Fundamentos del XML-Schema o XSD

El XSD (XML Schema Definition) surgió como alternativa a la DTD y para superar algunas de las limitaciones de ésta. Las principales ventajas del XSD con respecto a la DTD son las siguientes:

- Usa sintaxis XML. De hecho, los XSD son documentos XML bien formados.
- Permite definir tipos de datos
- Presenta un modelo de datos abierto usando conceptos de la orientación a objetos.
- Soporta la integración de los espacios de nombres.
- Permite expresar conjuntos
- Permite construir tipos complejos de datos.
- No impone restricciones de repetición generales como la DTD

En XSD hay sólo dos categorías de elementos:

- Tipo simple: elementos con modelo de contenido simple y sin atributos. Sólo pueden contener información de caracteres.
- Tipo complejo: elementos con cualquier otro modelo.

Definición de elementos

Cada tipo de datos tiene un **espacio léxico**, un espacio de valores y un mapeo del espacio léxico al de valores. Dado un tipo de datos, su **espacio de valores** es el conjunto de sus valores posibles. Cada valor en el espacio de valores se puede representar por uno o más literales en su espacio léxico. El **espacio léxico** es el conjunto de literales válidos o representaciones léxicas para un tipo de datos.

schema.

Es el elemento raíz del XSD. Su sintaxis es la siguiente:

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

Este elemento define el espacio de nombres en el que están definidos todos los elementos y tipos de datos de XML-Schema. Como un XSD es una instancia o documento XML puede llevar un preámbulo:

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

element.

Define los elementos que contendrán los documentos XML asociados al XSD. La sintaxis para definir un elemento de tipo simple es la siguiente:

```
<xs:element name =" nombreElemento " type =" tipoElemento "/>
```

Veamos ahora cómo se define un elemento de tipo complejo

```
<xs:element name ="titulo ">
    <xs:complexType >
        <xs:simpleContent >
            <xs:extension base =" xs:string ">
                 <xs:attribute ref="leng "/>
            </ xs:extension >
        </ xs:simpleContent >
    </ xs:complexType >
</ xs:element >
```

El elemento *titulo* es de tipo complejo porque tiene un atributo, pero su contenido es simple porque es de tipo carácter o texto. Además en la definición hay que indicar que tiene el atributo *leng*.

otro ejemplo de definición de tipo complejo

```
<xs:element name =" archivo">
  <xs:complexType >
    <xs:sequence >
      <xs:element ref=" informe" maxOccurs =" unbounded "/>
    </ xs:sequence >
  </ xs:complexType >
</ xs:element >
```

Esta definición indica que el elemento *archivo* es de tipo complejo, está compuesto por una secuencia de 1 a n ocurrencias de elementos *informe*.

Cuando los elementos y atributos se definen directamente dentro del elemento documento xs:schema, se denominan **globales**. Los componentes globales se pueden referenciar en cualquier parte del esquema y en otros esquemas que lo importen. Si no fuera asi se denominan **locales**.

Definición de atributos

attribute.

Define los atributos que podrán contener los elementos. La sintaxis para definirlos es la siguiente:

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

La declaración de los atributos es independiente de la de los elementos.

Tipos de datos predefinidos

Tipos de cadena.

- **xs:string** Es una cadena de caracteres válidos Unicode e ISO.
- **xs:normalizedString** Al igual que el anterior también es una cadena de caracteres válidos Unicode e ISO, pero se reemplazan los caracteres tabulador (#x9), linefeed (#xA), y retorno de carro (#xD) por espacio (#x20).

- **xs:token** Es similar al tipo xs:normalizedString pero en éste se eliminan los espacios
  al principio y al final, y varios espacios contiguos se sustituyen por uno simple.
- **xs:language** Se deriva de xs:token. Se creó para aceptar otros códigos de lenguaje.
- **xs:NMTOKEN** Se corresponde con el tipo NMTOKEN visto en las DTD.
- **xs:Name** Es similar a xs:NMTOKEN con la restricción de que los valores deben comenzar con una letra o con los caracteres «:» o «_».
- **xs:NCName** Es similar a xs:Name con la restricción de que los valores deben comenzar con una letra o con el carácter «_».
- **xs:ID** Se deriva de xs:NCName. Su valor debe ser único en el documento ya que se
  trata de un identificador único.
- **xs:IDREF** Se deriva de xs:NCName. Su valor debe emparejarse con un ID definido en
  el mismo documento.
- **xs:ENTITY** Se deriva de xs:NCName. Su valor debe emparejarse con una entidad externa no analizada.
- **xs:QName** Soporta espacios de nombres con prefijo.
- **xs:anyURI** El valor deberá cumplir las limitaciones de los caracteres admitidos en
  XML.
- **xs:hexBinary** Permite codificar contenido binario como una cadena de caracteres traduciendo el valor de cada octeto binario en 2 dígitos hexadecimales.
- **xs:base64Binary** Corresponde a la codificación «base64» que agrupa series de 6 bits
  en un array de 64 caracteres imprimibles.

Tipos numéricos

- **xs:decimal** Representa números decimales arbitrariamente largos. El separador decimal es «.» y puede contener un signo inicial «+» o «-«.
- **xs:nonPositiveInteger** Es el subconjunto de xs:integer formado por negativos y cero.
- **xs:negativeInteger** Es el subconjunto de xs:nonPositiveInteger formado por negativos.
- **xs:nonNegativeInteger** Es el subconjunto de xs:integer formado por positivos y cero.
- **xs:positiveInteger** Es el subconjunto de xs:nonNegativeInteger formado por positivos.
- **xs:long** Enteros que pueden almacenarse en 64 bits.
- **xs:int** Enteros que pueden almacenarse en 32 bits.
- **xs:short** Enteros que pueden almacenarse en 16 bits.
- **xs:byte** Enteros que pueden almacenarse en 8 bits.
- **xs:unsignedLong** Enteros no negativos que pueden almacenarse en 64 bits.
- **xs:unsignedInt** Enteros no negativos que pueden almacenarse en 32 bits.
- **xs:unsignedShort** Enteros no negativos que pueden almacenarse en 16 bits.
- **xs:unsignedByte** Enteros no negativos que pueden almacenarse en 8 bits.
- **xs:float** Representa números en notación científica con potencias enteras de 10 de 32 bits de precisión.
- **xs:double** Igual que xs:float salvo que en este caso la precisión es de 64 bits.

- **xs:boolean** Puede tomar valores true y false (1 y 0).

Tipos fecha y hora.

- **xs:dateTime** Define un instante de tiempo concreto. El formato es: YYYY-MMDDThh:mm:ss.
- **xs:date** Define un día concreto del calendario Gregoriano.
- **xs:gYearMonth** Es xs:date sin la parte de día.
- **xs:gYear** xs:gYearMonth sin la parte del mes.
- **xs:time** Define una hora concreta.
- **xs:gDay** Es un día del calendario Gregoriano.
- **xs:gMonthDay** Es un día de un mes del calendario Gregoriano.
- **xs:gMonth** Es un mes del calendario Gregoriano.
- **xs:duration** Expresa una duración en un espacio de 6 dimensiones: número de años,
  meses, días, horas, minutos y segundos. El formato es PnYnMnDTnHnMnS.

Tipo lista.

- **xs:NMTOKENS** Lista de xs:NMTOKEN separada por espacios.
- **xs:IDREFS** Lista de xs:IDREF separada por espacios.
- **xs:ENTITIES** Lista de xs:ENTITY separada por espacios.

Tipo no definido

El tipo anySimpleType acepta cualquier valor.

Creación de nuevos tipos de datos

Se pueden crear nuevos tipos de datos tomando como punto de partida los tipos de datos existentes. Este tipo de creación de datos se denomina derivación. Existen 3 mecanismos de derivación:

Derivación por restricción.

Los tipos de datos se crean añadiendo restricciones a los posibles valores de otros tipos. El propio XML-Schema usa este mecanismo. Las restricciones de un tipo se definen mediante facetas. Una restricción se añade con el elemento xs:restriction y cada faceta se define utilizando un elemento específico dentro de xs:restriction. El tipo de dato que se restringe se denomina tipo base.

```
<xs:simpleType name ="IntegerMuyCorto">
  <xs:restriction base ="xs:integer">
    <xs:minInclusive value ="-100"/>
    <xs:maxExclusive value ="100"/>
  </ xs:restriction >
</ xs:simpleType >
```

Las facetas se clasifican en 3 categorías:

- Las que definen el procesamiento de espacios en blanco, tabuladores, etc. Estas facetas actúan entre el espacio de análisis y el léxico.
- Las que trabajan sobre el espacio léxico.
- Las que restringen el espacio de valores.

Facetas de cadenas con procesamiento de espacios en blanco

En estas facetas se eliminan los espacios iniciales y finales, se sustituyen tab, line feed y CR por espacios y n espacios consecutivos se sustituyen por uno. Los tipos sobre los que pueden actuar estas facetas son: xs:ENTITY, xs:ID, xs:IDREF, xs:language, xs:Name,xs:NCName, xs:NMTOKEN, xs:token, xs:anyURI, xs:base64Binary, xs:hex-Binary, xs:NOTATION y xs:QName.

- **xs:enumeration** define una lista de posibles valores.
- **xs:length** define una longitud fija en número de caracteres o bytes.
- **xs:maxLength** define una longitud máxima en número de caracteres o bytes.
- **xs:minLength** define una longitud mínima en número de caracteres o bytes.
- **xs:pattern** define un patrón que debe emparejarse con la cadena.

Facetas de tipos numéricos reales

Estas facetas actúan sobre los tipos xs:float y xs:double restringiendo el espacio de valores.

- **xs:enumeration** permite definir una lista de posibles valores.
- **xs:maxExclusive** define un valor máximo que no se puede alcanzar.
- **xs:maxInclusive** define un valor máximo que sí se puede alcanzar.
- **xs:minExclusive** define un valor mínimo que no se puede alcanzar.
- **xs:minInclusive** define un valor mínimo que sí se puede alcanzar.
- **xs:pattern** define un patrón que debe cumplir el valor léxico del tipo de datos.

Facetas de tipos de fecha y hora

Son las mismas que para los tipos numéricos reales.

Facetas de tipos enteros

Las mismas facetas que para los tipos numéricos reales más la siguiente:

- **xs:totalDigits** define el número máximo de dígitos. Actúa sobre el espacio de valores.

Derivación por lista

Es un mecanismo que permite derivar un tipo de datos de lista a partir de un tipo de datos atómico. Todos los datos de la lista tienen que ser del mismo tipo.

Solo se permiten las siguientes facetas en este tipo de derivación: xs:length referida al número de elementos, xs:enumeration, xs:maxLength, xs:minLength y

xs:whiteSpace.
Este tipo de derivación se realiza con el elemento xs:list.

```
<xs:simpleType name =" listaInteger ">
  <xs:list itemType =" xs:integer "/>
</ xs:simpleType >
```

Derivación por unión

Este mecanismo permite definir nuevos tipos de datos fusionando los espacios léxicos de varios tipos predefinidos o definidos por el usuario. El tipo de dato resultante pierde la semántica y facetas de los tipos miembro. Este tipo de derivación se realiza con el elemento xs:union. Solo se permiten 2 facetas en este tipo de derivación: xs:pattern y xs:enumeration.

```
<xs:simpleType name =" integerUnionDate ">
  <xs:union memberTypes =" xs:integer xs:date"/>
</ xs:simpleType >
```

Espacios de nombres en XML-Schema

El XML-Schema definido por el W3C asocia un espacio de nombres a todos los objetos (elementos, atributos, tipos simples, complejos, . . .) definidos en un XSD.

Para asociar a un XSD un espacio de nombres, y que por lo tanto pertenezca a dicho espacio de nombres, se usa el atributo targetNamespace del elemento schema.

la declaración del espacio de nombres objetivo (targetNamespace) permite:

- Definir elementos y atributos que pertenecen a dicho espacio de nombres y que serán qualified.
- Definir elementos y atributos que no pertenecen a ningún espacio de nombres y por tanto serán unqualified.

Cuando se ha definido un espacio de nombres objetivo está prohibido definir elementos globales que sean de tipo unqualified. La distinción entre elementos y atributos qualified y unqualified se hace a través del atributo form. Los valores por defecto de los atributos form se definen en el elemento schema mediante elementFormDefault y attributeFormDefault.