

5) uniones - xpath (cc = context)

Nodes: ②

Nodo como nodo text

Wildcard que excluye atributos

Str: ①

1.1 • Starts-with (s1, s2) 1 con 2

[@key] Starts-with (., 'a') or Starts-with (., 'e') or ...]

1.2 • Contains (s1, s2) 1 o 2

//Concept [Contains (@*, 'business')]

↳ " € - r @ ? "

1.3 • String-length(s)

1.4 • Concat (s1, s2, ...)

1.5 • Substring-before ('alaja', 'ja')

↳ 'ala'

1.6 • Substring-after ('alaja', 'ala')

↳ 'ja'

Math: ③

3.1 • Ceiling (n), floor(n), round(n)

3.2 • number (str) (Son math func basicas)

3.3 • Sum (Value-list)

"Sum (x/a/t) div count(x/a/t)"

3.4 • x div y

//@name [count(..//Concept) <
(Count(..//Concept) div Count(..//Profile))]

3.5 • x mod y

Keys Se usan cuando se quiere crear una relación de referencias de IDs.

• Unique

<Key name="nom_clave">
 <selector xpath="/" />
 <Field xpath="/" />+
</key>

<Keyref name="nom_ref" Refer="nom_clave">
 <!... Idem key ---> </Keyref>

K: Obligatorio (elemento / atributo);

U: Opcional (elemento / atributo);

2.1 • node(): lista de nodos cc. !Φ=1 !Φ=1

//x/Profile [.//ImplData [not (node ()) and not (@*))]]/@n...

2.2 • Count (node) → ej: 3.3, 3.4.

2.3 • Position(): Position of cc node / Use it for requirements.

//Profile [last ()] //Concept [position () = 1] /@Key

2.4 • last() → ej: 2.3

2.5 • Combinación de 2+ Node-sets

//*[contains(name(), 'w')] //attribute::#[contains(name(), 'ID')]

Bool: ④

4.1 • not (boolean) → ej: 2.1

 not (*empty-set*) = true

4.2 • =, !=, <, >, <=, >=

Selectors ⑤ A lo largo de él no son modo text.

5.1 • Self::, child:: Sirve para filtrar entre el cc.

5.2 • descendant::, descendant-or-self::

5.3 • ancestor::, ancestor-or-self::

5.4 • attribute::

5.5 • parent:: → Padre/n inmediato

5.6 • preceding:: → Adelante lo que este arriba del cc. (TODO)

//Profile [not (. / @name = preceding::Profile / @name)]

Si no hay un anterior con igual nombre, lo selecciona.

5.7 • preceding-sibling:: → Selección lo que este arriba del cc y es "hermano".

Si queremos seleccionar los que no se refieren dentro de un mismo "SCOPE", pero no

nos importa si están a aparecer en el resto

del DOC, es decir si se refiere en otro lado, elegir igual

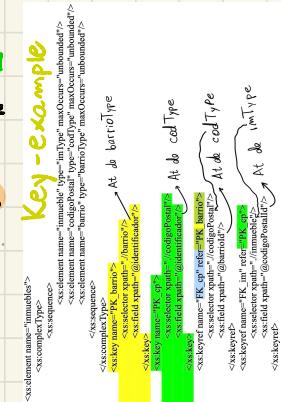
Anomalías

Redundancia de datos.

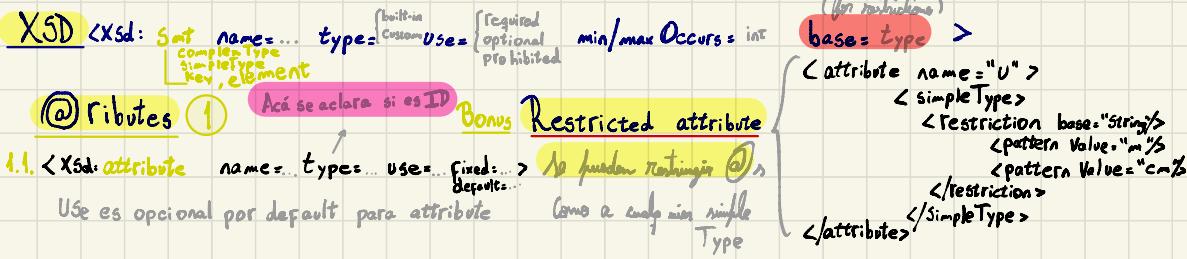
• de Actualización: 2 ramas distintas elementos con el mismo ID, si cambia algo en la rama 1, lo 2 (x)

• de Pobrando: Un objeto contiene elementos con ID's, si borro ese objeto, pierde el elemento.

Field: N-Campo || @... ; Selector: Path a campo / atributo



Para seleccionar
elementos distintos



Restrictions ② - base = SimpleType / complexType

2.1 Regex

<XSd:pattern value="regex"/>

2.1.1 ? $x == 0 \text{ || } x == 1$

2.1.5 Id = [0-9]

2.1.2 * $0 \leq x$

2.1.6 \n Newline

2.1.3 + $1 \leq x$

2.1.7 \r Carriage Return

2.1.4 . Wildcard

2.1.8 \t tab

2.2 Limits (for char limit use regex)

2.2.1 <min/max Inclusive Value = "Limit">

2.2.2 <totalDigits Value = "Limit"> FD included.

2.2.3 <fractionDigits Value = "Limit">

<restriction base="decimal">
<minInclusive value="100%"/> } Guarda 2 de los 5 dígitos para los
<max value="250%"/> } dígitos 000100.2000 ✓, 250.1 x
<totalDigits value="5"/>
<fractionDigits value="2"/> } se ignoran los 0 o extra.

</restriction>

2.2.4 <enumeration value="valor fijo"/>

2.4 List ↳ Una lista de enumeración - lista de valores posibles.

<list itemType="SimpleType"/>

Valido si se tiene una lista (separada por espacio c/o)
y todos cumplen

Si se tiene float, lo siguiente valido:

<Vary> 1 2 0.3 2.5 </Vary>

2.3 Union

<union memberTypes = "Type1 Type2 ..."/>

Valido si el elem cumple con uno de los Type i

<element name="Vary">
<SimpleType>

<UNION memberTypes="xsd:decimal xsd:date"/>

</SimpleType>

(<Vary> 0.2 </Vary> ✓; <Vary> 2025-12-30</Vary> ✓)

③ ComplexType: Sequence / choice / All

→ Se describen solo

3.1 Sequence : secuencia de elementos que DEBEN respetar el orden

Límite

3.2 Choice : Pool de elementos de los que se puede elegir 1 (si aparece `<choice maxOccurs="x">`)

3.3 All : Pool de elementos DEBEN estar todos pero en cualquier orden.

④ Extension (es la misma que restriccion pero el nombre, tambien se usa `base = "baseType"`)

5 DTD

4.1 De complexType a complexType 5.1 Def elementos: <!ELEMENT nombre Content> (5.2)

```
<complexType name="tipoDefinido">
  <complexContent>
    <extension base="ComplexTypeBase">
      <sequence>
        <sequence>
        </sequence>
      </sequence>
    </extension>
  </complexContent>
</complexType>
```

5.2 Content: 5.2.1 EMPTY; 5.2.2 (Lista de subelementos) (separados: , ; |)
5.2.3 (#PCDATA) TXT sin subelementos o φ (<...>); 5.2.4 (#PCDATA1...) Mixto.

5.3 Def attributes <!ATTLIST elementoNombre atributo tipo valorDef>

5.3.1 elementoNombre: Elemento al que pertenece el atributo

5.3.2 atr:Tipo: 5.3.2.1 CDATA . TXT; 5.3.2.3 ID: Si el valor es unico en el Doc;

5.3.2.2 (list): separados por | ; 5.3.2.4 IDREFS: ID/frente/s en otros elemento/s;

5.3.3 ValorDefault:

#REQUIRED: Obligatorio

#IMPLIED: Opcional

#FIXED "Valor": Siempre == Valor

"Valor": Valor es default (puede Sobreescribir)