Obtención de datos XML a partir de información almacenada en bases de datos

Roberto Berjón Gallinas[1], Ana M. Fermoso García[1], and María J. Gil Larrea[2]

[1] Universidad Pontificia de Salamanca, Escuela Universitaria de Informática.
Salamanca, España.
{rberjon,afermoso}@upsa.es
[2] Universidad de Deusto, E.S.I.D.E.
Bilbao, España.
marijose@eside.deusto.es

Abstract. XML se ha convertido en el estándar para la presentación de información en la Web y también para su intercambio en los flujos inter o intra empresas. Por ello en éstas resulta cada vez más necesario generar XML a partir de la información que tienen almacenada en sus bases de datos. Los sistemas de gestión de bases de datos(DBMS) empleados siguen siendo en su mayoría relacionales, aunque actualmente también se utilizan otros que sí permiten el almacenamiento y obtención de información en formato XML: los RDBMS habilitados para XML y los DBMS nativos XML. Esta diversidad de fuentes, unido a la consiguiente distribución de la información entre ellas, dificulta la generación de XML.

En este trabajo se presenta detalladamente las herramientas existentes que generan XML a partir de información relacional, así como también las características de los DBMS que permiten el almacenamiento XML. Una vez analizados todos estos sistemas, se planteará una propuesta que trate de solventar las limitaciones que éstos poseen.

1 Introducción

XML se ha convertido en el estándar para la presentación de información en la Web y también para su intercambio en los flujos inter o intra empresas. Por ello en éstas resulta cada vez más necesario generar XML a partir de toda la información que poseen. Esta información generalmente se encuentra distribuida entre varias fuentes de información, como documentos XML, las tradicionales bases de datos relacionales, y también otro tipo de bases de datos que permiten el almacenamiento y obtención de información en formato XML: los RDBMS habilitados para XML y los DBMS nativos XML. Esta diversidad de fuentes, unido al hecho de que la información no se encuentre siempre en el mismo formato, dificulta la generación de XML.

En este trabajo se va a realizar un estudio de las principales fuentes de información de que disponen las empresas y de cómo se puede obtener XML de ellas. Lógicamente la situación ideal sería la de contar con una herramienta que

pudiese consultar simultáneamente diversas fuentes de datos, sean éstas de tipo XML o no, y poder generar XML a partir del resultado de dichas consultas. Para abordar esta problemática, se va a realizar una clasificación de las distintas fuentes de información, así como un estudio de las herramientas existentes que realizan dicha tarea. En virtud de las conclusiones obtenidas, se planteará una nueva propuesta que intente solventar las limitaciones encontradas.

El trabajo se encuentra organizado en tres capítulos, en el primero se analizarán las fuentes de datos no XML y las características de las herramientas existentes que permiten la transformación de sus datos a este formato. En el segundo, se estudiarán las fuentes de datos XML y, finalmente, en el último se presentará la nueva propuesta.

2 Fuentes de datos no XML

La principal fuente de datos no XML con que cuentan las empresas son los RDBMS. Para convertir la información que almacenan a un formato XML se precisa de herramientas externas. Éstas forman una capa intermedia entre las aplicaciones cliente y la base de datos (ver Fig. 1), de tal forma que los clientes envían sus peticiones hacia la herramienta, ésta consulta la base de datos y transforma el resultado obtenido a una representación XML que devuelve como respuesta.

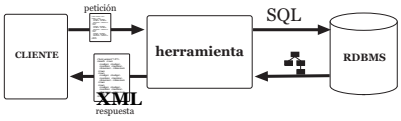

Fig. 1. Esquema gnral. de herramientas que convierten a XML datos de RDBMS

Estas herramientas se pueden clasificar, atendiendo a su patrón de diseño, en dos tipos: las que transforman el modelo relacional a una representación XML y aquéllas que lo que transforman es el resultado de consultas SQL a la base de datos.

2.1 Herramientas que transforman el modelo relacional a XML.

El esquema de funcionamiento de todas es similar al que se muestra en la Fig. 2. Crean de forma virtual una vista XML con información contenida en la base de datos. El hecho de que sea virtual quiere decir que la vista XML nunca llega a materializarse. Al ser esta vista lo único que realmente ve el usuario, éste debe emplear un lenguaje orientado a XML para consultar dicha información. Las herramientas traducen cada consulta a una o varias sentencias SQL que se emplearán para extraer la información necesaria de la base de datos. Posteriormente transforman a XML, siguiendo las indicaciones que el usuario haya definido en su petición, la información relacional obtenida. A continuación se describe las principales herramientas que siguen este modelo.

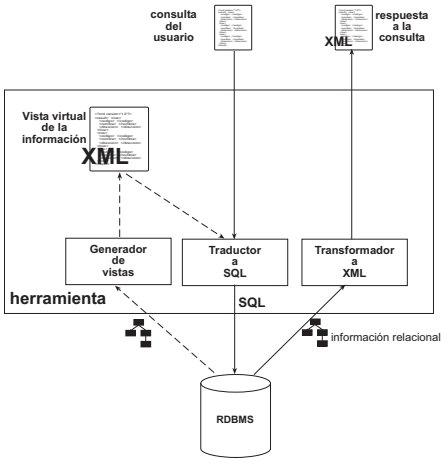

Fig. 2. Esquema de la transformación del modelo relacional a XML

XTABLES: [2](anteriormente conocida como XPERANTO [3]) crea automáticamente una vista canónica[3] denominada *default* que contiene la información de todas las tablas de la base de datos. Esta herramienta también permite crear nuevas vistas a través de consultas XQuery sobre la vista *default* o cualquier otra creada con posterioridad. El usuario final realiza consultas, también en XQuery, sobre el conjunto de vistas definidas.

Adicionalmente, XTABLES también permite la consulta simultánea tanto de datos relacionales como de otros documentos XML. Para ello, previamente se debe registrar dichos documentos XML en la herramienta, lo que implica el almacenamiento de estos documentos en la base de datos relacional. Este registro sigue los mismos conceptos enunciados por Shanmugasundaram en [4]. El proceso es el siguiente: a la herramienta se le debe facilitar el DTD de los documentos que se desee registrar. A partir de él crea un *DTD graph* que refleja la estructura jerárquica indicada por el DTD. Cada nodo de esta estructura representa un elemento, un atributo o un operador. Del *DTD graph* se crea un esquema relacional siguiendo las siguientes pautas: se crea una relación para el elemento raíz, todos los hijos de un elemento se representan como atributos de la

[3] recibe este nombre porque se aplica la misma regla para convertir cualquier tabla relacional a una representación XML. Esta regla determina que para cualquier relación con un esquema $R(A_1, A_2, \ldots, A_n)$ en la que existen las tuplas $(a_{11}, a_{21}, \ldots, a_{n1})$, $\ldots$, $(a_{1k}, a_{2k}, \ldots, a_{nk})$ su representación XML canónica tiene la siguiente estructura:

<R>
  <Tuple><A_1>a_{11}</A_1><A_2>a_{21}</A_2>...<A_n>a_{n1}</A_n></Tuple>
  ...
  <Tuple><A_1>a_{1k}</A_1><A_2>a_{2k}</A_2>...<A_n>a_{nk}</A_n></Tuple>
</R>

relación, excepto aquéllos que identifiquen los operadores + ó * ya que el modelo relacional no puede representar atributos con un conjunto de valores. En estos casos se crean otras nuevas relaciones manteniendo mediante restricciones del tipo *FOREIGN KEY* la referencia a su elemento padre.

Todas estas relaciones serán las tablas en donde se almacenen los datos contenidos en los documentos XML que posteriormente se registrarán en la herramienta.

Además de crear las tablas, la herramienta crea una nueva vista que reconstruye a partir de los datos almacenados, los documentos XML registrados. Esta vista, al igual que las otras, también podrá ser consultada por los usuarios finales, con lo que indirectamente se está consiguiendo consultar simultáneamente datos relacionales y datos XML.

El principal inconveniente de este planteamiento es la falta de sincronización entre el documento original y el almacenado en la base de datos. Si el documento original cambia, habría que actualizar también la base de datos, con lo que sería preciso conocer la ubicación exacta de de cada uno de los datos almacenados al ser preciso el uso de sentencias SQL, lo cual hace que la actualización sea una tarea excesivamente compleja.

SilkRoute: Su funcionamiento es similar a XTABLES salvo que esta herramienta sólo define una única vista XML y además no permite el registro de documentos XML en la base de datos. En [5][6] la vista XML se crea empleando el lenguaje RXL y los usuarios la consultan a través del lenguaje XML–QL [8]. Posteriormente en [7] los autores hicieron que XQuery fuese el lenguaje empleado tanto para definir la vista XML[4], como por los usuarios para consultarla.

XBD: [9] Al igual que las otras también define una vista que, a diferencia de las anteriores, muestra de forma arborescente la estructura de la base de datos siguiendo el siguiente planteamiento: si en un esquema relacional existen dos tablas T_1 y T_2 en donde T_2 define una relación *foreing key* respecto a la tabla T_1, la representación XML de cada registro de T_1 contiene además de los campos definidos en la tabla, todos los registros de la tabla T_2 relacionados con él. Los lenguajes que emplean los usuarios para acceder a la vista son *XSL adaptado* y *XQuery adaptado*, similares en apariencia a XSL y XQuery respectivamente, pero donde las expresiones XPath contenidas en ellos pueden, teniendo en cuenta la estructura de la vista, acceder fácilmente a registros de la base de datos relacionados entre sí.

La principal virtud de todas estas herramientas es que emplean un lenguaje de consulta basado en XML para acceder a la información y, por tanto, pueden

[4] Esta vista se denomina *public XML view* y se define a través de una consulta XQuery sobre una vista canónica denominada *canonical View* que automáticamente crea la herramienta y que engloba toda la información de la base de datos.

considerarse como una solución global ya que independientemente que la información resida en un RDBMS o en documentos XML independientes, se emplea siempre el mismo lenguaje de consulta. Sin embargo ésta es también su mayor desventaja ya que, en el caso de que la fuente de información sea una base de datos, dicho lenguaje tiene que ser traducido finalmente a SQL. Durante este proceso de traducción, estas herramientas se orientan únicamente en el modelo relacional de la información impidiendo por tanto hacer uso de las extensiones o facilidades que pueda ofrecer cada RDBMS (nuevos tipos de datos soportados, utilización de funciones estándares o definidas por el usuario) y que exigirían una sintaxis concreta y específica de SQL propia del RDBMS empleado.

Por otra parte, sólo SilkRoute define qué información concreta de la base de datos pueden ver los usuarios, ya que es necesario crear manualmente la vista XML que se les presenta. Esto puede considerarse una ventaja o un inconveniente, ya que la seguridad establecida se aplicará a todos los usuarios. La Tabla 2.1 resume las carácterísticas de estas herramientas.

Herramienta	**Lenguaje de vistas**	**Lenguaje de consulta**	**Se define la estructura del resultado XML**	**Permite consultar XML**	**Permite acotar la información accesible del RDBMS**
XTABLES	XQuery	XQuery	Si	Si	No
SilkRoute	{ RXL XQuery	{ XML-QL XQuery	Si	No	Si
XBD	– –	{ XSL adaptado XQuery adaptado	No	No	No

Table 1. Características de herramientas que transforman el modelo relacional a XML

2.2 Herramientas que transforman el resultado de consultas SQL a XML.

El otro patrón de diseño consiste en la utilización directa de SQL como lenguaje de consulta en las peticiones los clientes. El esquema general de funcionamiento de estas herramientas se representa en la Figura 3. Puede observarse cómo a la herramienta le llegan las peticiones de los clientes, de ellas extrae las sentencias SQL embebidas y las envía a la base de datos, transformando su resultado a XML.

La diferencia fundamental con respecto al anterior tipo de herramientas es que aquí las peticiones de los clientes incluyen las sentencias SQL con las que recuperar la información necesaria de la base de datos. Esta es precisamente su principal ventaja, ya que se puede emplear el SQL nativo del gestor de base de

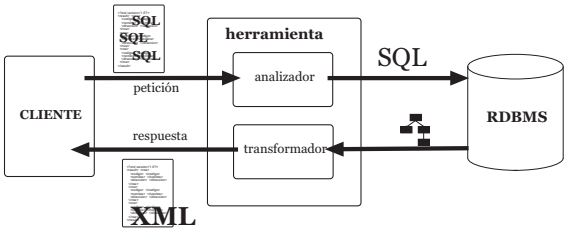

Fig. 3. Esquema general de herramientas que transforman SQL a XML

datos, pudiendo así solventar una de las desventajas analizadas en el anterior modelo.

Las herramimentas transforman los datos relacionales a XML ya sea siguiendo un modelo canónico, o bien según las indicaciones que especifique el usuario en sus peticiones. Según la estructura del XML que se desee obtener, esta transformación puede ser un inconveniente que no se encontraba en las otras herramientas puesto que empleaban lenguajes de consulta orientados a XML. XML es una estructura jerárquica (y como tal puede poseer varios niveles de anidamiento entre sus elementos), sin embargo la información a transformar, es decir, el resultado de consultas SQL, es relacional.

Según se indica en [29] "*En NF^2 las tuplas siguen una estructura jerárquica, lo que resulta útil para representar objetos que por su naturaleza están estructurados jerárquicamente*". Teniendo en cuenta esta afirmación, se puede concluir que la mejor opción para transformar los datos relacionales a XML es que éstos se encuentren siguiendo el modelo relacional anidado NF^2. Este modelo elimina la restricción de la primera forma normal (1NF) del modelo relacional básico (también conocido como modelo relacional plano) según el cual se prohíbe la definición de relaciones en las que estén presentes atributos multivaluados y/o atributos compuestos. Es decir, todos los atributos deben tener un único valor de entre los que define un dominio atómico (un dominio es atómico si sus elementos están considerados como unidades indivisibles[5]).

Por lo tanto, no habría tales problemas si las consultas SQL pudiesen obtener vistas NF^2. Sin embargo, la mayor parte de los RDBMS definen una base de datos como un conjunto de relaciones planas o tablas en primera forma normal. La base de datos Oracle es una excepción ya que permite el uso de UDT (User Defined Types), tablas anadidas, VARRAYS y además también permite obtener vistas NF^2 en las sentencias SQL mediante el uso de CURSORES.

Por lo tanto, las herramientas deberían o bien soportar vistas NF^2, o bien proporcionar la operación NEST[6], o bien facilitar cualquier otra opción que finalmente lo consiga.

[5] Por ello, a este modelo anidado también se le conoce como *modelo relacional no en primera forma normal*, *No-1NF*, *NFNF* ó *NF^2*.

[6] Se ha propuesto extensiones del álgebra relacional [27] [28] [30] y del cálculo relacional que permiten realizar la conversión de relaciones planas a vistas NF^2. Todas

Con respecto a lo expuesto anteriormente, se puede hacer una primera clasificación dentro de las herramientas que transforman SQL a XML atendiendo al criterio de anidamiento. De este modo, este tipo de herramientas se puede agrupar en:

(1) Aquéllas que permiten tratar directamente con vistas NF^2. *Oracle XML SQL Utility (XSU)* y *Oracle XSQL Pages* [10] son herramientas que siguen este patrón.
(2) Aquéllas que permiten convertir una vista plana (que será obtenida a través de la consulta SELECT embebida en la petición del usuario) a una vista NF^2 mediante el operador NEST: ejemplos de este tipo son *XML/SQL* [11] y *uR2X* [12].
(3) Aquéllas que permiten anidamiento definiéndolo en la propia sentencia SELECT: *SQLXML XML for SQL Server 2000* [14].
(4) Aquéllas en las que el anidamiento se simula ejecutando una sentencia SQL por cada una de las tuplas devueltas por otra. Por ejemplo: se ejecuta una sentencia SQL que devuelve la información de todos los alumnos, por cada uno de ellos se ejecuta otra sentencia SQL que devuelve las asignaturas de que está matriculado dicho alumno. Este tipo de herramientas deben permitir la ejecución de más de una sentencia SQL y además admitir que en la definición de las consultas SELECT se puedan incluir parámetros (con el único objeto de no tener que rescribir dichas consultas): *DB2XML* [21], *Net.Data* [22] [23] [22], *ODBC2XML / JDBCXML* [24], *JSP* [25] son herramientas que siguen este patrón.
(5) Aquéllas que no permiten trabajar con vistas NF^2, no tienen definida la operación de anidamiento NEST y tampoco pueden simular el anidamiento descrito anteriormente. Un ejemplo es *Extensiones SAX y DOM para JDBC* [26].

---

coinciden básicamente en la definición de dos operaciones básicas: anidar (NEST) y desanidar (UNNEST). Básicamente NEST, que se denota como: $\nu_{(A_1,A_2,\ldots,A_n)}(r)$, crea una nueva relación r' agrupando las tuplas de r que tienen el mismo valor en todos sus atributos excepto en los indicados por A_i.

Sea r una relación con un esquema $R = (A_1, A_2, \ldots, A_n)$, y $\nu_{(B_1,B_2,\ldots,B_m)}(r)$ la operación NEST sobre la relación r, donde cada B_i es un A_j distinto para algún j. Sea $C_1, C_2, \ldots, C_{n-m}$ las A_j que no son iguales a ningún B_i. Entonces, el resultado de la expresión anterior, es una relación r' basada en un esquema R'. Para calcular este esquema se siguen los siguientes pasos:

1. Se añade la regla $B = (B_1, B_2, \ldots, B_n)$.
2. Se añade la regla $R' = (C_1, C_2, \ldots, C_{n-m}, B)$.
3. Finalmente, la relación r' es el resultado de los siguientes pasos:
   (a) Dividir r en grupos de tuplas que coinciden en $C_1, C_2, \ldots, C_{n-m}$. Estos grupos están identificados por $G_1, G_2, \ldots, G_p$.
   (b) Incluir en r' una tupla t_i por cada grupo G_i, donde t_i toma sus valores de los $C_1, C_2, \ldots, C_{n-m}$ comunes a todas las tuplas en G_i y $t_i[B]$ es el conjunto de valores $(B_1, B_2, \ldots, B_n)$ de las tuplas en G_i.

Dentro de esta clasificación, se podría realizar una segunda atendiendo a otra serie de características enumeradas a continuación:

(a) Si la petición que realiza el usuario a la herramienta es a su vez un documento XML.
(b) Si la estructura del documento respuesta que genera la herramienta no está prefijada por ésta. Es decir, que sea el propio usuario quien, de alguna forma, pueda incluir en la petición la estructura con la que finalmente la herramienta genere el XML resultado, sin necesidad de realizar una transformación XSL posterior.
(c) Si la petición que realiza el usuario no tiene limitado el número de consultas SQL que se pueda incluir.
(d) Si las consultas SQL embebidas en la petición pueden contener parámetros cuyos valores también serán incluidos dentro de la petición.
(e) Si permite la consulta simultánea de distintas fuentes de datos (DBMSs)
(f) Si permite el acceso a UDTs (User Defined Types) y colecciones (VARRAYS o tablas anidadas)

Esta clasificación y subclasificación se resume en la Tabla 2.2:

Herramienta	NF^2					Otras características					
	(1)	(2)	(3)	(4)	(5)	(a)	(b)	(c)	(d)	(e)	(f)
DB2XML				⊙		⊙					
Net.Data				⊙		⊙	⊙	⊙	⊙		
SQLXML			⊙			⊙	⊙	⊙	⊙		
ODBC2XML/JDBC2XML				⊙		⊙	⊙	⊙	⊙		
Oracle XML SQL Utility	⊙								⊙		⊙
Oracle XSQL Pages	⊙					⊙		⊙	⊙	⊙	⊙
SAX and DOM Extensions					⊙						
JSP				⊙		⊙	⊙	⊙	⊙	⊙	
XML/SQL		⊙				⊙	⊙	⊙	⊙	⊙	
uR2X		⊙				⊙	⊙	⊙	⊙	⊙	

Table 2. Características de herramientas que transforman SQL a XML

3 Fuentes de datos XML

En la actualidad las empresas disponen de varios tipos de bases de datos que permiten el almacenamiento y consulta de información XML. Éstas se pueden clasificar en dos grandes grupos que se analizarán a continuación: las bases de datos relacionales habilitadas para XML y las bases de datos nativas XML.

3.1 RDBMS habilitados para XML:

Los RDBMS habilitados para XML (Oracle 9i Release 2 [10], DB2 XML Extender [15] y SQLServer 2005 [13]) son bases de datos tradicionales que definen un nuevo tipo de dato que permite el almacenamiento de información en formato XML. En todos ellos la información XML a almacenar sufre algún tipo de transformación, completamente transparente para el usuario, que en algunos sistemas implica la fragmentación del documento XML a fin de que éste pueda ser almacenado en tablas relacionales que posteriormente podrán ser indexadas y por tanto mejorar el rendimiento de la base de datos durante el proceso de consulta y extracción de este tipo de información. Esta transformación tiene dos planteamientos diferentes.

Uno es el aportado por Oracle y DB2. En ellos el tipo de dato XML (XMLType en el caso de Oracle y XMLVARCHAR en el de DB2) posee una tabla adjunta en donde se almacena la información contenida en el documento XML (posteriormente se puede crear índices en estas tablas para que su acceso sea más eficiente). Esto exige una previa asociación entre el contenido de los elementos y atributos XML con los campos de dicha tabla. En el caso de DB2 esta asociación se realiza a través de un *DAD file* en el que se describe la tabla con sus campos y, a través de expresiones XPath, el origen de esa información dentro del documento. En el caso de Oracle, se debe registrar previamente una versión extendida del XML Schema del documento XML, este proceso crea UDTs por cada complexType contenido en aquél (sus campos almacenarán la información de cada elemento o atributo XML) y finalmente se crea la tabla del UDT correspondiente al *root element* del documento.

El otro planteamiento es el utilizado por SQLServer 2005. En éste, el documento XML se almacena en un formato binario en el que los elementos se identifican a través de un número, que actúa a modo de índice, y la información embebida en el documento XML se convierte previamente al correspondiente tipo de dato en virtud de la naturaleza de dicha información.

El primer planteamiento es mejor, puesto que el usuario decide qué información en concreto desea indexar a fin de optimizar la búsqueda dentro de los datos XML. Además Oracle es quizá el mejor sistema ya que, a diferencia de lo que ocurre en DB2, no se produce una redundancia en la información almacenada (en DB2 además de almacenar la información fragmentada en tablas, también guarda el original y aunque mantiene ambas informaciones sincronizadas, esto redunda en emplear un mayor espacio de almacenamiento y un menor rendimiento en las actualizaciones).

En cuanto al lenguaje, embebido en las sentencias SQL, utilizado para consultar la información XML, sin duda SQLServer 2005 es el más potente. Esto se debe a que utiliza XQuery, en lugar de XPath como sucede en los otros sistemas.

En todos, además de permitir el almacenamiento XML, también se puede obtener este mismo formato a partir de su información puramente relacional. En este sentido, Oracle es quien mejor implementa esta característica ya que ofrece un conjunto de funciones, que se incluirán en la sentencia SELECT, encargadas de definir la estructura del documento XML de salida. En los otros sistemas, la

estructura del XML obtenido depende por completo de la estructura de la sentencia SQL (orden en que se seleccionen las tablas y los campos en la consulta). Se observa, por tanto, que no sólo la sintaxis de las sentencias SELECT es completamente diferente y particular en cada sistema (no siendo portables entre las distintas bases de datos) sino que también su formulación es muy compleja de definir.

La Tabla 3.1 muestra de forma resumida las principales características de los RDBMS analizados.

RDBMS	**Tipo de dato**	**Se emplea el lenguaje XML**	**La generación de XML a partir de datos relacionales se realiza empleando**
Oracle 9i R2	XMLType	XPath	Las funciones { XMLElement, XMLAttributes, XMLForest, XMLAgg, XMLConcat
SQLServer 2005	XML	XQuery	`SELECT ...` `FROM ...` `WHERE ...` `FOR XML, TYPE;`
DB2 XML Extender	XMLVarchar XMLCLOB XMLFile	XPath	{ *Compose XML*, *DAD file*

Table 3. Características de los RDBMS habilitados para XML

3.2 Bases de datos nativas XML

Permiten el almacenamiento, consulta y actualización de información XML. La diferencia fundamental entre ellas (Tamino [16], X-Hive/DB [17], dbXML [18], eXist [19], Xindice [20]) es el lenguaje empleado tanto para la consulta como para la actualización de la información almacenada. Se observa que en todos se emplea XQuery, XPath o ambos lenguajes para consultar la información. Sin duda en los sistemas donde se utilice XQuery podrá plantearse consultas mucho más complejas que en aquéllos que empleen XPath, al ser aquél un lenguaje más potente y versátil. En cuanto al lenguaje empleado para las actualizaciones, todavía no hay uno estandarizado aunque de momento XUpdate [1] es el más extendido. Sin embargo, el hecho de emplear dos lenguajes diferentes, uno para la consulta (XQuery) y otro para la actualización de la información (XUpdate) no parece la solución más acertada, teniendo en cuenta además que el formato del segundo es XML y el del primero no. De ahí que sea preferible utilizar, como sucede en la base de datos Tamino, una extensión no estandarizada de XQuery para realizar las actualizaciones.

En la Tabla 4 se muestra una comparativa entre los lenguajes de consulta y actualización que utiliza cada base de datos.

Gestor de base de datos	Lenguaje de consulta	Lenguaje de actualización
Tamino	XQuery, X–Query (XPath extendido)	extensión de XQuery
X-Hive/DB	XQuery, XPath	XUpdate
dbXML	XPath	XUpdate
eXist	XQuery, XPath	XUpdate
Xindice	XPath	XUpdate

Table 4. Lenguajes empleados por bases de datos nativas XML

4 Conclusiones y una nueva propuesta

Como puede observarse, no existe una herramienta que permita consultar simultáneamente fuentes de datos XML y no XML. En la actualidad, la única forma de conseguirlo sería emplear una única fuente que permitiese tanto el almacenamiento relacional como el XML. Sin embargo, la posible migración de la información podría ser una tarea mucho más costosa que emplear otro tipo de solución.

Para solucionar estos inconvenientes, se propone una herramienta, en la que estamos investigando, que permita realizar consultas a cada una de las fuentes de información analizadas, empleando para ello el lenguaje nativo de cada una. Estas consultas podrían estar o no parametrizadas. Debería también poder transformar a una representación XML el resultado de consultas a bases de datos relacionales permitiendo lógicamente las vistas NF^2. Al mismo tiempo y puesto que el intercambio de información exige que ésta cumpla con una determinada estructura o DTD, también sería conveniente que la herramienta permitiese al usuario especificar la estructura del XML resultante a fin de no tener que realizar transformaciones posteriores. Sería conveniente finalmente, que las peticiones que formulase el usuario a la herramienta estuviesen en formato XML, con el objetivo de poder implementar ésta como un servicio web y, de esta forma, conseguir que fuese el único punto de información de la empresa.

References

1. Laux A. and Martin L.: XUpdate Working Draft. Available at http://exist-db.org/xmldb/xupdate/xupdate-wd.html. (2000)
2. J. E. Funderburk, G. Kiernan, J. Shanmugasundaram, E. Shekita, C. Wei: XTABLES: Bridging relational technology and XML. IBM Systems Journal. (2002)
3. M. J. Carey, D. Florescu, Z.G. Ives, Y. Lu, J. Shanmugasundaram, E.J. Shekita, S. Subramanian: XPERANTO: Publishing Object-Relational Data as XML. IBM Research Report. (2001)
4. J. Shanmugasundaram, K. Tufte, C. Zhang, G. He, D.J. DeWitt, J.F. Naughton: Relational Databases for Querying XML Documents: Limitations and Opportunities. The VLDB Journal. (2001) 302–314
5. M. Fernández, W. Tan, D. Suciu: Silkroute: Trading between relations and XML. Proceedings of the Ninth International World Wide Web Conference. (2000)
6. M. Fernández, A. Morishima, D. Suciu, W.C. Tan: Publising relational data in XML: the SilkRoute approach. IEEE Data Engineering. (2001)

7. M. Fernández, Y. Kadiyska, A. Morishima, D. Suciu, W.C. Tan: SilkRoute: a framework for publising relational data in XML. ACM Transactions on Database Systems (TODS), 27(4) (2002)
8. A. Deutsch, M. Fernández, D. Florescu, A. Levy, D. Suciu: XML-QL: A Query Language for XML. In Proceedings of WWW The Query Language Workshop (QL) (1998)
9. A. Fermoso: XBD: Sistema de consulta basado en XML a bases de datos relacionales. PhD thesis, Facultad de Ingeniería E.S.I.D.E. Universidad de Deusto. (2003)
10. Oracle: Oracle 9i Release 2. XML Database Developers's Guide - Oracle XML DB. Oracle Corp. (2002)
11. C.M. Vittory, C.F. Dorneles, C.A. Heuser: Creating XML documents from relational data sources. In Proceedings of EC-WEB (Electronic Commerce and Web Technologies) (2001)
12. V. Braganholo: Updating Relational Databases through XML Views. Instituto de Informática. Univerdidade Federal Do Rio Grande do Sul (2002)
13. S. Pal, M. Fussell, I. Dolobowsky: XML Support in Microsoft SQL Server 2005. MSDN Library. Available at http://msdn.microsoft.com/xml/default.aspx?pull=/library/en-us/dnsql90/html/sql25xmlbp.asp (2004)
14. A. Conrad: A survey of Microsoft SQL Server 2000 XML Features. MSDN Library (2001)
15. IBM: IBM DB2 Universal Database. XML Extender Administration and Programming. Version 8. IBM Corp. (2002)
16. Software AG: Introducing Tamino. Tamino version 4.1.4. Software AG. (2003)
17. X-Hive Corporation: X-Hive/DB. Available at http://www.x-hive.com (2004)
18. The dbXML Group: dbXML. Available at http://www.dbxml.com/index.html (2004)
19. W. Meier: eXist. Available at http://exist.sourceforge.net (2004)
20. Apache Software Foundation: Xindice. Available at http://xml.apache.org/xindice/ (2004)
21. V. Turau: Making Legacy Data Accessible for XML Applications. Available at http://www.ti5.tu-harburg.de/Staff/Turau/pubs/legacy.pdf (1999)
22. J. Cheng and J. Xu: IBM DB2 XML Extender: an end-to-end solution for storing and retrieving XML documents. In Proceedings of ICDE'00 (2000)
23. IBM: IBM Net.Data for OS/2 Windows NT, and UNIX Administration and Programing Guide Version 7. IBM Corp. Available at http://www.ibm.com/software/netdata
24. Intelligent Systems Research: Merging ODBC Data into XML ODBC2XML. Available at http://www.intsysr.com/odbc2xml.htm (2003)
25. Apache group: Jakarta Proyect: DBTags Tag library. Available at http://jakarta.apache.org/taglibs/doc/dbtags-doc/index.html (2003)
26. R. Laddad: XML APIs for databases: blend the power of XML and databases using custom SAX and DOM APIs. Java World (2000)
27. M.A. Roth, H.F. Korth, A. Silberschatz: Extended algebra and calculus for nested relational databases. ACM Trans. Database Syst., 13(4):389–417 (1988)
28. P.C. Fischer, D. Van Gucht: Weak multivalued dependencies. In Proceedings of the 3rd ACM SIGACT-SIGMOD symposium on Principles of database systems (1984)
29. R. Elmasri, S. Navathe: Fundamentals of database systems. Addison Wesley (2002)
30. A. Silberschatz, H. Korth, S. Sudarshan: Database System Concepts. McGraw-Hill (1998)