



# TECNOLOGÍAS XML

## XQuery

### Descripción breve

Proyecto del Segundo Trimestre sobre las Tecnologías XML  
(XSD,XPath,XSLT,XQuery).

En este documento se representará las consultas XQuery realizadas y su resultado final.

Iván Hernández Fuentes

1ºCFSG Desarrollo de Aplicaciones Web

# Índice

1. Funciones Utilizadas.
2. Consultas – Funciones de Cadena
3. Consultas – Funciones de Numéricas
4. Consultas – Funciones de Uso General
5. Consultas – Cuantificadores existenciales.
6. Altas de nodos.
7. Modificación de nodos.
8. Eliminación de nodos.

# Funciones Utilizadas.

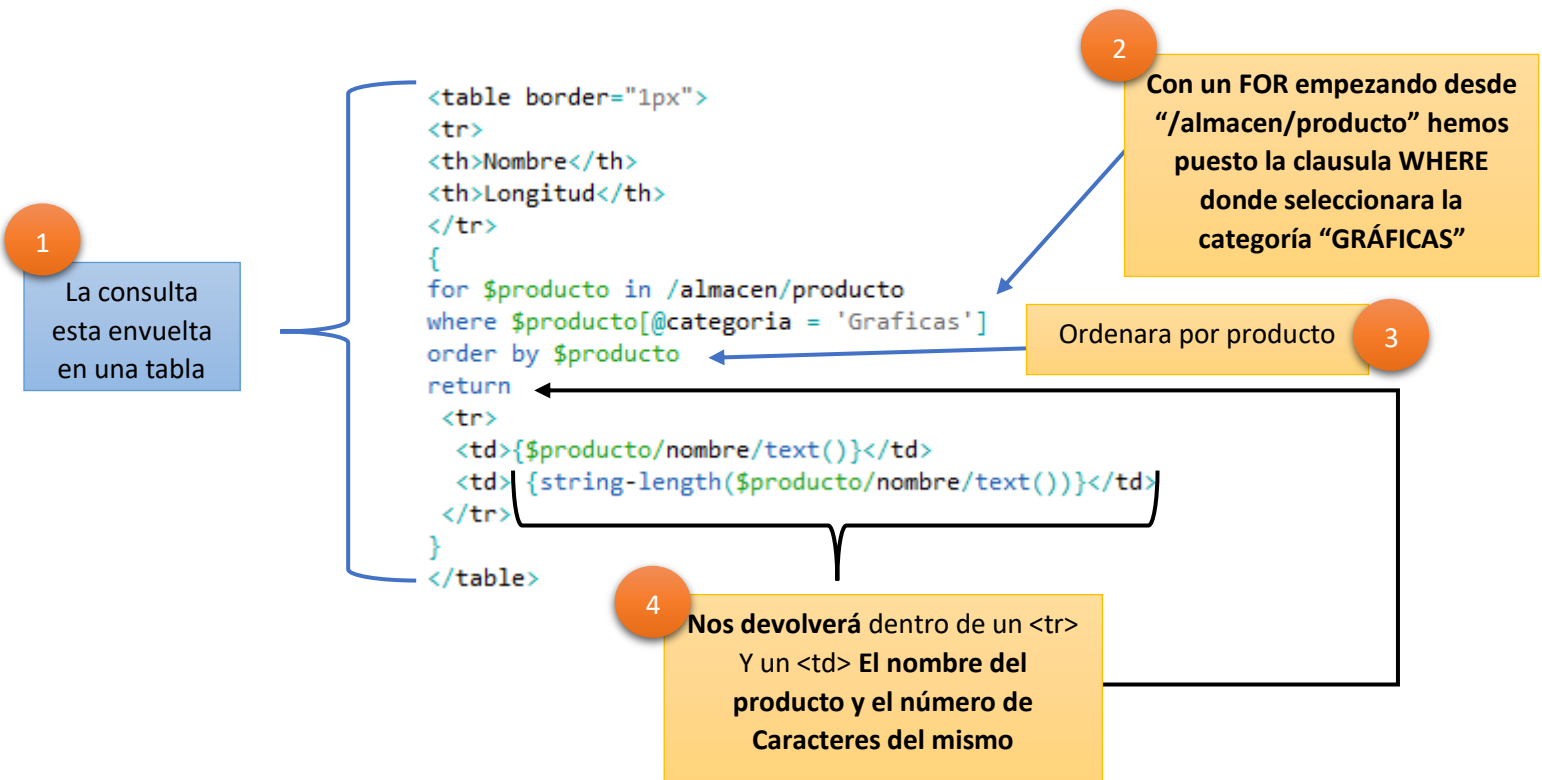
En las consultas XQuery realizadas se ha trabajado con las siguientes funciones y cuantificadores:

- String-length()
- Starts-with()
- Upper-Case()
- Lower-Case()
- Sum()
- Max()
- Min()
- Exists
- Every

# Consultas - Funciones de Cadena

La siguiente Consulta trabajaremos con la función *"string-length()"*

A continuación veremos las consultas hechas en XQuery, utilizando funciones de cadena donde el resultado se ha convertido en HTML haciendo uso de tablas.



El resultado de la Consulta XQuery es la siguiente:

Observamos claramente que el objetivo se ha cumplido **por un lado el Nombre del producto y por otro la longitud**

```
<table border="1px">
  <tr>
    <th>Nombre</th>
    <th>Longitud</th>
  </tr>
  <tr>
    <td>Asus Radeon R5 230</td>
    <td>18</td>
  </tr>
  <tr>
    <td>Gigabyte GeForce GT 710</td>
    <td>23</td>
  </tr>
  <tr>
    <td>Gigabyte GeForce GTX 1050Ti</td>
    <td>27</td>
  </tr>
  <tr>
    <td>MSI GeForce GT710</td>
    <td>17</td>
  </tr>
</table>
```

Una vez Convertido a formato HTML y aplicado estilos CSS nos queda de la siguiente manera:

## Gráficas-Carácteres - Funciones de cadena: Stringlength()

Nombre	Longitud
Asus Radeon R5 230	18
Gigabyte GeForce GT 710	23
Gigabyte GeForce GTX 1050Ti	27
MSI GeForce GT710	17

En esta tabla se representa El nombre de cada producto y la longitud de esa misma cadena de texto.

Hemos utilizado la función Stringlength()

La siguiente Consulta trabajaremos con la función *“Starts-with()”*.

El Objetivo ha mostrar, son los productos que empiecen por “I”, “S” y “C”.

4 Ordenamos los resultados de manera descendente según el nombre

1 Albergamos la consulta en una tabla.

```
<table>
{
  <tr>
  <tr>
  <th>"I"</th>
  {
    for $producto in /almacen/producto
    where starts-with ($producto/nombre, "I")
    order by $producto descending
    return
    <td>{$producto/nombre/text()}</td>
  }
</tr>
  <tr>
  <th>"S"</th>
  {
    for $s in(/almacen/producto)
    where starts-with ($s/nombre, "S")
    order by $s descending
    return
    <td>{$s/nombre/text()}</td>
  }
</tr>
  <tr>
  <th>"C"</th>
  {
    for $c in(/almacen/producto)
    where starts-with ($c/nombre, "C")
    order by $c descending
    return
    <td>{$c/nombre/text()}</td>
  }
</tr>
</tr>
}
</table>
```

2 Utilizamos un **FOR** que recorra desde el nodo “Producto”

3 En la cláusula **WHERE** Agregamos la función “starts-with”, donde tendrá que ser una “I” la letra por la que empiece.

5 Por último devolvemos dentro de un <td> el nombre del producto

La consulta XQuery tendría este resultado:

Cada Letra está agrupada con su resultado

```
<table>
  <tr>
    <tr>
      <th>"I"</th>
      <td>Intel i7 7700K</td>
      <td>Intel i3 4170</td>
    </tr>
    <tr>
      <th>"S"</th>
      <td>Seagate BarraCuda</td>
      <td>Samsung 860</td>
    </tr>
    <tr>
      <th>"C"</th>
      <td>Crucial MX500</td>
      <td>Crucial 2400 DDR4</td>
    </tr>
  </tr>
</table>
```

Convertido la consulta a un archivo HTML y con estilos CSS aplicados obtenemos lo siguiente:

### starts-with() Productos que empiezan por la letra "I", "S" y "C"

"I"	Intel i7 7700K	Intel i3 4170
"S"	Seagate BarraCuda	Samsung 860
"C"	Crucial MX500	Crucial 2400 DDR4

Haciendo uso de la función `starts-with()` hemos representado en una tabla los productos que empiezan por las letras descritas anteriormente.

La siguiente Consulta trabajaremos con la función “*Upper-case()* & *Lower-case()*”.

Mostraremos todos los nombres de los productos, **en mayúscula y minúscula**.

1  
Agrupamos  
en una tabla

```
<table>

  <tr>
    <th>Normal</th>
    <th>Mayúscula</th>
    <th>Minúscula</th>
  </tr>
  {
    for $producto in /almacen/producto

    return <tr>
      <td>{$producto/nombre/text()}</td>
      <td>{upper-case($producto/nombre/text())}</td>
      <td>{lower-case($producto/nombre/text())}</td>
    </tr>
  }
</table>
```

2

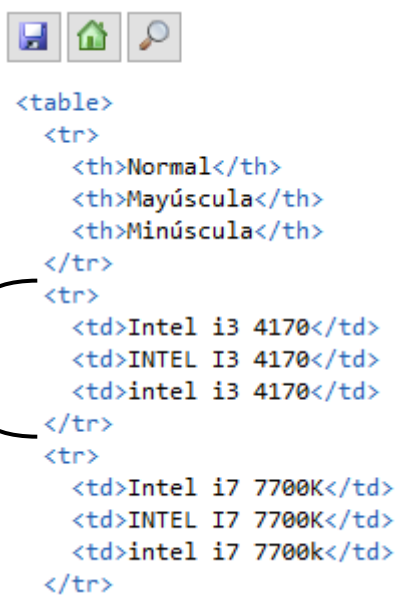
Con un **FOR** hacemos un recorrido desde “/almacen/producto”

3

Devolvemos en un **<td>** el nombre del producto haciendo uso de la función **UPPER** y **LOWER**

La consulta en XQuery nos da este resultado:

El nombre original, el nombre en mayúscula y en minúscula



```
<table>
  <tr>
    <th>Normal</th>
    <th>Mayúscula</th>
    <th>Minúscula</th>
  </tr>
  <tr>
    <td>Intel i3 4170</td>
    <td>INTEL I3 4170</td>
    <td>intel i3 4170</td>
  </tr>
  <tr>
    <td>Intel i7 7700K</td>
    <td>INTEL I7 7700K</td>
    <td>intel i7 7700k</td>
  </tr>
```



El resultado pasado HTML y aplicando estilos CSS:

### Funciones de cadena: Upper-case() / Lower-case()

Normal	Mayúscula	Minúscula
Intel i3 4170	INTEL I3 4170	intel i3 4170
Intel i7 7700K	INTEL I7 7700K	intel i7 7700k
AMD Ryzen 5 1600X	AMD RYZEN 5 1600X	amd ryzen 5 1600x
AMD Ryzen Threadripper 1950X	AMD RYZEN THREADRIPPER 1950X	amd ryzen threadripper 1950x
MSI Interceptor DS B1	MSI INTERCEPTOR DS B1	msi interceptor ds b1
Razer DeathAdder	RAZER DEATHADDER	razer deathadder
Logitech G203	LOGITECH G203	logitech g203
Logitech G2gkghkj03	LOGITECH G2GKGHKJ03	logitech g2gkghkj03
Gigabyte GeForce GTX 1050Ti	GIGABYTE GEFORCE GTX 1050TI	gigabyte geforce gtx 1050ti
Gigabyte GeForce GT 710	GIGABYTE GEFORCE GT 710	gigabyte geforce gt 710
Asus Radeon R5 230	ASUS RADEON R5 230	asus radeon r5 230
MSI GeForce GT710	MSI GEFORCE GT710	msi geforce gt710
Crucial 2400 DDR4	CRUCIAL 2400 DDR4	crucial 2400 ddr4
Kingston DDR3	KINGSTON DDR3	kingston ddr3
Kingston DDR4	KINGSTON DDR4	kingston ddr4
GskillAegis	GSKILLAEGIS	gskillaegis
Seagate BarraCuda	SEAGATE BARRACUDA	seagate barracuda
Kingston A400	KINGSTON A400	kingston a400
Samsung 860	SAMSUNG 860	samsung 860
Crucial MX500	CRUCIAL MX500	crucial mx500

# Consultas - Funciones Numéricas

Veremos las consultas hechas en XQuery, utilizando funciones numéricas donde el resultado se ha convertido en HTML haciendo uso de tablas.

La siguiente Consulta trabajaremos con la función “*SUM()*”.

El objetivo de la consulta es mostrar la **cantidad total de productos que existe por “Categoría”**

```
<table>
  <tr>
    <th>Categoría</th>
    <th>Total</th>
  </tr>
  {
    for $s in /almacen/producto
    let $categoria := $s/@categoria
    group by $categoria (:AGRUPAMOS SEGUN SU CATEGORIA LOS RESULTADOS:)
  }
  return
  <tr>
    <td>{$categoria}</td>
    <td>Suma Total: {sum($s/cantidad)}</td>
  </tr>
</table>
```

1

Agrupamos  
en una tabla

2

Utilizamos un **FOR** que recorra  
“/almacen/producto”

3

Utilizamos también el **LET**,  
donde obtendremos el  
**nombre de la categoría**

4


Agruparemos por  
categoría (*importante*)

5

Mostramos en la celda el **nombre de la  
Categoría** y utilizando **SUM()** sumamos la  
cantidad agrupada (según categoría)

El resultado sería el siguiente:

Cada categoría con su nombre y la suma total de esta



```
<table>
<tr>
<th>Categoría</th>
<th>Total</th>
</tr>
<tr>
<td>Procesadores</td>
<td>Suma Total: 185</td>
</tr>
<tr>
<td>Ratones</td>
<td>Suma Total: 110</td>
</tr>
<tr>
<td>Graficas</td>
<td>Suma Total: 112</td>
</tr>
<tr>
<td>MemoriaRAM</td>
<td>Suma Total: 76</td>
</tr>
<tr>
<td>DiscosDuros</td>
<td>Suma Total: 600</td>
</tr>
</table>
```

Convertido HTML y aplicado los estilos CSS tenemos la siguiente tabla:

### Funciones numéricas SUM(), y "group by"

Categoría	Total
Procesadores	Suma Total: 185
Ratones	Suma Total: 110
Graficas	Suma Total: 112
MemoriaRAM	Suma Total: 76
DiscosDuros	Suma Total: 600

Se muestra una tabla con la cantidad total que existe según la categoría

Para ello se ha utilizado la función SUM() y luego la suma se ha juntado según la categoría utilizando group by

La siguiente Consulta trabajaremos con la función “MAX(), MIN()”.

\*Se añadirá también una “Función de uso general”(exists)

Con las dos funciones anteriores obtendremos el precio máximo y mínimo de los productos de cada categoría, además trabajaremos con un condicional para que los precios mayores de 100 sean de un color.

4  
Agrupamos por categoría  
(importante)

1  
Agrupamos la consulta en una tabla

```
<table>
<tr>
  <th>Categoría</th>
  <th>Máximo</th>
  <th>Mínimo</th>
</tr>
{
  for $max in /almacen/producto
  let $categoria := $max/@categoria
  where exists($max/pvd)
  group by $categoria

  return
  <tr>
    <td>{$categoria}</td>
    <maximo>{if(max($max/pvd) > 100)
      then <td class="color">{max($max/pvd)}</td>
      else <td>Maximo:{max($max/pvd)}</td>
    }</maximo>
    <td>Minimo: {min($max/pvd)}</td>
  </tr>
}
</table>
```

2  
FOR que recorra  
“/almacen/producto”,  
también tenemos un LET  
para obtener la categoría.

3  
En la clausula **WHERE** hemos  
utilizado una **función de uso  
general(exists)**, donde exista el  
nodo PVD.

5  
Utilizamos un IF, si el  
**máximo precio (MAX)**  
es mayor a 100 entonces  
(**THEN**) al <td> le  
aplicaremos un estilo.

En otro caso (**ELSE**) el  
<td> sin estilo.

6  
Por último la función **MIN()**  
mostrará el precio mínimo.

El resultado de la consulta sería la siguiente:



Tenemos un precio máximo mayor a 100, al TD vemos que se aplicó correctamente la clase

Tenemos un precio máximo menor a 100, donde tenemos un TD normal

```
<table>
  <tr>
    <th>Categoría</th>
    <th>Máximo</th>
    <th>Mínimo</th>
  </tr>
  <tr>
    <td>Procesadores</td>
    <maximo>
      <td class="color">786</td>
    </maximo>
    <td>Mínimo: 100.99</td>
  </tr>
  <tr>
    <td>Ratones</td>
    <maximo>
      <td>Maximo:58</td>
    </maximo>
    <td>Mínimo: 20</td>
  </tr>
  <tr>
    <td>Graficas</td>
    <maximo>
      <td class="color">160</td>
    </maximo>
    <td>Mínimo: 40</td>
  </tr>
```

En HTML lo veríamos de la siguiente manera:

## Funciones numéricas MAX(),MIN() y utilización de condicional

Categoría	Máximo	Mínimo
Procesadores	786	Mínimo: 100.99
Ratones	Maximo:58	Mínimo: 20
Graficas	160	Mínimo: 40
MemoriaRAM	Maximo:90	Mínimo: 40
DiscosDuros	120	Mínimo: 30

En la tabla hemos representado el precio máximo y mínimo de cada producto según la categoría.

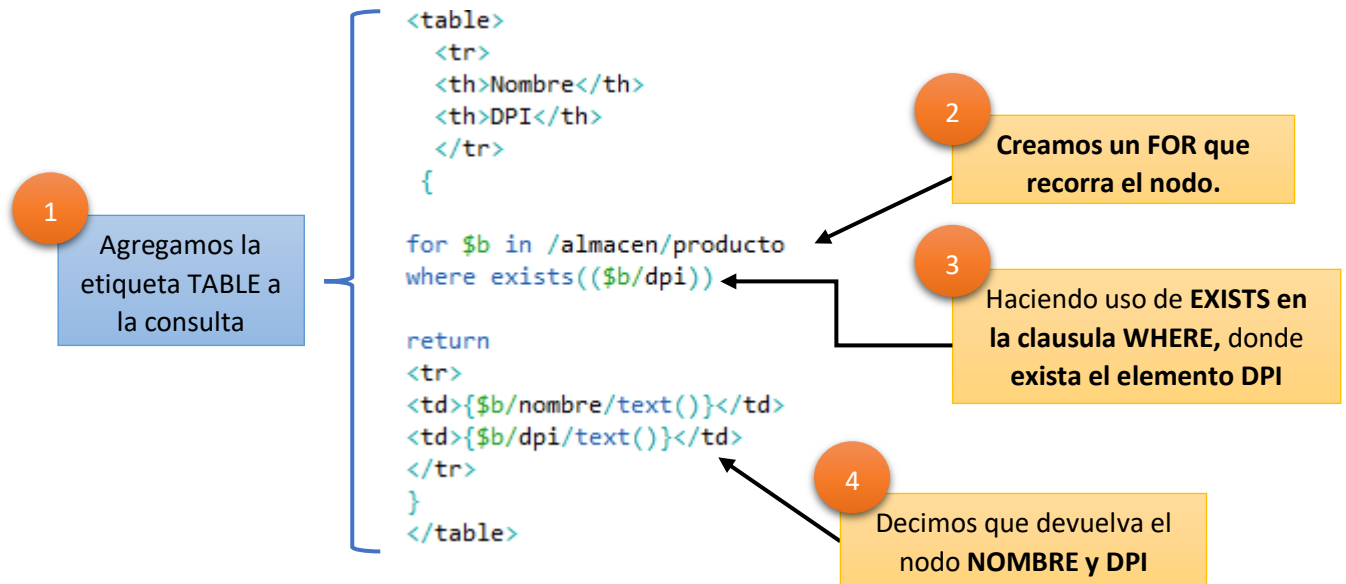
Gracias a la función MAX() y MIN()

Además hemos introducido un condicional como novedad, donde el precio mayor a 100 será de color verde

# Consultas - Funciones Uso General

La siguiente Consulta trabajaremos con la función “*exists*”.

El objetivo es mostrar el nombre del producto y en este caso el elemento DPI, ya que diremos en la consulta que nos lo muestre donde exista el elemento DPI.



El resultado seria el siguiente:



En HTML lo veríamos de la siguiente manera.

## Funciones de uso general exists()

Nombre	DPI
MSI Interceptor DS B1	1600
Razer DeathAdder	16000
Logitech G203	6000
Logitech G2gkghkj03	6000

Utilizando la función `exists()` hemos dicho que muestre el nombre, de aquellos productos compuesto por el elemento DPI

# Consultas-Cuantificadores Existenciales

La siguiente Consulta trabajaremos con el cuantificador *“every”*.

Con “every” muestra los productos donde en su precio haya un “2”

1  
Añadimos la consulta a una tabla

```
<table>
  <tr>
    <th>Nombre</th>
    <th>Precio</th>
  </tr>
  {
    for $i in /almacen/producto
    where every $numero in $i/pvd satisfies contains($numero,"2")

    return
    <tr>
      <td>{$i/nombre/text()}</td>
      <td>{$i/pvd/text()}</td>
    </tr>
  }
</table>
```


2  
FOR que recorra el nodo /almacen/producto

3  
En la **clausula WHERE** ponemos el cuantificador **EVERY** en donde una **nueva variable (\$numero)** in(en) la expresion(**\$i/pvd**) “satisfies contains” tenga el número 2

4  
Por último **devuelve el nombre y el PVD**

El resultado de la consulta seria este:

Vemos como tanto un producto como otro, el elemento PVD contiene un 2



```
<table>
  <tr>
    <th>Nombre</th>
    <th>Precio</th>
  </tr>
  <tr>
    <td>MSI Interceptor DS B1</td>
    <td>20</td>
  </tr>
  <tr>
    <td>Crucial MX500</td>
    <td>120</td>
  </tr>
</table>
```



El resultado en HTML lo tendríamos aquí:

## Cuantificadores existenciales "every"

Nombre	Precio
MSI Interceptor DS B1	20
Crucial MX500	120

Representamos en una tabla el nombre del producto donde el pvd contiene al menos un 2

# Altas de nodos

Vamos a insertar un nodo al principio en medio y al final.

## Insertar un nodo al principio.

```
insert node
<producto categoria="Procesadores">
  <nombre>Intel XEON E31220</nombre>
  <socket>1151 v2</socket><!--Opcional-->
  <cantidad>30</cantidad>
  <stock>Si</stock>
  <pvd>199</pvd>
  <imagen>imagenes/procesadores/xeon_e3.jpg</imagen>
  <informe>
    <comentario>Para servidores</comentario>
  </informe>
</producto>
as first into /almacen
```

1

Insertamos el siguiente nodo con "Insert node"

2

Escribimos todo el nodo completo

3

Y será al principio o al primero del nodo almacén con "as first into"

Aquí tendríamos el resultado, vemos que está el primero por delante de los anteriores productos.

```
<almacen xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:
noNamespaceSchemaLocation="almacen.xsd">
  <producto categoria="Procesadores">
    <nombre>Intel XEON E31220</nombre>
    <socket>1151 v2</socket>
    <!--Opcional-->
    <cantidad>30</cantidad>
    <stock>Si</stock>
    <pvd>199</pvd>
    <imagen>imagenes/procesadores/xeon_e3.jpg</imagen>
    <informe>
      <comentario>Para servidores</comentario>
    </informe>
  </producto>
  <producto categoria="Procesadores">
    <nombre>Intel i3 4170</nombre>
    <socket>1150</socket>
    <!--Opcional-->
    <cantidad>100</cantidad>
    <stock>Si</stock>
    <pvd>100.99</pvd>
    <imagen>imagenes/procesadores/i3_4170.jpg</imagen>
    <informe>
      <comentario/>
    </informe>
  </producto>
```

Vemos que es el primero

## Insertar un nodo al final.

insert node

1

Insertamos el siguiente nodo con "Insert node"

2

Escribimos todo el nodo completo

```
<producto categoria="DiscosDuros">
  <nombre>TOSHIBA3764</nombre>
  <tipo_disco>HDD</tipo_disco>
  <cantidad>79</cantidad>
  <stock>Si</stock>
  <pvd>45</pvd>
  <imagen>imagenes/discos/toshiba_3764.jpg</imagen>
  <informe>
    <comentario></comentario> <!--default en xsd-->
  </informe>
</producto>
```

3

Le decimos que lo inserte el último con "as last " into (en el nodo /almacen)

as last into /almacen

Comprobamos ejecutando la sentencia:

```
<producto categoria="DiscosDuros">
  <nombre>Crucial MX500</nombre>
  <tipo_disco>SSD</tipo_disco>
  <cantidad>110</cantidad>
  <stock>Si</stock>
  <pvd>120</pvd>
  <imagen>imagenes/discos/Crucial MX500.jpg</imagen>
  <informe>
    <comentario/>
    <!--default en xsd-->
  </informe>
</producto>
<producto categoria="DiscosDuros">
  <nombre>TOSHIBA3764</nombre>
  <tipo_disco>HDD</tipo_disco>
  <cantidad>79</cantidad>
  <stock>Si</stock>
  <pvd>45</pvd>
  <imagen>imagenes/discos/toshiba_3764.jpg</imagen>
  <informe>
    <comentario/>
    <!--default en xsd-->
  </informe>
</producto>
</almacen>
```

Observamos que se encuentra en la última posición

## Insertar un nodo en el medio

1 Insertamos el siguiente nodo con "Insert node"

```
insert node
```

2 Escribimos todo el nodo completo

```
<producto categoria="Graficas">
  <nombre>GTX 950M</nombre>
  <tipo>GDDR3</tipo>
  <cantidad>67</cantidad>
  <stock>Si</stock>
  <pvd>60</pvd>
  <imagen>imagenes/graficas/gtx_950.jpg</imagen>
  <informe>
    <comentario></comentario> <!--default en xsd-->
  </informe>
</producto>
```

3 Insertamos el nodo antes de la posición 11

```
before //producto[11]
```

Realizamos la consulta a la posición 11

```
//producto[11]
```

script.xq biblioteca.xml scr

```
190 as last into /almacen
191 :)
192
193 insert node
194
195 <producto categoria="Graficas">
196   <nombre>GTX 950M</nombre>
197   <tipo>GDDR3</tipo>
198   <cantidad>67</cantidad>
```

OK

### Result

```
<producto xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" categoria="Graficas">
  <nombre>GTX 950M</nombre>
  <tipo>GDDR3</tipo>
  <cantidad>67</cantidad>
  <stock>Si</stock>
  <pvd>60</pvd>
  <imagen>imagenes/graficas/gtx_950.jpg</imagen>
  <informe>
    <comentario/>
    <!--default en xsd-->
  </informe>
</producto>
```

El resultado vemos que es el correcto

# Modificación de nodos

Vamos a modificar uno de los nodos, en este caso el primero donde el producto cambia el precio a '220'

## Modificar el valor de un nodo

replace value of node /almacen/producto[1]/pvd with '220'

Mediante **"replace value"** reemplazamos el nodo

La posición que vamos a modificar, el elemento a modificar y el nuevo dato del elemento.

```
<producto xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
Procesadores">
  <nombre>Intel XEON E31220</nombre>
  <socket>1151 v2</socket>
  <!--Opcional-->
  <cantidad>30</cantidad>
  <stock>Si</stock>
  <pvd>220</pvd>
  <imagen>imagenes/procesadores/xeon_e3.jpg</imagen>
  <informe>
    <comentario>Para servidores</comentario>
  </informe>
</producto>
```

Comprobamos que el dato ha sido cambiado correctamente

## Modificar de forma completa el nodo

Cambiamos varios datos.

```
replace node /almacen/producto[@categoria = 'Graficas' and nombre = 'GTX 950M']  
with
```

```
<producto categoria="Graficas">  
  <nombre>GTX 1060</nombre>  
  <tipo>GDDR5</tipo>  
  <cantidad>30</cantidad>  
  <stock>Si</stock>  
  <pvd>300</pvd>  
  <imagen>imagenes/graficas/gtx_160.jpg</imagen>  
  <informe>  
    <comentario/>  
    <!--default en xsd-->  
  </informe>  
</producto>
```

1  
Introducimos la sentencia  
"replace node" y especificamos la  
condición XPath

2  
Escribimos el nodo completo  
con varios cambios

```
<producto xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" categoria="Graficas">  
  <nombre>GTX 1060</nombre>  
  <tipo>GDDR5</tipo>  
  <cantidad>30</cantidad>  
  <stock>Si</stock>  
  <pvd>300</pvd>  
  <imagen>imagenes/graficas/gtx_160.jpg</imagen>  
  <informe>  
    <comentario/>  
    <!--default en xsd-->  
  </informe>  
</producto>
```

Hacemos la consulta al  
nodo y vemos los cambios  
correctamente

# Eliminación de nodos

Para acabar eliminaremos los tres nodos creados anteriormente:

```
delete node doc("almacen.xml")//producto[@categoria = 'Procesadores' and nombre = 'Intel XEON E31220']
```

```
delete node doc("almacen.xml")//producto[10]
```

```
delete node doc("almacen.xml")//producto[19]
```

Con “**delete node**” seguido de la **expresión XPath** vamos eliminando los nodos