

---

# **Modelado de datos XML: XML-Schema**

*Ofimática Avanzada*

Profesor: Víctor Fresno Fernández  
curso 2006/07

# Fundamentos de XML-Schema

---

## Limitaciones de las DTD

- ◆ No permite definir tipos de datos
- ◆ No permite espacios de nombres
- ◆ Las DTD están limitadas a contenido textual
- ◆ Imponen restricciones de repetición (+, \*, ?)

# Fundamentos de XML-Schema

---

**Por estas limitaciones se propone un esquema con..**

- ♦ Sintaxis XML.. aunque resulta más complejo y menos legible que las DTD
- ♦ Con soporte de tipos de datos
- ♦ Introducción de conceptos de la programación orientada a objetos

# Fundamentos de XML-Schema

---

El W3C y su grupo de trabajo (<http://www.w3c.org/XML>) proponen un esquema con

- ◆ Datos XML inspirados en los tipos de datos SQL
- ◆ DCD (descripción del contenido del documento)
- ◆ SOX (esquema para XML orientado a objetos)
- ◆ DDML (lenguaje de marcado de definición de documentos)

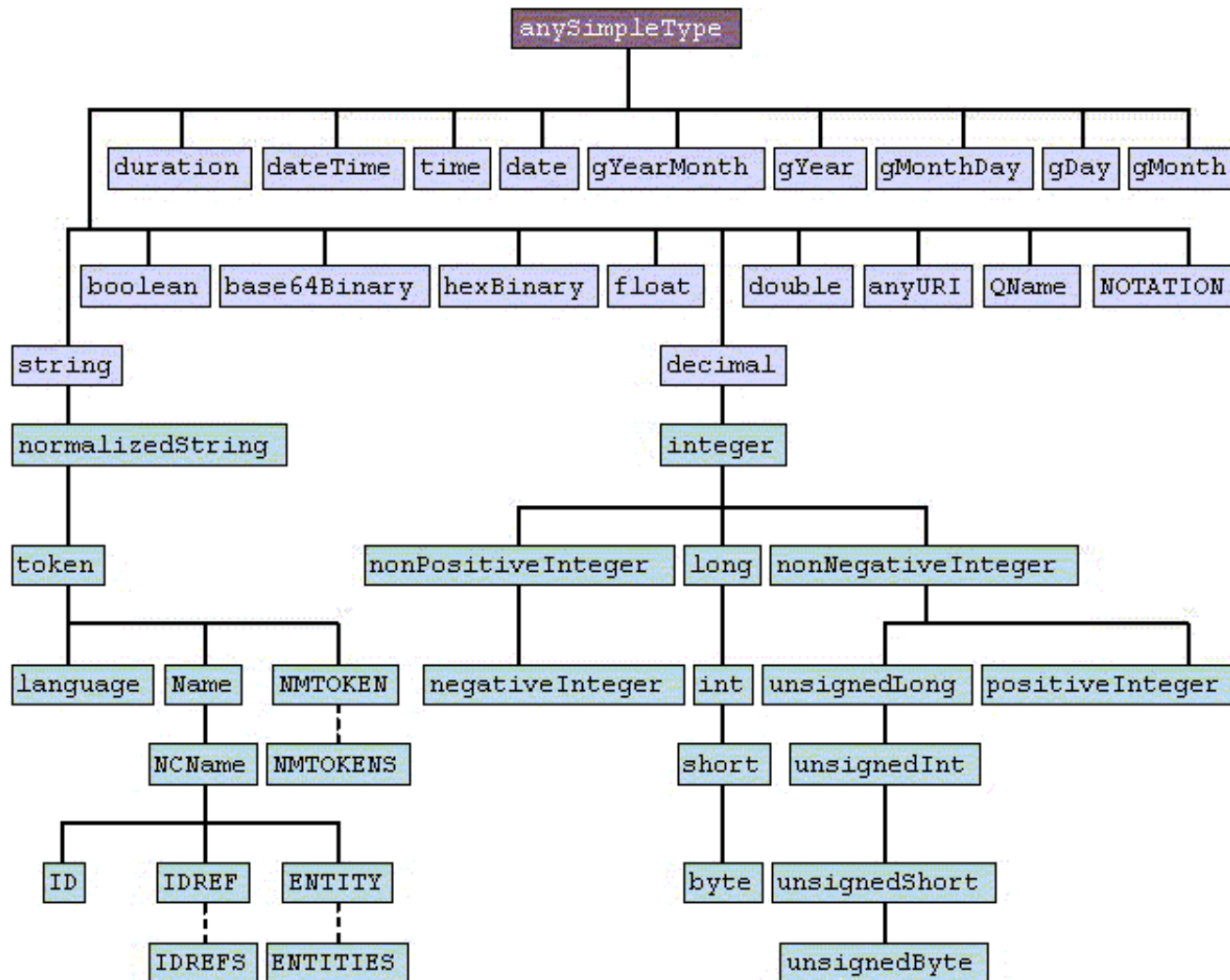
# Fundamentos de XML-Schema

---

- ◆ Los XML-Schemas son documentos XML bien formados
- ◆ Se crea alrededor del concepto de tipo (entero, fecha, ..) y de instancias a estos tipos
  - Existen tipos básicos simples
  - Permite definir tipos complejos de datos

Cabecera del XML-Schema

```
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema"/>
```



- ur types
- built-in primitive types
- built-in derived types
- complex types

- derived by restriction
- derived by list
- derived by extension or restriction

# Fundamentos de XML-Schema

---

## Tipos simples

- ◆ El primer uso de estos tipos es la declaración de elementos y atributos.
- ◆ Sólo pueden contener información de caracteres (no permiten ni atributos ni elementos)

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

```
<xsd:attribute name="identificacion" type="xsd:ID"/>
```

# Fundamentos de XML-Schema

---

## Tipos complejos

- ◆ Pueden contener cualquier combinación de contenido de elementos , información de caracteres y atributos

```
<xsd:complexType name="Programa" >
```

```
  <xsd:sequence>
```

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

```
    <xsd:element name="descripcion" type="xsd:string"/>
```

```
  </xsd:sequence>
```

```
  <xsd:attribute name="identificacion" type="xsd:ID"/>
```

```
</xsd:complexType>
```

# Fundamentos de XML-Schema

---

## Tipos complejos

```
<xsd:element name="programa" type="Programa"/>
```

Documento XML válido

```
<programa identificacion="g0039">  
  <nombre>La siesta de los leones</nombre>  
  <descripcion>documental de sobremesa.. </descripcion>  
</programa>
```

# Fundamentos de XML-Schema

---

## Tipos complejos

- ◆ Es posible anidar tipos complejos
- ◆ Podemos fijar patrones en cadenas de caracteres

```
<xsd:element name="sw-clave" type="xsd:string">
```

```
  <xsd:simpleContent>
```

```
    <xsd:restriction>
```

```
      <xsd:patter value="/d{4}-[A-Z]{3}-/d{5}"/>
```

```
    <xsd:/restriction>
```

```
  </xsd:simpleContent>
```

```
</xsd:element>
```

# Fundamentos de XML-Schema

---

## Tipos anónimos

- ◆ Se aplican cuando tenemos muchos tipos con poca diferencia y con pocos elementos que lo usen
- ◆ No son reutilizables
- ◆ En lugar de declarar un tipo complejo y declarar un elemento de este tipo complejo, se declara un tipo anónimo..

# Fundamentos de XML-Schema

---

## Tipos anónimos

```
<xsd:element name="sw-clave" >
  <xsd:simpleType>
    <xsd:restriction base="xsd:string">
      <xsd:patter value="/d{4}-[A-Z]{3}-/d{5}"/>
    </xsd:restriction>
  </xsd:simpleType >
</xsd:element>
```

# Fundamentos de XML-Schema

---

## Modelos de contenido

.. Elementos que contienen elementos

```
<xsd:complexType name="elemento" >  
  <xsd:sequence>  
    <xsd:element name="subelemento1" type="xsd:string"/>  
    <xsd:element name="subelement2" type="xsd:string"/>  
  </xsd:sequence>  
</xsd:complexType>
```

# Fundamentos de XML-Schema

---

## Modelos de contenido

.. Elementos que contienen alternativa de elementos

```
<xsd:complexType name="elementoTipoChoice" >  
  <xsd:choice>  
    <xsd:element name="subelement1" type="xsd:string"/>  
    <xsd:element name="subelement2" type="miTipoComplejo"/>  
  </xsd:choice>  
</xsd:complexType>
```

# Fundamentos de XML-Schema

---

## Modelos de contenido

.. Elementos con contenido mixto

```
<xsd:element nombre="elemetoMixto">
```

```
  <xsd:complexType mixed="true" >
```

```
    <xsd:choice>
```

```
      <xsd:element name="subelement1" type="xsd:string"/>
```

```
      <xsd:element name="subelement2" type="miTipoComplejo"/>
```

```
    </xsd:choice>
```

```
  </xsd:complexType>
```

```
</xsd:element>
```

# Fundamentos de XML-Schema

---

## Modelos de contenido

.. Los elementos vacíos se declaran como tipos complejos que sólo tengan atributos

```
<xsd:complexType name="elementoVacio" >  
    <xsd:attribute name="attribute1" type="xsd:string"/>  
    <xsd:attribute name="attribute2" type="xsd:date"/>  
</xsd:complexType>
```

# Fundamentos de XML-Schema

---

## Cardinalidades

- ◆ Atributos minOccurs y maxOccurs

```
<xsd:complexType name="pista_cd" >
```

```
  <xsd:sequence>
```

```
    <xsd:element name="subelement2" type="miTipoComplejo" minOccurs="1"
maxOccurs="*" />
```

```
  </xsd:sequence>
```

```
</xsd:complexType>
```

# Fundamentos de XML-Schema

---

## Definición de grupos

- ♦ Igual que en las DTD podíamos definir entidades para reciclar grupos de declaraciones

```
<!ENTITY % grupoDeElementos "subelement1,subelemento2">
```

# Fundamentos de XML-Schema

---

.. en los XML-Schemas se pueden declarar grupos

```
<xsd:group nombre="grupoDeElementos">
```

```
  <xsd:sequence>
```

```
    <xsd:element name="subelement1" type="xsd:string"/>
```

```
    <xsd:element name="subelement2" type="miTipoComplejo"/>
```

```
  </xsd:sequence ></xsd:group>
```

♦ Luego se reutilizan..

```
<xsd:complexType nombre="elemento">
```

```
  <xsd:group ref="grupoDeElementos"/>
```

```
</xsd:complexType>
```

# Fundamentos de XML-Schema

---

## Anotaciones

- ◆ XML-Schema proporciona 2 elementos de comentarios
  - Documentation (explicación al lector)
  - AppInfo (explicación al lector o a un programa)

```
<xsd:complexType nombre="elemento">
```

```
  <xsd:annotation">
```

```
    <xsd:documentation>este elemento es .. </xsd:documentation>
```

```
  </xsd:annotation">
```

```
</xsd:complexType>
```

