

XML Schema

DOCUMENTOS XML

Mecanismos de validación

- **DTD** (Definición de Tipo de Documento)
- **XML Schema** (Definición de esquema XML)

LIMITACIONES DE LOS DTD

- **Utilizan una sintaxis propia (no son XML)**
- **No podemos especificar mucho sobre los tipos de datos/atributos:**
 - ¿Un elemento que sólo contenga valores numéricos?
 - ¿Un elemento que contenga una fecha en formato correcto?
 - ¿Un elemento como NIF, que siga un patrón de 8 dígitos + 1 letra?
- **Orden rígido en los elementos**
 - (DNI, nombre, apellidos) : ¿podemos alterar el orden?
- **Los elementos no permiten valores enumerados ni valores por defecto**
- **Los elementos no permiten valores identificativos**
- **Cardinalidad muy limitada:**
 - opcional/obligatorio; simple/múltiple

EJEMPLO

```
<!ELEMENT garderia (neno*)>
<!ELEMENT neno (nome, dataNacimiento, peso, altura, vacunas)>
<!ATTLIST neno numeroExpedente CDATA #REQUIRED>
<!ELEMENT nome (#PCDATA)>
<!ELEMENT dataNacimiento (#PCDATA)>
<!ELEMENT peso (#PCDATA)>
<!ELEMENT altura (#PCDATA)>
<!ELEMENT vacunas (#PCDATA)>
```

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE garderia SYSTEM "garderia.dtd">
<garderia>
  <neno numeroExpedente="2344">
    <nome>Noa Blanco Coello</nome>
    <dataNacimiento>2010-10-12</dataNacimiento>
    <peso>13.5</peso>
    <altura>78</altura>
    <vacunas>true</vacunas>
  </neno>
</garderia>
```

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE garderia SYSTEM "garderia.dtd">
<garderia>
  <neno numeroExpedente="jejeje">
    <nome>1111/?@</nome>
    <dataNacimiento>00</dataNacimiento>
    <peso>zzzz</peso>
    <altura>no</altura>
    <vacunas>???</vacunas>
  </neno>
</garderia>
```

EJEMPLO CON XMLSchema

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
  <xs:element name="garderia">
    <xs:complexType>
      <xs:sequence>
        <xs:element name="neno" minOccurs="1" maxOccurs="30">
          <xs:complexType>
            <xs:sequence>
              <xs:element name="nome" type="xs:string"/>
              <xs:element name="dataNacimiento" type="xs:date"/>
              <xs:element name="peso" type="xs:decimal"/>
              <xs:element name="altura" type="xs:unsignedByte"/>
              <xs:element name="vacunas" type="xs:boolean"/>
            </xs:sequence>
            <xs:attribute name="numeroExpedente" type="xs:unsignedShort" use="required"/>
          </xs:complexType>
        </xs:element>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
</xs:schema>
```

EJEMPLO CON XMLSchema

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
  <xs:element name="garderia">
    <xs:complexType>
      <xs:sequence>
        <xs:element name="neno" minOccurs="1" maxOccurs="50" type="tipoNeno"/>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
  <xs:complexType name="tipoNeno">
    <xs:sequence>
      <xs:element name="nome" type="xs:string"/>
      <xs:element name="dataNacimiento" type="xs:date"/>
      <xs:element name="peso" type="xs:decimal"/>
      <xs:element name="altura" type="xs:unsignedByte"/>
      <xs:element name="vacunas" type="xs:boolean"/>
    </xs:sequence>
    <xs:attribute name="numeroExpedente" type="xs:unsignedShort"/>
  </xs:complexType>
</xs:schema>
```

DETECCION DE ERROR DE TIPOS

```
1 <?xml version="1.0" encoding="UTF-8"?>
2 <garderia xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
3   |xsi:noNamespaceSchemaLocation="garderia2.xsd">
4   |<nenos numeroExpedente="2344">
5   |   |<nome>Noa Blanco Coello</nome>
6   |   |<dataNascimento>2010-10-12</dataNascimento>
7   |   |<peso>13.5</peso>
8   |   |<altura>-78</altura>
9   |   |<vacunas>true</vacunas>
10  |</nenos>
```

Parado

Error at line 8, column 25: value '-78' must be greater than or equal to minInclusive facet value '0'

DETECCION DE ERROR DE TIPOS

```

1 <?xml version="1.0" encoding="UTF-8"?>
2 <garderia xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
3   |       |       |       |       |       |       |       |       |
4   |       |       |       |       |       |       |       |       |
5   |       |       |       |       |       |       |       |       |
6   |       |       |       |       |       |       |       |       |
7   |       |       |       |       |       |       |       |       |
8   |       |       |       |       |       |       |       |       |
9   |       |       |       |       |       |       |       |       |
10  |       |       |       |       |       |       |       |       |
11 </garderia>

```

Error at line 9, column 29: value 'truee' is invalid boolean

ENLACE A UN ESQUEMA

(Obviamos el espacio de nombres)

¿Cómo se define el XMLSchema?

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
  <!--... resto de la definición del esquema ...-->
</xs:schema>
```

¿Cómo se enlaza desde el documento XML?

```
<?xml version="1.0" encoding="UTF-8"?>
<elemento_raiz xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:noNamespaceSchemaLocation="nombre_del_archivo.xsd">
  <!--... resto de la definición del archivo XML ...-->
</elemento_raiz>
```

ENLACE A UN ESQUEMA

XMLSchema

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
  <!-- resto de la definición del esquema -->
</xs:schema>
```

archivo .xsd

"norma"

"molde"

"clase"...

Documentos \approx instancias XML

```
<?xml version="1.0" encoding="UTF-8"?>
<raiz xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:noNamespaceSchemaLocation="fichero.xsd">
  <!-- resto de la definición del archivo .xml -->
</raiz>
```

```
<?xml version="1.0" encoding="UTF-8"?>
<raiz xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:noNamespaceSchemaLocation="fichero.xsd">
  <!-- resto de la definición del archivo .xml -->
</raiz>
```

```
<?xml version="1.0" encoding="UTF-8"?>
<raiz xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:noNamespaceSchemaLocation="fichero.xsd">
  <!-- resto de la definición del archivo .xml -->
</raiz>
```

archivos .xml

Respetan la "norma" contenida en el esquema

TIPOS DE ELEMENTOS

ELEMENTOS SIMPLES

- Sólo contienen información
- No tienen elementos descendientes ni tampoco atributos
- Sintaxis
`<xs:element name="nnnn" type="xs:tipo"/>`
- Ejemplo
`<xs:element name="nome" type="xs:string"/>`
- Pueden tener un valor por defecto
`<xs:element name="cidade" type="xs:string" default="Vigo"/>`
- Pueden tener un valor por fijo (no se puede cambiar):
`<xs:element name="color" type="xs:string" fixed="blanco"/>`

TIPOS DE ELEMENTOS

ELEMENTOS SIMPLES (continuación)

- **Cardinalidad**

Se indica mediante los atributos **minOccurs** y **maxOccurs**

- **Ejemplo**

```
<xs:element name="neno" minOccurs="1" maxOccurs="30">
```

- Valor por defecto para ambos atributos es 1 → obligatorio y único
- Elemento opcional → minOccurs = 0
- Elemento múltiple → maxOccurs = "unbounded"

Tipos simples predefinidos

- **Existen múltiples tipos** (ver página 7 y 8 del documento base)
- **Algunos son:**
 - **De texto**
string
 - **Numéricos**
decimal
integer
negativeInteger – positiveInteger – nonPositiveInteger (negativos + 0)
byte
 - **Fecha y hora**
date
time
 - **Booleanos**
boolean

ATRIBUTOS

- Se declaran igual que los elementos simples

`<xs:attribute name="nombreAtributo" type="xs:tipoAtributo"/>`

- Ejemplo:

`<xs:attribute name="numeroExpedente" type="xs:unsignedShort"/>`

- También pueden tener valores por defecto y valores fijos

`<xs:attribute name="provincia" type="xs:NMTOKEN" default="Lugo"/>`

`<xs:attribute name="provincia" type="xs:NMTOKEN" fixed="Lugo"/>`

- Por defecto, son opcionales.

- Se puede variar la obligatoriedad mediante el atributo “use”

`<xs:attribute name="numExpedente" type="xs:unsignedShort" use="optional"/>`

`<xs:attribute name="numExpedente" type="xs:unsignedShort" use="required"/>`

TIPOS DE ELEMENTOS

ELEMENTOS COMPLEJOS

- Pueden contener elementos descendentes y/o atributos
- Primero se indican los elementos hijos y después los atributos (si existen)
- El nombre del elemento se indica como valor del atributo “name” y se emplea la indicación <xs:complexType>
- **Ejemplo** (cinco elementos hijos y además un atributo)

```
<xs:element name="neno" >
  <xs:complexType>
    <xs:sequence>
      <xs:element name="nome" type="xs:string"/>
      <xs:element name="dataNacimiento" type="xs:date"/>
      <xs:element name="peso" type="xs:decimal"/>
      <xs:element name="altura" type="xs:unsignedByte"/>
      <xs:element name="vacunas" type="xs:boolean"/>
    </xs:sequence>
    <xs:attribute name="numeroExpediente" type="xs:unsignedShort" use="required"/>
  </xs:complexType>
</xs:element>
```

INDICADORES DE ORDEN

Existen 3 indicadores de orden

- **<xs:sequence>**
Deben aparecer todos los elementos en el mismo orden indicado
- **<xs:choice>**
Lista de elementos alternativos (sólo puede aparecer uno de los indicados)
- **<xs:all>**
Lista de elementos que pueden aparecer en cualquier orden.
No es muy recomendable pero, si aparece, debe aparecer como **hijo único** en el nivel más alto del modelo de contenido (es decir, no puede aparecer dentro de <xs:sequence>, <xs:choice> o <xs:all>)

EJEMPLO - <xs:sequence>

```
<xs:element name="cliente" maxOccurs="unbounded">
  <xs:complexType>
    <xs:sequence>
      <xs:element name="cif" type="xs:string"/>
      <xs:element name="empresa" type="xs:string"/>
      <xs:element name="localidad" type="xs:string"/>
    </xs:sequence>
  </xs:complexType>
</xs:element>
```

```
<cliente>
  <cif>A11111111</cif>
  <empresa>Manuel Barros</empresa>
  <localidad>PONTEAREAS</localidad>
</cliente>
<cliente>
  <cif>P22222222</cif>
  <empresa>HERMANOS GARCÍA, S.A.</empresa>
  <localidad>LUGO</localidad>
</cliente>
```

EJEMPLO - <xs:choice>

```
<xs:element name="cliente" maxOccurs="unbounded">
  <xs:complexType>
    <xs:sequence>
      <xs:choice>
        <xs:element name="cif" type="xs:string"/>
        <xs:element name="nif" type="xs:string"/>
      </xs:choice>
        <xs:element name="empresa" type="xs:string"/>
        <xs:element name="localidad" type="xs:string"/>
    </xs:sequence>
  </xs:complexType>
</xs:element>
```

```
<cliente>
  <nif>11111111A</nif>
  <empresa>Manuel Barros</empresa>
  <localidad>PONTEAREAS</localidad>
</cliente>
<cliente>
  <cif>P22222222</cif>
  <empresa>HERMANOS GARCÍA, S.A.</empresa>
  <localidad>LUGO</localidad>
</cliente>
```

EJEMPLO - <xs:all>

```
<xs:element name="persoa" maxOccurs="unbounded">
  <xs:complexType>
    <xs:all>
      <xs:element name="nif" type="xs:string"/>
      <xs:element name="nome" type="xs:string"/>
    </xs:all>
  </xs:complexType>
</xs:element>
```

```
<persoa>
  <nif>12345678Z</nif>
  <nome>Pepe Pérez</nome>
</persoa>
<persoa>
  <nome>Pepa Gómez</nome>
  <nif>87654321Q</nif>
</persoa>
```

EJEMPLO - <xs:all> erróneo

```
<xs:element name="pessoa" maxOccurs="unbounded">
  <xs:complexType>
    <xs:sequence>
      <xs:all>
        <xs:element name="nif" type="xs:string"/>
        <xs:element name="nome" type="xs:string"/>
      </xs:all>
      <xs:element name="nss" type="xs:string"/>
    </xs:sequence>
  </xs:complexType>
</xs:element>
```

EJEMPLO

Elemento “**persoa**” compuesto de tres elementos hijos: “**nome**”, “**direccion**” y “**telefono_fijo**” o bien “**telefono_movil**”. Contiene también atributo “**dni**”

```
<persoa dni="98989898">
  <nome>Julio Suárez</nome>
  <direccion>Pontevedra</direccion>
  <telefono_movil>55522222</telefono_movil>
</persoa>
<persoa dni="32323232">
  <nome>María Sánchez</nome>
  <direccion>Vigo</direccion>
  <telefono_fijo>98611111</telefono_fijo>
</persoa>
```

SOLUCION (archivo .xsd)

```
<?xml:version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
  <xs:element name="persoas">
    <xs:complexType>
      <xs:sequence>
        <xs:element name="persona" maxOccurs="unbounded">
          <xs:complexType>
            <xs:sequence>
              <xs:element name="nome" type="xs:string"/>
              <xs:element name="direccion" type="xs:string"/>
              <xs:choice>
                <xs:element name="telefono_fijo" type="xs:string"/>
                <xs:element name="telefono_movil" type="xs:string"/>
              </xs:choice>
            </xs:sequence>
            <xs:attribute name="dni" type="xs:string"/>
          </xs:complexType>
        </xs:element>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
</xs:schema>
```

TIPOS DE ELEMENTOS

ELEMENTOS MIXTOS

- Pueden contener tanto texto como elementos hijos y atributos
- Se declaran especificando el atributo `mixed="true"` en la etiqueta `<xs:complexType>`

- Ejemplo:

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

- Fragmento xml válido

```
<carta>
  Estimado <nombre>Manuel Pérez</nombre>.
  Le notificamos que su hijo <hijo>Marcos
  Pérez</hijo> no ha asistido a clase en el día
  de hoy: <fecha>2021-02-28</fecha>.
</carta>
```

TIPOS DE ELEMENTOS

ELEMENTOS CON DATOS Y ATRIBUTO/S

- Ejemplo:

```
<racon unidad="gr">350</racon>
```

- Deben ir dentro de las etiquetas <xs:simpleContent> y <xs:extension>

```
<xs:element name="racon">
  <xs:complexType>
    <xs:simpleContent>
      <xs:extension base="xs:positiveInteger">
        <xs:attribute name="unidad" type="xs:string"/>
      </xs:extension>
    </xs:simpleContent>
  </xs:complexType>
</xs:element>
```


DECLARACION DE NUEVOS TIPOS

- Es posible crear un tipo de dato “personalizado”.
- Puede aplicarse sobre tipos simples: **simpleType** o compuestos: **complexType**
- Puede ser **reutilizado** en diferentes partes del mismo esquema
- Ejemplo:

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
  <xs:element name="garderia">
    <xs:complexType>
      <xs:sequence>
        <xs:element name="neno" minOccurs="1" maxOccurs="50" type="tipoNeno"/>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
  <xs:complexType name="tipoNeno">
    <xs:sequence>
      <xs:element name="nome" type="xs:string"/>
      <xs:element name="dataNascimento" type="xs:date"/>
      <xs:element name="peso" type="xs:decimal"/>
      <xs:element name="altura" type="xs:unsignedByte"/>
      <xs:element name="vacunas" type="xs:boolean"/>
    </xs:sequence>
    <xs:attribute name="numeroExpedente" type="xs:unsignedShort"/>
  </xs:complexType>
</xs:schema>
```

RESTRICCIONES

- Sirven para limitar el conjunto de valores posibles para un elemento o atributo.
- Siempre se aplican sobre un tipo existente, llamado “tipo base”
- Ejemplo:
 - **Valor comprendido en un rango determinado**
(edad entre 3 y 6 años, inclusive)
 - **Número de caracteres fijo, mínimo o máximo**
(contraseña con 8 caracteres alfanuméricos)
 - **Conjunto de valores permitidos**
(calificación: apto / no apto)
 - **Número de dígitos y/o número de decimales**
(DNI con 8 dígitos o precio con 2 decimales)
 - **Valor numérico mayor/menor que otro valor**
(nota > 5)
 - **Patrones**
(matrícula con 4 dígitos seguidos de 3 letras)
- Etc.

RESTRICCIONES

Extraído de documentación de Ricardo Eíto Brun

enumeration	Establece una lista de valores "aceptados"
fractionDigits	Número de cifras decimales
length	Número de caracteres obligatorios
maxExclusive y maxInclusive	Valor máximo de un rango
minExclusive y minInclusive	Valor mínimo en un rango
maxLength y minLength	Número máximo y mínimo de caracteres permitidos
pattern	Define una secuencia de caracteres permitida
totalDigits	Número exacto de dígitos permitidos
whiteSpace	Indica cómo se deben de tratar los espacios en blanco

DECLARACION DE RESTRICCIONES

RESTRICCION SOBRE ELEMENTO SIMPLE: Ejemplo1

```
<xs:element name="neno" minOccurs="1" maxOccurs="30">
  <xs:complexType>
    <xs:sequence>
      <xs:element name="nome" type="xs:string"/>
      <xs:element name="dataNascimento" type="xs:date"/>
      <xs:element name="peso" type="xs:decimal"/>
      <xs:element name="altura" type="xs:unsignedByte"/>
      <xs:element name="vacunas" type="xs:boolean"/>
    </xs:sequence>
    <xs:attribute name="numeroExpedente" type="xs:unsignedShort" use="required"/>
  </xs:complexType>
</xs:element>
```

```
<xs:element name="nome">
  <xs:simpleType>
    <xs:restriction base="xs:string">
      <xs:minLength value="3"/>
      <xs:maxLength value="15"/>
    </xs:restriction>
  </xs:simpleType>
</xs:element>
```

```
<neno numeroExpedente="2344">
  <nome>Noa Blanco Coello</nome>
  <dataNascimento>2010-10-12</dataNascimento>
  <peso>13.5</peso>
  <altura>78</altura>
  <vacunas>true</vacunas>
</neno>
```

Value 'Noa Blanco Coello' with length '17' exceeds maximum length facet of '15'.

DECLARACION DE RESTRICCIONES

RESTRICCION SOBRE ELEMENTO SIMPLE: Ejemplo2

```
<xs:element name="neno" minOccurs="1" maxOccurs="30">
  <xs:complexType>
    <xs:sequence>
      <xs:element name="nome" type="xs:string"/>
      <xs:element name="dataNascimento" type="xs:date"/>
      <xs:element name="peso" type="xs:decimal"/>
      <xs:element name="altura" type="xs:unsignedByte"/>
      <xs:element name="vacunas" type="xs:boolean"/>
    </xs:sequence>
    <xs:attribute name="numeroExpedente" type="xs:unsignedShort" use="required"/>
  </xs:complexType>
</xs:element>
```

```
<xs:element name="dataNascimento">
  <xs:simpleType>
    <xs:restriction base="xs:date">
      <xs:minInclusive value="2018-01-01"/>
    </xs:restriction>
  </xs:simpleType>
</xs:element>
```

```
<neno numeroExpedente="2344">
  <nome>Noa Blanco Coello</nome>
  <dataNascimento>2010-10-12</dataNascimento>
  <peso>13.5</peso>
  <altura>78</altura>
  <vacunas>true</vacunas>
</neno>
```

/value '2010-10-12' must be greater than or equal to MinInclusive '2018-01-01' .

DECLARACION DE RESTRICCIONES

RESTRICCIONES EQUIVALENTES “con nombre”

```
<xs:element name="neno" minOccurs="1" maxOccurs="30">
  <xs:complexType>
    <xs:sequence>
      <xs:element name="nome" type="tipo_nome"/>
      <xs:element name="dataNacimiento" type="tipo_dataNacimiento"/>
      <xs:element name="peso" type="xs:decimal"/>
      <xs:element name="altura" type="xs:unsignedByte"/>
      <xs:element name="vacunas" type="xs:boolean"/>
    </xs:sequence>
    <xs:attribute name="numeroExpedente" type="xs:unsignedShort" use="required"/>
  </xs:complexType>
</xs:element>
```

```
<xs:simpleType name="tipo_nome">
  <xs:restriction base="xs:string">
    <xs:minLength value="3"/>
    <xs:maxLength value="17"/>
  </xs:restriction>
</xs:simpleType>
```

```
<xs:simpleType name="tipo_dataNacimiento">
  <xs:restriction base="xs:date">
    <xs:minInclusive value="2018-01-01"/>
  </xs:restriction>
</xs:simpleType>
```

EXPRESIONES REGULARES

- Se refiere a **forzar que un dato se ajuste a un determinado “patrón”**
- Ejemplo, NIF con 8 dígitos seguidos de una letra
- Se definen empleando la restricción **<xs:pattern>**
- Resumen: *(información completa en pág. 15)*

SÍMBOLO	SIGNIFICADO
\d	Cualquier dígito (equivale a [0-9])
\D	Cualquier carácter no dígito
[abc]	Uno de los caracteres a, b o c
[A-Z]	Un carácter entre A y Z (rango)
[^abc]	Ningún carácter que sea a, b o c
[A-G-[D]]	Rango de A a G, excepto D

EXPRESIONES REGULARES

- **Cardinalidad:** se refiere siempre al carácter que precede al símbolo

SÍMBOLO	SIGNIFICADO
?	0 ó 1 apariciones
+	1 o varias apariciones
*	0 ó varias apariciones
{n}	n apariciones
{n,m}	De n a m apariciones
{n,}	n o más apariciones
()	Agrupación de elementos
\(Carácter “(“
\)	Carácter “)”

EXPRESIONES REGULARES

- Ejemplos

```
<xs:element name="telefono">
  <xs:simpleType>
    <xs:restriction base="xs:string">
      <xs:pattern value="[0-9]{3}-[0-9]{3}-[0-9]{3}"/>
    </xs:restriction>
  </xs:simpleType>
</xs:element>
```

- Interpretación

Tres dígitos numéricos seguidos de un carácter guión “-”

A continuación otros tres dígitos seguidos de otro guión

Y, nuevamente, otros tres dígitos para finalizar

P.ej. 555-111-222

TIPOS UNION

- Permiten que un único dato pueda combinar valores de diferentes tipos.
- P.ej. para una calificación: valores de 1 a 10 junto con expresiones como “PC” (pendiente de calificar), “CA” (calificado anteriormente) o “NP” (no presentado)
- Se utiliza el elemento **<xs:union>**
- **Sintaxis:**

```
<xs:union>
  <xs:simpleType>
    (...)
  </xs:simpleType>
  <xs:simpleType>
    (...)
  </xs:simpleType>
</xs:union>
```



```
<xs:union>
  <xs:simpleType>
    <xs:restriction base="xs:byte">
      <xs:minInclusive value="1"/>
      <xs:maxInclusive value="10"/>
    </xs:restriction>
  </xs:simpleType>
  <xs:simpleType>
    <xs:restriction base="xs:string">
      <xs:enumeration value="NP"/>
      <xs:enumeration value="CA"/>
      <xs:enumeration value="PC"/>
    </xs:restriction>
  </xs:simpleType>
</xs:union>
```

UNICIDAD Y CLAVES

- Son el **equivalente** a ID e IDREF en los DTDs
- Tienen **mayor potencia**.
- Permiten:
 - Indicar que un **elemento** es **único**
 - Indicar que un **atributo** es **único**
 - Definir **combinaciones de elementos y atributos** como únicos
 - Definir **claves**: valor único, obligatorias y no pueden tener valor nulo

UNICIDAD

- Permiten definir **valores únicos para elementos y atributos**.
- **Permite nulos** (a diferencia de las claves).
- Se representa con el **elemento <xs:unique>**
- Contiene obligatoriamente el atributo **“name”**
- Es un elemento compuesto, formado por dos subelementos:
 - <xs:selector>** ruta al elemento que contiene el valor único
 - <xs:field>** selecciona el campo con valor único dentro del elemento anterior
- Los valores de *“selector”* y *“field”* son **expresiones Xpath** (patrón similar a una ruta en una URL o dispositivo de almacenamiento)
- El elemento <xs:unique> siempre debe aparecer dentro de <xs:element>. Normalmente, **dentro del <xs:element> más externo** (en el elemento raíz → unicidad para todo el documento)

UNICIDAD: ejemplo

```
1 <?xml version="1.0" encoding="UTF-8"?>
2 <xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
3   <xs:element name="clientes">
4     <xs:complexType>
5       <xs:sequence>
6         <xs:element name="cliente" maxOccurs="unbounded">
7           <xs:complexType>
8             <xs:sequence>
9               <xs:element name="cif" type="xs:string"/>
10              <xs:element name="empresa" type="xs:string"/>
11              <xs:element name="localidad" type="xs:string"/>
12              <xs:element name="provincia" type="xs:string"/>
13              <xs:element name="facturacion" type="xs:decimal"/>
14            </xs:sequence>
15          </xs:complexType>
16        </xs:element>
17      </xs:sequence>
18    </xs:complexType>
19  </xs:element>
20 </xs:schema>
```

**Posición donde insertamos
código para unicidad**



UNICIDAD: ejemplo

```
6 <xs:element name="cliente" maxOccurs="unbounded">
7   <xs:complexType>
8     <xs:sequence>
9       <!-- unicidad en cif -->
10      <xs:element name="cif" type="xs:string"/>
11      <xs:element name="empresa" type="xs:string"/>
12      <xs:element name="localidad" type="xs:string"/>
13      <xs:element name="provincia" type="xs:string"/>
14      <xs:element name="facturacion" type="xs:decimal"/>
15    </xs:sequence>
16  </xs:complexType>
17 </xs:element>
18 </xs:sequence>
19 </xs:complexType>
20 <!-- dentro del elemento más externo: queremos unicidad en documento completo -->
21 <xs:unique name="cif_unico">
22   <xs:selector xpath="cliente"/><!-- camino al elemento que contiene el valor único -->
23   <xs:field xpath="cif"/><!-- campo donde está el valor único -->
24 </xs:unique>
25 </xs:element>
26 </xs:schema>
```

UNICIDAD: error de dato

```

1 <?xml version="1.0" encoding="UTF-8"?>
2 <clientes xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
3   xsi:noNamespaceSchemaLocation="clientes.xsd">
4   <cliente>
5     <cif>X11111111</cif>
6     <empresa>Empresa XXXX</empresa>
7     <localidad>Ponteareas</localidad>
8     <provincia>PONTEVEDRA</provincia>
9     <facturacion>24567.99</facturacion>
10  </cliente>
11  <cliente>
12    <cif>X11111111</cif>
13    <empresa>Empresa 7777</empresa>

```



Validation stopped at line 12, column 25: Duplicate unique value declared for identity constraint of element 'clientes'.

UNICIDAD: ejemplo xpath

```
<?xml version="1.0" encoding="UTF-8"?>
<clientes xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:noNamespaceSchemaLocation="clientes_xpath.xsd">
  <cliente>
    <dato>
      <otrodato>
        <cif>X11111111</cif>
      </otrodato>
    </dato>
    <empresa>Empresa XXXX</empresa>
    <localidad>Ponteareas</localidad>
    <provincia>PONTEVEDRA</provincia>
    <facturacion>24567.99</facturacion>
  </cliente>
```

Nuevo valor de xpath:
xpath="cliente/dato/otrodato"

```
<!-- dentro del elemento más externo: "clientes" -->
<xs:unique name="cif_unico">
  <xs:selector xpath="cliente/dato/otrodato"/><!--camino al elemento que contiene el valor único-->
  <xs:field xpath="cif"/><!--campo donde está el valor único-->
</xs:unique>
</xs:element>
</xs:schema>
```


UNICIDAD: ejemplo con atributo

```
<?xml version="1.0" encoding="UTF-8"?>
<clientes xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="clientes_unicidad_en_att.xsd">
  <cliente>
    <dato>
      <otrodato cif="X11111111"/>
    </dato>
    <empresa>Empresa XXXX</empresa>
    <localidad>Ponteareas</localidad>
    <provincia>PONTEVEDRA</provincia>
    <facturacion>24567.99</facturacion>
  </cliente>
```

```
<!-- dentro del elemento más externo: "clientes" -->
<xs:unique name="cif_unico">
  <xs:selector xpath="cliente/dato/otrodato"/><!-- camino al elemento que contiene el valor único-->
  <xs:field xpath="@cif"/><!-- campo donde está el valor único-->
</xs:unique>
```

CLAVE

- Se representa con el **elemento** `<xs:key>`
- Similar a `<xs:unique>`, con su misma estructura, pero **no permite valores nulos**.
- Contiene obligatoriamente el atributo **“name”** para su referencia posterior.
- Para **hacer referencia a una clave** se emplea el elemento `<xs:keyref>`
- El elemento `<xs:keyref>` contiene el **atributo “refer”** cuyo valor será el nombre que se le puso a la clave en su atributo **“name”**
- Una clave puede estar formada por varios elementos

CLAVE: ejemplo

```
1 <?xml version="1.0" encoding="UTF-8"?>
2 <clientes xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
3   ..... xsi:noNamespaceSchemaLocation="clientes_productos.xsd">
4   <producto>
5     <codigo>1234</codigo>...
6     <nombre>Folios</nombre>
7   </producto>
8   <producto>
9     <codigo>5678</codigo>...
10    <nombre>Plastilina</nombre>
11  </producto>
12  <cliente>
13    <suministra>1234</suministra>
14    <cif>X11111111</cif>
15    <empresa>Empresa XXXX</empresa>
16    <localidad>Ponteareas</localidad>
17    <provincia>PONTEVEDRA</provincia>
18    <facturacion>24567.99</facturacion>
19  </cliente>
20  <cliente>
21    <suministra>5678</suministra>
22    <cif>Z99999999</cif>
23    <empresa>Empresa ZZZZ</empresa>
```

← clave

→ referencia a...

CLAVE: ejemplo

```
1 <?xml version="1.0" encoding="UTF-8"?>
2 <clientes xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
3   xsi:noNamespaceSchemaLocation="clientes_productos.xsd">
4   <producto>
5     <codigo>1234</codigo>
6     <nombre>Folios</nombre>
7   </producto>
8   <producto>
9     <codigo>5678</codigo>
10    <nombre>Plastilina</nombre>
11  </producto>
12  <cliente>
13    <suministra>1234</suministra>
14    <cif>X11111111</cif>
15    <empresa>Empresa XXXX</empresa>
16    <localidad>Pontear</localidad>
17    <provincia>PONTE</provincia>
18    <facturacion>24567</facturacion>
19  </cliente>
20  <cliente>
21    <suministra>5678</suministra>
22    <cif>Z99999999</cif>
23    <empresa>Empresa ZZZZ</empresa>
```

Diagram illustrating XML Schema Key and Keyref declarations and their usage:

- Key Declaration (Lines 29-34):** Declares a key named "codigo_producto" on the "producto" element, with the "codigo" field as the key value.

```
<!-- declaramos la clave -->
<xs:key name="codigo_producto">
  <xs:selector xpath="producto"/>
  <xs:field xpath="codigo"/>
</xs:key>
```
- Keyref Declaration (Lines 65-73):** Declares a keyref named "referencia_producto" on the "cliente" element, with the "suministra" field as the key value, referencing the "codigo_producto" key.

```
<!-- declaramos la referencia -->
<xs:keyref name="referencia_producto" refer="codigo_producto">
  <xs:selector xpath="cliente"/>
  <xs:field xpath="suministra"/>
</xs:keyref>
```
- Usage:** The XML instance shows two "producto" elements with "codigo" values 1234 and 5678. Two "cliente" elements are shown, each with a "suministra" value (1234 and 5678) that matches the "codigo" of a "producto" element, thus satisfying the keyref constraint.