estudios de postgrado en computación

Bases de datos avanzadas

Universidad de Los Andes

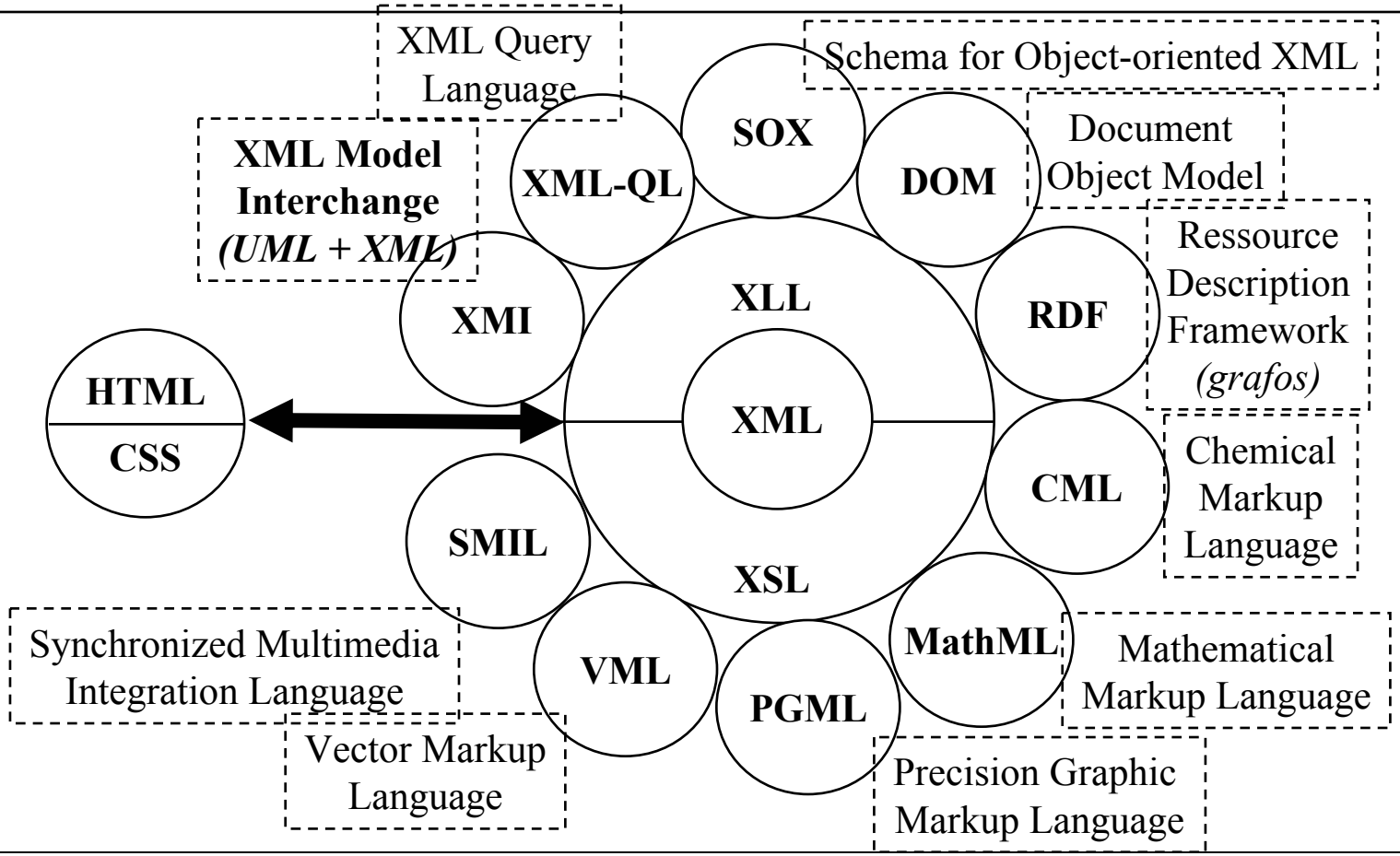
Postgrado en Computación

Prof. Isabel M. Besembel Carrera

Unidad II. Sesión 8. DOM, SAX, XSL, XPath.

UNIVERSIDAD DE LOS ANDES

La nebulosa XML



Tomado de J-M Pinon

UNIVERSIDAD DE LOS ANDES

no es un lenguaje de programación

¿Cómo tratar documentos XML?

- Dos facetas:
  - ✓ Acoplamiento con la ayuda de un API estándar
    - DOM (Document Object Model) es un API objeto (API DOM), se utiliza un analizador para transformar el documento XML en un árbol de objetos
    - SAX (Sample Api for Xml) es un API a eventos, que permite ver el documento XML como un flujo de eventos que invocan los procedimientos de tratamiento
  - ✓ Transformación del documento XML vía XSL (eXtensible Styling Language) que es un verdadero lenguaje de programación según el paradigma de las reglas de producción.
    - Se escriben reglas para probar condiciones en el documento XML y que al cumplirse producen los datos resultantes
    - XSL puede transformar documentos XML en HTML o WML
- Desarrollo de aplicaciones con XML gracias a DOM, SAX y XSL

UNIVERSIDAD DE LOS ANDES

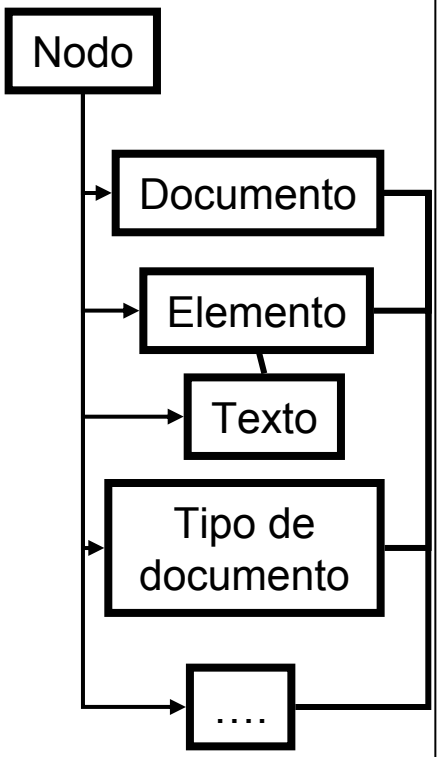
DOM …

- Document Object Model http://www.w3.org/DOM
  - ✓ DOM 1 estándar W3C para HTML y XML, 1998. DOM 2, 2000.
- API (Application Program Interface) permite el dialogo entre una aplicación y los servicios de menor nivel del sistema (red)
- DOM: Interfaz que permite a los programas y guiones acceder y modificar dinámicamente el contenido y la estructura de documentos XML, basado en IDL (Corba-OMG)
- Se utiliza un API-DOM para cada lenguaje (Java, JavaScript, C++, VisualBasic, etc.) Ejm: DOM-JavaScript para HTML
- Define la estructura lógica genérica del documento y la manera de acceder y tratar las instancias
- Descripción y manipulación de documentos HTML/XML

UNIVERSIDAD DE LOS ANDES

Tipos de nodos DOM 1

Objetos	Descripción
Element	Nodo elemento (Interfaz básica)
Attribute (attr)	Nodo atributo (Interfaz básica)
Text	Nodo texto, hijo de elemento (Interfaz básica)
CDATASection	Nodo CDATA (Interfaz extendida)
EntityReference	Referencia a entidad (Interfaz extendida)
Entity	Entidad (Interfaz básica)
ProcessingInstruction	Instrucción de procesamiento (Interfaz extendida)
Comment	Comentario (Interfaz básica)
Document	Nodo raíz del documento XML (Interfaz básica)
DocumentType	DTD o esquema del documento XML (Interfaz extendida)
DocumentFragment	Guarda secciones del documento XML (Interfaz básica)
Notation	Anotación del documento XML (Interfaz extendida)
CharacterData	Métodos y propiedades de los nodos Text, Comment y CDATASection.básica
DOMImplementation (b)	Acceso a los métodos y propiedades de la aplicación independiente del DOM



DOM 1

Interfaz de Node

- getNodeName() nombre del nodo actual
- getNodeType() tipo del nodo actual
- getNodeValue() valor del nodo actual
- getOwnerDocument() nodo raíz
- hasChildNodes() cierto si el nodo tiene hijos
- setNodeValue() cambia el valor del nodo
- cloneNode() copia el nodo
- insertBefore(nodoNuevo, nodoRef) inserta un nodo nuevo hijo antes del nodo referencia
- replaceChild(nodoNuevo, nodoViejo) reemplaza el nodo viejo por el nuevo
- removeChild(nodo) elimina el nodo
- appendChild(nodoNuevo) anexa el nodo nuevo al final de la lista de nodos
- getAttributes() atributos del nodo
- getChildNodes() hijos del nodo
- getFirstChild() primer nodo hijo
- getLastChild() último nodo hijo
- getParentNode() padre del nodo

UNIVERSIDAD DE LOS ANDES

DOM 1

- Interfaz de Document subclase de Node
  - ✓ createElement(nombreElemento) crea un nuevo elemento
  - ✓ createComment(comentario) crea una línea de comentario en el documento
  - ✓ createAttribute(nombreAt) crea un atributo nuevo
  - ✓ getElementsByTagName(marca) descendientes de los elementos correspondientes a la marca dada

Documento XML
<nom ciudad="Mérida">
<nombre>Juan>/nombre>
<apellido>Perez</apellido>
</nom>

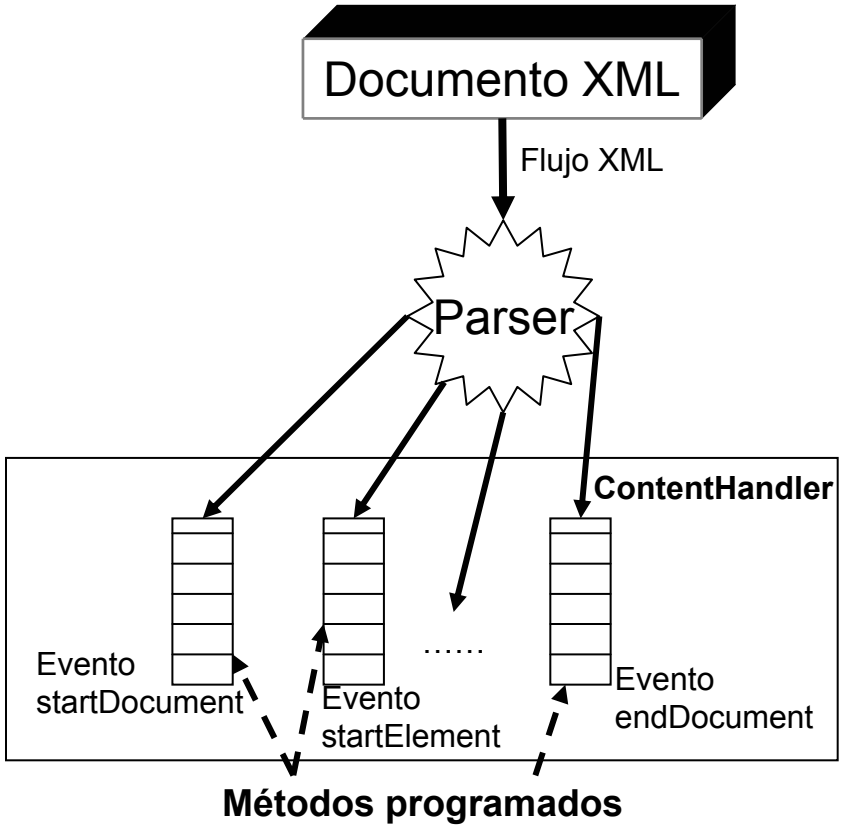
DOM

```
Public class EjemploDOM
Public static main(String argc[]) throws IOException, DOMException
{  XMLDocument xmlDoc=new XmlDocument();
                ElementNode nom = (ElementNode)
                xmlDoc.createElement("nom");
                ElementNode nombre = (ElementNode)
                xmlDoc.createElement("nombre");
                ElementNode apellido = (ElementNode)
                xmlDoc.createElement("apellido");
                xmlDoc.appendChild(nom);
                nom.appendChild(nombre);
                nombre.appendChild(xmlDoc.createTextNode("Juan");
                nom.appendChild(apellido);
                apellido.appendChild(xmlDoc.createTextNode("Perez");
                nom.setAttribute("ciudad", "Mérida");
                System.exit(0);   }   }
```

UNIVERSIDAD DE LOS ANDES

SAX …

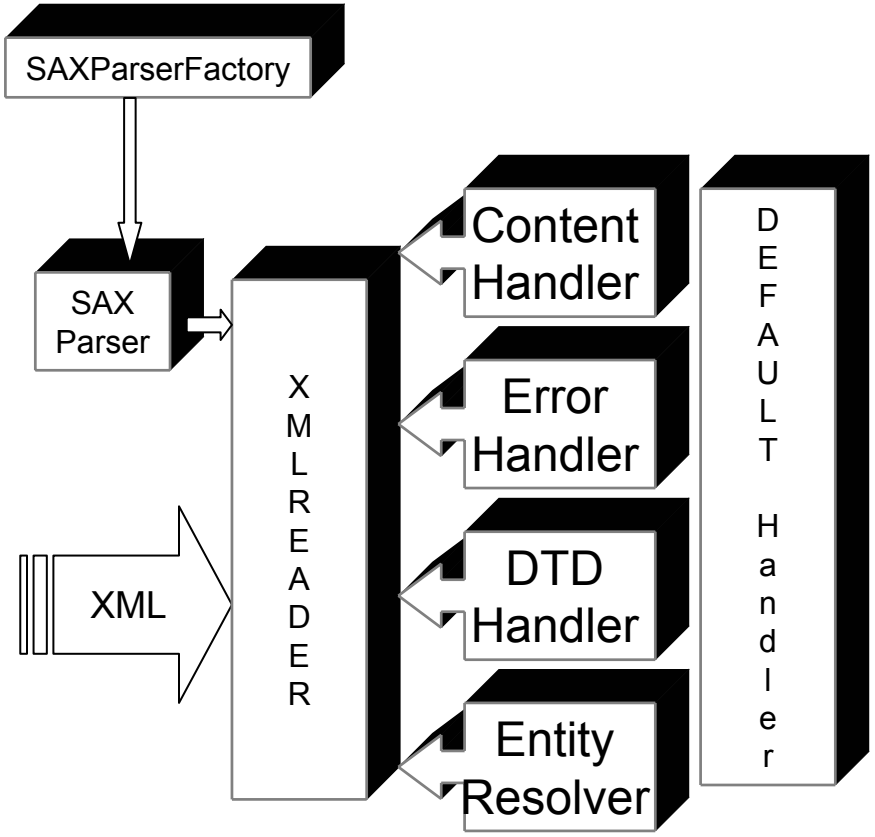
- Basado en el método de invocación (callback method)
- A medida que se recorre el documento se invoca el método apropiado
  - ✓ Eventos: inicio o fin de documento, inicio o fin de elemento, atributo, etc.
- Aplicación SAX: consiste en escribir la implementación de los métodos



UNIVERSIDAD DE LOS ANDES

API ligero y rápido, que no construye la imagen del documento en memoria permitiendo tratamiento al vuelo

SAX



- Crear instancia del parser con SAXParserFactory
- Manejador por omisión: DefaultHandler
- ContentHandler tratamiento normal
- ErrorHandler tratamiento de errores
- DTDHandler para las DTD
- EntityResolver resolver referencias externas

UNIVERSIDAD DE LOS ANDES

JAXP

- JAX API for XML Processing. En Java y J2EE
- Basada en SAX y DOM, última versión soporta XSLT
- Permite acceder a un parser SAX o DOM
- Definido en javax.xml.parsers que contiene dos fábricas: para SAX (SAXParserFactory) y para DOM (DocumentBuilderFactory)

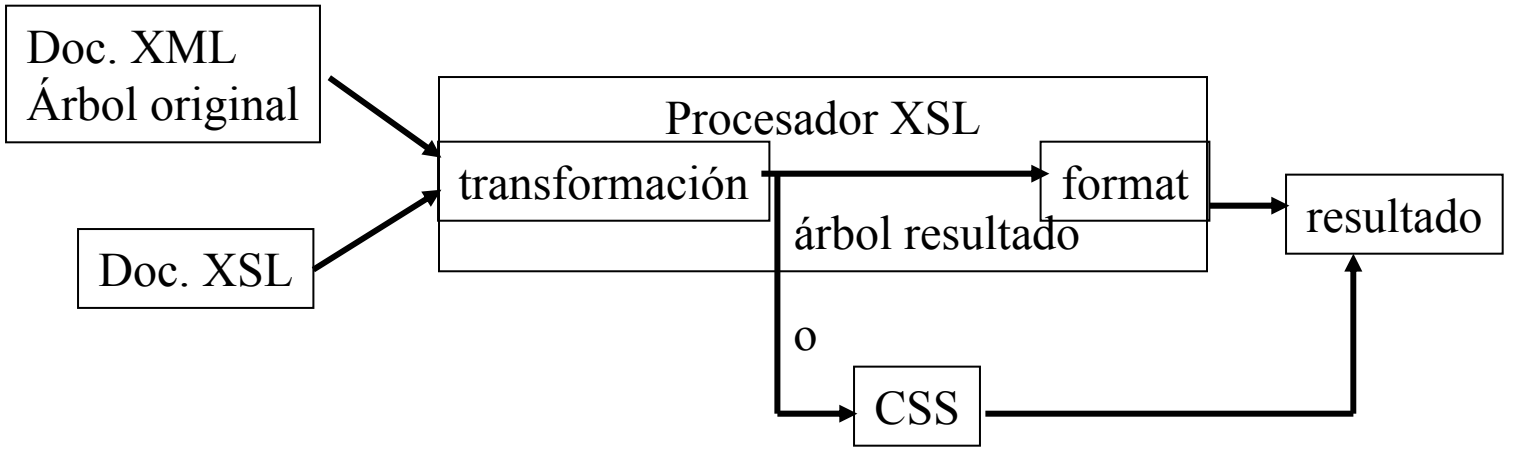
```
SAXParserFactory fabrica=SAXParserFactory.newInstance();
SAXParser saxParser = fabrica.newSAXParser();
saxParser.parse("ejemplo.xml", handler);
```

- El programador debe sobreescribir los métodos correspondientes a los eventos a los eventos que desea tratar

UNIVERSIDAD DE LOS ANDES

XSL …

- eXtensible Style Language http://www.w3.org/TR/xsl
  - ✓ Combinación de DSSSL y CSS, lenguaje de gestión de datos con formato en un documento XML
- XSLT (XSL Transformation): http://www.w3.org/TR/xslt
  - ✓ Transforma el árbol de origen del documento XML en el árbol resultado



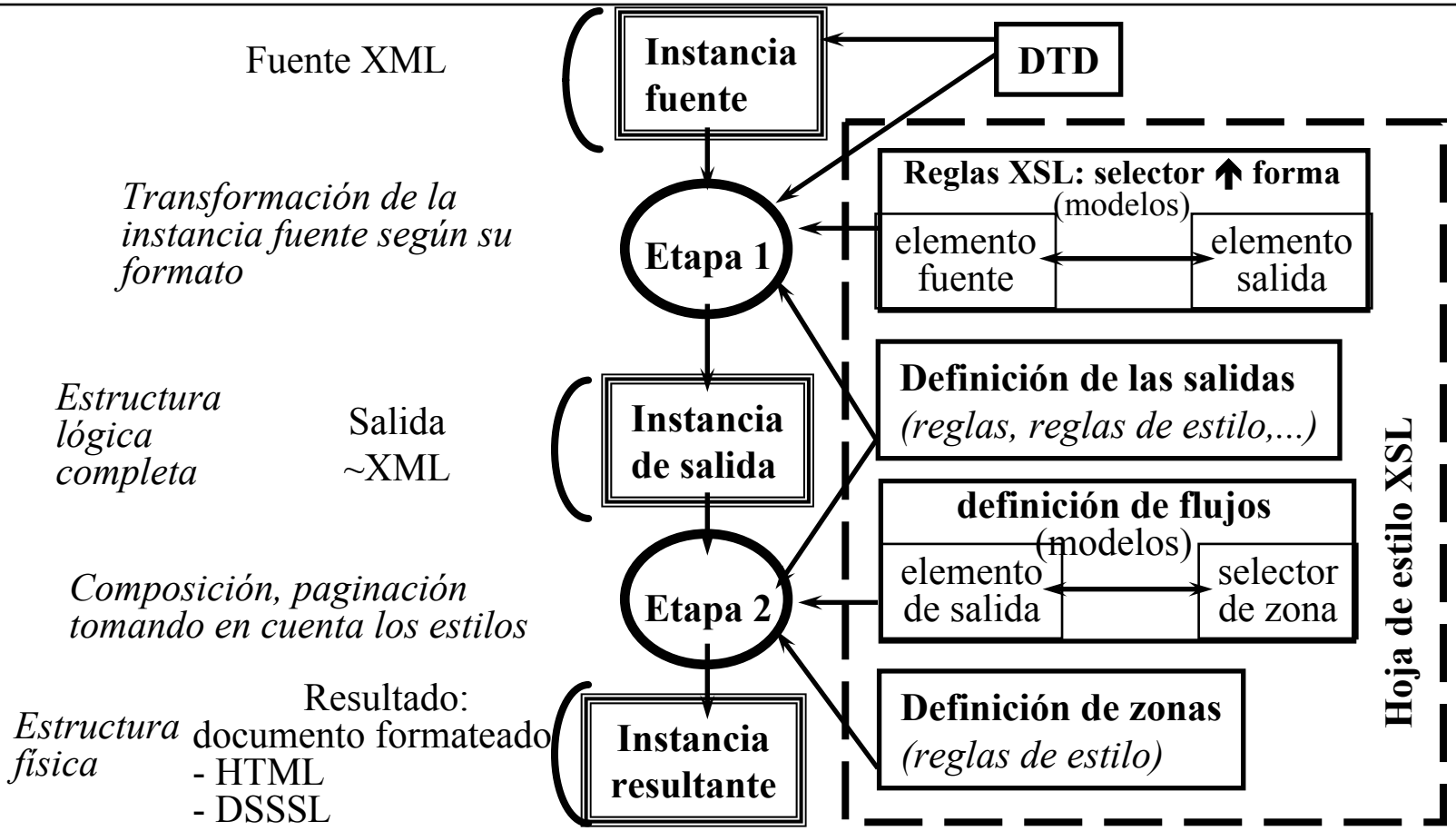
UNIVERSIDAD DE LOS ANDES

XSL …

- XSL-FO (XSL Formatting Objets)
  - ✓ Describe el formato de presentación de los elementos y objetos del documento XML
- Etapas:
  - ✓ Construcción de un nuevo árbol XML según el documento fuente. Regla: selector -> forma
    - Selector: detecta las formas dentro del documento fuente y activa la regla
    - Forma: estructura XML o HTML generada al activarse la regla
  - ✓ Análisis del árbol generado para interpretar los elementos y atributos que definen las propiedades del formato físico

UNIVERSIDAD DE LOS ANDES

Etapas XSL …



Tomado de J-M Pinon

UNIVERSIDAD DE LOS ANDES

Etapas XSL

- **Etapa 1:** transforma la instancia fuente XML en instancia de salida XML
  - Supresión de los elementos no autorizados
  - Inserción de los elementos externos (en vez de las referencias)
  - Duplicaciones necesarias (entidades)
  - Reagrupamiento
  - Informaciones para la etapa 2 (anotaciones)
- **Etapa 2:** transforma la instancia de salida XML en instancia resultado HTML, DSSSL, u otra
  - Definición de las zonas o cuadros
  - Inserción de los contenidos de los elementos en las zonas
  - Tomar en cuenta los estilos de:
    - Formato (dimensiones de las zonas, tamaño, etc.)
    - Presentación (colores, fuentes, negritas, etc.)

UNIVERSIDAD DE LOS ANDES

Nodos

- Documentos XSL estructurados en árbol como los XML
- Nodos: raíz
  - ✓ Elemento = cada elemento
  - ✓ Texto = cada conjunto de datos caracter (CDATA)
  - ✓ Atributo = cada atributo de elemento
  - ✓ Espacio de nombre = cada elemento posee un nodo con su espacio de nombre cuyo prefijo es xsl:
  - ✓ De procesador = cada instrucción de procesamiento
  - ✓ Comentario = cada comentario
- Proceso: recorrido desde la raíz, nodo por nodo, donde en nodo en procesamiento es el nodo actual. Nodo raíz $\neq$ elemento raíz.

UNIVERSIDAD DE LOS ANDES

Estructura XSL

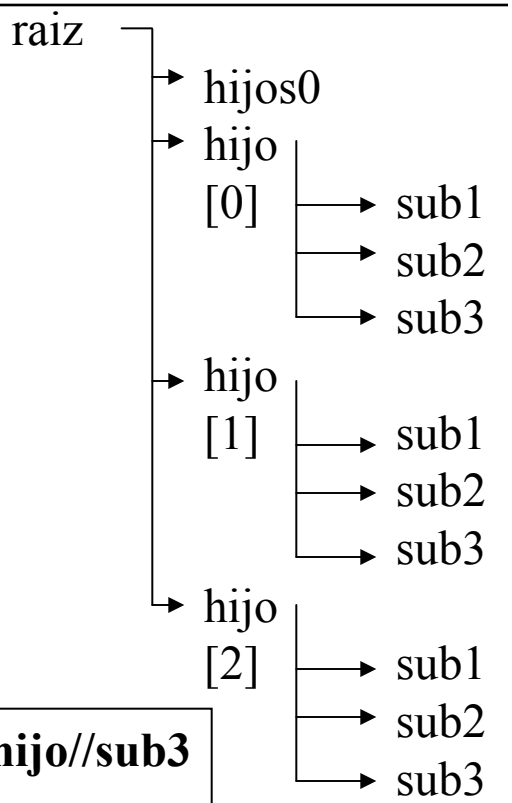
```
<?xml version='1.0' encoding='ISO-8859-1' ?>
<xsl:stylesheet version='1.0' language='Javascript'
  xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
  xmlns:fo=" http://www.w3.org/1999/XSL/Format"
  result-ns=" fo">
  xmlns="http://www.w3.org/TR/REC-html40"     (usando HTML)
  result-ns="">
.....
</xsl:stylesheet>
```

/	nodo raíz o los hijos del nodo actual	@	nodo atributo
//	descendientes del nodo actual	*	todos los nodos
.	Nodo actual	[ ]	operador de agrupamiento
..	Nodo padre		
|	alternativas		

UNIVERSIDAD DE LOS ANDES

Ejemplo 1

```
<?xml version='1.0' encoding='ISO-8859-1' ?>
<?xml-stylesheet type="text/xsl" href=" ex1.xsl " ?>
<raiz>
<hijos0>Pedro</hijos0>
<hijo id="1">
        <sub1>subtítulo 1</sub1>
        <sub2> 1</sub2>
        <sub3>otro 1</sub3>
</hijo>
<hijo id="2">
        <sub1>subtítulo 2</sub1>
        <sub2> 2</sub2>
        <sub3 ord='first'>otro 2 </sub3>
</hijo>
<hijo id="3">
        <sub1>subtítulo 3</sub1>
        <sub2> 3</sub2>
        <sub3>otro 3 </sub3>
</hijo>
</raiz>
```



hijo/*	***/hijo/sub2 | ../hijo//sub3**
hijo//*	**/raiz/hijo[1]**
***/@att**	**/raiz/hijo[end( )]**

UNIVERSIDAD DE LOS ANDES

Template XSL

- Contenedor de instrucciones para gestionar cada nodo del árbol original, describe su transformación y marca el nodo resultado

```
<xsl:template match="/">
    <p> <xsl:value-of  /> </p>
</xsl:template>
```

- Muestra el contenido del nodo **xsl:value-of**
- Filtros: para mejorar la salida **[**operador patrón**]**
- Operador: opcional, define como aplicar el patrón

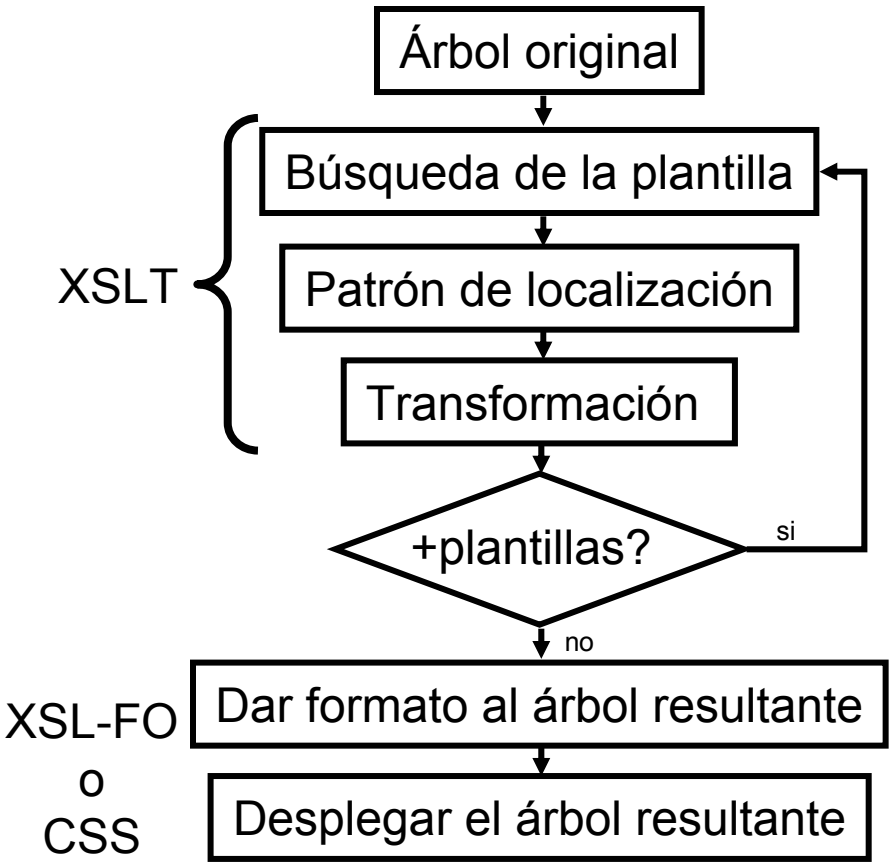
Insensible a mayúsculas o minúsculas

Operadores de comparación

=	**eq**		**ieq**	**=**
!=	**ne**	≠	**ine**	≠
<	**lt**		**ilt**	<
>	**gt**		**igt**	>
<=	**le**	≤	**ile**	≤
>=	**ge**	≥	**ige**	≥

UNIVERSIDAD DE LOS ANDES

Filtros XSL



Conflictos: se aplica la plantilla más precisa
Ejemplo: raiz/hijo/sub3 +precisa que //sub3

Combinación de filtros:

&&	**and**	∨
||	**or**	∧
!	**not**	¬

Ejemplo:
//*[version!='2' and not siglas='pp']

Filtros de conjunto:

any	cualquiera de los elementos
all	todos los elementos

Ejemplo:
<xsl:template match='fecha[any@fe]'>

UNIVERSIDAD DE LOS ANDES

Copia y lazos XSL

- Copia: de un nodo del árbol origen en el árbol resultante

  <xsl:copy> … **</xsl:copy>**

- Inserción del nombre del nodo actual: **<xsl:node-name />**
- Lazo: para cada elemento

  <xsl:for-each select=‘patron**’>** ….. **</xsl:for-each>**

Ejemplos:

```
<xsl:copy>
   <xsl:apply-templates />
</xsl:copy>
….
<xsl:for-each select=‘//hijo’>
   <xsl:value-of /> <br/>
</xsl:for-each>
```

UNIVERSIDAD DE LOS ANDES

Ordenar y decidir XSL

- Ordenamiento: usando el atributo **order-by** en los elementos **xsl:apply-templates** o en **xsl:for-each**, donde **+** ascendente y **–** descendente. Valores numéricos deben compararse con el mismo número de digitos.
- Decisión:
  - ✓ Simple: **<xsl:if match=‘**patrón**’>** bloque-si **</xsl:if>**
  - ✓ Múltiple:

<xsl:choose>
<xsl:when match=‘patrón1**’>** bloque del patrón 1 **</xsl:when>**
<xsl:when match=‘patrón2**’>** bloque del patrón 2 **</xsl:when>**
….
<xsl:otherwise> bloque de lo contrario **</xsl:otherwise>**
</xsl:choose>

UNIVERSIDAD DE LOS ANDES

Ejemplos

- Ordenamiento:

```
<xsl:for-each select='//libro' order-by='+autor'>
    <b> <xsl:value-of select='autor' /> </b> <br/>
</xsl:for-each>
```

- Decisiones:

```
<xsl:if match='//hijo[sub2 < '2']'> Es el mejor !!!
    <xsl:value-of select='sub1'  />
</xsl:if>
<xsl:choose>
    <xsl:when match='.[editorial='Prentice-Hall']'>
        <xsl:value-of select='titulo' /> <br />
    </xsl:when>
    <xsl:when match='.[editorial='McGraw-Hill']'>
        <xsl:value-of select='autor' /> <br />
    </xsl:when>
    <xsl:otherwise>    </xsl:otherwise>
</xsl:choose>
```

UNIVERSIDAD DE LOS ANDES

Ejemplo …

```
<?xml version="1.0" encoding= "ISO-8859-1"?>
<?xml-stylesheet type= "text/xsl" href= "ejemplo1.xsl"?>
<Agenda>
  <Contenido> Lista de direcciones Web </Contenido>
  <direccion>
      <Nom>HTML</Nom>
      <Pag>http://www.w3.org/TR/REC-html4</Pag>
      <Desc>Specification he HTML 4</Desc>
  </direccion >
  <direccion >
      <Nom>Escuela de Sistemas-ULA</Nom>
      <Pag>http://www.ing.ula.ve</Pag>
      <Figura>fac.jpg</Figura>
      <Desc>Mi escuela</Desc>
      <Correo>eisula@ula.ve</Correo>
  </direccion >
</Agenda>
```

Documento XML

UNIVERSIDAD DE LOS ANDES

Ejemplo

Documento XSL

```
<?xml version="1.0" encoding= "ISO-8859-1"?>
<xsl:stylesheet version0 "1.0"
  xmlns:xsl= "http://www.w3.org/TR/WD-xsl"
  xmlns= "http://www.w3.org/TR/REC-html40"
  result-ns= "">
<xsl:template match= "/">
<xsl:element name= "HTML" />
<xsl:element name= "HEAD" />
<xsl:element name= "TITLE" >
  <xsl:value-of select="//Contenido" />
</xsl:element>
<xsl:element name= "/HEAD" />
<xsl:element name= "BODY" />
<LINK rel="stylesheet" type="text/css" href="agenda.css" />
<xsl:comment> <xsl:value-of select="//Contenido" />
</xsl:comment>
<H1> <xsl:value-of select="//Contenido" /> </H1>
<xsl:for-each select="//direccion">
  <P> La página web de
  <B><xsl:value-of select="Nom" /> </B>
  es la siguiente: <xsl:apply-templates select="Pag" />
  <xsl:value-of select="Desc" />
  <xsl:if match="//Correo">
   <xsl:apply-templates select="Correo" />
</xsl:if>
</xsl:for-each>
<xsl:element name= "/BODY" />
<xsl:element name= "/HTML" />
</xsl:template>

<xsl:template match= "Pag">
<A> <xsl:attribute name="href">
  <xsl:value-of /> </xsl:attribute>
 <xsl:value-of /> </A>
</xsl:template>

<xsl:template match="Correo">
  <A> <xsl:attribute name="href">
  <xsl:eval>mailto: </xsl:eval>
  <xsl:value-of /> </xsl:attribute>
  <xsl:value-of /> </A>
</xsl:template>
</xsl:stylesheet>
```

Tomado de J-M Pinon

UNIVERSIDAD DE LOS ANDES

Creación de nodos en la salida …

- De instrucción de procesamiento:

  <xsl:pi name=‘instrucciónDeProcesamiento**’>** valorDeLaInstDeProc
  </xsl:pi>

- Comentario:

  <xsl:comment> comentarioDeseado **</xsl:comment>**

- Elemento:

  <xsl:element name=‘nombreDelElemento**’>** valorDelElemento
  </xsl:element>

- Atributo:

  <xsl:attribute name=‘nombreDelAtributo**’>** valorDelAtributo
  </xsl:attribute>

- Sección CDATA de XML:

  <xsl:cdata> valorDeLaSecciónCDATA **</xsl:cdata>**

UNIVERSIDAD DE LOS ANDES

Creación de nodos en la salida

- Referencia a una entidad (alias) para incluir un conjunto de datos en el lugar donde se declara la referencia a la entidad:

  <xsl:entity-ref name='nombreDelAlias'> valor **</xsl:entity-ref>**

- Guiones en XSL: define una sección del documento XSL que contiene código de algún lenguaje de guiones (JavaScript, VScript, JScript)

  <xsl:script language='lenguaje'**>** código de función en ese lenguaje **</xsl:script>**

  - ✓ Si el código contiene caracteres reservados de XML se encierra el código en CDATA: **<![CDATA[** código de función en ese lenguaje **]]>**
  - ✓ Para hacer que se evalúe el código: las funciones pueden tener parámetros

    <xsl:eval language='lenguaje'**>** llamada a función en ese lenguaje **</xsl:script>**

UNIVERSIDAD DE LOS ANDES

Ejemplo

1. **<xsl:for-each select="//"> Nodo: <xsl:node-name />
 </xsl:for-each>**
2. **<xsl:pi name='xml'>** version='1.0' **</xsl:pi>**
3. **<xsl:element name=**'sub4'**>** 10 **</xsl:element>**
4. **<xsl:template match="**hijo**">**
   <xsl:copy>
   <xsl:attribute name='padre'**>** raiz **</xsl:attribute>**
   </xsl:copy>
   </xsl:template>
5. **<xsl:entity-ref name=**'copyright'**>** © **</xsl:entity-ref>**
6. **<xsl:script>**

```
function pulgadasAcm(pulg)
        pulgadasAcm=pulg*2.54
end function
```

   </xsl:script>
7. **<xsl:eval>** pulgadasAcm(19) **</xsl:eval>**

UNIVERSIDAD DE LOS ANDES

XSL-FO

- Especificación para una parte de la información: paginación, formato y estilo que serán aplicados al objeto en el árbol resultante.
- Se utilizan la mayoría de las propiedades de CSS.
- Prefijo **fo:**propiedad.
- Propiedades agrupadas por su comportamiento:
  - Posicionamiento: fo:title, **fo:root**, **fo:block-container**
  - Borde, relleno, fondo, tipo y estilo de letra:
  - Corte de palabras (hyphenation): **fo:block**, **fo:character**
  - Margen, alineación, color: **fo:color-profile**
  - Tablas, viñetas, enumerados: **fo:table**, **fo:table-column**, **fo:table-caption**

<fo:inline> guru **</fo:inline**>

<fo:inline> Ap

<fo:inline baseline-shift='super**'>** 1*ji* **</fo:inline>**

</fo:inline>

fo:table-cell, **fo:region-body**, **fo:region-before**, **fo:region-after**, **fo:region-start**, **fo:region-end**, **fo:block-container**, **fo:inline-container**, **fo:external-graphic**, **fo:instream-foreign-object**

UNIVERSIDAD DE LOS ANDES

XPath …

- Objetivo: definir un lenguaje que señale partes del documento XML
- Partes:
  - ✓ Sintaxis de las expresiones de rutas
  - ✓ Conjunto de funciones o biblioteca central de XPath
- Modelo de información: árbol con nodos de comportamiento
  - Raíz
  - Elemento
  - Texto
  - Atributo
  - Espacio de nombres
  - Instrucción de procesamiento
  - comentario

UNIVERSIDAD DE LOS ANDES

XPath …

- Rutas: referencia un nodo o un grupo de nodos. Se construyen en forma similar a las rutas de ubicación de directorios con “/”
- Tipos:
  - ✓ Absoluta: siempre se inicia en el nodo raíz
  - ✓ Relativa: secuencia de 1 o más pasos de ubicación separados con “/”, cada paso es un nodo relativo a partir del nodo contexto
- Paso: contiene 3 partes
  - ✓ Eje: relación entre los nodos seleccionados y el contexto
  - ✓ Nodo prueba: nombre de los nodos seleccionados
  - ✓ Conjunto opcional: predicado que refina el conjunto de nodos seleccionados

UNIVERSIDAD DE LOS ANDES

XPath …

- Ejes:
  - ✓ Ancestro (ancestor)
  - ✓ Ancestro o el mismo (ancestor-or-self)
  - ✓ Atributo (attribute ⇔ @)
  - ✓ Hijo (child)
  - ✓ Descendiente (descendant)
  - ✓ Descendiente o el mismo (descendant-or-self ⇔ //)
  - ✓ Siguiente (following)
  - ✓ Siguiente hermano (following-sibling)
  - ✓ Espacio de nombres (namespace)
  - ✓ Principal (parent ⇔ .)
  - ✓ Precedente (preceding)
  - ✓ Hermano precedente (preceding-sibling)
  - ✓ El mismo (self ⇔ .)

Ejemplos:
✓child:: autor selecciona todas las marcas autor de la marca contexto equivalente a //autor
✓ child::autor/text() nodos texto del elemento autor ⇔ //autor/text()
✓ child::entrada/child::fecha selec-ciona todos los elementos fecha de los elementos entrada ⇔ //entrada/fecha
✓ child::entrada[position()=2]/child::fecha selecciona el elemento fecha de la segunda entrada
✓ child::fecha/attribute::*[position() >1 and position() !=last()] selecciona todos los atributos excepto el primero y el último del elemento segundario fecha, sin importar su nombre ⇔ fecha/@*[position() >1 and position() != last()]

UNIVERSIDAD DE LOS ANDES

XPath …

- Funciones:
  - ✓ position():numero
  - ✓ count(conj-nodos):numero
  - ✓ last():numero
  - ✓ id(objeto):conj-nodos
  - ✓ local-name(conj-nodos):cadena
  - ✓ namespace-uri(conj-nodos):cadena
  - ✓ name(conj-nodos):cadena
  - ✓ concat(cad, cad1*):cadena
  - ✓ starts-with(cad, cad1):booleano
  - ✓ contains(cad, cad1):booleano
  - ✓ substring-before(cad, cad1):cadena
  - ✓ substring-after(cad, cad1):cadena
  - ✓ substring(cad, ini, num?):cadena la posición se cuenta a partir de 1
  - ✓ string-length(cadena?):numero
  - ✓ normalize-space(cadena?):cadena normaliza a 1 espacio entre palabras y elimina espacios al inicio y al final
  - ✓ translate(cad, cad1, cad2):cadena reemplaza en cad los caracteres que aparecen en cad1 en la misma posición de los de cad2
  - ✓ boolean(objeto):booleano convierte el objeto en booleano, número es verdadero si es diferente de 0, un conjunto de nodos o una cadena es verdadero si no está vacío, boolean(/) regresa verdadero si es un DBF

UNIVERSIDAD DE LOS ANDES

XPath

- ✓ false():booleano regresa falso
- ✓ true():booleano regresa verdadero
- ✓ not(booleano):booleano
- ✓ lang(cadena):booleano
  regresa verdadero si el lenguaje del nodo contexto es el mismo o es uno secundario del especificado en el argumento
- ✓ number(objeto?):numero convierte el objeto en número. Ver IEEE 754.
- ✓ sum(conj-nodos):numero
- ✓ floor(numero):numero
- ✓ ceiling(numero):numero
- ✓ round(numero):numero

Ejemplos:

- ✓ concat(“ca”,“sa”,”5”) “casa5”
- ✓ starts-with(“valor”,”alo”) falso
- ✓ contains(“valor”,”alo”) true
- ✓ substring-before(“La casa”,” “) “La”
- ✓ substring(“casa”,2,2) ”as”
- ✓ normalize-space(“ M o l o k o “)
  “M o l o k o”
- ✓ translate(“arbol”, “xyzr”, “mngh”)
  “ahbol”
- ✓ boolean(-2) true
- ✓ boolean(“”) false
- ✓ number(“ -23”) -23
- ✓ floor(6.5) 6
- ✓ ceiling(6.05) 7
- ✓ round(6.7) 7