - \d{4}"/>
```

Para una cadena de 8 caracteres que debe seguir el patrón ddd-dddd

```
   <xs:restriction  base="xs:string">
      <xs:length value="8"/>
      <xs:pattern value="\d{3}-\d{4}"/>
   </xs:restriction>
```

TIPOS COMPLEJOS

Los tipos de datos complejos permiten especificar la estructura interna de un elemento XML, es decir, sus atributos y elementos hijos.

- **Contenido vacío**: puede contener atributos
- **Contenido texto con atributos**: un elemento con atributos, pero sin elementos hijos
- **Contenido de elementos**: contiene otros elementos hijos y también pueden contener atributos

TIPOS COMPLEJOS

Los tipos de datos complejos permiten especificar la estructura interna de un elemento XML, es decir, sus atributos y elementos hijos.

- **Contenido vacío**: puede contener atributos

```
<xs:attribute name="nombreAtributo" type="tipoSimple"
      [use="uso"] [default="valor"] [fixed="valor"]/>
```

use: Indica si la existencia del atributo es opcional (optional), obligatoria (required) o prohibida (prohibited). Por defecto opcional.

```
<xs:complexType name="antiguedad" >
   <xs:attribute name="edad" type="xs:int"/>
</xs:complexType>
```

- Si es vacío, pero sin atributos:

```
<xs:complexType name="tipovacio" />
```

TIPOS COMPLEJOS

Elemento vacío. **Ejemplo** bola bingo: definimos un elemento vacío **bola**, no pudiendo contener ni otros elementos ni texto. Ahora bien, 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>
```

TIPOS COMPLEJOS

- **Contenido texto con atributos**: un elemento con atributos, pero al no tener elementos hijos se indica <xsd:simpleContent>

```
<xs:complexType  name="tipoPuesto">
  <xs:simpleContent>
     <xs:extension base="xs:string">
        <xs:attribute name="categoria"
type="xs:string" use="required"/>
     </xs:extension>
  </xs:simpleContent>
</xs:complexType>
```

XML:

```
<puesto categoria="senior">programador</puesto>
```

TIPOS COMPLEJOS

- **Contenido de elementos**: contiene otros elementos (sequence, choice…) y también pueden contener atributos

```
<lugar>
    <ciudad>
        <pais>Italia</pais>
        <nombre>Florencia</nombre>
    </ciudad>
</lugar>
```

Indicadores en XSD: 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**).

TIPOS COMPLEJOS- INDICADORES DE ORDEN

- 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 hijos de un elemento, **xs:all** sirve para indicar que dichos elementos pueden aparecer en cualquier orden.

```
<xs:element name="lugar">
    <xs:complexType>
        <xs:sequence>
            <xs:element name="ciudad" type="tipoCiudad"/>
        </xs:sequence>
    </xs:complexType>
</xs:element>
<xs:complexType name="tipoCiudad">
    <xs:all>
        <xs:element name="nombre" type="xs:string"/>
        <xs:element name="pais" type="xs:string"/>
    </xs:all>
</xs:complexType>
```

TIPOS COMPLEJOS- INDICADORES DE ORDEN

- Indicadores de orden (xs:**sequence**, **xs:all**, **xs:choice**)

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

Ejemplo2:

```
<xsd:complexType name="vehiculoMotor">
    <xsd:choice>
        <xsd:element name="coche" type="xsd:string"/>
        <xsd:element name="moto" type="xsd:string"/>
        <xsd:element name="furgoneta" type="xsd:string"/>
        <xsd:element name="camion" type="xsd:string"/>
    </xsd:choice>
 </xsd:complexType>
```

TIPOS COMPLEJOS-INDICADORES DE OCURRENCIA

- Indicadores de ocurrencia (**maxOccurs**, **minOccurs**) Permiten establecer, respectivamente, el número máximo y mínimo de veces que puede aparecer un determinado elemento. El valor por defecto para ambos es 1.

```
<?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>
```

- "país" puede aparecer una o ilimitadas veces.
- "nombre" tiene que escribirse obligatoriamente, y solo una vez, dentro de "país".
- De cada "país" puedan escribirse de cero a cinco "ciudades".

TIPOS COMPLEJOS-INDICADORES DE OCURRENCIA

- Indicadores de ocurrencia (**maxOccurs, minOccurs**)

```
<?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>
```

TIPOS COMPLEJOS

- **Tipos complejos con contenido mixto**

Un elemento XML **mixto** es el que contiene texto y otros elementos:

```
<carta> Querido Sr.
  <nombre>Juan Abad</nombre>.
  su pedido <idpedido>1032</idpedido>
  se mandará en la fecha
  <fecha>2014-07-13</fecha>.
</carta>
```

TIPOS COMPLEJOS

- **Tipos complejos con contenido mixto**

Para permitir que los datos de texto aparezcan entre los elementos hijos de la "carta", el atributo mixed debe estar establecido en "true". La etiqueta <xs:sequence> significa que los elementos definidos (nombre, idpedido y fecha) debe aparecer en este orden en el interior del elemento "carta".

```
<xs:element name="carta">
<xs:complexType mixed="true">
  <xs:sequence>
     <xs:element name="nombre" type="xs:string"/>
     <xs:element name="idpedido"
           type="xs:positiveInteger"/>
     <xs:element name="fecha" type="xs:date"/>
  </xs:sequence>
</xs:complexType>
</xs:element>
```

TIPOS ANÓNIMOS Y CON NOMBRE

A la hora de definir tipos complejos puede optarse por dos formas: tipos **anónimos** y **con nombre**

- **Tipos anónimos:** no tienen un nombre de tipo, son más legibles pero no pueden utilizarse más tarde

Para permitir que los datos de texto aparezcan entre los elementos hijos de la "carta", el atributo mixed debe estar establecido en "true". La etiqueta <xs:sequence> significa que los elementos definidos (nombre, idpedido y fecha) debe aparecer en este orden en el interior del elemento "carta".

```
<xsd:element name ="alumno">
        <xsd:complexType>
              <xsd:sequence>
                    <xsd:element name="nombre"
type="xsd:string"/>
                  <xsd:element name="apellido"
type="xsd:string"/>
              </xsd:sequence>
      </xsd:complexType>
</xsd:element>
```

TIPOS ANÓNIMOS Y CON NOMBRE

- **Tipos con nombre:** se complica un poco la sintaxis, pero pueden reutilizarse más tarde

```
<xsd:element name ="alumno" type="TipoPersona"/>
……
<xsd:complexType name="TipoPersona">
    <xsd:sequence>
        <xsd:element name="nombre" type="xsd:string"/>
        <xsd:element name="apellido"
type="xsd:string"/>
    </xsd:sequence>
</xsd:complexType>
```

REFERENCIAS

Se puede hacer referencia a un elemento que ya se ha definido, aunque no se haya especificado un tipo de dato con nombre, utilizando la etiqueta ref.

```
<xs:element name="nota">
  <xs:complexType>
    <xs:sequence>
      <xs:element ref="to"/>
      <xs:element ref="from"/>
      <xs:element ref=“cabecera"/>
      <xs:element ref="body"/>
    </xs:sequence>
  </xs:complexType>
</xs:element>

<xs:element name="to" type="xs:string"/>
<xs:element name="from" type="xs:string"/>
<xs:element name="cabecera" type="xs:string"/>
<xs:element name="body" type="xs:string"/>
```

VER EJEMPLOS