Bases de Datos nativas en XML

Las bases de datos nativas definen un modelo lógico para el documento XML (a diferencia de dicho modelo), además, almacena y recupera documentos de la misma manera que los XML.

Ejemplos de estos modelos son XPath, XML Infoset y modelos que implican DOM y SAX

Base de datos centrada en documentos

Todas las bases de datos relacionales son centradas en los Datos, pues lo que ellos almacenan en sus campos son datos simples más conocidos como datos atómicos. Una base de datos nativa en XML, no posee campos, ni almacena datos atómicos, lo que ella almacena son documentos XML, por lo tanto a este tipo de bases de datos se les denomina bases de datos centradas en documentos.

Características

Diversos productos brindan diferentes características para las bases de datos nativas en XML, pero generalmente tienen las siguientes características:

Procesamiento de datos

El procesamiento de datos en este tipo de bases de datos parecería ser algo muy beneficioso, pero realmente no es así, debido al formato jerárquico en el que está almacenada la información. Muchas bases de datos necesitan que el usuario recupere todo el documento XML, lo actualice con el XML API de su preferencia y posteriormente vuelva a almacenar el documento en el repositorio. Esto se debe a que aún no existe un lenguaje estándar que permita la actualización, inserción o eliminación de elementos de un documento XML. Existe un lenguaje que permite realizar actualizaciones en un documento XML pero aún no es un estándar y muchos gestores de este tipo de bases de datos no lo soportan, este lenguaje es Xupdate.

Almacenamiento

Por deducción lógica, una base de datos nativa en XML almacena la información en formato XML, pero esto es solamente una deducción lógica, pues este tipo de bases de datos tienen repositorios con un formato "tipo XML", como puede ser DOM o Infoset. En este mismo "repositorio" (paquete de archivos) se almacenan los índices que se generan por cada documento XML almacenado.

Búsquedas

Este tipo de bases de datos no utiliza SQL como lenguaje de consulta. En lugar de ello utilizan Xpath. Algunas bases de datos permiten seleccionar los elementos que deberán tener índices mientras que otras bases de datos indexan todo el contenido del documento. El problema que tienen las búsquedas en este tipo de bases de datos es que no permiten realizar búsquedas muy complicadas, como por ejemplo ordenamiento y cross join, debido a que Xpath no fue

creada realmente para búsquedas en bases de datos, sino simplemente para búsquedas en un solo documento.

Muchas bases de datos permiten realizar búsquedas utilizando la tecnología Full-Text Search, de esta manera se puede agilizar la búsqueda de datos en los documentos XML.

Áreas de aplicación

Solo existe un requisito para cualquier aplicación donde se requiera usar NXD: la aplicación debe usar XML. Después de esto no existe ninguna restricción o regla para no utilizar NXD en una aplicación. En general, cualquier NXD es una excelente herramienta para almacenar documentos orientados a datos (Ej. XHTML o DocBook), la información que tiene una compleja estructuración anidada profundamente y la información que esta semi-estructurada en la naturaleza.

Básicamente, si la información es representada por XML un NXD probablemente seria una buena solución. Un NXD puede almacenar cualquier tipo de información XML, pero probablemente no seria la mejor herramienta para algo como un sistema de la contabilidad donde los datos son muy bien definidos y rígidos.

Algunas áreas potenciales de aplicación podrían ser:

- Portales de información corporativa
- Información de catálogos
- BD en partes de manufactura
- Información medica
- BD personalizadas

Ejemplos

Una gran parte de usos necesita encontrar documentos enteros. Por ejemplo, un portal Web podría permitir a usuarios buscar todos los documentos sobre una empresa particular y un sistema de dirección podría permitir a usuarios encontrar todos los documentos que se relacionen con una cierta parte.

El modo menos complejo de buscar documentos es con búsquedas texto completas. En bases de datos nativas XML, estos son XML-aware. Es decir esto distingue entre el contenido (que es buscado) y el margen (que no es).

Búsquedas más complejas estructuralmente, que pueden preguntar el margen, el texto, o ambos.

XPath y XQuery son los ejemplos en lenguajes de búsquedas estructuradas; bases de datos nativas XML apoyan un número de lenguajes propietarios también.

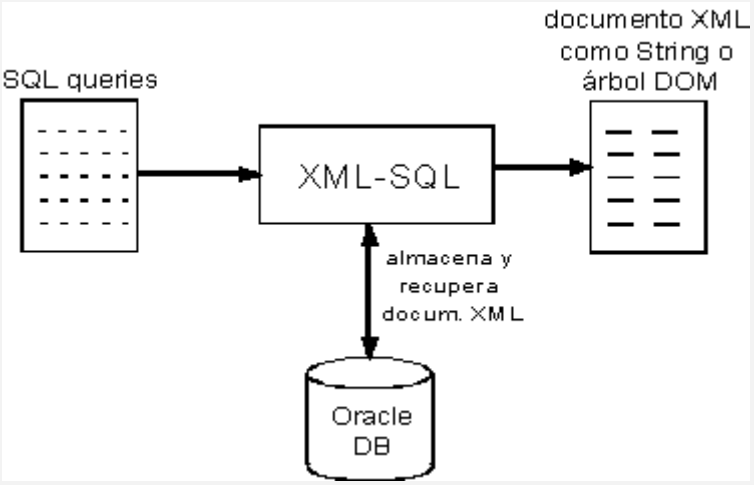
Soluciones para XML y Bases de Datos en Java

XML proporciona un marco estándar para intercambio de datos entre múltiples aplicaciones. Concretamente, XML ha sido asociado mayormente a aplicaciones en Internet debido a que el contenido de un documento XML es simplemente texto, lo cual facilita el intercambio de documentos a través de los distintos protocolos de Internet, sistemas operativos y "cortafuegos" (*firewalls*). Por otro lado, las aplicaciones que manejan datos, se benefician del uso de bases de datos (BD), para almacenar, consultar y modificar dichos datos. Es por ello que se están realizando esfuerzos para poder manejar documentos XML almacenados en una base de datos.

A continuación se describirán diferentes formas de acceso a bases de datos XML java

1) Oracle XML SQL Utility (XSU)

La utilidad XSU de Oracle consiste en un conjunto de clases Java que aceptan *queries* desde las aplicaciones, acceden a la BD mediante JDBC, y devuelven el resultado del *query* en formato XML. Como proceso complementario, XSU acepta un documento XML que se adapta al esquema de la BD y guarda los datos en función de dicho esquema, tal y como se muestra a continuación.



La estructura del documento XML resultante está basado en la estructura interna de la base de datos, de forma que:

- Las columnas de las tablas pasan a ser elementos del nivel más alto
- Los valores escalares pasan a ser elementos con contenido de sólo texto

La utilidad XML SQL genera:

- Una representación string del documento XML. Se suele usar esta representación si se debe devolver el documento XML como respuesta a una petición de un usuario
- Un árbol DOM en memoria de elementos. Se suele usar esta representación si es necesario utilizarlo para transformalo utilizando métodos DOM o modificar el XML de alguna manera

Mediante la utilidad XSU podemos realizar lo siguiente:

- Transformar datos recuperados desde tablas de una base de datos relacional en documentos XML
- Extraer datos de un documento XML e insertarlos en las columnas correspondientes de una tabla
- Extraer datos de un documento XML y utilizarlos para actualizar o borrar valores de columnas de una tabla

El mapeado por defecto de SQL a XML (y viceversa), consiste en incluir las filas devueltas por un *query* SQL en una etiqueta <ROWSET>, que constituye el elemento <ROWSET>. Est elemento es también el elemento raíz del documento XML generado. Dicho elemento <ROWSET> contiene uno o más elementos <ROW>. Cada uno de los elementos <ROW>

contiene los datos de una de las filas devueltas de la base de datos. Específicamente, cada elemento <ROW> contiene uno o más elementos cuyos nombre y contenido son los de las columnas de la base de datos especificados en la lista SELECT del *query* SQL.

Por ejemplo, consideremos la siguiente tabla denominada emp:

```
CREATE TABLE emp
{ EMPNO NUMBER,
  ENAME VARCHAR2 (20),
  JOB VARCHAR2 (20),
  MGR NUMBER,
  HIREDATE DATE,
  SAL NUMBER,
  DEPTNO NUMBER
};
```

XSU puede generar el siguiente documento especificando el *query* select * from emp:

```
<?xml version='1.0'?>
<ROWSET>
  <ROW num="1">
    <EMPNO>7369</EMPNO>
    <ENAME>Smith</ENAME>
    <JOB>CLERK</JOB>
    <MGR>7902</MGR>
    <HIREDATE>12/17/1980 0:0:0</HIREDATE>
    <SAL>800</sal>
    <DEPTNO>20</DEPTNO>
  <ROW>
  <!-- filas adicionales ... -->
</ROWSET>
};
```

Las clases Java que implementan la utilidad XSU son:

- oracle.xml.sql.query.OracleXMLQuery: API para generación de documentos XML
- oracle.xml.sql.dml.OracleXMLSave: API para guardar, insertar, actualizar y borrar

Los pasos a seguir para generar documentos XML utilizando la clase OracleXMLQuery, son los siguientes:

- Crear una conexión

```
//importar el driver de Oracle
import oracle.jdbc.driver.*;
//cargar el driver JDBC de Oracle
DriverManager.registerDriver(new oracle.jdbc.driver.OracleDriver());
```

```
//crear la conexión
Connection conn=
   DriverManager.getConnection
```

- ("jdbc:oracle:thin:@www.rvg.ua.es:1521:j2eebd","schema","passwd");Crear una instancia de OracleXMLQuery proporcionando un *string* SQL o un objeto ResultSet

```
import oracle.xml.sql.query.OracleSMLQuery;
OracleXMLQuery qry = new OracleXMLQuery (conn, "select * from emp*");
```

- Obtener el resultado como un árbol DOM o como un *string* XML

```
//para obtener el resultado como un string
String xmlString = qry.getXMLString;
//para obtener el resultado como un árbol DOM
org.w3c.DOM.Document domDos = qry.getXMLDOM();
```

Ejemplo. Obtención de un *String* a partir de una sentencia SQL.

Se trata de recuperar datos de una tabla mediante una sentencia SQL y obtener el resultado como un documento XML, representado por un String.

```
import oracle.jdbc.driver.*;
import oracle.xml.sql.query.OracleXMLQuery;
import java.lang.*;
import java.sql.*;


class testXMLSQL {

  public static void main(String[] argv)
  {

    try{
     // create the connection
     Connection conn  = getConnection("scott","tiger");

     // Create the query class.
     OracleXMLQuery qry = new OracleXMLQuery(conn, "select * from emp");

     // Get the XML string
     String str = qry.getXMLString();

     // Print the XML output
```

```
      System.out.println(" The XML output is:\n"+str);
      qry.close();
     }catch(SQLException e){
      System.out.println(e.toString());
     }
   }

   // Creación de la conexión dado un nombre de usuario y una clave
   private static Connection getConnection(String username, String password)
     throws SQLException
   {
       DriverManager.registerDriver(new oracle.jdbc.driver.OracleDriver());
      // Create the connection using the JDBC driver
       Connection conn =
        DriverManager.getConnection
          ("jdbc:oracle:thin:@www.rvg.ua.es:1521:j2eebd", username,password);
      return conn;
   }
       }
```

2) SAX y DOM

En líneas generales hay otras dos grandes formas de usar un lenguaje de programación como Java para leer o escribir archivos XML.

- DOM: significa Document Object Model (o Modelo del objeto documento). DOM en general almacena los archivos en memoria lo que es mucho más rápido y eficiente. DOM es un estándar y sus clases y métodos existen en muchos otros lenguajes.
- SAX: en algunos casos, los archivos muy grandes, pueden no caber en memoria. SAX proporciona otras clases y métodos distintos para ir procesando un archivo por partes. SAX significa Simple Access for XML, pero en general es un poco más complicado.

Otras soluciones: XQuery (lenguaje de programación especializado en consultas XML)

Para empezar **XQuery es un superconjunto de XPath** por lo que las expresiones básicas de XQuery también servirán en XQuery. La primera diferencia es que en XQuery tendremos que usar la función `doc` para cargar un archivo y luego navegar por él. Supongamos que tenemos un archivo de ejemplo como este:

```
<datos>
    <proveedores>
        <proveedor numprov="v1">
            <nombreprov>Smith</nombreprov>
            <estado>20</estado>
            <ciudad>Londres</ciudad>
        </proveedor>
        <proveedor numprov="v2">
            <nombreprov>Jones</nombreprov>
            <estado>10</estado>
            <ciudad>Paris</ciudad>
        </proveedor>
        <proveedor numprov="v3">
            <nombreprov>Blake</nombreprov>
            <estado>30</estado>
            <ciudad>Paris</ciudad>
        </proveedor>
        <proveedor numprov="v4">
            <nombreprov>Clarke</nombreprov>
            <estado>20</estado>
            <ciudad>Londres</ciudad>
        </proveedor>
        <proveedor numprov="v5">
            <nombreprov>Adams</nombreprov>
            <estado>30</estado>
            <ciudad>Atenas</ciudad>
        </proveedor>
    </proveedores>
    <partes>
        <parte numparte="p1">
            <nombreparte>Tuerca</nombreparte>
            <color>Rojo</color>
            <peso>12</peso>
            <ciudad>Londres</ciudad>
        </parte>
        <parte numparte="p2">
```

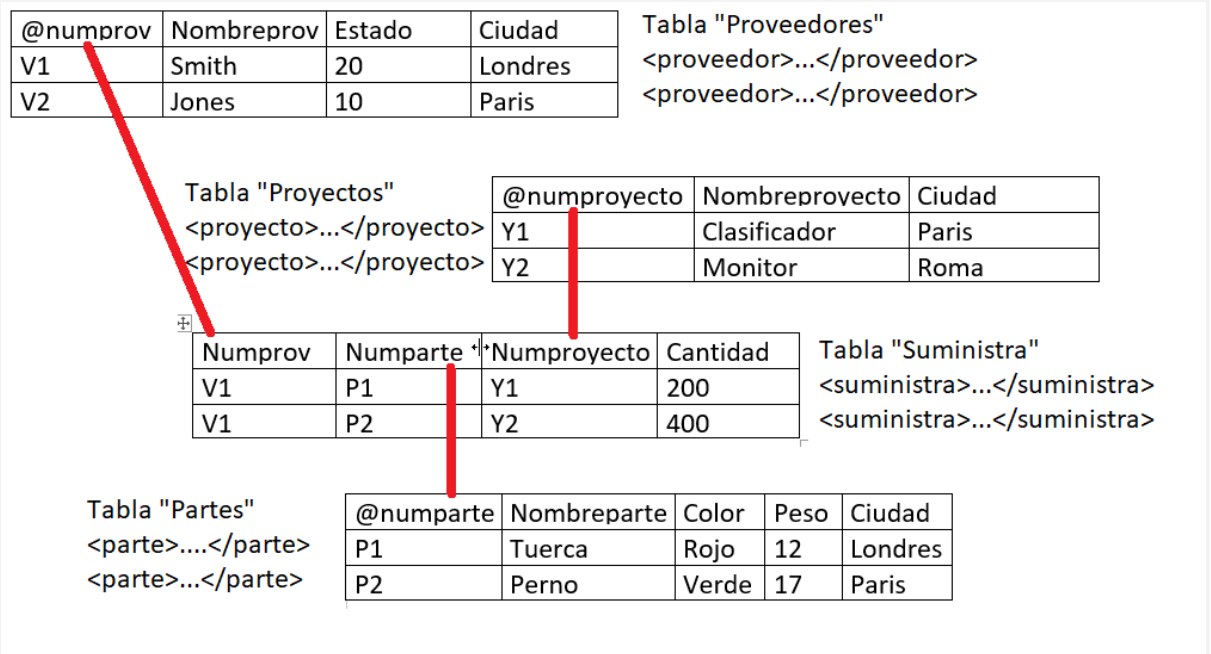
```
            <nombreparte>Perno</nombreparte>
            <color>Verde</color>
            <peso>17</peso>
            <ciudad>Paris</ciudad>
        </parte>
        <parte numparte="p3">
            <nombreparte>Tornillo</nombreparte>
            <color>Azul</color>
            <peso>17</peso>
            <ciudad>Roma</ciudad>
        </parte>
        <parte numparte="p4">
            <nombreparte>Tornillo</nombreparte>
            <color>Rojo</color>
            <peso>14</peso>
            <ciudad>Londres</ciudad>
        </parte>
        <parte numparte="p5">
            <nombreparte>Leva</nombreparte>
            <color>Azul</color>
            <peso>12</peso>
            <ciudad>Paris</ciudad>
        </parte>
        <parte numparte="p6">
            <nombreparte>Engranaje</nombreparte>
            <color>Rojo</color>
            <peso>19</peso>
            <ciudad>Londres</ciudad>
        </parte>
    </partes>
    <proyectos>
        <proyecto numproyecto="y1">
            <nombreproyecto>Clasificador</nombreproyecto>
            <ciudad>Paris</ciudad>
        </proyecto>
        <proyecto numproyecto="y2">
            <nombreproyecto>Monitor</nombreproyecto>
            <ciudad>Roma</ciudad>
        </proyecto>
        <proyecto numproyecto="y3">
            <nombreproyecto>OCR</nombreproyecto>
            <ciudad>Atenas</ciudad>
        </proyecto>
        <proyecto numproyecto="y4">
            <nombreproyecto>Consola</nombreproyecto>
            <ciudad>Atenas</ciudad>
        </proyecto>
        <proyecto numproyecto="y5">
            <nombreproyecto>RAID</nombreproyecto>
            <ciudad>Londres</ciudad>
        </proyecto>
        <proyecto numproyecto="y6">
            <nombreproyecto>EDS</nombreproyecto>
            <ciudad>Oslo</ciudad>
        </proyecto>
        <proyecto numproyecto="y7">
            <nombreproyecto>Cinta</nombreproyecto>
            <ciudad>Londres</ciudad>
        </proyecto>
```

```
    </proyectos>
    <suministros>
        <suministra>
            <numprov>v1</numprov>
            <numparte>p1</numparte>
            <numproyecto>y1</numproyecto>
            <cantidad>200</cantidad>
        </suministra>
        <suministra>
            <numprov>v1</numprov>
            <numparte>p1</numparte>
            <numproyecto>y4</numproyecto>
            <cantidad>700</cantidad>
        </suministra>
        <suministra>
            <numprov>v2</numprov>
            <numparte>p3</numparte>
            <numproyecto>y1</numproyecto>
            <cantidad>400</cantidad>
        </suministra>
        <suministra>
            <numprov>v2</numprov>
            <numparte>p3</numparte>
            <numproyecto>y2</numproyecto>
            <cantidad>200</cantidad>
        </suministra>
        <suministra>
            <numprov>v2</numprov>
            <numparte>p3</numparte>
            <numproyecto>y3</numproyecto>
            <cantidad>300</cantidad>
        </suministra>
        <suministra>
            <numprov>v2</numprov>
            <numparte>p3</numparte>
            <numproyecto>y4</numproyecto>
            <cantidad>500</cantidad>
        </suministra>
        <suministra>
            <numprov>v2</numprov>
            <numparte>p3</numparte>
            <numproyecto>y5</numproyecto>
            <cantidad>600</cantidad>
        </suministra>
        <suministra>
            <numprov>v2</numprov>
            <numparte>p3</numparte>
            <numproyecto>y6</numproyecto>
            <cantidad>400</cantidad>
        </suministra>
        <suministra>
            <numprov>v2</numprov>
            <numparte>p3</numparte>
            <numproyecto>y7</numproyecto>
            <cantidad>600</cantidad>
        </suministra>
        <suministra>
            <numprov>v2</numprov>
            <numparte>p5</numparte>
```

```
            <numproyecto>y2</numproyecto>
            <cantidad>100</cantidad>
        </suministra>
        <suministra>
            <numprov>v3</numprov>
            <numparte>p3</numparte>
            <numproyecto>y1</numproyecto>
            <cantidad>200</cantidad>
        </suministra>
        <suministra>
            <numprov>v3</numprov>
            <numparte>p4</numparte>
            <numproyecto>y2</numproyecto>
            <cantidad>500</cantidad>
        </suministra>
        <suministra>
            <numprov>v4</numprov>
            <numparte>p6</numparte>
            <numproyecto>y3</numproyecto>
            <cantidad>300</cantidad>
        </suministra>
        <suministra>
            <numprov>v4</numprov>
            <numparte>p6</numparte>
            <numproyecto>y7</numproyecto>
            <cantidad>300</cantidad>
        </suministra>
        <suministra>
            <numprov>v5</numprov>
            <numparte>p2</numparte>
            <numproyecto>y2</numproyecto>
            <cantidad>200</cantidad>
        </suministra>
        <suministra>
            <numprov>v5</numprov>
            <numparte>p2</numparte>
            <numproyecto>y4</numproyecto>
            <cantidad>100</cantidad>
        </suministra>
        <suministra>
            <numprov>v5</numprov>
            <numparte>p5</numparte>
            <numproyecto>y5</numproyecto>
            <cantidad>500</cantidad>
        </suministra>
        <suministra>
            <numprov>v5</numprov>
            <numparte>p6</numparte>
            <numproyecto>y2</numproyecto>
            <cantidad>200</cantidad>
        </suministra>
        <suministra>
            <numprov>v5</numprov>
            <numparte>p1</numparte>
            <numproyecto>y4</numproyecto>
            <cantidad>100</cantidad>
        </suministra>
        <suministra>
            <numprov>v5</numprov>
```

```
            <numparte>p3</numparte>
            <numproyecto>y4</numproyecto>
            <cantidad>200</cantidad>
        </suministra>
        <suministra>
            <numprov>v5</numprov>
            <numparte>p4</numparte>
            <numproyecto>y4</numproyecto>
            <cantidad>800</cantidad>
        </suministra>
        <suministra>
            <numprov>v5</numprov>
            <numparte>p5</numparte>
            <numproyecto>y4</numproyecto>
            <cantidad>400</cantidad>
        </suministra>
        <suministra>
            <numprov>v5</numprov>
            <numparte>p6</numparte>
            <numproyecto>y4</numproyecto>
            <cantidad>500</cantidad>
        </suministra>
    </suministros>
</datos>
```

En el esquema siguiente se muestra el mismo fichero en forma de tablas.



Estructura de tablas del XML de la base de datos de proveedores, partes y proyectos

En él podríamos ejecutar consultas como estas: * Recuperar todos los proveedores con `doc("datos.xml")/datos/proveedores` * Recuperar todos los datos con `doc("datos.xml")/datos/` * Recuperar todas las partes con `doc("datos.xml")/datos/partes`.

De hecho, podemos usar las mismas consultas con predicados XPath y así por ejemplo extraer los datos del proveedor cuyo numprov es “v1” con la consulta siguiente:

```
doc("datos.xml")/datos/proveedores/proveedor[@numprov='v1']
```

Que devuelve este resultado:

```
<proveedor numprov="v1">
    <nombreprov>Smith</nombreprov>
    <estado>20</estado>
    <ciudad>Londres</ciudad>
</proveedor>
```

Extraer los datos de partes cuyo color sea “Rojo”. La consulta XQuery sería:

```
doc("datos.xml")/datos/partes/parte[color='Rojo']
```

Y el resultado sería:

```
<parte numparte="p1">
    <nombreparte>Tuerca</nombreparte>
    <color>Rojo</color>
    <peso>12</peso>
    <ciudad>Londres</ciudad>
</parte>
<parte numparte="p4">
    <nombreparte>Tornillo</nombreparte>
    <color>Rojo</color>
    <peso>14</peso>
    <ciudad>Londres</ciudad>
</parte>
<parte numparte="p6">
    <nombreparte>Engranaje</nombreparte>
    <color>Rojo</color>
    <peso>19</peso>
    <ciudad>Londres</ciudad>
</parte>
```

En XQuery se pueden mezclar las marcas con el programa. Sin embargo, para poder distinguir lo que se tiene que ejecutar de lo que no, tendremos que encerrar nuestras sentencias XQuery entre llaves y dejar las marcas fuera de las llaves. Así, esta consulta consigue generarnos un XML valido añadiendo un elemento raíz al conjunto de partes:

```
<partesrojas>
{
    doc("datos.xml")/datos/partes/parte[color='Rojo']
}
</partesrojas>
```

El resultado devuelto es este:

```
<partesrojas>
    <parte numparte="p1">
        <nombreparte>Tuerca</nombreparte>
        <color>Rojo</color>
        <peso>12</peso>
        <ciudad>Londres</ciudad>
    </parte><parte numparte="p4">
        <nombreparte>Tornillo</nombreparte>
        <color>Rojo</color>
        <peso>14</peso>
        <ciudad>Londres</ciudad>
    </parte><parte numparte="p6">
        <nombreparte>Engranaje</nombreparte>
        <color>Rojo</color>
        <peso>19</peso>
        <ciudad>Londres</ciudad>
    </parte>
</partesrojas>
```

Antes se ha mencionado que se puede usar `WHERE` para crear condiciones. ¿Como cambiar entonces la consulta anterior para poner la condición en un `WHERE` y no meterla entre corchetes?

XQUERY es un potente lenguaje de consulta para Bases de datos, se puede integrar con diferentes lenguajes de programación.