# Acceso a Datos Relacionales Bajo un Entorno XML. Un Caso Práctico

Ana Fermoso García, Roberto Berjón Gallinas

Escuela Universitaria de Informática, Universidad Pontificia de Salamanca, Compañía 5, 37002 Salamanca, España
{afermosoga, rberjonga}@upsa.es

**Abstract.** En nuestros días y especialmente con la explosión de Internet, las organizaciones tienen que manejar multitud de datos y además datos de fuentes heterogéneas, como por ejemplo datos procedentes de las tradicionales bases de datos relacionales y otros procedentes de nuevas tipos de datos aparecidos con la Web, como los documentos HTML y XML. Las organizaciones tienen que trabajar con todos estos tipos de datos al mismo tiempo, por lo que se hacen necesarios sistemas que actúen como "mediadores" para facilitar la integración de esta información. En nuestra investigación nos centraremos en los datos XML, estándar de facto para el intercambio de información en la Web, y las bases de datos relacionales, que continúan siendo uno de los principales sistemas de almacenamiento. El propósito es que los datos necesarios en cada momento acaben en el mismo formato, XML en este caso al ser este el lenguaje de la Web, antes de manejarlos de forma conjunta, y para ello proponemos un nuevo sistema al que hemos denominado XBD. XBD es un sistema de acceso eficiente a datos relacionales bajo un entorno XML. XBD incluye un modelo de adaptación para que cualquier base de datos relacional pueda ser consultada en nuestro sistema, dos nuevos lenguajes para realizar las consultas y la herramienta software que implementa la funcionalidad de este sistema de consulta, y todo ello, tanto el interfaz de consulta, como los lenguajes, así como los resultados de las consultas, tendrán una apariencia XML.

## 1 Introducción

Internet esta provocando grandes cambios en las organizaciones y su gestión. Toda organización tiene que acabar migrando a la Web. Por otro lado, la sociedad actual es también la sociedad de la información y el conocimiento, la información es poder y los datos tienen que acabar convirtiéndose en "información", que es un concepto más amplio. Cuando un dato se convierte en información quiere decir que puede servirnos para la toma de decisiones del negocio, por ejemplo. Además la cantidad de información que se maneja ha sufrido un gran incremento en cuanto a su volumen y heterogeneidad, los datos pueden venir de fuentes muy diferentes.

Los nuevos entornos de trabajo han hecho aparecer nuevos tipos y formatos de datos, como los utilizados para la publicación de información en la Web, HTML o XML [8]. Sin embargo, en paralelo se tiene que seguir trabajando con los datos almacenados en las tradicionales bases de datos, como las bases de datos relacionales.

En conclusión, en la organización se tiene que trabajar al mismo tiempo con datos procedentes de fuentes heterogéneas más o menos estructuradas, y por tanto será necesario en muchos casos, algún método intermedio que sirva como mediador y que permita integrar estos datos con diferentes formatos para trabajar de forma conjunta con ellos.

En cuanto a los nuevos tipos de datos creados como respuesta a las necesidades de la Web, destaca XML como estándar de facto para la representación e intercambio de información en este entorno. Una de las consecuencias de trabajar en este medio suele ser la necesidad de intercambiar información con otros sistemas y para ello el formato XML se ha convertido en el formato más usado. Incluso están apareciendo sistemas de almacenamiento específico para este tipo de datos, que van más allá de los documentos XML tratados de forma independiente, nos referimos a las bases de datos nativas XML.

El mundo de las bases de datos también se ha visto afectado por los nuevos cambios. Por un lado, las tradicionales bases de datos relacionales siguen almacenando la mayor parte de información de la organización, pero en las últimas versiones de los sistemas de gestión de base de datos, se han visto obligados también a adaptarse a las nuevas necesidades, por ejemplo permitiendo también tratar y almacenar información en formato XML, aunque el tema del almacenamiento de información XML en bases de datos relacionales quede fuera del ámbito de este trabajo.

También en relación con las bases de datos han aparecido nuevos tipos como los data warehouse, que se han convertido en sistemas de almacenamiento de datos de fuentes heterogéneas, desde los que luego se analizan esos datos para convertirlos en información y conocimiento y así ayudar a la toma de decisiones de la organización. En estos nuevos sistemas se harán también necesarios métodos para permitir integrar estos datos como paso previo  a su análisis.

En nuestro trabajo, demostrada la importancia de XML en el medio Web por un lado, y la gran cantidad de información que continua siendo almacenada en las bases de datos relacionales, estudiaremos como podemos tratar de forma conjunta datos procedentes de ambos entornos. Si los datos procedentes de uno u otro entorno pudiesen estar en el mismo formato común, XML al ser éste el lenguaje de la Web, sería más fácil la integración de los mismos, tanto si se necesitan para el intercambio de información a través de la red, como si se utilizan para su posterior almacenamiento y tratamiento dentro de un data warehouse, como acabamos de comentar.

El objeto de nuestra investigación será precisamente, proporcionar un entorno de acceso común para datos en formato XML y relacionales al que hemos denominado XBD (acceso a Bases de Datos con XML) . Para el usuario resultará un sistema transparente en cuanto a que accederá a datos XML y relacionales de una forma muy similar, con lo que para él es como si toda la información estuviera en formato XML y sólo estuviese consultando datos de este tipo. Se trata por tanto de acceder a datos relacionales bajo un sistema con apariencia de consulta a datos XML. El resultado obtenido tras las consultas a la base de datos relacional, también sería XML.

Según lo expuesto, una vez vistas en esta introducción las necesidades que las nuevas tecnologías plantean a las organizaciones, queda con ello justificado la utilidad del nuevo entorno de consulta a bases de datos relacionales que proponemos.

En el siguiente apartado se tratará de hacer un estudio de la situación actual de otros sistemas que también tienen relación con el tema de las bases de datos y XML, permitiendo obtener datos XML de fuentes relacionales. A continuación, y una vez analizados las limitaciones de los sistemas estudiados con relación a nuestros objetivos, ya se pasaría por fin a describir en detalle las características de nuestro nuevo sistema de consulta XBD que hemos diseñado como respuesta a las nuevas necesidades de gestión de datos heterogéneos. La descripción del sistema incluirá su arquitectura, el modelo de adaptación que se propone para que cualquier base de datos relacional pueda ser consultada en el entorno XBD, los lenguajes de consulta que en él se utilizan y el procesamiento en general de las consultas en este sistema. Finalmente se demostrarán sus ventajas y las pruebas que con él se han realizado y por último se expondrían las nuevas líneas de investigación en las que estamos ya trabajando y que siguen relacionadas con el área de la integración entre datos relacionales y XML, pero tratada desde una nueva perspectiva.

## 2 Obtención de datos XML de Fuentes Relacionales

El propósito de nuestra investigación es ayudar a integrar bases de datos relacionales con datos XML, obteniendo datos XML a partir de información almacenada en base de datos relacionales. Una vez logrado que tanto la información procedente de fuentes XML como de fuentes relacionales esté en el mismo formato, en nuestro caso XML, resultará más fácil trabajar de forma conjunta con ella, tal como se hace necesario en muchos sistemas actuales.

Las últimas versiones de los Sistemas de Gestión de Base de Datos (SGBD) más utilizados como Oracle [6][7], Microsoft SQL Server [5] o IBM DB2 [4], han tenido que adaptarse a los nuevos tipos de información permitiendo trabajar de una forma más o menos directa con datos XML.

En consecuencia, estos SGBD ya permiten almacenar datos XML junto a sus tradicionales datos relacionales, definiendo nuevos tipos de datos como XMLType en Oracle y SQL Server, o XML Column y XML Collection en DB2. Una vez almacenada la información XML en la base de datos habrá que disponer también de herramientas para acceder a ella. Si la información en XML almacenada en la base de datos tiene una estructura más o menos fija (documento XML centrado en los datos) y reconocida por la base de datos a través de una especie de esquema, conocido por ejemplo como DAD (Definición de Acceso a Datos) en IBM DB2, se podrá acceder en detalle a la información XML almacenada. Esto se consigue en la mayor parte de los SGBD ampliando el lenguaje de consulta estándar SQL con construcciones de XPath embebidas en dichas consultas SQL y dirigidas a los datos XML. Si no hay una estructura reconocida de la información XML almacenada, es decir, se trata de documentos XML centrados en el documento sin una estructura prefijada, no en los datos, los SGBD permitirán acceder a todo el documento XML en su conjunto, al documento XML completo, no a una parte en particular del mismo.

Sin embargo, todos los aspectos de las nuevas versiones de los SGBD que acabamos de comentar en relación a XML se refieren al almacenamiento de este tipo de información en las bases de datos relacionales y nosotros estamos más interesados no en el almacenamiento, sino en la obtención de datos en formato XML generados a partir de información que no está en este formato. En este sentido también los SGBD están proporcionando nuevas herramientas que permiten realizar consultas a las bases de datos en el tradicional lenguaje de consultas de datos relacionales SQL, y obtiene los resultados de la consulta en XML.

Los mismos resultados que con estas últimas herramientas se pueden conseguir con las páginas activas como las JSP (Java Server Pages), ASP (Active Server Pages) o PHP. En todas ellas es posible incluir consultas a base de datos en SQL y obtener los resultados de dichas consultas en XML. Estos resultados se añadirán al resto del contenido de la página Web a la hora de mostrar su información final.

En los dos casos anteriores, tanto los de la herramientas de los SGBD que retornan los resultados en XML como en las páginas activas, la consulta se realiza utilizando un lenguaje propio del entorno de base de datos como SQL. Sin embargo, nuestro objetivo es que para facilitar la integración de datos relacionales con datos XML y que el usuario no tenga que preocuparse ni del formato, ni del tipo de fuente al que accede, y ni siquiera del lenguaje de consulta, sino que para él es como si todo lo que consulta fuese información en XML, por esta razón el lenguaje de consulta también debería estar basado en XML, no ser un lenguaje del entorno de base de datos. Esto por  tanto representa una desventaja de las herramientas anteriores si tenemos en cuenta nuestras metas.

En este sentido, existen ya algunos sistemas que permiten la consulta a bases de datos relacionales obteniendo los resultados en XML, pero además usando lenguajes de consulta propios del entorno XMLm y sin que los datos originales de la base de datos tengan que sufrir ningún tipo de transformación ni traducción a ningún formato. Dentro de estos sistemas destacan SilkRoute [1] y XTABLES [2].

En ambos trabajos se propone la creación de una vista virtual en XML de los datos de la base de datos y sobre esta vista se realizan las consultas. La vista se considera virtual porque en realidad los datos de la base de datos no se transforman a XML, se le presentan al usuario con apariencia XML, pero en realidad ni se traducen, ni se almacenan en XML. En los dos sistemas se realiza una consulta inicial a partir de la cual se genera esa vista virtual XML, que contendrá a los datos de la base de datos afectados por la consulta y el resto de consultas tendrán que hacerse sobre los datos de esa vista virtual y en un lenguaje de consulta basado en XML, es decir, se condiciona al resto de consultas a los resultados obtenidos en esa consulta inicial.

En XML la vista XML virtual recibe el nombre de “default view” y como único lenguaje de consulta, tanto para la consulta inicial como para el resto, se utiliza XQuery. En SilkRoute la vista virtual obtenida tras la primera consulta recibe el nombre de “query view” y para obtenerla se utiliza un lenguaje propio de este sistema denominado RXL. El resto de consultas se realizarán sobre los datos de la vista virtual pero utilizando otro lenguaje de consulta denominado XML-QL.

El fin último de estos dos sistemas coinciden con el nuestro, se consulta una base de datos relacional obteniendo los resultados de la consulta en formato XML y además utilizando también lenguajes basados en XML para hacer las consultas y sin necesidad de realizar ninguna transformación de los datos a consultar. El inconveniente sin embargo de estos dos sistemas, es que en ambos existe una consulta inicial que condiciona al resto de consultas, tal que sólo podrán ser consultados los datos afectados por esta primera consulta. En el sistema que nosotros proponemos se consiguen los mismos resultados, consultar datos relacionales con lenguajes de consulta basados en XML y obtener también los resultados en XML, pero de forma que en cualquier momento se pueda consultar cualquier parte o dato de la base de datos, no existirá una consulta inicial que condicione el resto de consultas, y esta es otra de sus ventajas. Como luego veremos en este nuevo sistema al que denominamos XBD ya no existe ninguna vista virtual porque el proceso de adaptación y consulta de cualquier base de datos relacional bajo este entorno es distinto a la idea de la "vista virtual XML", en su lugar se crea el "Esquema XML De Base de Datos" que es un documento XML con un significado distinto al de la vista virtual.

Analizadas las características de estos sistemas que permiten obtener datos XML de fuentes relacionales y expuestas sus limitaciones e inconvenientes de acuerdo a nuestros objetivos, a continuación exponemos las características del nuevo sistema que nosotros proponemos denominado XBD, y después demostraremos sus ventajas y bondades con relación a los sistemas analizados y a nuestras metas.

## 3. El nuevo Sistema de Consulta XBD

La idea fundamental del nuevo sistema XBD (XML para Bases de Datos) que nosotros proponemos, y del que podemos encontrar un estudio más detallado en [1], trata de solventar los principales problemas de los sistemas analizados en relación con el tratamiento de datos relacionales y XML en el entorno Web.

En nuestro sistema será posible consultar datos relacionales bajo un entorno con apariencia XML, de hecho, para el usuario la apariencia es prácticamente igual a la consulta a datos XML incluido el lenguaje de consulta, como luego veremos, y los resultados obtenidos, que también serán XML.

Para poder hacer efectivas estas consultas además, la gran ventaja es que no se necesita hacer ninguna conversión de información. No es necesario traducir los datos relacionales a XML, y ni siquiera será necesario mostrar estos datos como una vista XML.

Para conseguir estos objetivos hay que afrontar dos aspectos fundamentales, por una parte, cómo se van a consultar los datos relacionales, con qué lenguaje, y en segundo lugar, cuál debe ser la arquitectura interna del sistema que permita que hechas las consultas a la base de datos en el lenguaje del sistema, los resultados obtenidos en formato ya XML, sean los correctos.

Para permitir consultar una base de datos desde el entorno XBD serán necesarios dos procesos básicos. El primero permitirá adaptar la base de datos de modo que esta sea consultada en nuestro sistema. Para ello, ahora no se obtendrá una vista virtual en XML como en los casos anteriores, sino que se propondrá otra solución distinta de modo que como resultado del proceso se obtendrá el documento en formato XML que denominamos "Esquema XML de BD" cuyo significado explicaremos luego. Una vez adaptada la base de datos, el siguiente proceso ya nos permitirá que sea consultada. Para ello será necesario especificar por un lado, las características del lenguaje de consulta y por otro, el proceso interno que permita que estas consultas se ejecuten sobre la base de datos y se obtengan sus resultados en XML. De esta forma, internamente XBD estará organizado en dos subsistemas principales que luego detallaremos al describir su arquitectura, el subsistema de adaptación y el de consulta.

En primer lugar analizaremos el tema del lenguaje a utilizar para realizar las consultas a la base de datos y luego ya el de la arquitectura del sistema con los dos subsistemas que lo forman.

## 4. Lenguajes de Consulta en el Sistema XBD

Si se desea que todo el sistema de consulta, incluido el lenguaje que se utiliza para dichas consultas, tenga una apariencia XML, tenemos dos alternativas: o crear un nuevo lenguaje de consulta basado en XML o reutilizar alguno de los ya existentes adaptándolo a nuestras necesidades.

Como deseamos que para el usuario resulte un sistema transparente y que de hecho para él es como si consultase información XML, la primera solución quedaría descartada porque se trataría de que el usuario no tuviese que emplear apenas tiempo de aprendizaje utilizando algún lenguaje que ya conozca del entorno XML.

En cuanto a lenguajes de consulta del entorno XML, el lenguaje de consulta por excelencia es XQuery [12], sin embargo XSL [11], aunque realmente es un lenguaje más de presentación que de consulta, indirectamente también nos va a permitir consultar información. En definitiva, en nuestro sistema se van a utilizar estos dos lenguajes, previa adaptación de los mismos a la consulta de base de datos, lo que implicará añadir alguna modificación, pero muy pequeña, a la sintaxis de las versiones estándar de estos lenguajes.

A los lenguajes obtenidos de esta adaptación, que son los que se van a usar para hacer las consultas en XBD les denominaremos *XQuery Adaptado* y *XSL Adaptado*. Su sintaxis es muy semejante a la de los lenguajes XQuery y XSL estándar, la diferencia fundamental radica en el significado que ahora toman las construcciones de estos lenguajes al utilizarlas para la consulta a base de datos. Cambia la semántica, pero la sintaxis es prácticamente la misma.

A continuación explicamos las características fundamentales de ambos lenguajes.

### 4.1 XSL Adaptado

Para adaptar el lenguaje XSL a nuestro sistema XBD sólo necesitamos algunas de la construcciones del lenguaje XSL estándar: `stylesheet`, `apply-templates`, `template`, `value-of`, `if`, `for-each`, `when` y `short`. A todas estas construcciones se les añade el prefijo “`xsl_adapt`” para señalar que forman parte de este nuevo lenguaje y que tienen un significado especial respecto a la versión estándar.

En relación con la consulta a bases de datos y teniendo en cuenta que al final, como resultado de la consulta en XSL adaptado, se obtiene una sentencia SELECT de SQL de consulta a base de datos, la equivalencia o significado de las construcciones en XSL para obtener la SELECT de SQL equivalente, sería la siguiente:

- *xsl_adapt:stylesheet:*

Contiene información acerca del fichero que contiene el *esquema XML de base de datos*, que es un documento XML obtenido a partir del proceso de adaptación de la base de datos al sistema XBD. Este fichero contiene información acerca de la estructura de la base de datos: sus entidades, atributos y relaciones. El nombre de este fichero resultado del proceso de adaptación, aparece como valor del atributo "`Schema_db`" de esta construcción.

Además esta construcción también contiene información sobre como conectarse a la base de datos a la que se refiere la consulta.

- *xsl:adapt:apply-template / xsl_adapt:template*
  Cada `template` equivale a una sentencia SELECT de SQL. Ambas construcciones actúan de forma combinada. En el `apply-template` determinamos la tabla principal de la consulta y dentro del `template` asociado aparecerá el resto del contenido de la sentencia SELECT.
- *xsl_adapt:value-of*
  Muestra los campos consultados de la base de datos. Estos campos aparecerán en la sentencia SELECT cuando la consulta se traduce a SQL.
- *xsl_adapt:if*
  Añade una condición a la consulta. Esta construcción se traducirá a una cláusula WHERE en la sentencia SELECT de SQL asociada.
- *xsl_adapt:for-each*
  Permite agrupar los resultados de la consulta. Se traduce a la cláusula GROUP-BY de la SELECT de SQL.
- *xsl_adapt:when*
  Añade una condición a la cláusula GROUP-BY de agrupamiento. Equivale a la cláusula HAVING de SQL.
- *xsl_adapt:sort*
  Permite clasificar la consulta. Equivale a la cláusula ORDER BY de SQL.

Todas estas construcciones tienen diferentes atributos: `select, match, test,` ..., en los que pueden aparecer expresiones de camino que utilizan la sintaxis de XPath [9], a través de las cuales referenciaremos a tablas y campos de la base de datos.

Además será posible anidar sentencias SELECT, definiendo un "`template`" dentro de otro, es decir, insertando una sentencia "`template`" dentro de un "`apply-template`", pues cada "`template`" equivale a una sentencia SELECT.

A la hora de traducir las consultas anidadas a la correspondiente sentencia de consulta en SQL se pueden dar dos posibilidades. Si el "`apply-template`" aparece aislado, se traducirá a una sentencia SELECT anidada en la cláusula FROM de la sentencia SELECT inicial, es decir, como una subconsulta de SQL. Si la construcción "`apply-template`" tuviera a su vez anidada una construcción "`value-of`", entonces la anidación se traduciría a un "CURSOR" dentro de la sentencia SELECT equivalente. A los resultados asociados a este cursor habría que darles un nombre en el resultado de la consulta en XML para identificarlos. Este nombre estará formado por la unión del nombre de la tabla principal de la consulta anidada y la tabla principal de la sentencia SELECT inicial.

Para entender mejor el modelo de traducción descrito de cualquier consulta a base de datos en el lenguaje XSL Adaptado, a su equivalente sentencia SELECT de consulta en SQL, a continuación detallamos el Data Type Definiyion (DTD) que contendrá implícitas las reglas de este proceso de traducción.

```
<!ELEMENT xsl_adapt:if ( (xsl_adapt:value-of)+,
(xsl_adapt:apply-templates)?)>
          <!ATTLIST xsl_adapt:if
             test CDATA #REQUIRED>
<!ELEMENT xsl_adapt:apply-templates (xsl_adapt:value-
of)*>
          <!ATTLIST xsl_adapt:apply-templates
             select CDATA #REQUIRED>
<!ELEMENT xsl_adapt:value-of EMPTY>
          <!ATTLIST xsl_adapt:value-of
             select CDATA #REQUIRED>
<!ELEMENT xsl_adapt:for-each ((xsl_adapt:when)?)>
          <!ATTLIST xsl_adapt:for-each
             select CDATA #REQUIRED>
<!ELEMENT xsl_adapt:when EMPTY>
          <!ATTLIST xsl_adapt:when
             test CDATA #REQUIRED>
<!ELEMENT xsl_adapt:sort EMPTY>
          <!ATTLIST xsl_adapt:sort
             select CDATA #REQUIRED>
```

Según este DTD y deduciendo a partir de él las reglas de traducción del lenguaje XSL Adaptado a SQL, por ejemplo la sentencia SQL: “`SELECT nom_prod FROM producto`”, en XSL adaptado equivaldría a la siguiente:

```
<?xml versión=”1.”>
<!DOCTYPE xsl_adapt:stylesheet system “xsl_adapt.dtd”>
<xsl_adapt:stylesheet
version = “1.0”
schema_db= “schema_db_produc.xml”>
  <xsl_adapt:template match=/”>
     <xsl_adapt:apply_template
          select=”Producto”>
  </xsl_adapt:template>
  <xsl_adapt:template
          match=//Producto”>
     <xsl_adapt:value_of
          select=”nom_prod”>
  </xsl_adapt:template>
</xsl:stylesheet>
```

### 4.2 XQuery Adaptado

Para adaptar el lenguaje XQuery a la consulta a base de datos, sólo se necesitan las sentencias FOR-LET-WHERE-RETURN (FLWR) de este lenguaje. La equivalencia con las cláusulas de la sentencia SELECT de SQL sería la siguiente:

- FROM: determina las tablas de la base de datos a consultar.
- RETURN: especifica los atributos del resultado de la consulta
- WHERE: añade una condición, cláusula WHERE, a la consulta.
- LET: permite agrupar los resultados de la consulta. Equivale al GROUP-BY de SQL.

Las expresiones de camino con XPath que aparecen en las sentencias FLWR, van a determinar las tablas y campos de la consulta.

Según esto, la sentencia del ejemplo anterior "`SELECT nom_prod FROM producto`", en XQuery Adaptado, tendría la siguiente sintaxis.

```
FOR $p IN document("schema_db_produc.xml")
    RETURN Producto/nom_prod
```

## 5. Arquitectura Sistema XBD

El sistema XBD implementa dos tareas principales. La primera permitirá adaptar la base de datos a consultar, para que de hecho pueda ser consultada en el sistema XBD. La segunda, una vez adaptada la base de datos, permitirá consultarla utilizando cualquiera de los lenguajes del apartado anterior.

Según esto la arquitectura del sistema tendría dos componentes principales, el de adaptación y el de consulta. Esta arquitectura se representa en la Figura 1. En ella podemos observar que el documento "Esquema XML de BD" se obtiene como resultado del proceso de adaptación y se utiliza como entrada para el subsistema de consulta.

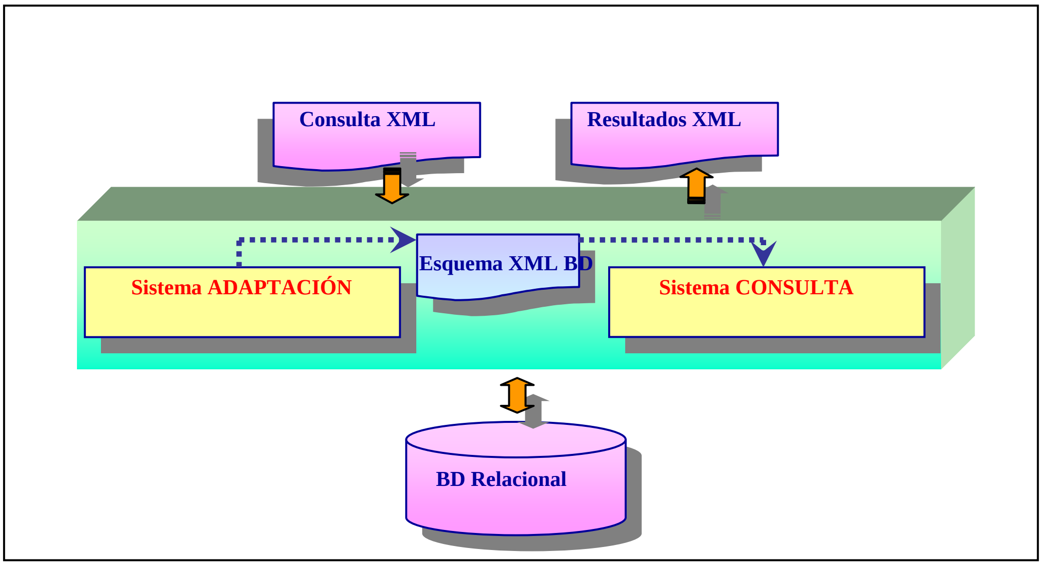


Figura 1. Arquitectura Sistema de Consulta XBD

## 5.1 Adaptación de una base de datos relacional a XBD

Esta tarea permite que cualquier base de datos relacional sea consultada en el sistema XBD. Para conseguirlo es necesario obtener un documento XML especial denominado *Esquema XML de Base de Datos* (Esquema XML de BD) que contendrá información acerca de la estructura de la base de datos: tablas, relaciones y campos. Este documento se utilizará como entrada del siguiente componente de XBD, el que se encarga de hacer las consultas.

La apariencia del "Esquema XML de BD" es similar a la de un Schema XML clásico [10].

En XML hay dos tipos de esquemas principales, el formato XSD del W3C que es el más utilizado, y el formato XDR de Microsoft que fue el primero. En el sistema XBD se podrá optar por utilizar un formato similar a uno u otro, para crear el documento resultado del proceso de adaptación, el Esquema XML de BD.

Los elementos de estos documentos dependen del formato utilizado. En XDR los principales elementos son "`element`" y "`ElementType`". En XSD, "`SimpleType`", "`element`", "`ComplexType`" y "`sequence`". Estos elementos contendrán información acerca de la base de datos: tablas, relaciones con otras tablas y cardinalidad, campos de cada tabla con sus tipos de datos y si dichos campos son o no clave.

Si usamos el formato XSD, con `SimpleType` definimos cada campo o columna de la base de datos junto al tipo de datos, a través de los atributos "`name`" y "`type`", de estos campos. Cada tabla se describe con un elemento `ComplexType`. Dentro de cada `Complextype` existirá un elemento "`sequence`" a través del que definiremos la relación y cardinalidad de la tabla con el resto de tablas del sistema con las que esta relacionado, y varios elementos “`element`” anidados dentro de “`sequence`” para definir cada uno de los campos que forman la tabla, junto a información sobre si ese campo es clave primaria de dicha tabla, clave secundaria cuando sirve para relacionar a la tabla con otra, o ninguna de las anteriores.

En el formato XDR, con `element` se define cada campo de cada tabla y con `ElementType` se define cada tabla. Anidado dentro de cada `ElementType`, se define el contenido de cada tabla, campos que lo forman y otras tablas con las que se relaciona junto a su cardinalidad.

Para entender mejor la estructura de cualquier Esquema XML de Base de Datos en formato XSD y reflejar lo que acabamos de comentar sobre cómo este esquema describe internamente la estructura de la base de datos, a continuación exponemos el DTD que deberían seguir los documentos de este tipo.

```
<!ELEMENT Schema_bd (conexion, (SimpleType)+,
(ComplexType)+)>
<!ELEMENT conexion EMPTY>
         <!ATTLIST conexion
type (basedatos|docxml) "basedatos"
           owner CDATA #REQUIRED
           cad_conex CDATA #REQUIRED>
<!ELEMENT SimpleType EMPTY>
         <!ATTLIST SimpleType
           name CDATA #REQUIRED
type CDATA #REQUIRED>
<!ELEMENT ComplexType ((sequence)+)>
         <!ATTLIST ComplexType
           name CDATA #REQUIRED>
<!ELEMENT sequence ((element)+)>
         <!ATTLIST sequence
           minOccurs CDATA #IMPLIED
maxOccurs (1|unbounded) "unbounded">
<!ELEMENT element EMPTY>
         <!ATTLIST element
           type CDATA #REQUIRED
           pk CDATA #IMPLIED
           fk CDATA #IMPLIED
           ref CDATA #IMPLIED>
```

De la misma forma exponemos también el DTD que se debería seguir si se utiliza el formato de Esquema XML de Base de Datos más parecido al XDR de Microsoft.

```
<!ELEMENT Schema_bd (conexion, (ElementType)+)>
<!ELEMENT conexion EMPTY>
<!ATTLIST conexion
type (basedatos|docxml) "basedatos"
owner CDATA #REQUIRED
cad_conex CDATA #REQUIRED>
<!ELEMENT ElementType (element)*>
         <!ATTLIST ElementType
           name CDATA #REQUIRED
content (columna|tabla) "columna"
           type CDATA #IMPLIED>
```

```
<!ELEMENT element EMPTY>
         <!ATTLIST element
          type CDATA #REQUIRED
          minOccurs CDATA #IMPLIED
          maxOccurs (1|*) "*"
          pk CDATA #IMPLIED
          fk CDATA #IMPLIED
          ref CDATA #IMPLIED >
```

Finalmente, y para visualizar la utilización de los Esquemas XML de Base de Datos, como ejemplo se da a continuación la definición de las tablas de PRODUCTOS y PEDIDOS. Además se sabe que la tabla de PRODUCTOS guarda una relación de 1 a N con PEDIDOS. Según esto el contenido del Esquema XML de BD en formato XSD para la tabla PRODUCTOS sería el que aparece después de la definición de las tablas.

```
PRODUCTOS (cod_prod, nom_prod, stock_prod)
PEDIDOS (cod_ped, fecha_ped, cod_prod, cant_ped)
```

```
<SympleType name="cod_prod" type="NUMBER">
<SympleType name="nom_prod" type="VARCHAR2">
<SympleType name="stock_prod" type="NUMBER">
<ComplexType name="PRODUCT0S">
 <sequence minOccurs="1"
   maxoccurs="1">
    <element type= "cod_prod"
     pk="true">
    <element type= "nom_prod>
    <element type= "stock_prod>
  </sequence>
  <sequence maxoccurs="unbounded">
    <element type="PEDIDOS">
  </sequence>
</ComplexType>
```

## 5.2 Consulta a una base de datos relacional en XBD

Esta tarea permite consultar la base de datos, previamente adaptada con la tarea anterior y gracias a la utilización del documento *Esquema XML de BD* obtenido en el proceso anterior.

Las consultas se podrán realizar en cualquiera de los lenguajes de consulta del sistema XBD: XSL o XQuery Adaptado.

Además, el componente que implementa esta tarea tiene una apariencia muy semejante a la consulta a un documento XML, en cuanto a que el interfaz de consulta presenta la información de la base de datos, su estructura, en un formato arborescente similar al árbol DOM de un documento XML.

En esta estructura arborescente cada tabla se representa como un nodo que parte del nodo raíz. Cada rama asociada a cada nodo a su vez contendrá a los campos que forman la tabla y otras tablas con las que se relaciona. Junto a esta estructura arborescente, el interfaz también suministra ayuda para editar la consulta, con información sobre la sintaxis de los lenguajes de consulta a utilizar.

Por último, además de la opción de edición de las consultas, el componente de consulta también suministra otras dos opciones, las de validación y ejecución de la consulta.

La validación permitirá comprobar si la consulta es correcta tanto desde el punto de vista de los datos consultados en la base de datos: tablas, campos y relaciones, como de la sintaxis de la consulta en relación a los lenguajes de consulta del sistema XBD.

Finalmente, la opción de ejecución nos permitirá ejecutar la consulta sobre la base de datos. Esto supone traducir internamente la consulta a una única sentencia SELECT de SQL, siguiendo las reglas o semántica de las construcciones del lenguaje de consulta, con relación a la equivalencia con las cláusulas de la sentencia SELECT de SQL vistas en el apartado 4 de este trabajo. La consulta SQL obtenida como resultado del proceso de traducción, se ejecutará sobre la base de datos y el propio DBMS que contiene a la base de datos ya dispone de herramientas para convertir los resultados de la base de datos obtenidos de la consulta, a XML. De esta forma, tal como pretendíamos, tanto el lenguaje de consulta como el formato del resultado será XML.

En la figura 2 se muestra la apariencia del interfaz de consulta. En él, tal como hemos comentado, podemos observar ala izquierda el árbol que muestra la estructura de la base de datos, y a la derecha la pantalla en la que editamos la consulta. En la parte inferior podemos utilizar la ayuda que se nos ofrece para la edición de la sintaxis de las construcciones de la consulta, y también podemos observar la ruta seleccionada en el árbol. Esta ruta utiliza la sintaxis de XPath y podemos utilizarla para añadirla como valor de alguno de los atributos de las construcciones de la consulta que se esta editando.

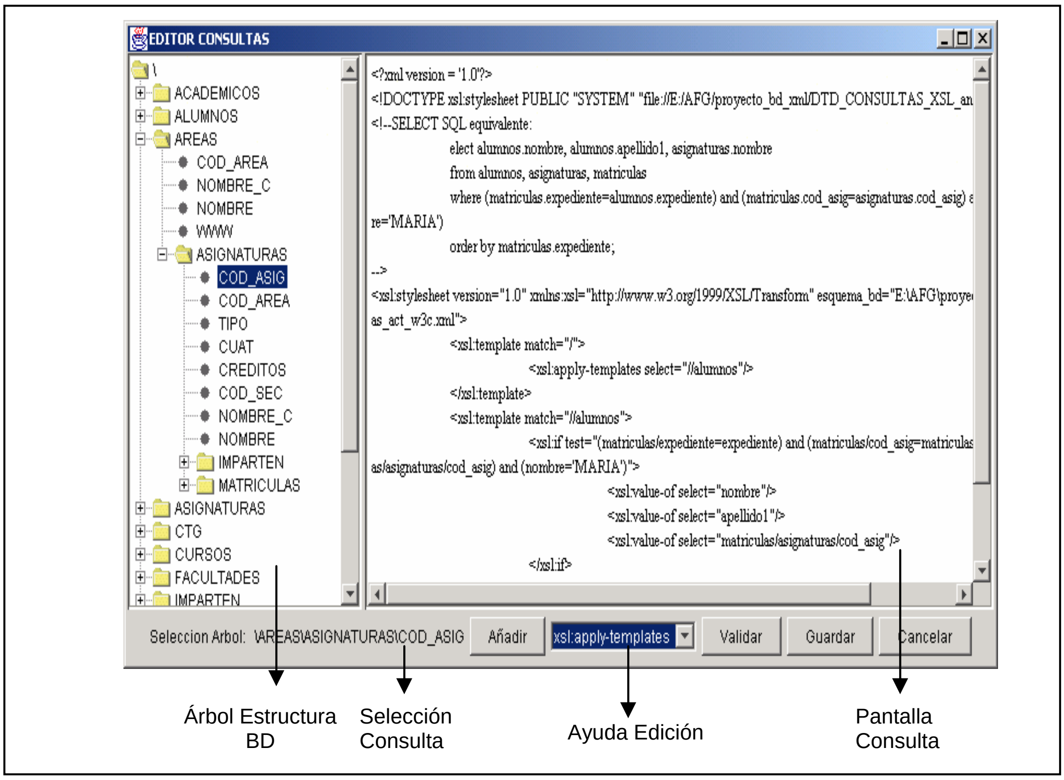


Figura 2. Interfaz de Consulta del Sistema XBD

## 6. Conclusiones y Líneas Futuras

Se demuestra, una vez descritas las características de nuestro sistema, que se consiguen los objetivos que nos habíamos propuesto. XBD permitirá consultas a base de datos con un formato y lenguajes semejantes a la consulta a documentos XML, para que al usuario le resulte un sistema transparente y considere que está consultando únicamente datos XML, tal como si estuviera en un entorno Web.

Para conseguir este efecto, no se necesita transformar la información de la base de datos a ningún formato, como ocurría en otros sistemas. Además se podrá consultar en cualquier momento cualquier campo de la base de datos, por que en otros sistemas esta consulta estaba restringida a la parte de la base de datos afectada por una consulta inicial.

Por último, el sistema resulta muy eficiente porque quien realiza la consulta final en SQL sobre la base de datos, obtenida tras el proceso detraducción de la consulta realizada en XSL o XQuery Adaptado, es el propio DBMS que contiene a dich abase de datos y sólo se añade el paso especial de traducción de la consulta a SQL, cuyo tiempo de ejecución es despreciable en comparación con la ejecución de la consulta completa.

Esta eficiencia ha quedado demostrada también al utilizar nuestro sistema dentro de una aplicación de nuestra Facultad cuya tarea es permitir la consulta de las calificaciones de nuestros alumnos y mostrarlas a través de la Web junto a otras informaciones académicas referidas al alumnado que existen ya en formato XML.

Una vez demostradas las bondades de XBD, sin embrago, también se podría mejorar. Nuestras líneas de investigación actuales giran entorno a esta idea. Se trataría de conseguir que en la misma consulta y al mismo tiempo se pudiesen consultar distintas fuentes, diferentes bases de datos y/o documentos XML, y además aprovechando la máxima potencia de cada una. Es decir, en vez de poder obtener sólo sentencias SQL en formato estándar, que también se pudiesen obtener sentencias de consulta que aprovechasen las ampliaciones de este lenguaje SQL propias de cada DBMS, e igual para los sistemas de consulta a documentos XML.

## Referencias


1. Fermoso, A, Berjón, R. Acceso a datos relacionales en entornos web. Un caso Práctico. Ed. Universidad Pontificia de Salamanca (2004)
2. Fernández, M., Kadiyska, Y., Morishima, A., Suciu, D., Tan., W.C., SilkRoute: a framework for publishing relational data in XML. ACM Transactions on Database Systems (TODS), Vol. 27, Nª 4, December (2002) pp. 438-493.
3. Funderburk, J. E., Kiernan, G., Shanmugasundaram, J. , Shekita, E., Wei, C., XTABLES: Bridging relational technology and XML. IBM Systems Journal, Vol. 41 , Nª 4 (2002)
4. IBM. IBM DB2 Universal Database. XML Extender Administration and Programming. Version 8. IBM Corporation. (2002)
5. Microsoft, SQL Server 2000. XML and Internet Support. Microsoft Corp. (2002)
6. Oracle, Oracle 9i Release 2. Database Concepts. Oracle Corp., March. (2002)
7. Oracle, Oracle 9i Release 2. XML Database Developers's Guide-Oracle XML DB. Oracle Corp., October 2002.
8. World Wide Web Consortium, Extensible Markup Languaje (XML). http://www.w3c.org/xml. (2004)
9. World Wide Web Consortium, XML Path Language (XPath). http://www.w3c.org/TR/xpath. (2004)
10. World Wide Web Consortium, XML Schema. http://www.w3c.org/2001/XMLSchema (2004)
11. World Wide Web Consortium, Extensible Style Language (XSL). http://www.w3c.org/Style/XSL (2004)
12. World Wide Web Consortium, XQuery: A Query Language for XML. http://www.w3c.org/TR/xquery (2004)