Introducción a XML Schema

Roberto Gómez Cárdenas
rogomez@itesm.mx
http://homepage.cem.itesm.mx/rogomez

Limitantes DTDs

- Documentos DTD no se encuentran en formato XML
  - Más trabajo para los parsers
- No expresa tipos de datos (tipeo de datos débil)
- No soporta espacio de nombres.
- Documento puede sobre-escribir definiciones externas DTD.
- No soporte de DOM,
- Opción: XML Schema,

XML Schema: Esquema XML

- Recomendación W3C liberada en mayo 2001
  - http://www.w3.org/TR/xmlschema-0
  - http://www.w3.org/TR/xmlschema-1
  - http://www.w3.org/TR/xmlschema-2
  - Dependa en las siguientes operaciones
    - XML-Infoset, XML-Namespaces, XPath
- Ventajas con respecto a los DTDs
  - Usan sintaxis de XML, al contrario que los DTDs.
  - Permiten especificar los tipos de datos.
  - Son extensibles.
- También conocido por las siglas XSD

Lámina 3 Roberto Gómez C.

¿Qué define un XML schema'

- Elementos que pueden aparecer en un documento.
- Atributos que pueden aparecer en un documento.
- Cuales elementos son elementos hijos.
- El orden de los elementos hijos.
- El número de elementos hijos.
- Si un elemento esta vacío o puede incluir texto.
- Tipos de datos para elementos y atributos
- Valores por defecto y valores fijos para elementos y atributos.

Lámina 4 Roberto Gómez C.

Ventajas del soporte de datos por para de XML Schema

- Es más fácil describir el contenido admisible de los documentos.
- Es más fácil validar que los datos sean correctos.
- Es más fácil de trabajar con datos provenientes de una base de datos.
- Es más fácil definir facetas (restricciones en los datos)
- Es más fácil definir patrones (formatos de datos).
- Es más fácil convertir datos entes diferentes tipos de datos.

Lámina 5 Roberto Gómez C.

Ventajas

- No es necesario aprender un nuevo lenguaje.
- Posible usar el editor XML para editor los archivos Schema.
- Posible usar el parser XML para parsear los archivos Schema.
- Posible manipular su Schema con XML DOM.
- Posible transformar su esquema con XSLT.

Lámina 6 Roberto Gómez C.

Documento XML y archivo DTD

note.xml

```
<?xml version="1.0"?>
<note>
  <to>Tove</to>
  <from>Jani</from>
  <heading>Reminder</heading>
  <body>Hello World</body>
</note>
```

note.dtd

```
<!ELEMENT note (to, from, heading, body)>
<!ELEMENT to (#PCDATA)>
<!ELEMENT from (#PCDATA)>
<!ELEMENT heading (#PCDATA)>
<!ELEMENT body (#PCDATA)>
```

Lámina 7 Roberto Gómez C.

Documento XML y XML Schema

note.xml

```
<?xml version="1.0"?>
<note>
  <to>Tove</to>
  <from>Jani</from>
  <heading>Reminder</heading>
  <body>Hello World</body>
</note>
```

note.xsd →

```
<?xml version="1.0"?>
<xs:schema
xmlns:xs="http://www.w3.org/2001/XMLSchema"
targetNamespace="http://www.acme.com"
xmlns="http://www.acme.com"
elementFormDefault="qualified">

<xs:element name="note">
  <xs:complexType>
    <xs:sequence>
      <xs:element name="to" type="xs:string"/>
      <xs:element name="from" type="xs:string"/>
      <xs:element name="heading" type="xs:string"/>
      <xs:element name="body" type="xs:string"/>
    </xs:sequence>
  </xs:complexType>
</xs:element>

</xs:schema>
```

Lámina 8 Roberto Gómez C.

El elemento <schema>

- Es el elemento raíz de cada XML Schema

```
<?xml version="1.0"?>
<xs:schema >
  …
  …
</xs:schema>
```

- Puede contener algunos atributos
- Ejemplo:

```
<?xml version="1.0"?>

<xs:schema
xmlns:xs="http://www.w3.org/2001/XMLSchema"
targetNamespace="http://www.acme.com"
xmlns="http://www.acme.com"
elementFormDefault="qualified">
...
...
</xs:schema>
```

Lámina 9 Roberto Gómez C.

Atributos del elemento <schema>

- xmlns:xs=http://www.w3.org/2001/XMLSchema
  - Indica que los elementos y tipos de datos en el esquema vienen del espacio de nombres especificado.
- targetNamespace="http://www.acme.com"
  - Indica que los elementos definidos por este esquema vienen de www.acme.com
- xmlns="http://www.acme.com"
  - Indica que el espacio de nombres por default es www.acme.com
- elementFormDefault="qualified“
  - Indica que cualquier elemento usado por el documento XML declarados en este esquema debe contar con un espacio de nombres calificado.

Lámina 10 Roberto Gómez C.

Ejemplo referencia de un Schema dentro de un documento XML

```
<?xml version="1.0"?>

<note xmlns="http://www.acme.com"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://www.acme.com note.xsd">

  <to>Tove</to>
  <from>Jani</from>
  <heading>Reminder</heading>
  <body>Hello World</body>
</note>
```

Lámina 11 Roberto Gómez C.

Fragmentos referencia

- xmlns=http://www.acme.com
  - Especifica la declaración del espacio de direcciones por default.
  - Le indica al validador que todos los elementos usados en este documento XML están declarados en el espacio de direcciones www.acme.com
- xmlns:xsi
  - Atributo con dos valores:
    - http://www.w3.org/2001/XMLSchema-instance
      - Espacio de nombres a utilizar.
    - http://www.acme.com note.xsd
      - Ubicación del XML schema a usar para el espacio de direcciones referenciado anteriormente

Lámina 12 Roberto Gómez C.

Elementos simples

- Elemento que solo contiene texto.
- No puede contener otro elemento o atributos.
- Sintaxis

  <xs:element name="*xxx*" type="*yyy*"/>
- Donde *xxx* es el nombre del elemento e *yyy* es el tipo del elemento
- Los datos más comunes son

  xs:string
  xs:decimal
  xs:integer
  xs:boolean
  xs:date
  xs:time

Lámina 13 Roberto Gómez C.

Ejemplos elementos simples

- Elementos XML

```
<apellido>Garcia</apellido>
<edad>36</edad>
<fecha>1970-03-27</fehca>
```

- Definiciones

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

Lámina 14 Roberto Gómez C.

Valores por default y fijos para elementos simples

- Elementos simples pueden tener un valor por defecto o un valor especificado.
- Un valor por defecto es automáticamente asignado al elemento cuando ningún otro valor es especificado.
- Ejemplo valor por defecto *rojo*

```
<xs:element name="color" type="xs:string" default="rojo"/>
```

- Un valor por fijo también es asignado de forma automática al elemento, y no es posible especificar otro valor.
- Ejemplo valor fijo *rojo*

```
<xs:element name="color" type="xs:string" fixed="rojo"/>
```

Lámina 15 Roberto Gómez C.

Atributos XSD

- Los elementos simples no pueden contar con atributos.
- Si un elemento cuenta con atributos, se considera como un tipo complejo.
  - Sin embargo el atributo en sí siempre es declarado como un tipo simple.
- Sintaxis atributo

```
<xs:attribute name="xxx" type="yyy"/>
```

- Donde *xxx* es el nombre del atributo e *yyyy* especifica el tipo de dato del atributo.
- Tipos más comunes son:

xs:string	xs:decimal
xs:integer	xs:boolean
xs:date	xs:time

Lámina 16 Roberto Gómez C.

Ejemplo

- Elemento XML con un atributo

```
<apellido leng="EN”>Perez</apellido>
```

- La definición del atributo es:

```
<xs:attribute name="leng" type="xs:string"/>
```

Lámina 17 Roberto Gómez C.

Valores por defecto y fijos atributos

- Valor por defecto es automáticamente asignado a un atributo cuando ningún otro valor es especificado.
- Ejemplo de valor por defecto *EN*

```
<xs:attribute name="leng" type="xs:string" default="EN"/>
```

- Un valor por fijo también es asignado de forma automática al elemento, y no es posible especificar otro valor.
- Ejemplo de valor fijo *EN*

```
<xs:attribute name="leng" type="xs:string" fixed="EN"/>
```

Lámina 18 Roberto Gómez C.

Atributos opcionales y requeridos

- Los atributos son opcionales por defecto
- Para especificar que el atributo es requerido, se debe utilizar el atributo "*use*":

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

Lámina 19 Roberto Gómez C.

Restricciones en contenido

- Cuando un elemento XML o un atributo cuentan con tipos de datos definidos, este pone restricciones en el contenido del elemento o atributo.
- Si un elemento es de tipo "*xs:date*" y contiene un string como "*Hola Mundo*", el elemento no será válido.
- Con los XML Schemas, es posible añadir restricciones a los elementos y atributos XML.
- Estos elementos se conocen como *facetas*.

Lámina 20 Roberto Gómez C.

Facetas y restricciones XSD

- Las restricciones son usadas para definir valores aceptables para los elementos XML o atributos.
- Restricciones en elementos XML son llamadas facetas.
- Se cuenta con los siguientes tipos de restricciones
  - Restricciones en los valores
  - Restricciones en un conjunto de valores

Lámina 21 Roberto Gómez C.

Restricciones en valores

- Ejemplo definición de un elemento llamado age con una restricción.
  - El valor de *edad* no puede ser menor que 0 o mayor que 120

```
<xs:element name="edad">
  <xs:simpleType>
    <xs:restriction base="xs:integer">
      <xs:minInclusive value="0"/>
      <xs:maxInclusive value="120"/>
    </xs:restriction>
  </xs:simpleType>
</xs:element>
```

Lámina 22 Roberto Gómez C.

Restricciones en un conjunto de valores

- Para limitar el contenido de un elemento XML a un conjunto de valores validos, es necesario usar la clausula *enumeration*.
  - Ejemplo define un elemento llamado *coche* con una restricción.
  - Los únicos valores aceptables son Audi, Golf, Peaugout

```
<xs:element name="coche">
  <xs:simpleType>
    <xs:restriction base="xs:string">
      <xs:enumeration value="Audi"/>
      <xs:enumeration value="Golf"/>
      <xs:enumeration value="Peaugout"/>
    </xs:restriction>
  </xs:simpleType>
</xs:element>
```

Lámina 23 :oberto Gómez C.

Restricciones en conjunto de valores

- Otra versión del ejemplo anterior es:

```
<xs:element name="coche" type="TiposCoches"/>

<xs:simpleType name="TiposCoches">
  <xs:restriction base="xs:string">
    <xs:enumeration value="Audi"/>
    <xs:enumeration value="Golf"/>
    <xs:enumeration value="Peaugout"/>
  </xs:restriction>
</xs:simpleType>
```

- Nota:
  - En este caso el tipo "*TiposCoches*" puede ser usado por otros elementos ya que no es parte del elemento coche.

Lámina 24 Roberto Gómez C.

Expresiones regulares en XML Schema

Expresión	Coincidencias	Significado
Capitulo\d	Capitulo**0**, Capitulo**1**, Capitulo**2**	\d representa cualquier dígito, es decir 0, 1, 2, 3, 4 , 5, 6, 7, 8, 9
Apéndice\D	Apéndice**A**, Apéndice**B**, Apéndice**C** ...	\D representa cualquier valor **no** dígito
Capitulo\s\d	Capitulo **0**, Capitulo **1**, Capitulo **2**	\s representa espacio en blanco, retorno de carro, nueva línea o intro
Apéndice\s\w	Apéndice **A**, Apéndice **B**, Apéndice **C** ...	\w representa a cualquier carácter
a*x	x, a**x**, a**a**x, a**aa**x	* indica que el carácter al que sigue puede aparecer cero, una, o más veces
a?x	a**x**, x	? indica que el carácter al que sigue puede aparecer como mucho una vez
a+x	a**x**, a**a**x, a**aa**x	+ indica que el carácter al que sigue debe aparecer al menos una vez
(a|b)+x	a**x**, b**x**, a**a**x, a**b**x, b**a**x, b**b**x, a**aa**x, a**ab**x, a**ba**x, a**bb**x, b**aa**x, b**ab**x, b**ba**x, b**bb**x ...	(|) cualquier combinación de los caracteres incluidos en el grupo
[abcde]x	a**x**, b**x**, c**x**, d**x**, e**x**	[abcde] sólo un carácter de la lista
[a-e]x	a**x**, b**x**, c**x**, d**x**, e**x**	[a-e] sólo un carácter en el intervalos de la **a** hasta la **e**
[-ae]x	-**x**, a**x**, e**x**	[-ae] sólo un carácter de la lista
[ae-]x	a**x**, e**x**, -**x**	[ae-] sólo un carácter de la lista
[^0-9]x	a**x**, B**x** ...	[^0-9] cualquier carácter no-dígito
\Dx	a**x**, B**x** ...	\D cualquier carácter no-dígito
.x	a**x**, B**x**, 2**x**...	. cualquier carácter
.*abc.*	**1x2**abc, abc**1x2**, **z3456**abc**hooray**	.* caulquier carácter puede aparecer cero, una, o más veces
ab{2}x	ab**bx**	{n} el valor anterior se repite n veces
ab{2,4}x	ab**bx**, ab**bbx**, ab**bbbx**	{n,m} por lo menos n veces pero no mas de m veces
ab{2,}x	ab**bx**, ab**bbx**, ab**bbbx**	{n,} por lo menos n veces
(ab){2}x	**abab**x	(ab){n} el grupo **ab** aparece n veces

Lámina 25 Roberto Gómez C.

Sintaxis expresiones regulares (1)

Expresión	Ejemplos coincidencia	Significado
Cap\d	Cap0, Cap1, Cap2	\d representa cualquier dígito, es decir 0, 1, 2, 3, 4 , 5, 6, 7, 8, 9
Apen\D	ApenA, ApenB, ApenC ...	\D representa cualquier valor no dígito
Cap\s\d	Cap 0, Cap 1, Cap 2	\s representa espacio en blanco, retorno de carro, nueva línea o intro
Apen\s\w	Apen A, Apen B, Apen C ...	\w representa a cualquier carácter
a*x	x, ax, aax, aaax	* indica que el carácter al que sigue puede aparecer cero, una, o más veces
a?x	ax, x	? indica que el carácter al que sigue puede aparecer como mucho una vez
a+x	ax, aax, aaax, …	+ indica que el carácter al que sigue debe aparecer al menos una vez

Lámina 26 Roberto Gómez C.

Sintaxis expresiones regulares (2)

Expresión	Ejemplos coincidencia	Significado
(a|b)+x	ax, bx, aax, abx, bax, bbx, aaax, aabx, abax, abbx, baax, babx, bbax, bbbx ...	(|) cualquier combinación de los caracteres incluidos en el grupo
[abcde]x	ax, bx, cx, dx, ex	[abcde] sólo un carácter de la lista
[a-e]x	ax, bx, cx, dx, ex	[a-e] sólo un carácter en el intervalos de la a hasta la e
[-ae]x	-x, ax, ex	[-ae] sólo un carácter de la lista
[ae-]x	ax, ex, -x	[ae-] sólo un carácter de la lista
[^0-9]x	ax, Bx ...	[^0-9]cualquier carácter no-dígito
\Dx	ax, Bx ...	\D cualquier carácter no-dígito
.x	ax, Bx, 2x...	. cualquier carácter
.*abc.*	1x2abc, abc1x2, z3456abchooray	.* cualquier carácter puede aparecer cero, una, o más veces

Lámina 27 Roberto Gómez C.

Sintaxis expresiones regulares (3)

Expresión	Ejemplos coincidencia	Significado
ab{2}x	abbx	{n} el valor anterior se repite n veces
ab{2,4}x	abbx, abbbx, abbbbx	{n,m} por lo menos n veces pero no mas de m veces
ab{2,}x	abbx, abbbx, abbbbx	{n,} por lo menos n veces
(ab){2}x	ababx	(ab){n} el grupo ab aparece n veces

Lámina 28 Roberto Gómez C.

Restricciones en series de valores

- Define un elemento llamado "*letra*" con una restricción.
- El único valor aceptable es UNA de las letras minúsculas entre a y z.

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

Lámina 29 Roberto Gómez C.

Segundo ejemplo restricciones en serie de valores

- Definición de un elemento llamado "*iniciales*" con una restricción.
- El único valor aceptado son TRES de las letras mayúsculas entre la A y la Z.

```
<xs:element name="iniciales">
  <xs:simpleType>
    <xs:restriction base="xs:string">
      <xs:pattern value="[A-Z][A-Z][A-Z]"/>
    </xs:restriction>
  </xs:simpleType>
</xs:element>
```

Lámina 30 Roberto Gómez C.

Tercer ejemplo restricciones en serie de valores

- Elemento: iniciales
- Restricción:
  - Los únicos valores aceptados son tres letras minúsculas o mayúsculas entre la A y la Z

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

Lámina 31 Roberto Gómez C.

Cuarto ejemplo restricciones en serie de valores

- Elemento: choice
- Restricción
  - Una de las siguientes letras: x, y O z

```
<xs:element name="choice">
  <xs:simpleType>
    <xs:restriction base="xs:string">
      <xs:pattern value="[xyz]"/>
    </xs:restriction>
  </xs:simpleType>
</xs:element>
```

Lámina 32 Roberto Gómez C.

Quinto ejemplo restricciones en serie de valores

- Elemento: prodid
- Restricción
  - Cinco dígitos en una secuencia, y cada dígito debe tener un valor entre 0 y 9

```
<xs:element name="prodid">
  <xs:simpleType>
    <xs:restriction base="xs:integer">
      <xs:pattern value="[0-9][0-9][0-9][0-9][0-9]"/>
    </xs:restriction>
  </xs:simpleType>
</xs:element>
```

Lámina 33 Roberto Gómez C.

Otras restricciones en series de valores

- Elemento: letra
- Valor aceptado: cero o más ocurrencias de letras minúsculas ente a y z.

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

Lámina 34 Roberto Gómez C.

Otras restricciones en series de valores

- Elemento: letra
- Restricción: valor aceptable es uno o más pares de letras cada uno consistiendo de un letra minúscula seguida de una letra mayúscula.

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

- Ejemplos:
  - sToP: válido
  - Stop, STOP o stop: inválido

Lámina 35 Roberto Gómez C.

Otras restricciones en series de valores

- Elemento: genero
- Restricción: masculino o femenino

```
<xs:element name="gender">
  <xs:simpleType>
    <xs:restriction base="xs:string">
      <xs:pattern value="masculino|femenino"/>
    </xs:restriction>
  </xs:simpleType>
</xs:element>
```

Lámina 36 Roberto Gómez C.

Otras restricciones en series de valores

- Elemento: password
- Restricción: ocho caracteres, que deben ser letras minúsculas o mayúsculas entre a y z, o un número entre el 0 y 9.

```
<xs:element name="password">
  <xs:simpleType>
    <xs:restriction base="xs:string">
      <xs:pattern value="[a-zA-Z0-9]{8}"/>
    </xs:restriction>
  </xs:simpleType>
</xs:element>
```

Lámina 37 Roberto Gómez C.

Restricciones de caracteres de espacio en blanco

- Elemento: direccion
- Restricción: la palabra *whiteSpace* tiene asignado el valor de *preserve*, lo que significa que el procesador XML no borrará ningún carácter de espacio en blanco que encuentre

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

Lámina 38 Roberto Gómez C.

Segundo ejemplo restricciones espacio en blanco

- Elemento: direccion
- Restricción: la palabra *whiteSpace* se le asigna el valor de *replace*, esto significa que el procesador XML reemplazará los caracteres en blanco por espacios.

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

Lámina 39 Roberto Gómez C.

Tercer ejemplo restricciones espacio en blanco

- Elemento: dirección
- Restricción: : la palabra *whiteSpace* se le asigna el valor de *collapse*, lo que significa que el procesador XML removerá todos los caracteres en blanco y los substituirá por unos solo.

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

Lámina 40 Roberto Gómez C.

Restricciones longitud

- El límite de la longitud de un elemento se puede definir con las palabras: length, maxLength, y minLength.
- Elemento: password
- Restricción: valor debe contar con exactamente ocho caracteres.

```
<xs:element name="password">
  <xs:simpleType>
    <xs:restriction base="xs:string">
      <xs:length value="8"/>
    </xs:restriction>
  </xs:simpleType>
</xs:element>
```

Lámina 41 Roberto Gómez C.

Segundo ejemplo restricciones con longitud

- Elemento: password
- Restricción: elemento debe contar con mínimo cinco caracteres y máximo ocho.

```
<xs:element name="password">
  <xs:simpleType>
    <xs:restriction base="xs:string">
        <xs:minLength value="5"/>
        <xs:maxLength value="8"/>
    </xs:restriction>
  </xs:simpleType>
</xs:element>
```

Lámina 42 Roberto Gómez C.

Restricciones para tipos de datos

Restricción	Descripción
enumeration	Define una lista de valores aceptables
fractionDigits	Especifica el número máximo de dígitos decimales permitido. Debe ser igual o mayor a cero.
length	Especifica el número exacto de caracteres o elementos permitido. Debe ser igual o mayor que cero.
maxExclusive	Especifica el límite superior para valores numéricos (el valor debe ser menor que este valor).
maxInclusive	Especifica el límite superior para valores numéricos (el valor debe ser menor o igual a este valor).
maxLength	Especifica el número máximo de caracteres o elementos permitido. Debe ser igual o mayor a cero.
minExclusive	Especifica el límite inferior para valores numéricos (el valor debe ser mayor que este valor).

Lámina 43 Roberto Gómez C.

Restricciones para tipos de datos

Restricción	Descripción
minInclusive	Especifica el límite inferior para valores numéricos (el valor debe ser mayor o igual a este valor).
minLength	Especifica el mínimo número de caracteres o elementos permitidos. 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 a cero.
whiteSpace	Especifica como manejar los espacios en blanco (avances de línea, tabuladores, retornos de carro, y espacios).

Lámina 44 Roberto Gómez C.

Elementos complejos

- Un elemento complejo es un elemento XML que contiene otros elementos o atributos.
- Cuatro tipo de elementos complejos
  - Elementos vacíos.
  - Elementos que solo contienen otros elementos.
  - Elementos que solo contienen texto.
  - Elementos que contienen texto y otros elementos.
- Nota:
  - Cada uno de estos elementos también pueden contener atributos.

Lámina 45 Roberto Gómez C.

Ejemplos de elementos complejos

- Un elemento XML, “producto”, que esta vacío

```
<producto pid="1345"/>
```

- Un elemento XML, “empleado” que solo contiene otros elementos

```
<empleado>
  <firstname>John</firstname>
  <lastname>Smith</lastname>
</empleado>
```

- Un elemento XML , “comida” que solo contiene texto.

```
<comida type=“postre">Ice cream</comida>
```

Lámina 46 Roberto Gómez C.

Ejemplos de elementos complejos

- Un elemento, descripción, que contiene elementos y texto.

```
<descripcion>
It happened on <date lang="norwegian">03.03.99</date> ....
</descripcion>
```

Lámina 47 Roberto Gómez C.

Definiendo un elemento complejo

- Elemento complejo empleado que solo contiene otros elementos.

```
<empleado>
  <firstname>John</firstname>
  <lastname>Smith</lastname>
</empleado>
```

- Posible definir elemento complejo en un XML Schema, de dos formas:
  - Primera forma: elemento empleado puede ser declarado directamente

```
<xs:element name="empleado">
  <xs:complexType>
    <xs:sequence>
      <xs:element name="firstname" type="xs:string"/>
      <xs:element name="lastname" type="xs:string"/>
    </xs:sequence>
  </xs:complexType>
</xs:element>
```

Lámina 48 Roberto Gómez C.

Definiendo elementos complejos

– Segunda forma: elemento empleado puede contar con un tipo de atributo que se refiere al nombre del tipo complejo a usar:

```
<xs:element name="empleado" type="personinfo“/>

<xs:complexType name="personinfo">
  <xs:sequence>
    <xs:element name="firstname" type="xs:string"/>
    <xs:element name="lastname" type="xs:string"/>
  </xs:sequence>
</xs:complexType>
```

Lámina 49 Roberto Gómez C.

Ventajas usar atributo

- Si se usa el método anterior, varios elementos se puede referir al mismo tipo complejo.
- Por ejemplo:

```
<xs:element name="empleado" type="personinfo“/>
<xs:element name="student" type="personinfo“/>
<xs:element name="member" type="personinfo“/>

<xs:complexType name="personinfo">
  <xs:sequence>
    <xs:element name="firstname" type="xs:string"/>
    <xs:element name="lastname" type="xs:string"/>
  </xs:sequence>
</xs:complexType>
```

Lámina 50 Roberto Gómez C.

Otra forma

- También es posible que un elemento complejo se base en un elemento complejo existente y añadir otros elementos complejos

```
<xs:element name="empleado" type="fullpersoninfo"/>

<xs:complexType name="personinfo">
  <xs:sequence>
    <xs:element name="firstname" type="xs:string"/>
    <xs:element name="lastname" type="xs:string"/>
  </xs:sequence>
</xs:complexType>

<xs:complexType name="fullpersoninfo">
  <xs:complexContent>
    <xs:extension base="personinfo">
      <xs:sequence>
        <xs:element name="address" type="xs:string"/>
        <xs:element name="city" type="xs:string"/>
        <xs:element name="country" type="xs:string"/>
      </xs:sequence>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>
```

Lámina 51 Roberto Gómez C.

Elementos vacíos XSD

- Un elemento XML vacío

```
<producto prodid="1345" />
```

- El elemento producto no cuenta con ningún contenido.
- Se define un tipo que permite elementos en su contenido, pero no se declaran elementos.

```
<xs:element name="producto">
  <xs:complexType>
    <xs:complexContent>
      <xs:restriction base="xs:integer">
        <xs:attribute name="prodid" type="xs:positiveInteger"/>
      </xs:restriction>
    </xs:complexContent>
  </xs:complexType>
</xs:element>
```

Lámina 52 Roberto Gómez C.

Una forma más de hacerlo.

- Se definió un tipo complejo con un contenido complejo.
- El elemento complexContent implica que se intenta restringir o extender el modelo de contenido de un tipo complejo, y la restricción de un entero declara un atributo pero no introduce ningún contenido de elemento.
- Posible declarar el elemento producto de forma más compacta

```
<xs:element name="producto">
  <xs:complexType>
    <xs:attribute name="prodid" type="xs:positiveInteger"/>
  </xs:complexType>
</xs:element>
```

Lámina 53 Roberto Gómez C.

Otro forma más.

- Es posible asignarle al elmento complexType un nombre y dejar que el elemento “producto” tengo un tipo de atributo que se refiere al nombre del complexType
  - Si se usa este método, varios elementos puede referirse al mismo tipo complejo:

```
<xs:element name="product" type="prodtype"/>

<xs:complexType name="prodtype">
  <xs:attribute name="prodid" type="xs:positiveInteger"/>
</xs:complexType>
```

Lámina 54 Roberto Gómez C.

Elementos que solo contienen elementos

- Elemento persona que solo contiene otros elementos

```
<persona>
  <firstname>John</firstname>
  <lastname>Smith</lastname>
</persona>
```

<xs:sequence>: Los elementos deben aparecerer en dicho orden

- Definiendo elemento “persona”

```
<xs:element name="persona">
 <xs:complexType>
  <xs:sequence>
    <xs:element name="firstname" type="xs:string"/>
    <xs:element name="lastname" type="xs:string"/>
  </xs:sequence>
 </xs:complexType>
</xs:element>
```

Lámina 55 Roberto Gómez C.

Otra forma de hacerlo

- Posible asignar al elemento complexType un nombre y dejar que el elemento persona cuente con tipo de atributo que se refiera al nombre del complexType.

```
<xs:element name="persona" type="persontype"/>

<xs:complexType name="persontype">
 <xs:sequence>
   <xs:element name="firstname" type="xs:string"/>
   <xs:element name="lastname" type="xs:string"/>
 </xs:sequence>
</xs:complexType>
```

Lámina 56 Roberto Gómez C.

Elementos de solo texto

- Este tipo solo esta compuesto de contenido simple
  - Texto y atributos.
- Por lo que solo se añade un elemento simpleContent alrededore del contenido.
- Cuando se una un contenido simple, se debe definir una extensión o una restricción dentro del elemento simpleContent

```
<xs:element name="somename">
  <xs:complexType>
    <xs:simpleContent>
      <xs:extension base="basetype">
        ....
        ....
      </xs:extension>
    </xs:simpleContent>
  </xs:complexType>
</xs:element>
```

```
<xs:element name="somename">
  <xs:complexType>
    <xs:simpleContent>
      <xs:restriction base="basetype">
        ....
        ....
      </xs:restriction>
    </xs:simpleContent>
  </xs:complexType>
</xs:element>
```

Lámina 57 Roberto Gómez C.

Ejemplo

- Elemento XML, *shoesize*, que solo contiene texto

```
<shoesize country="france">35</shoesize>
```

- Delaración de complexType *shoesize*.
  - El contenido es definido como un valor entero y el elemento *shoesize* tambien contiene un atributo de nombre *country*

```
<xs:element name="shoesize">
  <xs:complexType>
    <xs:simpleContent>
      <xs:extension base="xs:integer">
        <xs:attribute name="country" type="xs:string" />
      </xs:extension>
    </xs:simpleContent>
  </xs:complexType>
</xs:element>
```

Lámina 58 Roberto Gómez C.

Otra opción

- Se le da al elemento complexType un nombre con un tipo de atributo que hace referencia al tipo complejo.
  - Si se usa este método varios elementos pueden hacer referencia al mismo tipo complejo.

```
<xs:element name="shoesize" type="shoetype"/>

<xs:complexType name="shoetype">
  <xs:simpleContent>
    <xs:extension base="xs:integer">
      <xs:attribute name="country" type="xs:string" />
    </xs:extension>
  </xs:simpleContent>
</xs:complexType>
```

Lámina 59 Roberto Gómez C.

Elementos mezclados

- Elemento XML

```
<carta>
  Estimado Sr.<nombre>Jose Perez</nombre>.
  Su orden <norden>1032</norden>
  sera embarcada el <fechaembarque>2001-07-13</fechaembarque>.
</carta>
```

- Declaración

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

Lámina 60 Roberto Gómez C.

Otra opción

- Declarar un elemento complexType

```
<xs:element name="carta" type="tipocarta"/>

<xs:complexType name="tipocarta" mixed="true">
  <xs:sequence>
    <xs:element name="nombre" type="xs:string"/>
    <xs:element name="norden" type="xs:positiveInteger"/>
    <xs:element name="fechaembarque" type="xs:date"/>
  </xs:sequence>
</xs:complexType>
```

Lámina 61 Roberto Gómez C.

Indicadores

- Controlar como serán utilizados los elementos dentro de los documentos.
- Existen siete indicadores
  - Indicadores de orden: definen el orden de los elementos
    - All
    - Choice
    - Sequence
  - Indicadores de ocurrencia: cuantas veces puede ocurrir un elemento
    - maxOccurs
    - minOccurs
  - Indicadores de grupo: definir conjuntos elementos relacionados
    - Group name
    - attributeGroup name

Lámina 62 Roberto Gómez C.

Indicador all

- Elementos hijos pueden aparecer en cualquier orden.

```
<xs:element name="persona">
  <xs:complexType>
    <xs:all>
      <xs:element name="nombre" type="xs:string"/>
      <xs:element name="apellido" type="xs:string"/>
    </xs:all>
  </xs:complexType>
</xs:element>
```

Lámina 63 Roberto Gómez C.

Indicadores choice

- Puede ocurrir un hijo u otra cosa

```
<xs:element name="persona">
  <xs:complexType>
    <xs:choice>
      <xs:element name="empleado" type="empleado"/>
      <xs:element name="miembro" type="miembro"/>
    </xs:choice>
  </xs:complexType>
</xs:element>
```

Lámina 64 Roberto Gómez C.

Indicador sequence

- Orden hijos obligatorio

```
<xs:element name="persona">
  <xs:complexType>
   <xs:sequence>
     <xs:element name="nombre" type="xs:string"/>
     <xs:element name="apellido" type="xs:string"/>
   </xs:sequence>
  </xs:complexType>
</xs:element>
```

Lámina 65 Roberto Gómez C.

Indicador maxOccurs

- Máximo numero de ocurrencias elemento

```
<xs:element name="persona">
  <xs:complexType>
    <xs:sequence>
      <xs:element name="nombre_completo" type="xs:string"/>
      <xs:element name="nombre_hijo" type="xs:string" maxOccurs="10"/>
    </xs:sequence>
  </xs:complexType>
</xs:element>
```

Lámina 66 Roberto Gómez C.

Indicador minOccurs

- Mínimo número de veces elementos ocurre

```
<xs:element name="persona">
  <xs:complexType>
    <xs:sequence>
      <xs:element name=" nombre_completo " type="xs:string"/>
      <xs:element name=" nombre_hijo " type="xs:string“ maxOccurs="10" minOccurs="0“/>
    </xs:sequence>
  </xs:complexType>
</xs:element>
```

Lámina 67 Roberto Gómez C.

Ejemplo Myfamily.xml

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<personas xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="family.xsd">

<persona>
  <nombre_completo>Agustin Lopez</nombre_completo>
  <nombre_hijo>Carmen</child_name>
</persona>

<persona>
  <nombre_completo>Luis Sandoval</nombre_completo>
  <nombre_hijo>Alberto</nombre_hijo>
  <nombre_hijo>Gabriel</nombre_hijo>
  <nombre_hijo>Yolanda</nombre_hijo>
  <nombre_hijo>Erika</nombre_hijo>
</persona>

<persona>
  <nombre_completo>Juan Corral</nombre_completo>
</persona>

</personas>
```

Lámina 68 Gómez C.

Archivo family.xsd

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
elementFormDefault="qualified">

<xs:element name="personas">
  <xs:complexType>
    <xs:sequence>
      <xs:element name="persona" maxOccurs="unbounded">
        <xs:complexType>
          <xs:sequence>
            <xs:element name="nombre_completo" type="xs:string"/>
            <xs:element name="nombre_hijo" type="xs:string"
            minOccurs="0" maxOccurs="5"/>
          </xs:sequence>
        </xs:complexType>
      </xs:element>
    </xs:sequence>
  </xs:complexType>
</xs:element>

</xs:schema>
```

Lámina 69 Roberto Gómez C.

Indicadores de grupo

- Usados para definir grupos relacionados de elementos.
- Los grupos de elementos son definidos por una declaración de grupo que los engloba.

```
<xs:group name="groupname">
...
</xs:group>
```

- Cuando se incluye una declaración de grupo, deben definirse dentro de ella un indicador de secuencia (sequence), de elección (choice) o de todos (all).
- En esta secuencia se debe hacer referencia al grupo definido anteriormente, utilizando la fórmula xs:group ref='(nombre)'.

Lámina 70 Roberto Gómez C.

Ejemplo

- Ejemplo grupo de nombre *persongroup*, que define un grupo de elementos que debe ocurrir en una secuencia exacta.

```
<xs:group name="persongroup">
 <xs:sequence>
  <xs:element name="firstname" type="xs:string"/>
  <xs:element name="lastname" type="xs:string"/>
  <xs:element name="birthday" type="xs:date"/>
 </xs:sequence>
</xs:group>
```

Lámina 71 Roberto Gómez C.

Referenciando a un grupo

- Después de definir un grupo, este se puede referenciar de la siguiente forma

```
<xs:group name="persongroup">
 <xs:sequence>
  <xs:element name="firstname" type="xs:string"/>
  <xs:element name="lastname" type="xs:string"/>
  <xs:element name="birthday" type="xs:date"/>
 </xs:sequence>
</xs:group>

<xs:element name="person" type="personinfo"/>

<xs:complexType name="personinfo">
 <xs:sequence>
  <xs:group ref="persongroup"/>
  <xs:element name="country" type="xs:string"/>
 </xs:sequence>
</xs:complexType>
```

Lámina 72 Roberto Gómez C.

Ejemplo 2

```
<xs:group name='catalogoLibros'>
  <xs:sequence>
 <xs:element name='titulo' type='xs:string'/>
 <xs:element name='autor' type='xs:string'/>
  </xs:sequence>
</xs:group>

<xs:element name='libro' type='libroInfo'/>
  <xs:complexType name='libroInfo'>
  <xs:sequence>
 <xs:group ref='catalogoLibros'/>
 <xs:element name='NOReg' type='xs:string'/>
  </xs:sequence>
  </xs:complexType>
</xs:element>
```

```
<catalogoLibros>
 <libro>
    <titulo>Catálogo de Arte</titulo>
    <autor>Manuel Alonso Santos</autor>
    <NOReg>001</NOReg>
 </libro>
 <libro>
    <titulo>Las Misas de San Gregorio</titulo>
    <autor>Patricia Sela del Pozo</autor>
    <NOReg>002</NOReg>
 </libro>
 <libro>
    <titulo>El Arte Kitch</titulo>
    <autor>Carlos Delgado Mayordomo</autor>
    <NOReg>003</NOReg>
 </libro>
</catalogoLibros>
```

Lámina 73 Roberto Gómez C.

Atributos de grupos

- Atributos grupos son definidos con la declaración attributeGroup

```
<xs:attributeGroup name="groupname">
...
</xs:attributeGroup>
```

- Después de definir un **grupo de atributos**, se debe referenciar a ese grupo desde otro grupo o elemento de tipo compuesto.
- El siguiente ejemplo define un atributo de grupo de nombre “personattrgroup”

```
<xs:attributeGroup name="personattrgroup">
  <xs:attribute name="firstname" type="xs:string"/>
  <xs:attribute name="lastname" type="xs:string"/>
  <xs:attribute name="birthday" type="xs:date"/>
</xs:attributeGroup>
```

Lámina 74 Roberto Gómez C.

Referenciando al atributo creado

- Una vez que se definió un atributo, se puede referenciar en otra definición.

```
<xs:attributeGroup name="personattrgroup">
  <xs:attribute name="firstname" type="xs:string"/>
  <xs:attribute name="lastname" type="xs:string"/>
  <xs:attribute name="birthday" type="xs:date"/>
</xs:attributeGroup>

<xs:element name="person">
  <xs:complexType>
    <xs:attributeGroup ref="personattrgroup"/>
  </xs:complexType>
</xs:element>
```

Lámina 75 Roberto Gómez C.

Ejemplo atributo grupo

```
<xs:attributeGroup name='atributosLibro'>
   <xs:sequence>
  <xs:attribute name='codigo' type='xs:string'/>
  <xs:attribute name='publicacion' type='xs:date'/>
   </xs:sequence>
</xs:attributeGroup>

<xs:element name='libro'/>
   <xs:complexType>
   <xs:sequence>
    <xs:attributeGroup ref='atributosLibro'/>
   </xs:sequence>
   </xs:complexType>
</xs:element>
```

```
<libro codigo='02416' publicacion='1965'>
Smith, Paul: 'Mi vida'</libro>>
```

Lámina 76 Roberto Gómez C.

El elemento ANY

- Permite extender el documento XML con elementos no especificados en el esquema.
- Ejemplo:

```
<xs:element name="persona">
  <xs:complexType>
    <xs:sequence>
      <xs:element name="nombre" type="xs:string"/>
      <xs:element name="apellido" type="xs:string"/>
      <xs:any minOccurs="0"/>
    </xs:sequence>
  </xs:complexType>
</xs:element>
```

El uso del elemento any permite extender (después de <apellido>) el contenido de persona con cualquier elemento.

Lámina 77 Roberto Gómez C.

Extendiendo la definición

- Es posible extender el elemento persona con un elemento *hijo*.
- Es posible, aunque el esquema anterior no tenga declarado ningún elemento *hijo*.
- Ejemplo
  - Considerar la siguiente definición del elemento hijo:

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<xs:schema
xmlns:xs="http://www.w3.org/2001/XMLSchema"
targetNamespace="http://www.acme.com"
xmlns="http://www.acme.com"
elementFormDefault="qualified">

<xs:element name="hijos">
  <xs:complexType>
    <xs:sequence>
      <xs:element name="nombrehijo" type="xs:string"
      maxOccurs="unbounded"/>
    </xs:sequence>
  </xs:complexType>
</xs:element>

</xs:schema>
```

Lámina 78 Roberto Gómez C.

Ejemplo de XML con esquema anterior.

```
<?xml version="1.0" encoding="ISO-8859-1"?>

<personas xmlns="http://www.microsoft.com"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:SchemaLocation="http://www.microsoft.com family.xsd
http://www.acme.com hijos.xsd">

<persona>
  <nombre>Hege</nombre>
  <apellido>Refsnes</apellido>
  <hijos>
    <nombrehijo>Cecilie</nombrehijo>
  </hijos>
</persona>

<persona>
  <nombre>Stale</nombre>
  <apellido>Refsnes</apellido>
</persona>

</personas>
```

Lámina 79 Roberto Gómez C.

El elemento <anyAttribute>

- Permite extender documentos XML con atributos no especificados por el esquema
- Usando <anyAttribute> se puede añadir cualquier número de atributos al elemento "*persona*"

```
<xs:element name="persona">
  <xs:complexType>
    <xs:sequence>
      <xs:element name="nombre" type="xs:string"/>
      <xs:element name="apellido" type="xs:string"/>
      <xs:anyAttribute/>
    </xs:sequence>
  </xs:complexType>
</xs:element>
```

Lámina 80 Roberto Gómez C.

Expandiendo los atributos

- Extendiendo la definición del elemento “persona” con el atributo “sexo”.
- Posible hacerlo a pesar de que el esquema no declaro ningún atributo “sexo”.

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
targetNamespace="http://www.w3schools.com"
xmlns="http://www.w3schools.com"
elementFormDefault="qualified">

<xs:attribute name=“sexo">
  <xs:simpleType>
    <xs:restriction base="xs:string">
      <xs:pattern value="masculino|femenino"/>
    </xs:restriction>
  </xs:simpleType>
</xs:attribute>

</xs:schema>
```

Lámina 81 Roberto Gómez C.

Ejemplo XML con esquema anterior: Myfamily.xml

```
<?xml version="1.0" encoding="ISO-8859-1"?>

<personas xmlns="http://www.microsoft.com"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:SchemaLocation="http://www.microsoft.com family.xsd
http://www.w3schools.com attribute.xsd">

<persona sexo="femenino">
  <nombre>Gilda</nombre>
  <apellido>Osorio</apellido>
</persona>

<persona sexo="masculino">
  <nombre>Ricardo</nombre>
  <apellido>Rodriguez</apellido>
</persona>

</personas>
```

Lámina 82 Roberto Gómez C.

Sustitución elementos

- Posibilidad de asignar nombres diferentes a un mismo elemento.
- Uso de la primitiva: substitutionGroup en el esquema XML.
- Primero se declara un elemento de encabezado y después los otros elementos que pueden ser substituidos por el elemento de encabezado.

Elemento de encabezado

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

Elemento susbtiruible

Lámina 83 Roberto Gómez C.

Ejemplo sustitución elementos

- Elementos nombre y name, hacen referencia a lo mismo.
- Elementos cliente y customer, también.

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

<xs:complexType nombre="infocliente">
  <xs:sequence>
    <xs:element ref="nombre"/>
  </xs:sequence>
</xs:complexType>

<xs:element name="cliente" type="infocliente"/>
<xs:element name="customer" substitutionGroup="cliente"/>
```

Lámina 84 Roberto Gómez C.

Ejemplo XML con sustitución

- Un documento valido XML, de acuerdo al esquema anterior es:

```
<cliente>
  <nombre>John Smith</nombre>
</cliente>
```

- Otro documento válido es

```
<customer>
  <name>John Smith</name>
</customer>
```

Lámina 85 Roberto Gómez C.

Bloqueando substituciones

- Para prevenir substituciones se puede usar el atributo de bloqueo

```
<xs:element name="name" type="xs:string" block="substitution"/>
```

- Ejemplo de uso

```
<xs:element name="nombre" type="xs:string" block="substitution"/>
<xs:element name="name" substitutionGroup="nombre"/>

<xs:complexType nombre="infocliente">
  <xs:sequence>
    <xs:element ref="nombre"/>
  </xs:sequence>
</xs:complexType>

<xs:element name="cliente" type="infocliente" block="substitution"/>
<xs:element name="customer" substitutionGroup="cliente"/>
```

Lámina 86 Roberto Gómez C.

Documento XML con esquema anterior

- Documento válido

```
<cliente>
  <nombre>John Smith</nombre>
</cliente>
```

- Documento inválido

```
<customer>
  <name>John Smith</name>
</customer>
```

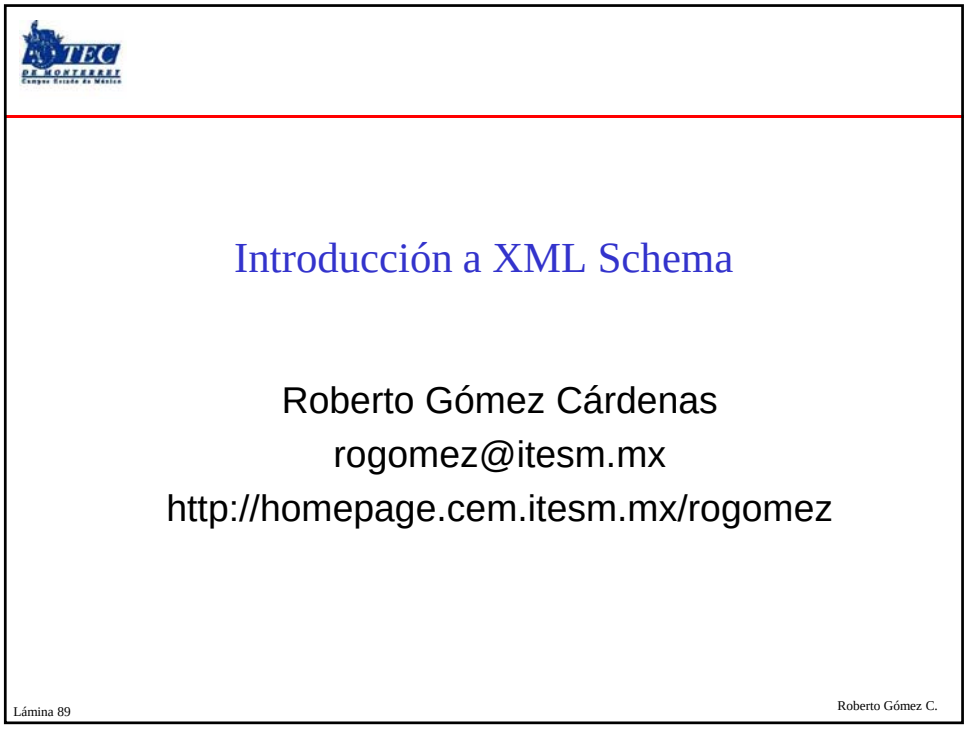
Lámina 87 Roberto Gómez C.

Tabla comparativa

DTD	XML Schema	Comentarios
ELEMENT	<element>	Crea un vínculo entre un nombre y unos atributos, modelos de contenido y anotaciones
#PCDATA	Soportado como parte de un tipo simple	
ANY	<any>	Posee distintos comodines para un mayor conjunto de posibilidades. Existe también <anyAttribute> con comodines similares.
EMPTY	Soportado	Se elimina la existencia de elementos descendientes del actual, diferenciando de la presencia de un *string* vacío en un elemento.
Modelo de Contenido	<complexType>	
, (Conector de secuencia)	<sequence>	
| (Conector de alternativas)	<disjunction>	
? (Opcional)	Soportado	Se han de emplear los atributos predefinidos de maxOccurs y minOccurs
+(Requerido y Repetible)	Soportado	Se han de emplear los atributos predefinidos de maxOccurs y minOccurs
*(Opcional y Repetible)	Soportado	Se han de emplear los atributos predefinidos de maxOccurs y minOccurs
ATTLIST	<attributeGroup>	Se pueden agrupar declaraciones de <attributes>
Tipo de atributo CDATA, ID, IDREF, NOTATION...	Tipos <simpleType>predefinidos	
ENTITY	NO Soportado	Las entidades son declaradas en declaraciones de marcas en el XML
ENTITY%Parameter	NO Soportada	Las entidades paramétricas ofrecen un mecanismo de bajo nivel que permite distintas cosas, algunas de estas se han intentado cubrir en XML Schema: .. La separación entre <element> y <complexType> .. Grupos de atributos .. Grupos de modelos con nombre .. Mecanismos de extensión y restricción de tipos .. Los mecanismos de <import> e <include> para componer esquemas .. El mecanismo de redefinición de elementos

Referencia: http://www.hipertexto.info/documentos/dtds.htm

Lámina 88 Roberto Gómez C.


Introducción a XML Schema
Roberto Gómez Cárdenas
rogomez@itesm.mx
http://homepage.cem.itesm.mx/rogomez
Lámina 89
Roberto Gómez C.