Benemérita Universidad Autónoma de Puebla

Facultad de Ciencias de la Computación

Materia: Proyectos de Investigación y Desarrollo

**Proyecciones XML**

**Patrón de Transformación**

Por

Ricardo Bermúdez Bermúdez

**Proyecciones de datos XML**

**Resumen**

Nuestra época queda marcada por la constante necesidad de integración de sistemas de toda índole dentro del marco del desarrollo socioeconómico, que en la fase ulterior de su desarrollo esta necesidad se expresa en la integración de diversos sistemas informáticos que en conjunto forman una gran red de sistemas de cómputo distribuido donde cada aplicación, cada sistema informático, cada componente que forma está red de sistemas utiliza distintas estructuras de datos, programados en distintos lenguajes y se ejecutan en las más diversas plataformas de hardware y software.

El lenguaje XML es hoy día uno de los estándares más utilizados como norma para estructurar y codificar datos que se envían entre distintos sistemas informáticos, es el común denominador de la representación de datos en la Capa de Aplicación respecto al Modelo OSI. En este contexto, el lenguaje de etiquetas XML cobra gran relevancia en la necesidad de la construcción de sistemas de arquitectura distribuida que se orienta específicamente hacia los servicios web; es y será por mucho tiempo la base de intercambio de datos entre distintos sistemas informáticos, de importancia equivalente a la norma de diseño de Bases Datos Entidad Relación surgida en la necesidad que se dio de representar sistemáticamente los registros de un sistema [3]. Sin embargo, el uso de las estructuras de datos muy difícilmente se normalizara, pues esta se adecua siempre a las particularidades del campo de aplicación de determinado sistema, y cuando este determinado sistema se integra con otros entonces surge otro problema: *transformar una estructura de datos de un formato a otro*. El Consorcio del World Wide Web (W3C por sus siglas en inglés) vio en esta problemática la necesidad de crear y estandarizar un lenguaje que describa como es la transformación de una estructura de datos en otra y surge entonces XSLT (Extensible Stylesheet Language Transformations) y XPATH.

Como se ha visto, el tema de integración de sistemas es muy amplio, variados modelos de integración han surgido como arquitecturas de integración por ejemplo REST, SOAP, RMI, CORBA, etc, así como lenguajes de codificación y estructuración de datos como HTML, XML, JSON, SOAP Envelope, etc. En este estudio nos enfocaremos al desarrollo de una herramienta facilite la fase de abstracción y transformación de datos contenidos en un XML de un formato a otro; simplificar la expresión de las operaciones necesarias, reducir el número de ellas para crear una estructura de datos análoga y útil a las particularidades de cada operación requerida por un sistema determinado es la meta buscada y clave del desarrollo de este proyecto. Visto desde el contexto de XSLT, se busca realizar un prototipo de lenguaje que disminuya la necesidad de operadores de control y repetición aprovechando que su uso se dará junto a otros lenguajes de programación y con ello a futuro proponer una mejora a esta norma de transformación.

Para poner a prueba las capacidades del marco de trabajo que se pretende desarrollar, se mostrará su implementación en la integración de un Servicio Web con un sistema de servicios turísticos llamado Travelnet, además de apoyarnos en los estándares SOAP y WSDL que nos facilitarán esta tarea.

**Índice**

* Portada…………………………………………………………………………………….. 1
* Resumen…………………………………………………………………………………… 2
* Índice…….…………………………………………………………………………………… 3
* Aplicaciones similares………………………………………………………………… 4
* Investigación……………………………………………………………………………… 10

**Aplicaciones Similares**

El resultado de este proyecto será un marco de trabajo sobre las transformaciones de datos basado en XPATH y en este contexto analizaremos 2 normas de la W3C por separado que se orientan hacia el procesamiento (consulta y transformación) de un documento XML: XQUERY y XSLT. La definición completa de estos lenguajes es muy extensa y su especificación está formada por varios documentos basados en normas y estándares de la W3C, IEEE, ISO, IETF, ACM, ANSI, entre otras instituciones encargadas de normar el desarrollo de sistemas y comunicaciones a través de la World Wide Web. De forma general es esta la ventaja principal y signo de robustez y seguridad de estos marcos de referencia; son estándares, en su condición de lenguajes formales, las estructuras lexicográficas, sintácticas y semánticas que los conforman corresponden al trabajo de décadas de ciencia e ingeniería aplicada al desarrollo de normas de trabajo y estándares internacionales. Por todo ello, el correspondiente trabajo al respecto y nuevas propuestas de normas de trabajo o modelos de procesamiento de un XML debe tener fundamento en estas recomendaciones. En este contexto esperamos el desarrollo de este proyecto sea un indicador de mejoras a estos marcos de trabajo.

Debido a lo anterior en esta sección sólo se comentarán las características más destacadas con el fin de cubrir los siguientes rubros:

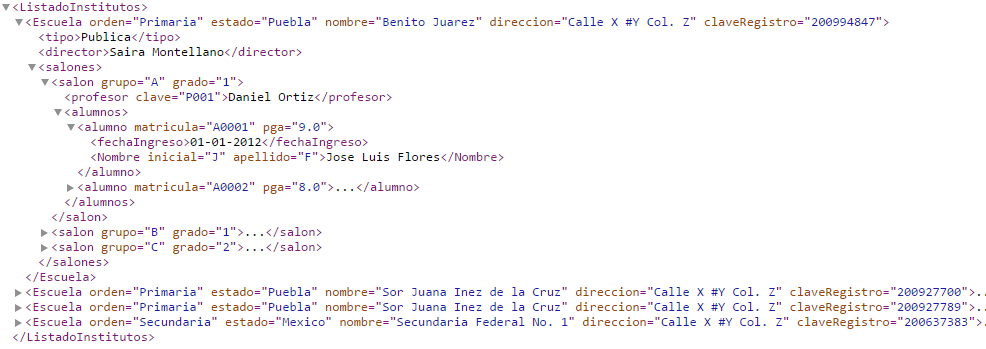
1. Funciones del sistema similar analizado.
2. Ventajas del sistema similar analizado.
3. Desventajas del sistema similar analizado.
4. Requisitos no funcionales del sistema similar analizado.
   1. Usabilidad
   2. Robustez
   3. Seguridad

Se eligió el software XMLSpy de Altova, para evaluar expresiones XQUERY, mientras que el servicio en línea de XPATH TESTER disponible en [J] y la implementación hecha en PHP del evaluador de expresiones XSLT [K] ayudaron en la ocasión haciendo las veces de evaluadores de expresiones XPATH y XSLT respectivamente. Se realizaron algunos experimentos en estas herramientas, hacemos hincapié en que funcionaron como evaluadores mientras que el enfoque de interés a la investigación son las estructuras sintácticas y semánticas de los lenguajes antes mencionados y no las implementaciones realizadas. Por el espacio reservado a este documento no nos es posible mostrarlos por completo. Sin embargo, los experimentos realizados nos han servido para formular los comentarios siguientes, en caso de ser requeridos se podrán solicitar al correo de los desarrolladores participantes de este proyecto.

**XQUERY**

1. Se define como un lenguaje de consulta de datos a un sistema de datos XML, versátil e inteligible. Construye un marco de trabajo: FLWOR (acrónimo de los términos anglosajones "For, Let, Where, Order by, Return") que sirve para expresar consultas a un XML, mostramos el siguiente ejemplo acerca de su funcionalidad.

Figura 1



\*\*

Ejemplo 1: Supongamos que en una estructura XML llamada <ListadoInstitutos> se guardan los registros de todas las escuelas de México, la Figura (1) muestra la estructura de los registros XML con los datos de las escuelas y someramente el contenido de la misma, de la cual solo necesitamos consultar los registros de las escuelas públicas que pertenecen al estado de Puebla con los siguientes datos: nombre de la escuela, grado y su clave de registro, nombre, promedio y matricula de sus alumnos inscritos (los datos necesitados se señalan con rojo en la Figura (1)) y los registros resultantes de la consulta deben representarse con la estructura del esquema XSD de la Figura (2):



Figura 2

Un breve análisis de la estructura descrita en la Figura (2) aclarará la estructura de los registros resultantes: el nodo <consultaPadronNacionalSEP> será el nodo raíz de la estructura XML, mientras que por cada escuela que coincida con la consulta deberá agregarse un nodo hijo del tipo <instituto> a la raíz, que a su vez deberá contener los como nodos hijo a las etiquetas <nombre>, <orden>, <claveRegistro> con sus correspondientes datos y una etiqueta <alumno> por cada alumno registrado en la escuela con sus datos solicitados. Así la expresión XQUERY que se aproxima a lo solicitado es la siguiente:

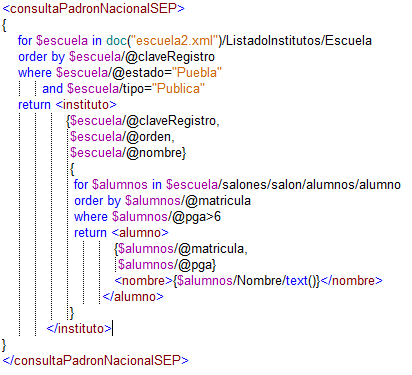
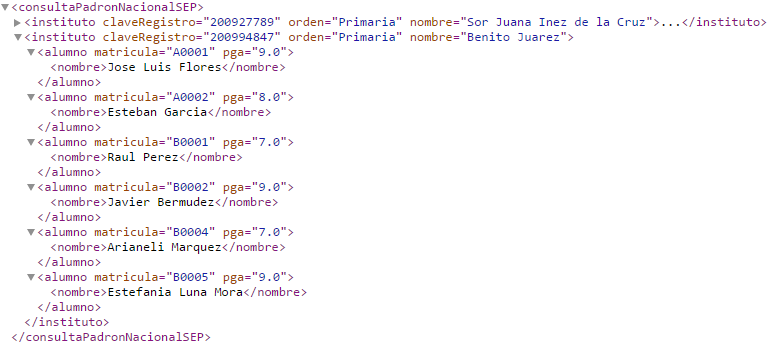


Figura 3

Y el resultado de evaluarla es el que se muestra a continuación:

Figura 4



Como hemos visto en el ejemplo, junto a la estructura FLOWR, se agregan una serie de operadores aritméticos, booleanos, funciones para el manejo de cadenas de texto y de búsqueda de valores específicos [H]. Cimienta su funcionamiento en el direccionamiento XPATH con capacidades para realizar “sub consultas” a través de la definición recursiva de expresiones FLOWR y la declaración de funciones con subrutinas de procesamiento. A diferencia de la norma XSLT las expresiones XQUERY no utilizan etiquetas con prefijos (nombres de espacio XML) para definir las palabras reservadas. Un tutorial sobre el uso de este lenguaje se encuentra en [B] y los documentos que tratan detalladamente la norma en [7].

1. Una de las ventajas de XQUERY es su autonomía, su definición es independiente de cualquier implementación que se haga en cualquier lenguaje de programación y es una recomendación de la W3C, utiliza un lenguaje “intuitivo” para quienes ya han manejado una sintaxis SQL y es capaz de evaluar las expresiones regulares XPATH. A pesar de que dentro de las expresiones XPATH se pueden utilizar operadores booleanos, de comparación y filtrar así resultados, la cláusula WHERE de XQUERY permite separar de la expresión de direccionamiento los filtros realizados.
2. Una de las desventajas del uso de XQUERY es que si bien, es cierto que su sintaxis es bastante similar a la de cualquier lenguaje SQL, en el manejo de la selección de datos de salida: la combinación de instrucciones FLOWR junto a la definición de la estructura de salida dificulta el seguimiento e intelección de la estructura resultante. Entre mayor profundidad tienen la estructura inicial y la estructura final este problema es más notorio. Y salvo algunas excepciones como el software XMLSpy, este lenguaje de consulta será implementado junto a otro lenguaje de programación donde muchos programadores eligen utilizar las sentencias de control y repetición propias del lenguaje para procesar los datos dentro del XML que utilizar este lenguaje de consultas. Al utilizar XQUERY en la implementación hecha en XMLSpy, el equipo de desarrollo notó que no es posible redefinir el atributo contenido en una etiqueta de una estructura inicial como etiqueta de la estructura resultante (véase solución al Ejemplo 1 en la Figura (4)), lo cual es sinónimo de que habrá estructuras finales en las que no es posible proyectar los datos iniciales (el mismo problema se resolvió con XSLT como se verá más adelante y se llegó a una solución satisfactoria –véase la Figura (6)-).
3. Dentro de los requisitos no funcionales del sistema nos encontramos con lo siguiente, su uso requiere de una implementación de las normas DOM, XPATH y XML sobre de la plataforma de trabajo, además de contemplar los siguientes puntos propuestos por la W3C como requerimientos de la norma XQUERY (un listado completo de los requisitos de desarrollo e implementación se encuentra en [9]):
4. Usabilidad:

* Compatibilidad con versiones anteriores
* Extensión de compatibilidad
* XML funcionalidad de consulta
* Definición de funciones externas

1. Robustez

* Creación dinámica de las consolidaciones de espacio de nombres
* Extensiones de tipos estáticos
* La invocación de funcionalidad externa
* Manejo de funciones de orden superior
* Apoyo a los nuevos tipos de datos

1. Seguridad

* Definición de errores
* Procesamiento de errores y recuperación
* Validación
* Módulo de restricciones

**XSLT**

1. Acrónimo del concepto ingles Extensible Stylesheet Language Transformations, es una especificación que define la sintaxis y semántica de operaciones de transformación a un documento XML expresadas a manera de hojas de estilos XSL que siguen la norma de documentos bien formados del estándar XML conforme a los nombres de espacios. Utiliza etiquetas XML para definir las clausulas especiales de sus operaciones, el enfoque principal de las estructuras XSLT es definir la transformación, busquemos resolver el Ejemplo 1 mostrado en la anterior subsección con las herramientas de la norma XSLT.



Figura 5

El resultado de ejecutar la transformación descrita en la Figura (5) sobre de la estructura <ListadoInstitutos> contenida en el documento XML mostrado en la Figura (1), se muestra a continuación:

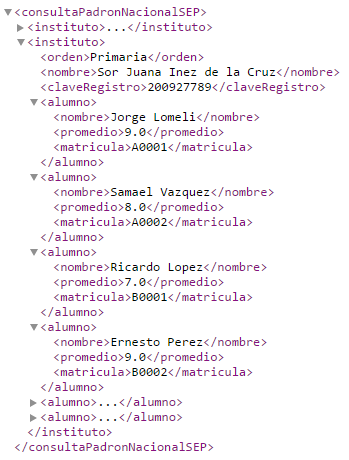


Figura 6

1. A diferencia de XQUERY, XSLT utiliza el enfoque de “Transformación”, inicialmente pensado para mapear los datos de un XML en un documento HTML y mediante un lenguaje basado en la normas XML, XSL y NS [L] define las cláusulas que lo integran. La estructura de repetición <xsl:for-each> es utilizada para recorrer cada nodo del conjunto de datos referenciado por la expresión XPATH, <xsl:if> y <xsl:choose> como una cláusulas de control para evaluar expresiones booleanas y proposiciones aritméticas sobre del contenido de un nodo (sea este del tipo atributo o etiqueta) y en caso de cumplirse ejecutar cierto segmento de código. A diferencia de XQUERY utiliza la definición de las clausulas <xsl:attribute> y una definición explicita de las etiquetas que pertenecerán a la estructura transformada para mapear los datos elegidos de la estructura inicial a una estructura final. Tiene la capacidad de definir subrutinas de procesamiento mediante la cláusula <xsl:template>. Un tutorial sobre el uso de este lenguaje se encuentra en [C] y los documentos que tratan detalladamente la norma en [8].
2. Una de las ventajas principales de utilizar XSLT es elegir detalladamente como serán los mapeos de los datos requeridos y a pesar de no poder definir en este lenguaje rutinas tan complejas como se lograrían con XQUERY, los “templates” permiten modular el código, además XSLT hereda todas las funciones definidas en el estándar XPATH para su uso.
3. Entre sus desventajas nos encontramos con el mismo caso de XQUERY, redefine estructuras de control y repetición que bien pueden ser sustituidas por las sentencias del lenguaje de programación junto al cual se implementa, a medida que complejidad de las estructuras iniciales y finales aumenta es más difícil detallar una transformación con el dialecto XSLT.
4. Al igual que con XQUERY para la recomendación XSLT la W3C sugiere una lista de requisitos que debe cumplir el desarrollo e implementación de la norma a continuación agrupamos los más importantes, aunque una lista completa de ellos se encuentra en [D].
5. Usabilidad:

* Compatibilidad con versiones anteriores
* Extensión de compatibilidad
* Trabajo con valores nulos
* Soporte para métodos XHTML
* Proveer de algoritmos eficientes de transformación de documentos largos
* Soporte para aplicar operación inversa de Transformación

1. Robustez

* Añadir un formato de funciones
* Soporte a la normalización de cadenas del estándar Unicode
* Buscar compatibilidad con la definición de nombres de espacios

1. Seguridad

* Definición de errores
* Procesamiento de errores y recuperación
* Rutinas de manejo de errores de tipos externos
* Validación de resultados

**Metodología de desarrollo**

EL World Wide Web Consortium hace pública la metodología de desarrollo de marcos de referencia. Una serie de pasos que de forma general describiremos a continuación aseguran la calidad de la especificación. Este proceso permite la intervención de todos los usuarios de las tecnologías, con el objetivo de que puedan aportar su conocimiento y opiniones a la mejora de los marcos de referencia.

Tras la conclusión de este proceso, elaborado por especialistas en la materia, se obtienen unos estándares de calidad que buscan estar disponibles en todo el mundo. Así en el desarrollo de un estándar destacamos 4 momentos que definen la etapa de madurez de la nueva norma de trabajo:

* **Borrador de trabajo**. Es un documento publicado para su revisión por parte de la comunidad.
* **Candidato a recomendación**. En este estadio los miembros especialistas de la W3C consideran que el documento ha sido ampliamente revisado y satisface una serie de requisitos preestablecidos.
* **Propuesta de recomendación**. Una vez el documento ha demostrado ser implementable y estar maduro, se envía al Comité Asesor del World Wide Web Consortium buscando su aceptación.
* **Recomendación de la W3C (estándar).** Aquí el documento alcanza el estado de estándar o norma de trabajo, después de que el proceso seguido hasta aquí le haya garantizado un extenso apoyo de todos los actores involucrados, incluidos los miembros de la W3C y su director.

En el proceso de desarrollo pueden llegar a participar más de 400 instituciones entre ellas educativas, privadas, gubernamentales, etc., que se encuentran asociadas a la W3C. Cabe mencionar que pese a esta síntesis sobre la metodología de normalización de procesos de la W3C, que tiene base en la argumentación rigurosa, objetiva y racional, es decir en la base de la metodología de la ciencia, el desarrollo de una recomendación es un proceso costoso en tiempo y dinero.

**Investigación**

Este documento especifica los conceptos de los operadores de manipulación y transformación de árboles representados por estructuras XML. El módulo -- Protocolo perteneciente a la Capa de Servicios Web -- de procesamiento de estructuras XML, Proyecciones XML, es una herramienta implementada junto con el lenguaje de programación de PHP que se enfoca en hacer explícito el *patrón de transformación* de una estructura de árbol en otra. Para el entendimiento y uso de esta herramienta se requiere un dominio del concepto de estructuras de árbol, el lenguaje de consultas sobre de un XML: XPATH y una noción básica de recursividad.

**Introducción**

Acceder y tomar ciertos datos de un XML puede ser una tarea bastante trivial cuando su estructura es sencilla (poca amplitud y profundidad de la estructura de árbol que representa), apenas es necesario utilizar una expresión XPATH sin definir operaciones sobre las etiquetas o atributos y unas cuantas estructuras de control para describir cómo será el cómputo del acceso a esos datos. Sin embargo, a medida que aumentan la amplitud y profundidad del árbol que el XML representa y las relaciones entre sus nodos son más complejas, la descripción del procesamiento del XML se puede tornar en cientos de líneas de código y de una complejidad exponencial se habla al intentar evaluar las operaciones necesarias para crear una estructura de datos a partir de un XML (cuando de dar mantenimiento al proceso se trata), durante la etapa de desarrollo/programación del procesamiento, el programador se debe concentrar más en el uso adecuado de las estructuras de control para el acceso a los datos que en el entendimiento de las relaciones de los árboles con que está trabajando. Existe una necesidad de *describir eficientemente* el procesamiento de un XML; creando procesos de desarrollo de mayor calidad que permitan concentrar el esfuerzo en el análisis e intelección de las determinadas estructuras de datos con que se trabaja, que hagan explícita la relación existente entre los datos de los distintos servicios/operaciones y permitir centralizar el manejo de errores al trabajar con estructuras XML; estos son los objetivos que se buscan resolver con este marco de trabajo.

**¿Cambio de tipos de datos o transformación?**

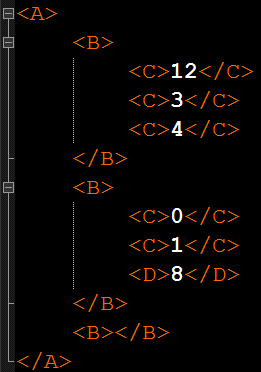
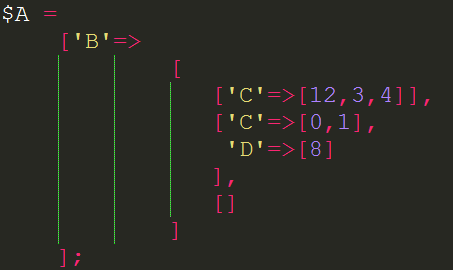
Se ha pensado que el procesar un XML se trata de un somero cambio de un formato/tipo de datos a otro (Véase el Apéndice A). Es cierto, para el procesamiento que se practica en el diseño de sistemas programados en PHP y otros lenguajes en la gran mayoría de los casos es necesario cambiar el tipo de datos, por ejemplo cambiar un XML por una descripción de arreglos anidados, pero quienes solo prestan atención a este punto se olvidan de que también es necesario ***transformar*** la “estructura interna” del XML. Esto se expresa de otra forma, si bien es cierto que los datos utilizados en determinados sistemas pueden ser por ejemplo “arreglos asociativos” (tablas hash capaces de anidar a otras tablas hash) no solo basta con ser un arreglo para poder ser procesado por el sistema, este debe definir *claves estándar* que *contienen los datos* requeridos por la operación del sistema. Esto le da una complejidad mayor al problema del procesamiento de un XML.

A menos de que se diga lo contrario al decir “arreglo” se hace referencia al tipo de datos arreglo como se define en PHP con todas las capacidades que este conlleva. Si durante el manejo de una estructura XML se busca que el *formato de salida* de los datos sea de *tipo* arreglo, en este nivel se habla de un *cambio de* *tipos de datos*. Por otro lado, si se hace una abstracción más general, al comparar una estructura XML y un arreglo es casi intuitivo que se trata de dos *tipos de datos equivalentes*. La equivalencia que se da entre un arreglo y un XML es que ambos son capaces de representar las mismas estructuras de datos. Prestamos atención a la estructuras de uso común con que se trabaja en el marco del XML: Estructuras de árbol con nodos rama: etiqueta y nodos hoja: etiqueta o etiqueta-valor.

A continuación se muestra un ejemplo de lo dicho anteriormente, cada nodo del árbol D1 define una etiqueta y un valor que en caso de tenerlo está entre corchetes:

C 1 Representación con Notación de Corchetes de PHP

C 2 Representación con XML

****

Como se observa, C1 y C2 son dos posibles representaciones de la estructura de árbol D1, es decir, no son las únicas pero si *equivalentes* por representar la misma estructura abstracta. Una demostración formal de que toda representación de una estructura de árbol descrita en XML tiene al menos una representación equivalente en arreglos con notación de corchetes no está en los propósitos de este documento, baste con entender que esta relación es posible.

**Conceptos de Transformación**

**Árboles**

Existen distintas clasificaciones sobre los tipos de estructuras de árbol, como mencione anteriormente las estructuras que representa un XML (así como también otros lenguajes como JSON) son Estructuras de Árbol donde los nodos rama pueden contener solo una “Etiqueta” y solo los nodos hoja pueden contener un par “Etiqueta-Valor” o solamente una “Etiqueta”. Se señalan en el siguiente árbol con raíz R los nodos hoja y nodos rama:

****

**Transformación**

Se echa abstracción del tipo de datos y dicho lo anterior, se define:

La ***transformación*** sobre un árbol es un conjunto de operaciones realizadas para construir una estructura de árbol a partir de otra.

Está muy bien podría ser la definición del Lenguaje de Transformaciones XSLT e independientemente de que el árbol está representado por un XML. Resolvemos así que siempre que se hable de transformación estaremos hablando de una operación aplicada para cambiar la estructura abstracta que tienen los datos y no al cambio del tipo de datos tarea que cuando menos en PHP es bastante trivial si solo fuera ese el problema.

El siguiente ejemplo muestra la relación de transformación entre dos árboles.

Ejemplo 1: Supongamos el árbol A1 representado por el diagrama D2, y su correspondiente representación A1’ donde se agrupan los nodos por nombre de etiqueta que llamaremos ***notación de agrupamiento por etiqueta***, el operador **\*** (Cerradura de Kleene)indica que puede haber 0 o más nodos con etiqueta N:

D 1 Árbol A1

D 1 Árbol A1’

A partir de A1 necesitamos generar un árbol A2 con la siguiente *descripción*:

*Por cada nodo C del tercer nivel de árbol A1 agregar a la raíz con etiqueta R del árbol A2 un nodo hijo H, donde los hijos de cada nodo C con etiqueta F y K serán hijos de su correspondiente representación como nodo H en el árbol A2, los de etiqueta F serán renombrados como P y los de etiqueta K como I.*

DT 1 [Enunciado]

El resultado de modificar el árbol A1 de acuerdo a la anterior descripción es la siguiente:

D 3 Árbol A2

A2

D 2 Árbol A2'

Emparejamos los arboles A1’ y A2’ para resaltar las operaciones realizadas según la descripción de transformación:

D 5 Relación de Transformación entre A1 y A2

**Función de Transformación**

Generalizamos la operación de Transformación sobre de cualquier árbol con cualquier descripción de transformación con la siguiente función:

*AF = transformación (DT, AI)*

Donde:

* *AI* *(Árbol inicial)* estructura a la que se aplicará la transformación según DT
* *DT (Descriptor de la Transformación)* es la estructura que describe cómo será la transformación sobre del AI
* *AF (Árbol final)* es la estructura resultante de la transformación aplicada a AI

Una vez identificados los elementos necesarios de la transformación, vemos que XML es el lenguaje que codifica la estructura de árbol. En cuanto al *descriptor de transformación* su codificación no es tan clara, la primera noción perceptible al respecto es la de *señalar los nodos* *en el árbol inicial* *a los cuales se aplicarán las operaciones* y esta es la función de XPATH en el marco de las Proyecciones XML y es la herramienta esencial y fundamental en que se apoya este trabajo.

**Lenguaje de direccionamiento de nodos: XPATH**

XPATH es un lenguaje de direccionamiento de elementos de una estructura XML, diseñado para ser utilizado por las normas de procesamiento XQuery y XSLT. A su vez este estándar de direccionamiento está fundamentado en la sintaxis del Uniform Resource Identifiers (URI RFC 3986, RFC 2396) al cual XPATH añade los operadores necesarios y propios de un lenguaje de consulta de datos sobre un sistema de almacén de datos.

Por motivos de espacio no es posible realizar una explicación profunda sobre una norma tan completa como lo es el estándar XPATH, además no es el objetivo de este documento enseñar a utilizarlo, un sencillo tutorial por parte de la W3C ha sido preparado en [[1]](http://www.w3schools.com/xml/xpath_intro.asp) a manera de introducir a los programadores en su uso (se recomienda su lectura antes de continuar leyendo este documento). Sin embargo, para propósitos de ingeniería la descripción completa de esta recomendación se encuentra en [[2]](https://www.w3.org/TR/xpath/). De forma somera se puede decir sin exagerar, que XPATH utilizado junto a otro estándar de procesamiento (como XQuery o XSLT) tiene la misma capacidad de abstracción y semántica de extracción de datos que un leguaje SQL.

Se retoma el ejemplo 1 para mostrar la funcionalidad de XPATH en la transformación. El descriptor de transformación sobre el árbol A1 contiene varios enunciados que señalan los nodos del árbol de entrada *(enunciados de consulta)* con los que se ha de operar la transformación, a continuación se muestran en el siguiente diagrama:

D 6 Elementos agrupados según el descriptor





Cambiamos esos enunciados por sus equivalentes expresiones XPATH y el resultado sería el siguiente:



Queda claro ahora la potencia de expresión de XPATH bastan unos cuantos símbolos para expresar los elementos que se necesitan direccionar o *apuntar.* Se introduce el concepto de ***subconjunto de trabajo*** como el conjunto de todos los nodos del árbol inicial que serán proyectados en el árbol final

**Descriptor de Transformación: Proyección**

Ya se tiene el lenguaje que permite tomar los elementos requeridos del árbol de entrada de la función de transformación, el siguiente paso será reconocer la estructura de salida que se busca, definir el lenguaje que permita expresar la estructura general del árbol de salida y la relación que tiene con los elementos tomados del árbol de entrada.

Se definimos al descriptor de transformación como ***proyección de transformación*** o simplemente proyección, por el hecho de que el descriptor debe contener la *traza de la transformación* de una estructura de árbol AI en otro AF y por otro lado en el descriptor se definirá el *alcance* de las operaciones de transformación porque solo se proyectaran sobre de AF las estructuras del árbol AI que validen con las expresiones Xpath del subconjunto de trabajo.

Continuamos analizando el Ejemplo 1. En el diagrama D6 que muestra la relación entre los árboles de entrada y salida de la operación de transformación, se hace notar la asociación directa que existe entre las raíces de los árboles A1 y A2. Analizando este concepto resulta lo siguiente:

En toda proyección de un árbol A en B las raíces de A y B siempre estarán asociadas.

Así, vemos que cualquier proyección comenzará por definir cómo será la transformación de un árbol en otro desde la asociación entre sus raíces, este será el eje central de la transformación.

Ya se estudió los enunciados de consulta del descriptor de transformación que en adelante llamaremos proyección. Sin embargo estos no se encuentran solos sino acompañados de un *enunciado de colocación,* que es aquel que describe la relación de los elementos del árbol entrada con los del árbol resultante. El primer enunciado “Por cada nodo C del tercer nivel de árbol A1” está unido con un segundo “agregar a la raíz con etiqueta R del árbol A2 un nodo hijo H”, y en ellos se mencionan los conceptos “tercer nivel” y “raíz” respectivamente, es decir, este enunciado establece las relaciones entre los arboles A1 y A2 desde sus ra. En general podemos definir lo siguiente:

A la proposición en el descriptor de transformación que describa la relación de los elementos de la transformación desde la raíz se le llamará ***trayectoria de la proyección.*** Al conjunto de las proposiciones que no son parte de la trayectoria de la proyección se les conocerá como ***mapa de la proyección.***

Agrupamos los enunciados del descriptor de transformación (DT1) del ejemplo 1 de acuerdo a los conceptos anteriores y tenemos lo siguiente:

**P = [**

**Trayectoria de la proyección:** Por cada nodo C del tercer nivel de árbol A1 agregar a la raíz con etiqueta R del árbol A2 un nodo hijo H,

**Mapa de la proyección: {**los hijos de cada nodo C con etiqueta F y K serán hijos de su correspondiente representación como nodo H en el árbol A2, los de etiqueta F serán renombrados como P y los de etiqueta K como I.**}**

**]**

Sustituimos los enunciados de (PT2) por las expresiones XPATH del diagrama D8 que señalan el direccionamiento de los elementos tomados de A1 y la notación de agrupamiento del diagrama D4 que señalan la estructura general del árbol A2 y resulta lo siguiente:

**PT = [****Trayectoria de la proyección:** //A/B/C --> H\*

**Mapa de la proyección:** **{**F --> P\*,

K --> I\***}]**

**Descriptor >\_FROOT** **y Descriptor >\_MAP**.

Se empieza a hacer presente la sintaxis del descriptor de transformación, falta aclarar algunos detalles más antes de definir formalmente las expresiones regulares que la validan. La palabra reservada **>\_FROOT** (**F**rom r**OOT**, *desde la raíz*)simbolizará a la trayectoria de la proyección mientras que el mapa de la proyección estará simbolizado por la palabra reservada **>\_MAP** (**MAP**ping, *trazado del mapa*). Sustituimos esto en (PT3) y resulta

**DT = [ >\_FROOT:** H\* <-- //A/B/C,

**>\_MAP:** **{**P\* <-- F,

I\* <-- K**}**

**]**

Falta señalar a la raíz del árbol A2 y esta ocupará el siguiente lugar:

**DT =** //R **[>\_FROOT:** H\* <-- //A/B/C,

**>\_MAP:** **{**P\* <-- F,

I\* <-- K**}**

**]**

A través de la necesidad de simbolizar los enunciados del descriptor de transformación se han señalado sus elementos constituyentes que son clave de la propuesta de sintaxis del descriptor de transformación que se hará a continuación. Señalamos que el prototipo de lenguaje que estamos proponiendo es declarativo, pues aquí no importa señalar los mecanismos computacionales requeridos para realizar la transformación sino solo se requiere señalar como serán las relaciones entre los elementos de los arboles AI y AF.

**Resultados: Implementación en PHP**

La implementación de la Función de Transformación en PHP ha sido realizada a través de la notación de corchetes de este lenguaje, lo cual nos ha facilitado la tarea al no tener que realizar un analizador léxico, sintáctico y semántico de la proyección solo se ha programado el motor de la traza de transformaciones el cual se basa en un recorrido por profundidad a través de la estructura de la proyección con la mecánica de una función recursiva.

**Sintaxis**

**Modulo principal**

La estructura de proyección será un arreglo asignado a una variable, que define los índices ‘>\_FROOT’ y ‘>\_MAP’ con la respectiva funcionalidad que describimos anteriormente, obedeciendo la siguiente estructura:

$main = [‘>\_MAP’=> [IF1 => A1,

IF2 => A2

IFn => An],

‘>\_FROOT’=> *expresión Xpath*];

**Índices del Mapa**

Los índices finales IFn serán de tipo String en PHP que pueden contener cualquier palabra mientras que las asignaciones An serán String que contendrán una expresión Xpath que apunta a los nodos hoja del subconjunto de trabajo, así si tomamos a PTf de la anterior sección y queremos expresarlo con esta codificación tenemos lo siguiente:

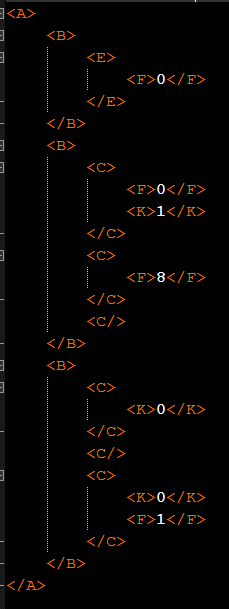
$main = [‘>\_MAP’=> [‘P’ => ‘F’,

‘I’ => ‘K’],

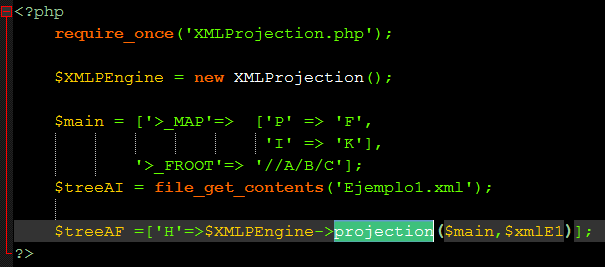
‘>\_FROOT’=> ‘//A/B/C’];

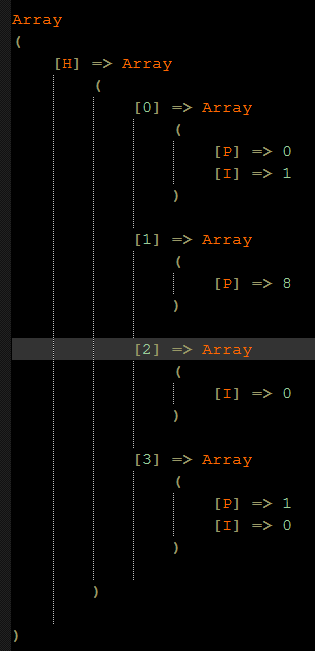
Adelante mostramos el ejemplo de su codificación:

Código 1 Ejemplo1.xml



Código 2 Programa Principal en PHP





Código 3 Resultado de Transformación

**Conclusiones**

Vemos en la capacidad de expresión de Xpath un lenguaje potente para describir transformaciones sin la necesidad de utilizar los símbolos de etiquetas propios del lenguaje de transformación XSLT, lo cual haría más fácil la labor de entendimiento de relaciones entre transformaciones, reduciendo el número de líneas de código necesarias para su implementación y aumentando la productividad en el desarrollo de integraciones entre distintos sistemas.

**Bibliografía**

[1] W3C Working Group. (s.n.). XPath Tutorial. 14-10-2016, de World Wide Web Consortium Sitio web: <http://www.w3schools.com/xml/xpath_intro.asp>

[2] XML Working Group. (16 de Noviembre de 1999). XML Path Language (XPath). 10-10-2016, de World Wide Web Consortium Sitio web: <https://www.w3.org/TR/xpath/>

[3] Network Working Group . (Enero, 2008). Augmented BNF for Syntax Specifications: ABNF. 10-10-.2016, de The Internet Engineering Task Force (IETF®) Sitio web: <https://www.ietf.org/rfc/rfc5234.txt>

[4] XML Working Group. (1999). XSL Transformations (XSLT) Version 1.0. 10-10-2016, de World Wide Web Consortium Sitio web: <https://www.w3.org/TR/xslt>