Tecnologías XML y Web Semántica

Departamento de Informática
Universidad de Oviedo

Sesión 2
Diseño de Vocabularios XML

Espacios de Nombres

XML Schema

Diseño de Vocabularios

Departamento de Informática

Universidad de Oviedo

Espacios de Nombres
Problema de la Homonimia

Homonimia: Mismo nombre con diferentes propósitos

```
<país nombre="Francia">
<capital>París</capital>
</país>
```

```
<inversión>
  <capital>7000€</capital>
</inversión>
```

¿Cómo combinar en el mismo documento estos vocabularios?

```
<inversiones>
  <país nombre="Francia">
    <capital>París</capital>
    <capital>1200€</capital>
  </país>
  . . .
</inversiones>
```

Ambigüedad

Espacios de Nombres
Posibles Soluciónes

Asignar un nombre único a cada etiqueta...
Posibles soluciones:

Crear una autoridad mundial que asigne nombres...

... o usar un mecanismo ya existente: URIs

Una URI es un identificador global único

Ejemplo: **`http://www.aulanet.uniovi.es`**

SOLUCIÓN:

Asociar a cada etiqueta una URI que indica a qué espacio de nombres pertenece...

`[http:www.bolsa.com]:capital`
`[http:www.geog.es]:capital`

Espacios de Nombres
Posibles soluciones

Solución fácil...

Asociar a cada etiqueta una URI

```
<[http://www.bolsa.com]:inversiones>
  <[http://www.geog.es]:país
      [http://www.geog.es]:nombre="Francia">
    <[http://www.geog.es]:capital>París
    </[http://www.geog.es]:capital>
    <[http://www.bolsa.com]:capital>1200€
    </[http://www.bolsa.com]:capital>
  </[http://www.bolsa.com]:país>
  . . .
</[http://www.bolsa.com]:inversiones>
```

Legibilidad...

Espacios de Nombres Definición

Solución: Asociar un alias a los elementos de un espacio de nombres dentro de un ámbito

xmlns:*alias* define *alias* en el ámbito de un elemento

```
<bolsa:inversiones
          xmlns:bolsa="http://www.bolsa.com"
          xmlns:geog="http://www.geog.es">
  <geog:país geog:nombre="Francia">
    <geog:capital>París</geog:capital>
    <bolsa:capital>1200€</bolsa:capital>
  </geog:país>
  . . .
</bolsa:inversiones>
```

NOTA: Las URIs sólo se utilizan para que el nombre sea único, no son enlaces, ni tienen que contener información

Espacios de Nombres
Asignación Dinámica

Es posible ir asociando espacios de nombres a los elementos según van apareciendo

```
<bolsa:inversiones
        xmlns:bolsa="http://www.bolsa.com">
  <geog:país
        xmlns:geog="http://www.geog.es"
        geog:nombre="Francia">
    <geog:capital>París</geog:capital>
    <bolsa:capital>1200€</bolsa:capital>
  </geog:país>
  . . .
</bolsa:inversiones>
```

Espacios de Nombres
Espacio de nombres por defecto

Mediante **xmlns="..."** se define un espacio de nombres por defecto (sin alias)

```
<inversiones
        xmlns="http://www.bolsa.com">
  <geog:país
        xmlns:geog="http://www.geog.es"
        geog:nombre="Francia">
    <geog:capital>París</geog:capital>
    <capital>1200€</capital>
  </geog:país>
  . . .
</inversiones>
```

Se refiere a
http://www.bolsa.com

Espacios de Nombres
Validación con DTDs

Posteriores a los DTDs, por tanto, los DTDs no dan soporte a Espacios de Nombres

Hay que definir los espacios de nombre usados

```
<!DOCTYPE inversiones [
<!ELEMENT inversiones (geog:país*)>
<!ELEMENT geog:país (geog:capital,capital) >
<!ELEMENT geog:capital (#PCDATA)>
<!ELEMENT capital (#PCDATA)>
<!ATTLIST inversiones
        xmlns CDATA #FIXED "http://www.bolsa.com">
<!ATTLIST geog:país
        geog:nombre CDATA #REQUIRED
        xmlns:geog CDATA #FIXED "http://www.geog.es">
]>
```

Espacios de Nombres
Valoración

- Ampliamente utilizados para combinar vocabularios
- Facilitan la incorporación de elementos no previstos inicialmente
- Sintaxis *extraña* al principio
  - Uso de prefijos
  - URIs como elemento diferenciador...pero las URLS también sirven para acceder a recursos
- Difícil combinación con DTDs

XML Schema

XML Schema

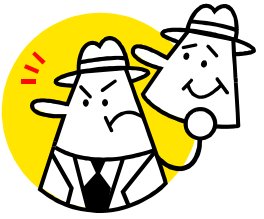
Lenguajes de Esquemas

- Esquema = definición de estructura de un conjunto de documentos XML
- Validar = Chequear que un documento sigue un esquema
  - Principal Ventaja: Protección de errores
  - Otras aplicaciones: Edición, compresión, enlaces de programación, etc.
- Originalmente se utilizaron los DTDs
- Posteriormente se ha desarrollado XML Schema
- Existen Otros:
  - RELAX-NG, Schematron, etc.

XML Schema
Características de los DTD's

- Especifican estructura del documento:
  - Elementos, atributos, anidamientos, etc.
- Integridad referencial mínima (ID, IDREF)
- Mecanismo sencillo de abstracción
  - Entidades ≈ Macros
- Inclusión de documentos externos
- Integrados en XML (Parte de la especificación)
- Sencillos de comprender (≈ Expresiones regulares)

XML Schema
Limitaciones de los DTD's



La Sintaxis ***no es XML*** (difíciles de manipular)

No soportan ***Espacios de nombres***

No permiten especificar ***tipos de datos*** (por ejemplo: enteros, flotantes, fechas, etc.

No permiten especificar ***secuencias no ordenadas***

((e1,e2,e3)|(e1,e3,e2)|(e2,e1,e3)|...(e3,e2,e1))

No hay soporte para declaraciones ***sensibles al contexto***: Los elementos se definen todos a nivel de documento, ejemplo, contenido con el mismo nombre cuya estructura cambia en diferentes contextos

Soporte limitado para ***Referencias cruzadas,*** no es posible formar claves a partir de varios atributos o de elementos

No son extensibles (una vez definido, no es posible añadir nuevos vocabularios a un DTD)

XML Schema

Objetivos de Diseño

- Sintaxis XML
- Soporte para Espacios de Nombres
- Mayor expresividad
  - Restricciones numéricas
  - Integridad dependientes del contexto
- Tipos de datos
  - Gran cantidad de tipos de datos predefinidos
  - Creación de tipos de datos por el usuario
- Extensibilidad
  - Inclusión/Redefinición de esquemas
  - Herencia de tipos de datos
- Soporte a Documentación

XML Schema

Ejemplo

alumnos.xsd

```
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
           targetNamespace="http://www.uniovi.es/alumnos"
           xmlns="http://www.uniovi.es/alumnos">
<xs:element name="alumnos">
 <xs:complexType>
  <xs:sequence>
   <xs:element name="alumno" minOccurs="1" maxOccurs="200"
               type="TipoAlumno"/>
  </xs:sequence>
 </xs:complexType>
</xs:element>
<xs:complexType name="TipoAlumno">
  <xs:sequence>
    <xs:element name="nombre" type="xs:string"/>
    <xs:element name="apellidos" type="xs:string"/>
    <xs:element name="nacim" type="xs:gYear"/>
  </xs:sequence>
  <xs:attribute name="dni" type="xs:string"/>
 </xs:complexType>
</xs:schema>
```

Elemento raíz schema y espacio de nombres determinado

Permite especificar rangos de inclusión

Permite especificar tipos

XML Schema
Validación

alumnos.xsd

```
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
           targetNamespace="http://www.uniovi.es/alumnos"
           xmlns="http://www.uniovi.es/alumnos">
<xs:element name="alumnos">
 <xs:complexType>
  <xs:sequence>
   <xs:element name="alumno" minOccurs="1" maxOccurs="200"
               type="TipoAlumno"/>
  </xs:sequence>
 </xs:complexType>
</xs:element>
<xs:complexTy
  <xs:sequenc
    <xs:eleme
    <xs:eleme
    <xs:eleme
  </xs:sequen
  <xs:attribu
 </xs:complex
</xs:schema>
```

alumnos.xml

```
<alumnos
  xmlns="http://www.uniovi.es/alumnos"
  xsi:SchemaLocation="http://www.uniovi.es/alumnos
                      alumnos.xsd"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
. . .
</alumnos>
```

Los espacios de nombres deben coincidir.
También puede usarse:
xsi:noNameSpaceLocation

XML Schema
Tipos Anónimos vs. Con nombre

```
<xs:element name="alumno">
<xs:sequence>
    <xs:element name="nombre" type="xs:string"/>
    <xs:element name="apellidos" type="xs:string"/>
  </xs:sequence>
</xs:element>
```

+ legible

```
…
<xs:element name="alumno" type="TipoAlumno"/>
…
<xs:ComplexType name="TipoAlumno">
<xs:sequence>
    <xs:element name="nombre" type="xs:string"/>
    <xs:element name="apellidos" type="xs:string"/>
  </xs:sequence>
</xs:ComplexType>
```

+ Reutilizable

XML Schema

Otra posibilidad: Referencias

```
<xs:element name="alumno">
<xs:sequence>
    <xs:element name="nombre" type="xs:string"/>
    <xs:element name="apellidos" type="xs:string"/>
  </xs:sequence>
</xs:element>
```

```
<xs:element name="alumnos">
<xs:sequence>
  <xs:element ref="alumno" />
</xs:sequence>
</xs:element>
```

XML Schema

Agrupaciones

Es posible nombrar agrupaciones de elementos y de atributos para hacer referencias a ellas

```
<xs:group name="nombApell">
  <xs:sequence>
    <xs:element name="nombre" type="xs:string"/>
    <xs:element name="apellidos" type="xs:string"/>
  </xs:sequence>
</xs:group>
```

```
<xs:complexType name="TipoAlumno">
 <xs:group ref="nombApell" />
 <xs:element name="carrera" type="xs:string"/>
</xs:complexType>
```

XML Schema
Tipos Complejos: Secuencia

Tipos Complejos: Son tipos que pueden contener elementos o atributos

Construcción básica mediante enumeración de elementos

```
<xs:complexType name="TipoAlumno">
  <xs:sequence>
    <xs:element name="nombre"    type="xs:string"/>
    <xs:element name="apellidos" type="xs:string"/>
    <xs:element name="nacim" type="xs:gYear"
                minOccurs="0" maxOccurs="1"/>
  </xs:sequence>
  <xs:attribute name="dni" type="xs:integer"/>
 </xs:complexType>
```

```
<alumno dni="9399390">
 <nombre>Juan</nombre>
 <apellidos>García García</apellidos>
 <nacim>1985</nacim>
</alumno>
```

XML Schema

Tipos Complejos:Alternativa

choice: Representa alternativas

OJO: Es una o-exclusiva

```
<xs:complexType name="Transporte">
   <xs:choice>
    <xs:element name="coche" type="xs:string"/>
     <xs:element name="tren" type="xs:string"/>
     <xs:element name="avión" type="xs:string"/>
   </xs:choice>
 </xs:complexType>
```

```
<transporte>
<coche>Renault R23</coche>
</transporte>
```

XML Schema
Tipos Complejos: Contenido Mixto

El contenido Mixto permite mezclar texto con elementos

```
 <xs:complexType name="TCom" mixed="true">
  <xs:choice minOccurs="0" maxOccurs="unbounded">
      <xs:element name="emph" type="xs:string"/>
  </xs:choice>
</xs:complexType>

<xs:element name="comentarios" type="TCom" />
```

```
<comentarios>
 Es un poco <emph>listillo</emph>
</comentarios>
```

XML Schema
Tipos Complejos: Secuencias no ordenadas

all = Todos los elementos en cualquier orden

En DTDs requería enumerar las combinaciones:

(A,B,C)|(A,C,B)|...|(C,B,A)

```
<xs:complexType name="TipoLibro">
  <xs:all>
      <xs:element name="autor" type="xs:string"/>
      <xs:element name="título" type="xs:string"/>
  </xs:all>
</xs:complexType>
<xs:element name="libro" type="TipoLibro" />
```

```
<libro>
 <autor>Juanita la Loca</autor>
 <título>No estoy loca</título>
</libro>
```

```
 <libro>
   <título>El kigote</título>
   <autor>Cerbantes</autor>
  </libro>
```

XML Schema
Tipos Simples

No pueden contener elementos o atributos

Pueden ser:

- Predefinidos o *built-in* (Definidos en la especificación)
  - Primitivos
  - Derivados
- Definidos por el usuario (a partir de tipos predefinidos)

XML Schema
Tipos Primitivos

string

boolean

number, float, double

duration, dateTime, time, date, gYearMonth, gYear, gMonthDay, gDay, gMonth

hexBinary, base64Binary

anyURI

QName = Nombre cualificado con espacio de nombres

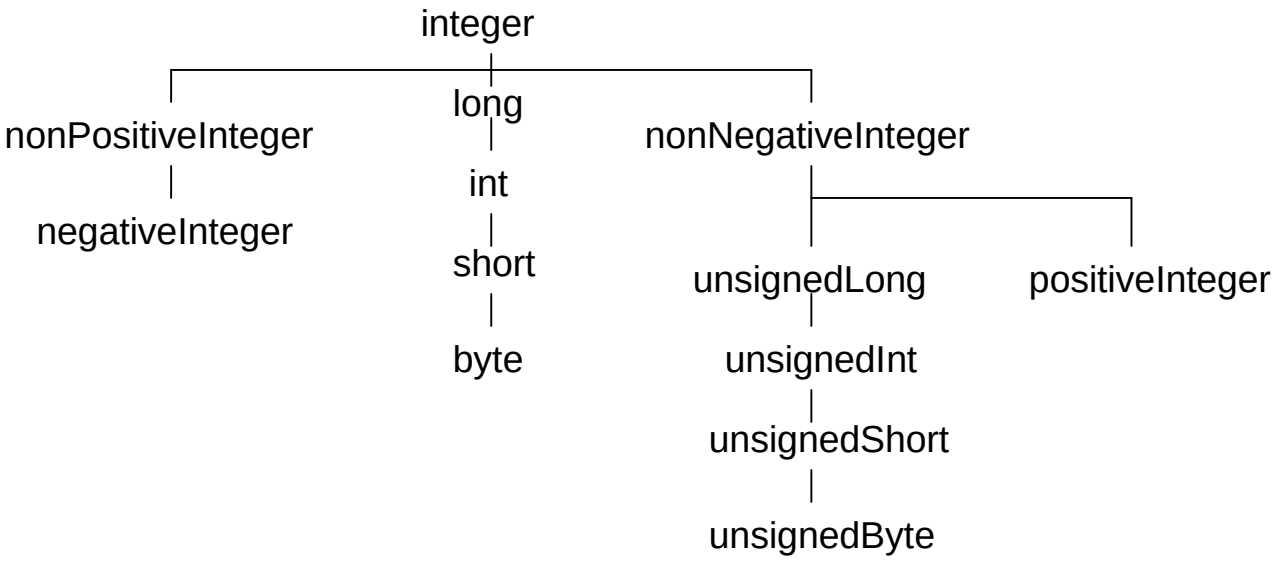
NOTATION = Notación binaria (similar a DTD)

XML Schema
Tipos Derivados

normalizedString, token, language

IDREFS, ENTITIES, NMTOKEN, NMTOKENS, Name, NCName, ID, IDREF, ENTITY

integer, nonPositiveInteger, negativeInteger, long, int, short, byte, nonNegativeInteger, unsignedLong, unsignedInt, unsignedShort, unsignedByte, positiveInteger



Esquemas XML
Facetas de Tipos

Facetas fundamentales:

- *equal:* Igualdad entre valores de un tipo de datos
- *ordered:* Relaciones de orden entre valores
- *bounded:* Límites inferiores y superiores para valores
- *cardinality:* Define si es finito o infinito (contable, no contable)
- *numeric:* Define si es numérico o no

Facetas de restricción

- *length, minlength, maxlength:* Longitud del tipo de datos
- *pattern:* Restricciones sobre valores mediante expresiones regulares
- *enumeration:* Restringe a una determinada enumeración de valores
- *whitespace:* Define política de tratamiento de espacios (preserve/replace, collapse)
- *(max/min)(in/ex)clusive:* Límites superiores/inferiores del tipo de datos
- *totaldigits, fractionDigits:* número de dígitos totales y decimales

XML Schema
Enumeraciones y Restricciones

Enumeración

```
<xs:simpleType name="TipoCarrera">
<xs:restriction base="xs:token">
 <xs:enumeration value="Gestión"/>
 <xs:enumeration value="Sistemas"/>
</xs:restriction>
</xs:simpleType>
```

Restricciones sobre valores

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

XML Schema
Listas

```
<xs:simpleType name="ComponentesRGB">
 <xs:list itemType="ComponenteRGB"/>
</xs:simpleType>

<xs:simpleType name="ComponenteRGB">
<xs:restriction base="xs:nonNegativeInteger">
   <xs:maxInclusive value="255" />
</xs:restriction>
</xs:simpleType>
```

Se pueden aplicar las facetas: length, maxLength, minLength, enumeration

```
<xs:simpleType name="ColorRGB">
   <xs:restriction base="ComponentesRGB">
     <xs:length value="3" />
   </xs:restriction>
 </xs:simpleType>
```

```
<color>255 255 0</color>
```

XML Schema
Uniones

```
<xs:simpleType name="TipoNota">
<xs:union>
  <xs:simpleType>
    <xs:restriction base="xs:float">
      <xs:maxInclusive value="10" />
      <xs:minInclusive value="0" />
    </xs:restriction>
  </xs:simpleType>
  <xs:simpleType>
    <xs:restriction base="xs:string">
      <xs:enumeration value="No presentado" />
    </xs:restriction>
  </xs:simpleType>
</xs:union>
</xs:simpleType>

<xs:element name="nota" type="TipoNota" />
```

```
<nota> 5.75 </nota>
```

```
<nota> No presentado </nota>
```

XML Schema
Expresiones regulares

```
<xs:simpleType name="NIF">
 <xs:restriction base="xs:token">
   <xs:pattern value="\d{7,8}[A-Z]" />
 </xs:restriction>
</xs:simpleType>

<xs:element name="nif" type="NIF" />
```

```
<nif>9394173J</nif>
```

```
<nif>11079845M</nif>
```

Ejemplos de expresiones regulares

Expresión	Posibles valores
Elemento \d	Elemento 2
a*b	b, ab, aab, aaab, …
[xyz]b	xb, yb, zb
a?b	b, ab
a+b	ab, aab, aaab, …
[a-c]x	ax, bx, cx

XML Schema
Expresiones Regulares

[a-c]x	ax, bx, cx
[^0-9]x	Carácter ¹ dígito seguido de x
\Dx	Carácter ¹ dígito seguido de x
(pa){2}rucha	paparucha
.abc	Cualquier carácter (1) seguido de abc
(a|b)+x	ax, bx, aax, bbx, abx, bax,...
a{1,3}x	ax, aax, aaax
\n	Salto de línea
\p{Lu}	Letra mayúscula
\p{Sc}	Símbolo de moneda

XML Schema
Tipos Derivados por Extensión

Similar a las subclases de POO. Consiste en añadir elementos a un tipo base

```
<xs:complexType name="Figura" >
  <xs:attribute name="color" type="Color"/>
 </xs:complexType>

<xs:complexType name="Rectángulo">
 <xs:complexContent>
 <xs:extension base="Figura">
 <xs:attribute name="base" type="xs:float" />
 <xs:attribute name="altura" type="xs:float" />
 </xs:extension>
 </xs:complexContent>
</xs:complexType>

<xs:complexType name="Círculo">
...similar pero incluyendo el radio
</xs:complexType>
```

XML Schema
Tipos Derivados por Extensión

Los tipos derivados pueden utilizarse en los mismos sitios que la clase base

```
<xs:element name="figuras">
   <xs:complexType>
     <xs:sequence>
      <xs:element name="figura" type="Figura"
                  maxOccurs="unbounded" />
     </xs:sequence>
   </xs:complexType>
  </xs:element>
```

```
<figuras>
<figura base="23" altura="3" xsi:type="Rectángulo" />
<figura radio="3" xsi:type="Círculo" />
</figuras>
```

Es necesario especificar el tipo mediante **`xsi:type`**

XML Schema
Tipos Abstractos

Mediante abstract="true" se declara un tipo como abstracto.

Ese tipo no puede usarse directamente

```
<xs:complexType name="Figura" abstract="true">
  <xs:attribute name="color" type="Color"/>
 </xs:complexType>
```

También es posible limitar la derivación de tipos

final="restriction"

XML Schema
Declaración de Atributos

```
<xs:complexType name="Círculo">
 <xs:attribute name="radio"
               type="xs:float"
               use="required" />

 <xs:attribute name="color"
               type="Color"
               default="255 0 0"/>

 <xs:attribute name="tipo"
               type="xs:string"
               fixed="jpeg" />
</xs:complexType>
```

Por defecto los atributos son opcionales. Indicar que son obligatorios: **`use="required"`**

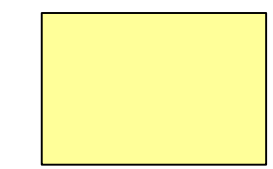
Valor por defecto de un atributo. Podría definirse otro valor.

Valor fijo de un atributo. Si no se define, se utiliza ése. Si se define, debe coincidir.

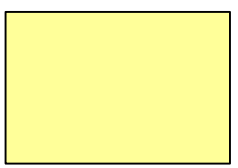
XML Schema
Inclusión de Esquemas

include permite incluir elementos de otros esquemas

- Los elementos deben estar en el mismo espacio de nombres
- Es como si se hubiesen tecleado todos en un mismo fichero



Alumnos.xsd



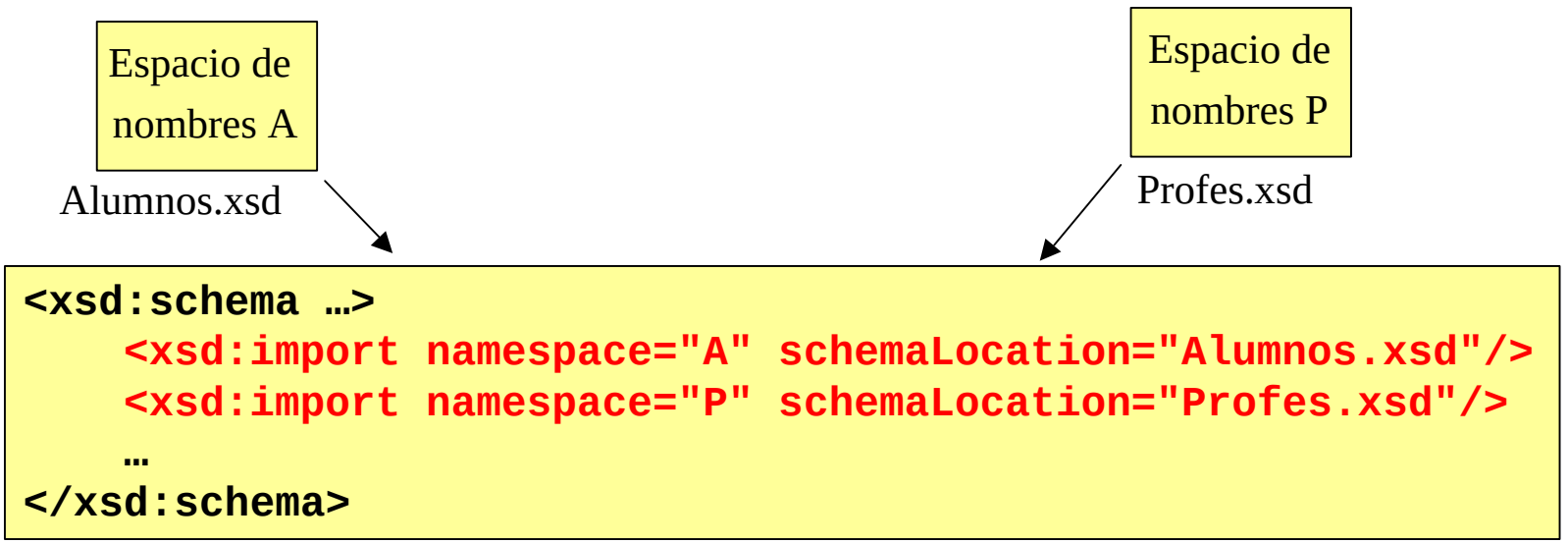
Profesores.xsd

```
<xsd:schema …>
    <xsd:include schemaLocation="Alumnos.xsd"/>
    <xsd:include schemaLocation="Profesores.xsd"/>
    …
</xsd:schema>
```

Universidad.xsd

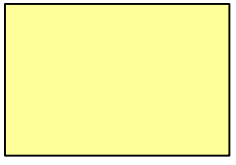
XML Schema
Importación de Esquemas

import permite incluir elementos de otros esquemas con distintos espacios de nombres



XML Schema
Redefinición de Esquemas

redefine es similar a *include* pero permite modificar los elementos incluidos.

Alumnos.xsd 

Añade el elemento `nota`

```
<xs:redefine schemaLocation="Alumnos.xsd">
  <xs:complexType name="TipoAlumno">
    <xs:complexContent>
      <xs:extension base="TipoAlumno">
         <xs:sequence>
           <xs:element name="nota" type="Nota" />
         </xs:sequence>
      </xs:extension>
    </xs:complexContent>
  </xs:complexType>
</xs:redefine>
```

AlumnosConNotas.xsd

XML Schema
Claves y Unicidad

Los DTDs proporcionaban el atributo ID para marcar la unicidad (un valor ID era único en todo el documento)

XML Schema tiene más posibilidades:

- Indicar que un elemento es único (**unique**)
- Definir atributos únicos
- Definir combinaciones de elementos y atributos como únicos
- Distinción entre unicidad y claves (**key)**
  - Clave = además de ser único, debe existir y no puede ser nulo.
- Declarar el rango de un documento en el que algo es único

XML Schema
Claves y Unicidad

```
<xs:complexType name="Alumnos">
 <xs:sequence>
  <xs:element name="Alumno" type="TipoAlumno"/>
 </xs:sequence>
  <xs:key name="DNI">
    <xs:selector xpath="a:alumno"/>
    <xs:field xpath="a:dni"/>
  </xs:key>
</xs:complexType>
```

Es necesario incluir el espacio de nombres (XPath)

La clave puede formarse para atributos y elementos

```
<xs:key name="DNI">
    <xs:selector xpath="a:alumno"/>
    <xs:field xpath="a:nombre"/>
    <xs:field xpath="a:nombre"/>
</xs:key>
```

Una clave puede estar formada por varios elementos

XML Schema
Claves y Unicidad

```
<xs:complexType name="Alumnos">
 <xs:sequence>
  <xs:element name="Alumno" type="TipoAlumno"/>
 </xs:sequence>
  <xs:unique name="DNI">
    <xs:selector xpath="a:alumno"/>
    <xs:field xpath="a:dni"/>
  </xs:unique>
</xs:complexType>
```

Unique especifica que debe ser único, pero podría no existir

XML Schema
Referencias a Claves

keyref especifica que debe hacer referencia a una clave (Claves Externas)

```
<xs:element name="clase">
 <xs:sequence>
  <xs:element name="alumnos" ...
  <xs:element name="delegado" ...
 </xs:sequence>

<xs:key name="DNI">
 <xs:selector xpath="a:alumnos/a:alumno"/>
 <xs:field xpath="a:dni"/>
</xs:key>

<xs:keyref name="Delegado" refer="DNI">
  <xs:selector xpath="a:delegado"/>
  <xs:field xpath="a:dni"/>
</xs:keyref>
```

XML Schema
Valores Nulos

Indicar que un elemento puede ser nulo sin estar vacío.

Vacío (Empty): Un elemento sin contenido

Nulo (Nil): Un elemento que indica que no hay valor

```
<xsd:element name="Persona">
  <xsd:complexType>
  <xsd:element name="nombre" type="xsd:NMTOKEN"/>
  <xsd:element name="primerApell" type="xsd:NMTOKEN"/>
  <xsd:element name="segundoApell" type="xsd:NMTOKEN"
                                   nillable="true"/>
 </xsd:complexType>
</xsd:element>
```

```
<persona>
      <nombre>John</nombre>
      <primerApell>Smith</primerApell>
      <segundoApell xsi:nil="true"/>
</persona>
```

El segundo apellido puede ser un NMTOKEN o estar indefinido

XML Schema
Incluir cualquier contenido...

any indica cualquier contenido de un determinado espacio de nombres

anyAttribute cualquier atributo de un espacio de nombres

```
<xs:complexType name="Comentario">
 <xs:sequence>
  <xs:any namespace="http://www.w3.org/1999/xhtml"
          minOccurs="1"
          processContents="skip" />
 </xs:sequence>
 <xs:anyAttribute
      namespace="http://www.w3.org/1999/xhtml"/>
</xs:complexType>
```

También puede declararse ##any, ##local, ##other

Otros valores
strict = obliga a validar
lax = valida si es posible

```
<comentarios>
  <html:p>Es un
     <html:emph>Listillo</html:emph>
  </html:p>
</comentarios>
```

XML Schema Limitaciones

No soporta entidades. Mecanismo para crear macros

<!ENTITY &texto; "Esto texto se repite muchas veces" >

Es necesario seguir usando los DTDs ☹

Lenguaje de Restricciones limitado

Ejemplo: ¿Verificar valor total = suma de valores parciales?

Sensibilidad al contexto limitada

Por ejemplo: Especificar que el contenido depende del valor de un atributo

<transporte tipo="coche"> ...</transporte>

<transporte tipo="avión"> ...</transporte>

Tamaño de archivos XML Schema puede ser excesivo

Legibilidad de las especificaciones...XML no siempre es legible

Complejidad de la especificación:

Muchas situaciones/combinaciones excepcionales

Otras propuestas: Relax-NG, Schematron, etc.

Ejercicios

Creación ficheros XML y validación mediante Esquemas

Herramientas:

- xsv (http://www.ltg.ed.ac.uk/~ht/xsv-status.html)
  - Herramienta desarrollada en Python
  - Funcionamiento a través de Web o en línea de comandos
  - Salida en formato XML (difícil de leer al principio)
- Xerces (Apache)
  - Librerías XML en Java y C++
  - Contiene diversas utilidades de prueba
  - Ejemplo: SAXCount cuenta el número de elementos pero también valida el Schema:

```
SAXCount -v=always -s  -n fichero.xml
```

Diseño de Vocabularios XML

Diseño de Vocabularios XML

- Separación tradicional de dos mundos
  - Sistemas orientados a Datos
    - Información uniforme y fuertemente estructurada (ej. Tablas)
    - Mucha cantidad de información repetida
    - Objetivo: Procesamiento eficiente (Almacenes de datos)
  - Sistemas orientados a Documentación
    - Información poco uniforme y entrelazada (ej. Libros)
    - No existe un patrón uniforme
    - Objetivo: Comunicación, Presentación (Mensajes)
- Se podría añadir un tercer mundo:
  - Programación Orientada a Objetos
  - Propuestas para añadir capacidad de programación a documentos
- XML: Información semi-estructurada (Lugar intermedio)
  - Estructuras jerárquicas entrelazadas

Diseño de Vocabularios XML

- Características a tener en cuenta
  - Tamaño de documentos
  - Facilidad de escritura
  - Facilidad de procesamiento
  - Flexibilidad (ej. HTML es muy flexible, Bases de Datos = menos)
  - Consistencia: Evitar características incoherentes
  - Nivel de abstracción: Buscar término medio en nivel de detalle

    <fecha>10 Marzo 2003</fecha>

    <fecha><día>10</dia><mes>Marzo</mes><año>2003</año></fecha>
- Patrones de diseño:
  - www.xmlpatterns.com

Diseño de Vocabularios XML

Ejemplo de Discusión

Representación de propiedades

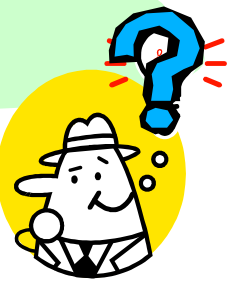
```
<pizza
    nombre="Margarita"
    precio="6" />
```

¿Atributos o Elementos?

```
<pizza>
    <nombre>Margarita </nombre>
    <precio>6</precio>
</pizza>
```

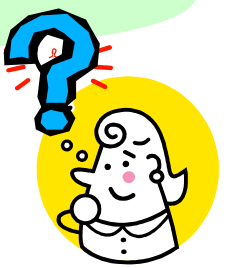
Razones filosóficas:

Atributos: valores asociados con objetos sin identidad propia (edad)
Subelementos: valores con identidad propia (fecha-nacimiento)



Orígenes (SGML):

Atributos: meta-información (información sobre el contenido)
Subelementos: Contenido



Diseño de Vocabularios XML

Ejemplo de Discusión

Representación de propiedades

```
<pizza
    nombre="Margarita"
    precio="6" />
```

¿Atributos o Elementos?

```
<pizza>
    <nombre>Margarita </nombre>
    <precio>6</precio>
</pizza>
```

En los DTDs
- Pueden incluirse restricciones sobre su valor
  - Ej. valor “si” o “no”
- Pueden definirse valores por defecto
- Pueden validarse los valores ID e IDREF
- Pueden definirse restricciones sobre espacios en blanco (NMTOKENS)

Ocupan menos espacio
Más fáciles de procesar (SAX y DOM)
Acceso a entidades externas (datos binarios)

Soportan valores arbitrariamente complejos y repetidos
Establecen un orden
Soportan *atributos de atributos*
Mayor flexibilidad ante modificaciones

Diseño de Vocabularios XML

En resumen...

...Aparición de una nueva torre de Babel...

Algunos Consejos:

- Estudiar dominio de la Aplicación (ver estándares ya definidos!!!)
- Considerar futuras ampliaciones (extensibilidad)
- Validar antes de que sea tarde
- Usar espacios de nombres
- etc. etc.

Fin