ReDaX (Relational to XML data publishing) un framework liviano para publicar información relacional

Emilio G. Ormeño (*), Fabián R. Berón (*)

(*) {eormeno, frberon}@iinfo.unsj.edu.ar
Instituto de Informática
Universidad Nacional de San Juan
Av. Ignacio de la Roza 590, San Juan Argentina.
Teléfono: (54 264) 423-4129

Resumen

Quizás uno de los mayores inconvenientes que posee XML, es que no ha sido pensado para almacenar información, en vez de ello, ha sido diseñado para permitir la publicación y el intercambio de información a través de la especificación XSL (eXtensible Stylesheet Languaje). Sin embargo, la mayor parte de la información de una empresa se encuentra en bases de datos relacionales. La publicación de información vía XML, es el proceso de transformar la información relacional en un documento XML para ser intercambiado a través de internet. Los productores de bases de datos ofrecen varias soluciones, sin embargo, al no existir un estándar, ellas difieren considerablemente entre sí; además de consumir una gran cantidad de recursos del servidor. Presentamos en este paper, un framework liviano llamado ReDaX que transforma el cursor resultante de una consulta SQL, en un documento XML. Un simple lenguaje llamado DXL (Data to XML Language) que define los elementos XML, sus relaciones de paternidad, y la traducción de los campos del cursor a atributos de dichos elementos, maneja dicha transformación; siendo el cursor, en la actual implementación, un objeto ResultSet de la interface JDBC (Java DataBase Connectivity).

Palabras clave

Data exchange, XML, Relational databases, XML Queries, Web Services.

1. Introducción

La necesidad de transferir información relacional vía XML ha crecido, y este hecho se hace más evidente en la actualidad con el advenimiento de los Web Services [12] tales como SOAP, WSDL, o UDDI. Sin embargo, la mayor parte de la información que una empresa posee, se encuentra almacenada en bases de datos relacionales. Esto habla respecto a la necesidad de poseer herramientas que traduzcan la información relacional en documentos XML, a fin de ser intercambiados a través de Internet.

Si bien los motores de bases de datos relacionales son especialmente buenos para almacenar y manipular información altamente estructurada, no lo son tanto cuando la información que se pretende almacenar es semiestructurada tal como la información que un documento XML posee. Es decir, que mientras la información relacional está normalizada a través de relaciones, la información XML está anidada y no normalizada. Con lo cual vemos que ambos modelos difieren muy significativamente.

En este paper describimos un framework liviano llamado ReDaX que transforma el cursor resultante de una consulta SQL en un documento XML. El proceso de transformación es definido con un simple lenguaje llamado DXL (Data to XML Language) que define los elementos XML, la forma en que se encuentran anidados, y la traducción de los campos del cursor a atributos de los elementos XML; siendo el cursor un objeto ResultSet de la interface JDBC.

ReDaX no es un lenguaje de consulta: utiliza el lenguaje DXL para estructurar la información de una vista de la base de datos. La salida XML resultante de la transformación es un simple anidamiento de campos dentro de registros, y dentro de tablas tal como el draft [1] de W3C propone.

El lenguaje DXL establece el significado estructural del cursor en términos de un árbol XML. Posee dos partes, en la primera, se definen los elementos XML (el conjunto de tags) y el conjunto de los campos del cursor que le corresponden; en la segunda parte, se definen la relaciones de paternidad entre los elementos definidos en la primera parte. En la implementación actual, traducimos cada campo del cursor como un atributo del Tag.

Este paper se organiza de la siguiente forma: en la sección 2, presentamos la arquitectura de ReDaX, en la sección 3, mostramos algunos trabajo relacionados. En la sección 4 y 5, presentamos un ejemplo simple y otro más sofisticado respectivamente. En la sección 6, explicamos la gramática y sentencias que se utilizan en el leguaje DXL. Las conclusiones y el trabajo a futuro se presentan en la sección 7.

2. Arquitectura de ReDaX

ReDaX es un middleware entre una interface JDBC y una aplicación que accede a la información a través de internet. La figura 1 muestra en detalle su arquitectura.

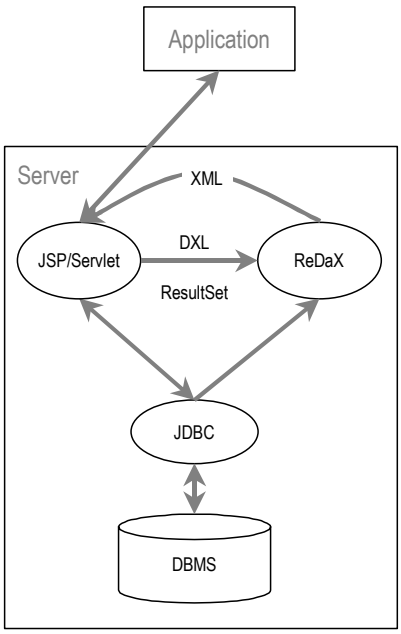


Figura 1: Arquitectura ReDaX.

La consulta SQL es ejecutada a través de un Servlet o de un JSP (Java Server Pages) que se ejecuta en el servidor, obteniéndose un cursor en la forma de un objeto ResultSet. Este objeto y la

definición de su significado estructural utilizando DXL es pasado a un objeto ReDaX, el cual procesa y retorna un documento XML. Dicho documento, puede, luego ser transformado utilizando una hoja de estilo para ser publicado en un navegador.

3. Trabajos relacionados

Existen varios frameworks relacionados a la publicación de información relacional. Uno de ellos es XQL [2,3] que es un lenguaje de consulta diseñado para representar una consulta a una base de datos relacional en formato XML. Aunque aún no ha sido formalmente aceptado por la W3C, constituye un método muy útil para especificar el acceso a la informacion almacenada en una base de datos, en un formato XML.

Otro framework que procesa información relacional a fin de ser publicada, es SilkRoute, que extrae informacion de una base de datos relacional y construye un documento XML al agregar los apropiados elementos, atributos, y sus anidamientos. Requiere que el administrador de la base de datos escriba una consulta en un lenguaje llamado RXL (Relational to XML Transformation Language) que define la vista de la base de datos. Este lenguaje, posteriormente, se transforma en una consulta SQL que es enviada al motor.

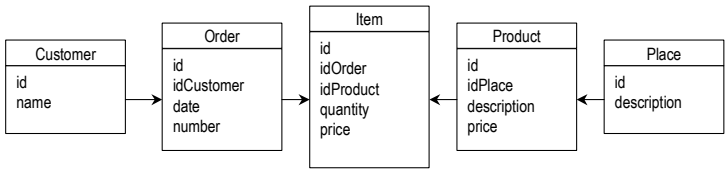
Ambos frameworks requieren el aprendizaje de un lenguaje completamente nuevo. En el primer caso, el lenguaje XQL es en sí mismo, una ampliación del lenguaje Xpath, con cuya complejidad, el desarrollador se tendrá que ver enfrentado. En el segundo caso, se requiere la utilización del lenguaje RXL además de un framework propietario y que requiere del servidor una gran cantidad de recursos.

Nuestra propuesta requiere solo de un cursor producido por una consulta realizada a la base de datos utilizando el lenguaje SQL y una simple definición respecto al árbol XML de salida que se requiere.

Otras soluciones provistas por diversos productores de bases de datos generan una salida XML siguiendo la misma estructura de la salida de la consulta e incluyendo Tags de elementos propietarios tales como “rowset” o “record”. De esta forma, el documento XML resultante necesita ser preprocesado con una hoja de estilo para que se ajuste a la estructura XML deseada a fin de ser publicada. El framework propuesto en este trabajo, además de ser independiente del motor de base de datos produce una salida XML que no necesita ser procesada nuevamente, a menos, claro está que deseemos un formato HTML de la misma.

4. Un ejemplo simple

El ejemplo completo con su código fuente puede ser visto en [6]. La Figura 2, muestra el clásico modelo de datos de ordenes, clientes, productos, y sus relaciones, junto a una información de ejemplo para que pueda apreciarse el trabajo del framework.



Customer	
id	name
1	John Smith
2	Robert O'Neill
3	Paul Mackenzie

Product			
id	idPlace	description	price
1	1	Pencil	1.00
2	2	Rubber	1.50
3	3	Paper	0.35

Place	
id	description
1	Storage #1
2	Storage #2
3	Storage #3

Order			
id	idCustomer	date	number
1	2	03-25-2003	0001
2	2	03-30-2003	0002
3	1	04-03-2003	0003
4	3	04-03-2003	

Item				
id	idOrder	idProduct	quantity	price
1	1	1	5	1.00
2	1	2	1	1.50
3	1	3	10	0.35
4	2	2	15	1.50
5	3	3	2	0.35
6	3	1	9	1.00
7	4	1	40	1.00
8	4	2	4	1.50

Figura 2: Un ejemplo de una base de datos.

Como ejemplo, si se tuviera que obtener la representación XML de las ordenes que aún no han sido cumplidas, en este caso, aquellas que no poseen un número de factura. Para ello, primeramente, se debería escribir la consulta SQL que se ve en la Figura 3 y cuyo cursor resultante se puede ver debajo de ella.

```
Select *
   from Order, Item, Customer, Product
   where Order.numer = null and
         Order.idCustomer = Customer.id and
         Item.idOrder = Order.id and
         Item.idProduct = Product.id
```

Order				Item					Customer		Product			
id	idCustomer	date	number	id	idOrder	idProduct	quantity	price	id	name	id	idPlace	description	price
4	3	04-03-2003		7	4	1	40	1.00	3	Paul Mackenzie	1	1	Pencil	1.00
4	3	04-03-2003		8	4	2	4	1.50	3	Paul Mackenzie	2	2	Rubber	1.50

Figura 3: Consulta de las ordenes no cumplidas y el cursor resultante.

En segundo lugar, y a fin de obtener el documento XML, ReDaX requiere de la definición de los elementos XML y de qué columnas del cursor se corresponde a cada atributo del elemento, además de las relaciones de anidamiento entre dichos elementos. Esto se realiza a través de la definición utilizando DXL que puede ser visto en la figura 4.

```
define Order from 1 as id, 4 as number idfield 1;
define Customer from 10 as id, 11 as name idfield 10;
define Item from 5 as id idfield 5;
define Product from 12 as id, 14 as description idfield 12;
parent Order of Customer;
parent Order of Item;
parent Item of Product;
```

Figura 4: La definición de la salida utilizando DXL.

La palabra clave define establece el conjunto de columnas del cursor que corresponden a un elemento XML. En el ejemplo, en la línea número 1 de la figura precedente, se está definiendo lo siguiente:

1. Que existe un elemento Order.
2. El conjunto de atributos del elemento Order está formado por las columnas número 1 y número 4 del cursor.

3. Cada valor de dichas columnas pasará a formar parte de un atributo cuyo nombre es la palabra que sigue luego de la palabra clave *as* en la definición del campo.
4. El campo número 1 es considerado identificatorio, en este caso la clave primaria.

Las relaciones se establecen en las tres últimas sentencias a través de la palabra clave parent que define que un elemento Order es padre de un elemento Customer y de un elemento Item, y a su vez, ese elemento Item es padre de un elemento Product.

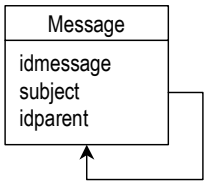
Dados, entonces, la definición DXL precedente y el cursor mostrado, ReDaX produce el documento XML que puede ser visto en la Figura 5.

```
<Order id='4' idCustomer='3' date='04-03-2003' number=''>
  <Customer id='3' name='Paul Mackenzie' />
  <Item id='7' idOrder='4' idProduct='1' quantity='40' price='1.00'>
    <Product id='1' idPlace='1' description='Pencil' price='1.00'/>
  </Item>
  <Item id='8' idOrder='4' idProduct='2' quantity='4' price='1.50'>
    <Product id='2' idPlace='2' description='Rubber' price='1.50'/>
  </Item>
</Order>
```

Figura 5: El documento XML resultante.

5. Un ejemplo recursivo

Sin importar la estructura de la base de datos y las limitaciones del lenguaje SQL, siempre será necesario extraer información anidada de ella. En este ejemplo, presentamos un foro de discusión simplificado en una sola tabla, la cual está relacionada consigo misma para almacenar la estructura de preguntas y respuestas. Permitiendo, de esa forma, representar una estructura de mensajes anidada. La Figura 6, muestra el modelo de datos de una sola tabla con un cursor de ejemplo.



idmessage	idparent	subject
1	[NULL]	How can I print ?
2	[NULL]	Is neccesary to print?
3	[NULL]	Printing is so important
4	1	You should have correctly configured the printer
5	2	Sometimes is useful to read from the paper.
6	5	Yes, I agree with that
7	1	And the correct drivers too..
8	5	Using a lot of papers is no so ecologic
9	3	Sometimes... Perhaps...
10	9	Can you explain me this again?
11	10	Sometimes is useful to read from the monitor.
12	8	I agree with that
13	12	I don't agree because...

Figura 6: Estructura de la tabla de mensajes.

A fin de extraer la estructura, se puede escribir una consulta SQL y obtener un cursor que puede ser visto en la Figura 7. Obviamente, debido a que SQL no es recursivo, es necesario limitar el número de niveles, en este ejemplo, se encuentra limitado a cinco niveles. La consulta SQL puede ser vista en [6].

Message				
id	id	id	id	id
1	4			
1	7			
2	5	6		
2	5	8	12	13
3	9	10	11	

Figura 7: El cursor de la consulta sobre la tabla de mensajes anidados.

La siguiente etapa consiste en definir los elementos XML y sus relaciones. La Figura 8, muestra la definición realizada con el lenguaje DXL. Lo nuevo respecto al anterior ejemplo es la palabra clave named después del nombre del elemento, el cual es utilizado como un alias. Otra diferencia es que en la primera línea se define un elemento vacío debido a que la especificación XML requiere que un documento posea un elemento raíz.

```
define Forum idfield 1;
define Message1 named Message from 2 as id idfield 2;
define Message2 named Message from 3 as id idfield 3;
define Message3 named Message from 4 as id idfield 4;
define Message4 named Message from 5 as id idfield 5;
define Message5 named Message from 6 as id idfield 6;
parent Forum of Message1;
parent Message1 of Message2;
parent Message2 of Message3;
parent Message3 of Message4;
parent Message4 of Message5;
```

Figura 8: La definición DXL del ejemplo de una tabla con anidamiento.

La Figura 9, muestra el documento XML resultante cuando la precedente definición es procesada por el framework ReDaX.

```
<Forum>
 <Message id='1'>
  <Message id='4'/>
  <Message id='7'/>
 </Message>
 <Message id='2'>
  <Message id='5'>
   <Message id='6'/>
   <Message id='8'>
    <Message id='12'>
     <Message id='13'/>
    </Message>
   </Message>
  </Message>
 </Message>
 <Message id='3'>
  <Message id='9'>
   <Message id='10'>
    <Message id='11'/>
   </Message>
  </Message>
 </Message>
</Forum>
```

Figura 9: El documento XML resultante del ejemplo de tabla con anidamiento.

6. Gramática de DXL

La Figura 10, muestra la gramática del lenguaje DXL, en las siguientes subsecciones, cada sentencia es explicada en detalle.

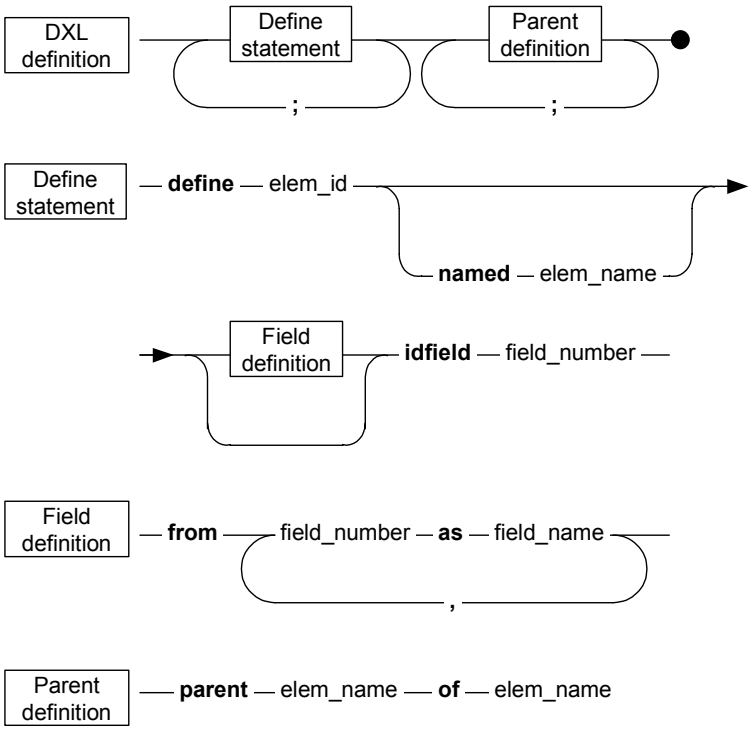


Figura 10: La gramática del lenguaje DXL.

6.1. La sentencia define

La sentencia define permite:

- Establece la existencia de un único elemento XML que está identificado por un nombre; el cual se representa en la gramática como elem_id. Este identificador será convertido a un tag XML, a menos que se encuentre especificada la cláusula named, en tal caso, el nombre que le siga será considerado un alias, el cual reemplazará al tag original. De cualquier forma, el nombre establecido será utilizado en las relaciones de anidamiento entre los elementos.
- Definir las columnas del cursor que corresponden al elemento XML que se está definiendo. Esto se representa en la gramática comenzando con la cláusula from seguida por una secuencia de pares field_number y un nombre. El primero, es la posición del campo en el cursor, el cual formará parte del elemento XML bajo la forma del valor de un atributo. El segundo, será el nombre de dicho atributo.
- Establecer cual de los campos definidos se considera identificador de cada elemento. Esto se realiza a través de la cláusula idfield, seguida del número de columna del cursor que lo identifica. No necesariamente, dicha columna o número de campo debe haber sido definida dentro de la cláusula from.

6.2. La sentencia parent

La sentencia parent permite establecer las relaciones de anidamiento o "paternidad" entre los elementos definidos. Como se puede ver en la figura de la gramática, una sentencia parent, comienza con dicha palabra clave seguida por el nombre que identifica al elemento hijo (no el alias).

7. Conclusiones y trabajo a futuro

Este trabajo forma parte de un framework más complejo llamado CDS (Course Development System) [7], el que se encuentra actualmente bajo construcción. El siguiente trabajo estará vinculado a la actualización de la base de datos utilizando un sistema de mensajería manejado por tokens. Dado que esos tokens se encuentran encapsulados bajo la forma de un documento XML, quedaría completo el escenario donde la base de datos puede ser procesada y actualizada.

En nuestra opinión, con herramienta tales como el framework presentado, se soluciona en parte uno de los mayores problemas que posee XML: no poder ser utilizado para almacenar información, en vez de ello, ha sido diseñado para intercambio, presentación, y configuración. De esa forma, queda delegada la tarea de acceder y actualizar físicamente la información a los motores de bases de datos, los cuales la ejecutan en forma efectiva y eficiente.

No proclamamos que este framework es la panacea, sin embargo, es lo suficientemente liviano como para ser utilizado en tecnologías tales como JSP, Servlet o inclusive PHP, que en su última versión incorpora sentencias para comunicarse con clases java.

8. Referencias

1. World Wide Web Consortium: *XML representation of a relational database*. URL: http://www.w3.org/XML/RDB.html (1997).
2. McLaughlin B. *Java and XML*. O'Reilly & Associates (2000).
3. Robie J., Lapp J., Schach D., World Wide Web Consortium: *XML Query Language*. URL: http://www.w3.org/TandS/QL/QL98/pp/xql.html (1998).
4. Dodds L. *XML and Databases? Follow Your Nose*. O'Reilly XML.com. URL: http://www.xml.com/pub/a/2001/10/24/follow-yr-nose.html (2001).
5. Fernandez M., Morishima A. *Publishing Relational Data in XML: the SilkRoute Approach*. In http://www.research.att.com/~mff/files (2001).
6. Instituto de Informática UNSJ. URL: http://www.idei-unsj.edu.ar
7. CDS (Courseware Development System). Ormeño, E.G., Ochoa, S.F. Un Ambiente de Desarrollo de Courseware. Memorias del VII Congreso Argentino de Ciencias de la Computación (CACIC 2001). Memorias en Formato Digital. El Calafate, Santa Cruz, Argentina. Oct. 16-20, 2001.
8. JSP (Java Server Pages), URL: http://java.sun.com/products/jsp
9. Java Servlets, URL: http://java.sun.com/products/servlet
10. JDBC (Java Database Connectivity), http://java.sun.com/products/jdbc
11. Ayala D., Browne C., Chopra V. *Professional Open Source Web Services*. Wrox Press 2002.