

# Introducción a XQuery

---



Jorge Castellanos Vega

# Qué es XQuery

---

Lenguaje diseñado para consultar datos en archivos XML.

Equivalente XML del SQL en Bases de Datos.

Se construye sobre expresiones XPath.

Puede usarse para:

- Extraer información para usar en servicios web.
- Generar resúmenes e informes.
- Transformar XML en XHTML.
- Buscar dentro de documentos web para obtener información de interés.

# Documento de ejemplo

---

## Archivo: canciones.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<MiBibliotecaMP3>
  <archivo almacenado="DISCO1">
    <canCIÓN>Hangar 18</canCIÓN>
    <artista>Megadeth</artista>
    <disco>Rust in Peace</disco>
    <puntuacion>9</puntuacion>
  </archivo>
  <archivo almacenado="DISCO2">
    <canCIÓN>Peace Sells</canCIÓN>
    <artista>Megadeth</artista>
    <disco>Peace Sells...But Who's
      Buying</disco>
    <puntuacion>9</puntuacion>
  </archivo>
```

```
<archivo almacenado="DISCO1">
  <canCIÓN>Master of Puppets</canCIÓN>
  <artista>Metallica</artista>
  <disco>Master of Puppets</disco>
  <puntuacion>10</puntuacion>
</archivo>
<archivo almacenado="DISCO2">
  <canCIÓN>Among The Living</canCIÓN>
  <artista>Anthrax</artista>
  <disco>Among The Living</disco>
  <puntuacion>8</puntuacion>
</archivo>
<archivo almacenado="DISCO1">
  <canCIÓN>For Whom The Bell Tolls</canCIÓN>
  <artista>Metallica</artista>
  <disco>Ride The Lightning</disco>
  <puntuacion>8</puntuacion>
</archivo>
</MiBibliotecaMP3>
```

Puedes descargar los ejemplos en el siguiente enlace: <http://cloud.educa.madrid.org/index.php/s/EPc1nNZtYrm5zZL>

# Software

Todos los ejemplos se han probado sobre Ubuntu utilizando el intérprete xQilla

```
jorge@Sirius: ~  
jorge@Sirius:~$ sudo apt install xqilla  
Leyendo lista de paquetes... Hecho  
Creando árbol de dependencias  
Leyendo la información de estado... Hecho  
Los paquetes indicados a continuación se instalaron de forma automática y ya no son necesarios.  
linux-headers-4.4.0-45 linux-headers-4.4.0-45-generic linux-headers-4.4.0-47  
linux-headers-4.4.0-47-generic linux-headers-4.4.0-51  
linux-headers-4.4.0-51-generic linux-headers-4.4.0-53  
linux-headers-4.4.0-53-generic linux-image-4.4.0-45-generic  
linux-image-4.4.0-47-generic linux-image-4.4.0-51-generic  
linux-image-4.4.0-53-generic linux-image-extra-4.4.0-45-generic  
linux-image-extra-4.4.0-47-generic linux-image-extra-4.4.0-51-generic  
linux-image-extra-4.4.0-53-generic  
Utilice «sudo apt autoremove» para eliminarlos.  
Se instalarán los siguientes paquetes adicionales:  
libxerces-c3.1 libxqilla6v5  
Se instalarán los siguientes paquetes NUEVOS:  
libxerces-c3.1 libxqilla6v5 xqilla  
0 actualizados, 3 nuevos se instalarán, 0 para eliminar y 667 no actualizados.  
Se necesita descargar 2.078 kB de archivos.  
Se utilizarán 10,7 MB de espacio de disco adicional después de esta operación.  
¿Desea continuar? [S/n] S
```

```
jorge@Sirius: ~/Documentos/xquery  
jorge@Sirius:~/Documentos/xquery$ which xqilla  
/usr/bin/xqilla  
jorge@Sirius:~/Documentos/xquery$
```

También puedes usar un servicio online como

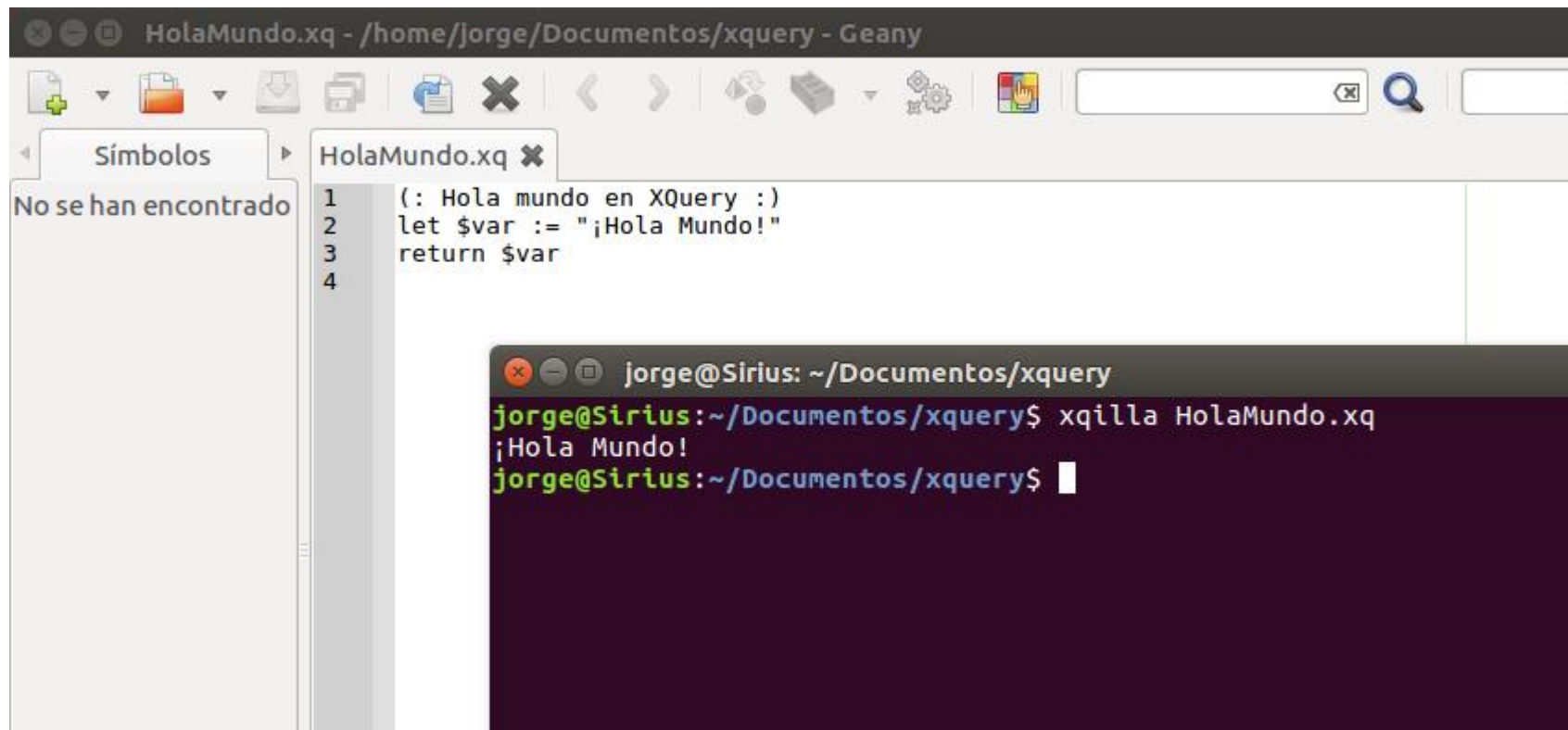
<http://www.xpathtester.com/xquery>

En ese caso no es necesario hacer referencia al archivo mediante la función doc

# Software

---

Podemos probar que todo está correcto utilizando cualquier editor y escribiendo un programa como el de la imagen. Posteriormente lo ejecutamos desde línea de comandos.



The image shows two overlapping windows. The top window is a Geany text editor titled 'HolaMundo.xq - /home/jorge/Documentos/xquery - Geany'. It contains the following XQuery code:

```
1 (: Hola mundo en XQuery :)  
2 let $var := "¡Hola Mundo!"  
3 return $var  
4
```

The bottom window is a terminal titled 'jorge@Sirius: ~/Documentos/xquery'. It shows the execution of the XQuery file using the 'xqilla' command:

```
jorge@Sirius:~/Documentos/xquery$ xqilla HolaMundo.xq  
¡Hola Mundo!  
jorge@Sirius:~/Documentos/xquery$
```

# Sintaxis XQuery

---

XQuery es **sensible a mayúsculas** y todos los elementos, atributos y variables deben ser identificadores válidos XML.

Las cadenas de caracteres pueden delimitarse tanto por comillas simples ('cadena') como por comillas dobles ("cadena").

Las variables se definen con un símbolo de dólar \$ seguido del nombre de la variable. Por ejemplo **\$contador**.

Los comentarios se delimitan mediante (: para la apertura y :) para el cierre.

- (: esto es un comentario en Xquery :)

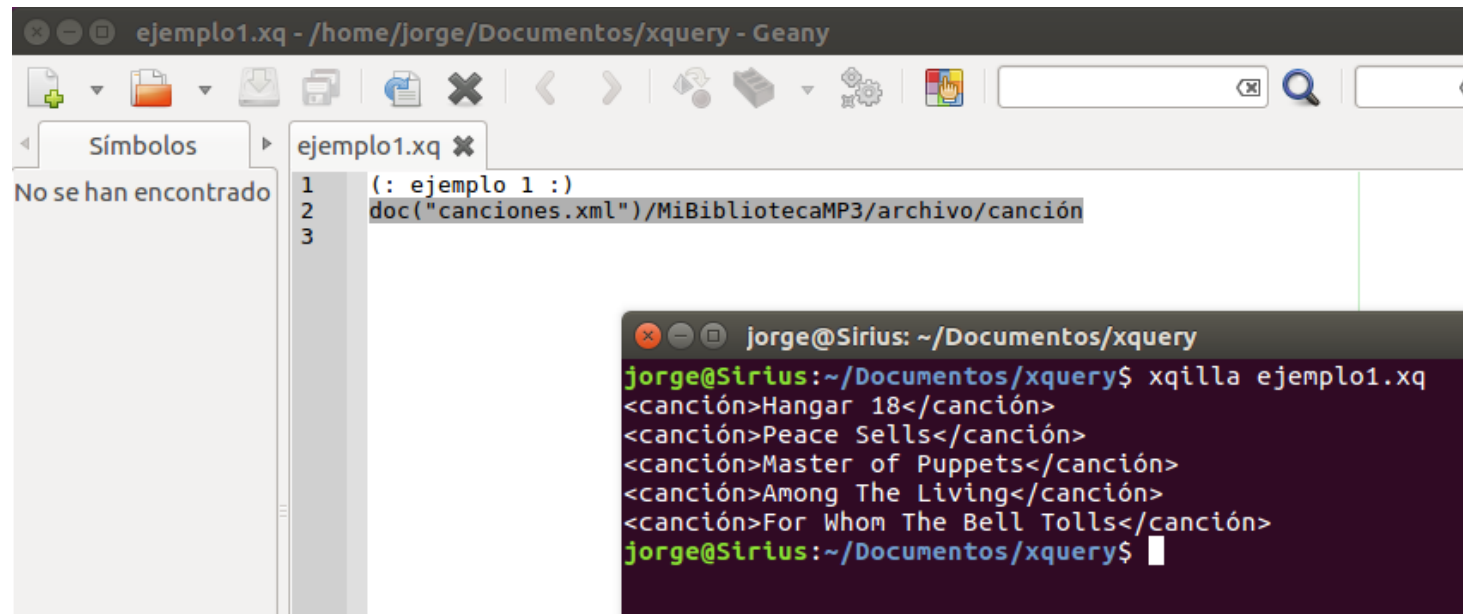
# Seleccionando nodos

Xquery utiliza funciones para extraer los datos de los documentos XML.

Para abrir un documento se usa la función **doc()**

Para navegar a través de un documento se usan expresiones de ruta. Por ejemplo:

- `doc("canciones.xml")/MiBibliotecaMP3/archivo/canción`



The screenshot shows the Geany IDE window titled 'ejemplo1.xq - /home/jorge/Documentos/xquery'. The editor contains the following Xquery code:

```
1 (: ejemplo 1 :)
2 doc("canciones.xml")/MiBibliotecaMP3/archivo/canción
3
```

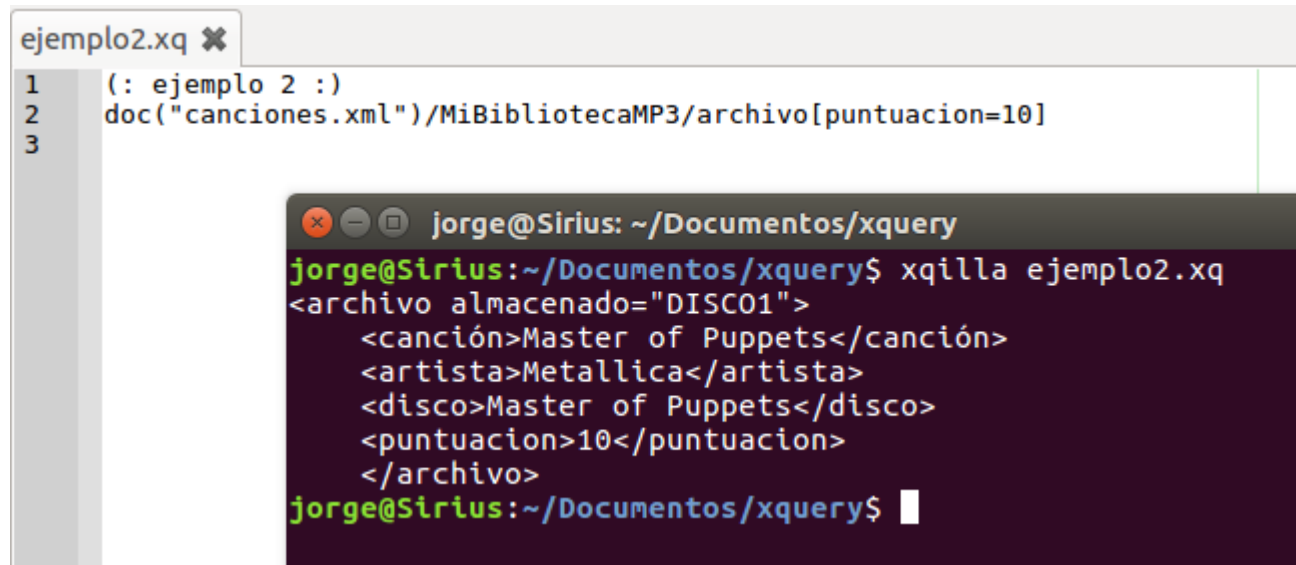
Below the IDE window, a terminal window shows the execution of the Xquery file:

```
jorge@Sirius: ~/Documentos/xquery
jorge@Sirius:~/Documentos/xquery$ xqilla ejemplo1.xq
<canción>Hangar 18</canción>
<canción>Peace Sells</canción>
<canción>Master of Puppets</canción>
<canción>Among The Living</canción>
<canción>For Whom The Bell Tolls</canción>
jorge@Sirius:~/Documentos/xquery$
```

# Seleccionando nodos

---

A las expresiones se les pueden añadir predicados, por ejemplo:



The image shows a code editor window titled 'ejemplo2.xq' with the following XQuery code:

```
1 (: ejemplo 2 :)
2 doc("canciones.xml")/MiBibliotecaMP3/archivo[puntuacion=10]
3
```

Below the editor is a terminal window showing the execution of the query using the 'xqilla' command. The terminal output displays the XML structure of the selected node.

```
jorge@Sirius: ~/Documentos/xquery
jorge@Sirius:~/Documentos/xquery$ xqilla ejemplo2.xq
<archivo almacenado="DISC01">
  <canCIÓN>Master of Puppets</canCIÓN>
  <artista>Metallica</artista>
  <disco>Master of Puppets</disco>
  <puntuacion>10</puntuacion>
</archivo>
jorge@Sirius:~/Documentos/xquery$
```



# Expresiones FLWOR

---

FLWOR es un acrónimo para **For**, **Let**, **Where**, **Order by**, **Return**. Cada una de las instrucciones tiene un significado concreto.

**For** – Selecciona una secuencia de nodos.

**Let** – Asigna un valor a una variable.

**Where** – Establece una condición que filtra los nodos.

**Order by** – Realiza una ordenación de los nodos que han pasado la condición.

**Return** – Valor de retorno, se evalúa una vez por cada nodo.

# Expresiones FLWOR

Por ejemplo la siguiente expresión.

- `doc("canciones.xml")/MiBibliotecaMP3/archivo[puntuacion>8]/canción`

Tiene su equivalente FLOWR

```
ejemplo4.xq ✕ ejemplo3.xq ✕
1 (: ejemplo 3 :)
2 doc("canciones.xml")/MiBibliotecaMP3/archivo[puntuacion>8]/canción
3
```

```
jorge@Sirius: ~/Documentos/xquery
jorge@Sirius:~/Documentos/xquery$ xqilla ejemplo3.xq
<canción>Hangar 18</canción>
<canción>Peace Sells</canción>
<canción>Master of Puppets</canción>
jorge@Sirius:~/Documentos/xquery$
```

Sin FLOWR

```
ejemplo4.xq ✕ ejemplo3.xq ✕
1 (: ejemplo 4 :)
2 for $i in doc("canciones.xml")/MiBibliotecaMP3/archivo
3 where $i/puntuacion>8
4 return $i/canción
5
```

```
jorge@Sirius: ~/Documentos/xquery
jorge@Sirius:~/Documentos/xquery$ xqilla ejemplo4.xq
<canción>Hangar 18</canción>
<canción>Peace Sells</canción>
<canción>Master of Puppets</canción>
jorge@Sirius:~/Documentos/xquery$
```

CON FLOWR

# Expresiones FLWOR

---

En el código:

```
for $i in doc("canciones.xml")/MiBibliotecaMP3/archivo
where $i/puntuacion>8
return $i/canción
```

La cláusula **for** va asignando a la variable **\$i** el valor de cada uno de los nodos del archivo que se ajustan a la expresión **/MiBibliotecaMP3/archivo**

Para cada elemento seleccionado se evalúa la condición mediante la instrucción **where** asegurándose que el valor de la puntuación del elemento sea mayor que 8.

Mediante **return** para cada elemento que cumple con la condición se devuelve el valor del elemento canción que es lo que se muestra en la salida.

# Expresiones FLOWR

En [xQuery tester online](#):

No es necesario incluir **doc()**

```
xQuery tester  XPath  XQuery  XSLT  Validate
XQuery:
for $i in /MiBibliotecaMP3/archivo
where $i/puntuacion>8
return $i/canción

XML: Test Save Color Format
<?xml version="1.0" encoding="UTF-8"?>
<MiBibliotecaMP3>
  <archivo almacenado="DISC01">
    <canción>Hangar 18</canción>
    <artista>Megadeth</artista>
    <disco>Rust in Peace</disco>
    <puntuacion>9</puntuacion>
  </archivo>
  <archivo almacenado="DISC02">
    <canción>Peace Sells</canción>
    <artista>Megadeth</artista>
    <disco>Peace Sells...But Who's Buying</disco>
    <puntuacion>9</puntuacion>
  </archivo>
  <archivo almacenado="DISC01">
    <canción>Master of Puppets</canción>
    <artista>Metallica</artista>
    <disco>Master of Puppets</disco>
    <puntuacion>10</puntuacion>
  </archivo>
  <archivo almacenado="DISC02">
    <canción>Among The Living</canción>
    <artista>Anthrax</artista>
    <disco>Among The Living</disco>
    <puntuacion>8</puntuacion>
  </archivo>
  <archivo almacenado="DISC01">
    <canción>For Whom The Bell Tolls</canción>
    <artista>Metallica</artista>
    <disco>Ride The Lightning</disco>
    <puntuacion>8</puntuacion>
  </archivo>
</MiBibliotecaMP3>
```

Resultado

```
xQuery tester  XPath  XQuery  XSLT  Validate
XQuery:
for $i in /MiBibliotecaMP3/archivo
where $i/puntuacion>8
XML: Test Save Color Format
INFO - XQuery completed (compiled in 1ms, evaluated in 3ms)
<?xml version="1.0" encoding="UTF-8"?>
<canción>Hangar 18</canción>
<canción>Peace Sells</canción>
<canción>Master of Puppets</canción>
```

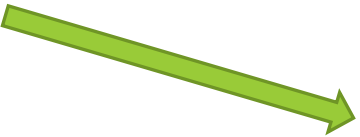
# Añadiendo HTML a FLWOR

Es posible combinar HTML con una Expresión FLOWR. Veamos un ejemplo:

```
ejemplo5.xq ✕
1 <html>
2 <head>
3   <title>Ejemplo 5</title>
4 </head>
5
6 <body>
7   <ol>
8   {
9     for $i in doc("canciones.xml")/MiBibliotecaMP3/archivo
10    where $i/puntuacion>8
11    order by $i/puntuacion
12    return <li> {$i/canción}({$i/puntuacion}) </li>
13  }
14 </ol>
15 </body>
16 </html>
17
```

Combinamos HTML con código XQuery, es importante tener en cuenta que **el código XQuery** se presenta **entre paréntesis {}**.

Por otro lado en la salida se incorporan los nombres de las etiquetas además de los datos ¿cómo hacemos para solucionarlo?

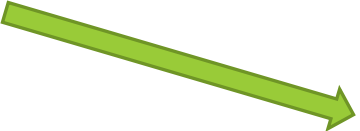


```
jorge@Sirius: ~/Documentos/xquery
jorge@Sirius:~/Documentos/xquery$ xqilla ejemplo5.xq
<html><head><title>Ejemplo 5</title></head><body><ol><li><canción>Master of Pupp
ets</canción>(<puntuacion>10</puntuacion>) </li><li><canción>Hangar 18</canción>
(<puntuacion>9</puntuacion>) </li><li><canción>Peace Sells</canción>(<puntuacion
>9</puntuacion>) </li></ol></body></html>
jorge@Sirius:~/Documentos/xquery$
```

# Añadiendo HTML a FLWOR

Para evitar que el return nos devuelva los títulos de las etiquetas hacemos uso de la función **data**.

```
ejemplo6.xq ✕
1 <html>
2 <head>
3   <title>Ejemplo 6</title>
4 </head>
5
6 <body>
7   <ol>
8   {
9     for $i in doc("canciones.xml")/MiBibliotecaMP3/archivo
10    where $i/puntuacion>8
11    order by $i/puntuacion
12    return <li> {data($i/canción)}({data($i/puntuacion)}) </li>
13  }
14 </ol>
15 </body>
16 </html>
```



```
jorge@Sirius: ~/Documentos/xquery
jorge@Sirius:~/Documentos/xquery$ xqilla ejemplo6.xq
<html><head><title>Ejemplo 6</title></head><body><ol><li>Master of Puppets(10) <
/li><li>Hangar 18(9) </li><li>Peace Sells(9) </li></ol></body></html>
jorge@Sirius:~/Documentos/xquery$
```

# Añadiendo HTML a FLWOR

Añadiendo un atributo XML como identificador de clase en el archivo HTML de salida.

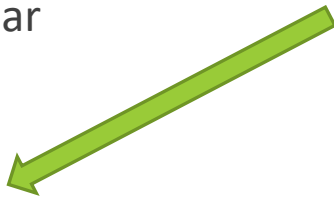
```
ejemplo7.xq ✕
1  <html>
2  <head>
3    <title>Ejemplo 7</title>
4  </head>
5  <body>
6    <ol>
7    {
8      for $i in doc("canciones.xml")/MiBibliotecaMP3/archivo
9      where $i/puntuacion>8
10     order by $i/puntuacion
11     return <li class="{data($i/@almacenado)}"> {data($i/canción)}({data($i/puntuacion)}) </li>
12   }
13 </ol>
14 </body>
15 </html>
```

```
jorge@Sirius: ~/Documentos/xquery
jorge@Sirius:~/Documentos/xquery$ xqilla ejemplo7.xq
<html><head><title>Ejemplo 7</title></head><body><ol><li class="DISC01">Master o
f Puppets(10) </li><li class="DISC01">Hangar 18(9) </li><li class="DISC02">Peace
Sells(9) </li></ol></body></html>
jorge@Sirius:~/Documentos/xquery$
```

# Expresiones condicionales

Es posible incorporar en XQuery la expresión **if-then-else**

- La expresión requiere paréntesis alrededor de la condición del if
- Al contrario que en otros lenguajes es obligatorio añadir la parte de **else** aunque no haya acción asociada, en ese caso podemos dejar simplemente **else ()**.



```
jorge@Sirius: ~/Documentos/xquery
jorge@Sirius:~/Documentos/xquery$ xqilla ejemplo8.xq
<html><head><title>Ejemplo 8</title></head><body><table><caption>DISCO 1 </caption>
<tr><td>Artista</td><td>Disco</td></tr><tr><td>Metallica</td><td>Master of Pu
ppets</td></tr><tr><td>Metallica</td><td>Ride The Lightning</td></tr><tr><td>Meg
adeth</td><td>Rust in Peace</td></tr></table><table><caption>DISCO 2 </caption><
tr><td>Artista</td><td>Disco</td></tr><tr><td>Anthrax</td><td>Among The Living<
td></tr><tr><td>Megadeth</td><td>Peace Sells...But Who's Buying</td></tr></table>
</body></html>
jorge@Sirius:~/Documentos/xquery$
```

```
ejemplo8.xq
1 <html>
2 <head>
3 <title>Ejemplo 8</title>
4 </head>
5
6 <body>
7 <table>
8 <caption>DISCO 1 </caption>
9 <tr><td>Artista</td><td>Disco</td></tr>
10 {
11 for $i in doc("canciones.xml")/MiBibliotecaMP3/archivo
12 order by $i/puntuacion
13 return if ($i/@almacenado="DISCO1")
14 then <tr><td>{data($i/artista)}</td><td>{data($i/disco)}</td></tr>
15 else ()
16 }
17 </table>
18
19 <table>
20 <caption>DISCO 2 </caption>
21 <tr><td>Artista</td><td>Disco</td></tr>
22 {
23 for $i in doc("canciones.xml")/MiBibliotecaMP3/archivo
24 order by $i/puntuacion
25 return if ($i/@almacenado="DISCO2")
26 then <tr><td>{data($i/artista)}</td><td>{data($i/disco)}</td></tr>
27 else ()
28 }
29 </table>
30 </body>
31 </html>
```



# Expresiones condicionales

## Ejemplo if then else completo

ejemplo9.xq ✕

```
1 <html>
2 <head>
3   <title>Ejemplo 9</title>
4 </head>
5
6 <body>
7   <table>
8     <caption>CANCIONES POR DISCO </caption>
9     <tr><td>Artista</td><td>Nombre disco</td><td>Grabada en</td></tr>
10    {
11      for $i in doc("canciones.xml")/MiBibliotecaMP3/archivo
12      order by $i/puntuacion
13      return if ($i/@almacenado="DISCO1")
14    then <tr><td>{data($i/artista)}</td><td>{data($i/disco)}</td>
15         <td>DISCO1</td></tr>
16    else <tr><td>{data($i/artista)}</td><td>{data($i/disco)}</td>
17         <td>DISCO2</td></tr>
18    }
19  </table>
20 </body>
21 </html>
```

```
jorge@Sirius: ~/Documentos/xquery
jorge@Sirius:~/Documentos/xquery$ xqilla ejemplo9.xq
<html><head><title>Ejemplo 9</title></head><body><table><caption>CANCIONES POR D
ISCO </caption><tr><td>Artista</td><td>Nombre disco</td><td>Grabada en</td></tr>
<tr><td>Metallica</td><td>Master of Puppets</td><td>DISCO1</td></tr><tr><td>Anth
rax</td><td>Among The Living</td><td>DISCO2</td></tr><tr><td>Metallica</td><td>R
ide The Lightning</td><td>DISCO1</td></tr><tr><td>Megadeth</td><td>Rust in Peace
</td><td>DISCO1</td></tr><tr><td>Megadeth</td><td>Peace Sells...But Who's Buying
</td><td>DISCO2</td></tr></table></body></html>
jorge@Sirius:~/Documentos/xquery$ xqilla ejemplo9.xq > ejemplo9.html
jorge@Sirius:~/Documentos/xquery$
```

Ejemplo 9 ✕ +

file:///home/jorge/Documentos/xquery/ejemplo9.html

Artista	Nombre disco	Grabada en
Metallica	Master of Puppets	DISCO1
Anthrax	Among The Living	DISCO2
Metallica	Ride The Lightning	DISCO1
Megadeth	Rust in Peace	DISCO1
Megadeth	Peace Sells...But Who's Buying	DISCO2

# Otras posibilidades de FLWOR

---

Tal y como hemos visto anteriormente en XQuery podemos filtrar elementos mediante expresiones de ruta o con expresiones FLWOR.

Recordamos que en las expresiones FLWOR la función de cada instrucción era:

- **for** – asocia una variable a cada elemento devuelto en la expresión.
- **let** – define una variable y la asocia un valor.
- **where** – especifica un criterio de selección de elementos
- **order by** – define cómo queremos ordenar los resultados
- **return** – indica qué queremos que se devuelva en el resultado

Ahora vamos a profundizar un poco más en las posibilidades de algunas de las instrucciones.

# Otras posibilidades de FLWOR

Recordemos su uso mediante un ejemplo

```
ejemplo10.xq ✕
1  <html>
2  <head>
3    <title>Ejemplo 10</title>
4  </head>
5
6  <body>
7    <table>
8      <caption>CANCIONES DE METALLICA </caption>
9      <tr><td>Canción</td><td>Disco</td><td>Grabada en</td></tr>
10     {
11       for $i in doc("canciones.xml")/MiBibliotecaMP3/archivo
12       where $i/artista="Metallica"
13       order by $i/puntuacion
14     } return <tr><td>{data($i/artista)}</td><td>{data($i/disco)}</td>
15           <td>{data($i/@almacenado)}</td></tr>
16     }
17   </table>
18 </body>
19 </html>
```

```
jorge@Sirius: ~/Documentos/xquery
jorge@Sirius:~/Documentos/xquery$ xqilla ejemplo10.xq
<html><head><title>Ejemplo 10</title></head><body><table><caption>CANCIONES DE M
ETALLICA </caption><tr><td>Canción</td><td>Disco</td><td>Grabada en</td></tr><tr>
<td>Metallica</td><td>Master of Puppets</td><td>DISC01</td></tr><tr><td>Metalli
ca</td><td>Ride The Lightning</td><td>DISC01</td></tr></table></body></html>
jorge@Sirius:~/Documentos/xquery$
```

# Otras posibilidades de FLWOR

## La cláusula for

- La cláusula for realiza una iteración de elementos según se haya indicado en su definición.
- Pueden existir múltiples cláusulas for en la misma expresión.
- Para especificar un número determinado de iteraciones se puede usar la palabra clave **to**.

```
ejemplo11.xq ✕  
1  (: ejemplo 11 :)  
2  
3  for $i in (1 to 10)  
4  return $i  
-
```



```
jorge@Sirius: ~/Documentos/xquery  
jorge@Sirius:~/Documentos/xquery$ xqilla ejemplo11.xq  
1  
2  
3  
4  
5  
6  
7  
8  
9  
10
```

# Otras posibilidades de FLWOR

## La cláusula for

Puesto que la variable asociada al bucle incorpora el elemento en el que estamos no podemos usarla para contabilizar las iteraciones. Para ello se usa la palabra clave **at**.

```
ejemplo12.xq ✕
1  <html>
2  <head>
3    <title>Ejemplo 12</title>
4  </head>
5
6  <body>
7    <ul>
8      {
9        for $i at $j in doc("canciones.xml")/MiBibliotecaMP3/archivo
10       where $i/puntuacion>8
11       order by $i/puntuacion
12       return <li>{$j}. {data($i/canción)}({data($i/puntuacion)}) </li>
13     }
14   </ul>
15 </body>
16 </html>
```

```
jorge@Sirius: ~/Documentos/xquery
jorge@Sirius:~/Documentos/xquery$ xqilla ejemplo12.xq
<html><head><title>Ejemplo 12</title></head><body><ul><li>3. Master of Puppets(10) </li><li>1. Hangar 18(9) </li><li>2. Peace Sells(9) </li></ul></body></html>
jorge@Sirius:~/Documentos/xquery$
```

# Otras posibilidades de FLWOR

## La cláusula for

- Se pueden especificar varias expresiones en la cláusula for, cada una de ellas irá separada por comas.

```
ejemplo13.xq ✕  
1  
2  
3 for $i in (1 to 5), $j in (1,2,3)  
4 return <resultado>i es {$i} j es {$j}</resultado>  
5
```

```
jorge@Sirius: ~/Documentos/xquery  
jorge@Sirius:~/Documentos/xquery$ xqilla ejemplo13.xq  
<resultado>i es 1 j es 1</resultado>  
<resultado>i es 1 j es 2</resultado>  
<resultado>i es 1 j es 3</resultado>  
<resultado>i es 2 j es 1</resultado>  
<resultado>i es 2 j es 2</resultado>  
<resultado>i es 2 j es 3</resultado>  
<resultado>i es 3 j es 1</resultado>  
<resultado>i es 3 j es 2</resultado>  
<resultado>i es 3 j es 3</resultado>  
<resultado>i es 4 j es 1</resultado>  
<resultado>i es 4 j es 2</resultado>  
<resultado>i es 4 j es 3</resultado>  
<resultado>i es 5 j es 1</resultado>  
<resultado>i es 5 j es 2</resultado>  
<resultado>i es 5 j es 3</resultado>  
jorge@Sirius:~/Documentos/xquery$
```

# Otras posibilidades de FLWOR

## La cláusula let

- Realiza definición y asignación de variables, es importante tener en cuenta que no realiza iteraciones por si misma.

```
ejemplo14.xq ✕  
1  
2  
3 let $i := (1 to 10)  
4 let $j := (1,2,3)  
5 let $k := 1  
6 return <resultado>i es {$i} j es {$j} k es {$k}</resultado>  
7
```

```
jorge@Sirius: ~/Documentos/xquery  
jorge@Sirius:~/Documentos/xquery$ xqilla ejemplo14.xq  
<resultado>i es 1 2 3 4 5 6 7 8 9 10 j es 1 2 3</resultado>  
jorge@Sirius:~/Documentos/xquery$ xqilla ejemplo14.xq  
<resultado>i es 1 2 3 4 5 6 7 8 9 10 j es 1 2 3 k es 1</resultado>  
jorge@Sirius:~/Documentos/xquery$
```

# Otras posibilidades de FLWOR

## La cláusula where

- Es posible establecer condiciones compuestas

```
ejemplo15.xq ✕
1  <html>
2  <head>
3    <title>Ejemplo 15</title>
4  </head>
5  <body>
6    <ol>
7      {
8        for $i in doc("canciones.xml")/MiBibliotecaMP3/archivo
9        where $i/puntuacion=8 or $i/puntuacion=10
10       order by $i/puntuacion
11       return <li class="{data($i/@almacenado)}"> {data($i/canción)}({data($i/puntuacion)}) </li>
12     }
13   </ol>
14 </body>
15 </html>
```

```
jorge@Sirius: ~/Documentos/xquery
jorge@Sirius:~/Documentos/xquery$ xqilla ejemplo15.xq
<html><head><title>Ejemplo 15</title></head><body><ol><li class="DISC01">Master
of Puppets(10) </li><li class="DISC02">Among The Living(8) </li><li class="DISC0
1">For Whom The Bell Tolls(8) </li></ol></body></html>
jorge@Sirius:~/Documentos/xquery$
```



# Otras posibilidades de FLWOR

## La cláusula where

- Un ejemplo con **and**

```
ejemplo16.xq ✕
1  <html>
2  <head>
3    <title>Ejemplo 16</title>
4  </head>
5  <body>
6    <ol>
7      {
8        (: canciones de grupos cuyo nombre empieza por "M" y
9          tienen puntuación de 9 :)
10     for $i in doc("canciones.xml")/MiBibliotecaMP3/archivo
11     where $i[substring(artista,1,1)="M"] and $i/puntuacion=9
12     order by $i/puntuacion
13     return <li> {data($i/canción)}({data($i/artista)}) </li>
14   }
15 </ol>
16 </body>
17 </html>
```

```
jorge@Sirius: ~/Documentos/xquery
jorge@Sirius:~/Documentos/xquery$ xqilla ejemplo16.xq
<html><head><title>Ejemplo 16</title></head><body><ol><li>Hangar 18(Megadeth) </
li><li>Peace Sells(Megadeth) </li></ol></body></html>
jorge@Sirius:~/Documentos/xquery$
```

# Otras posibilidades de FLWOR

## La cláusula where

- Un ejemplo con **and** y **not**

ejemplo17.xq ✕

```
1 <html>
2 <head>
3   <title>Ejemplo 17</title>
4 </head>
5 <body>
6   <ol>
7     {
8       (: canciones de grupos cuyo nombre empieza por "M" y
9        NO tienen puntuación de 9 : )
10    for $i in doc("canciones.xml")/MiBibliotecaMP3/archivo
11    where $i[substring(artista,1,1)="M"] and not($i/puntuacion=9)
12    order by $i/puntuacion
13    return <li> {data($i/canción)}({data($i/artista)}) </li>
14  }
15 </ol>
16 </body>
17 </html>
```

```
jorge@Sirius: ~/Documentos/xquery
jorge@Sirius:~/Documentos/xquery$ xqilla ejemplo17.xq
<html><head><title>Ejemplo 17</title></head><body><ol><li>Master of Puppets(Metallica) </li><li>For Whom The Bell Tolls(Metallica) </li></ol></body></html>
jorge@Sirius:~/Documentos/xquery$
```

# Otras posibilidades de FLWOR

## La cláusula order by

- Es posible ordenar por varios criterios y especificar si queremos ordenación ascendente (ascending) o descendente (descending)

```
ejemplo18.xq ✕
1 <html>
2 <head>
3   <title>Ejemplo 18</title>
4 </head>
5
6 <body>
7 <table>
8   <caption>CANCIONES POR DISCO </caption>
9   <tr><td>Artista</td><td>Nombre</td><td>Grabada en</td></tr>
10  {
11    for $i in doc("canciones.xml")/MiBibliotecaMP3/archivo
12    order by $i/artista ascending, $i/canción descending
13  } return <tr><td>{data($i/artista)}</td><td>{data($i/canción)}</td>
14    <td>{data($i/@almacenado)}</td></tr>
15  }
16 </table>
17 </body>
18 </html>
```

```
jorge@Sirius: ~/Documentos/xquery
jorge@Sirius:~/Documentos/xquery$ xqilla ejemplo18.xq
<html><head><title>Ejemplo 18</title></head><body><table><caption>CANCIONES POR DISCO
</caption><tr><td>Artista</td><td>Nombre</td><td>Grabada en</td></tr><tr><td>Anthrax</td><td>Among The Living</td><td>DISCO2</td></tr><tr><td>Megadeth</td>
<td>Peace Sells</td><td>DISCO2</td></tr><tr><td>Megadeth</td><td>Hangar 18</td>
<td>DISCO1</td></tr><tr><td>Metallica</td><td>Master of Puppets</td><td>DISCO1</td></tr><tr><td>Metallica</td><td>For Whom The Bell Tolls</td><td>DISCO1</td></tr></table></body></html>
```

CANCIONES POR DISCO		
Artista	Nombre	Grabada en
Anthrax	Among The Living	DISCO2
Megadeth	Peace Sells	DISCO2
Megadeth	Hangar 18	DISCO1
Metallica	Master of Puppets	DISCO1
Metallica	For Whom The Bell Tolls	DISCO1

# Funciones en XQuery

---

Xquery 1.0. comparten la librería de funciones con Xpath 2.0 y XSLT 2.0.

El método de llamada consiste en escribir el nombre la función acompañada de los argumentos entre paréntesis separados por comas. Se pueden usar en:

- Un elemento.
- Predicado de una expresión de ruta.
- Cláusula let.

Es posible anidar funciones una dentro de otra.

En la siguiente página tenemos un ejemplo de uso con varias llamadas a funciones.

# Funciones en XQuery

Ejemplo: en color verde las llamadas a función

```
ejemplo19.xq ✕
1 <html>
2 <head>
3 <title>Ejemplo 19</title>
4 </head>
5
6 <body>
7 <table>
8 <caption>CANCIONES POR DISCO </caption>
9 <tr><td>Artista</td><td>Nombre</td><td>Grabada en</td></tr>
10 {
11 for $i in doc("canciones.xml")/MiBibliotecaMP3/archivo
12 let $numero := (substring($i/@almacenado,6,1))
13 order by $i/artista ascending, $i/canción descending
14 return <tr><td>{upper-case(data $i/artista)}</td><td>{data $i/canción}</td>
15 <td>{$numero}</td></tr>
16 }
17 </table>
18 </body>
19 </html>
```

```
jorge@Sirius: ~/Documentos/xquery
jorge@Sirius:~/Documentos/xquery$ xqilla ejemplo19.xq
<html><head><title>Ejemplo 19</title></head><body><table><caption>CANCIONES POR
DISCO </caption><tr><td>Artista</td><td>Nombre</td><td>Grabada en</td></tr><tr><
td>ANTHRAX</td><td>Among The Living</td><td>2</td></tr><tr><td>MEGADETH</td><td>
Peace Sells</td><td>2</td></tr><tr><td>MEGADETH</td><td>Hangar 18</td><td>1</td>
</tr><tr><td>METALLICA</td><td>Master of Puppets</td><td>1</td></tr><tr><td>META
LLICA</td><td>For Whom The Bell Tolls</td><td>1</td></tr></table></body></html>
jorge@Sirius:~/Documentos/xquery$
```

# Funciones definidas por el usuario

---

En caso de que el usuario necesite realizar una función que no está en la [librería de funciones de XQuery](#) puede definir las suyas propias.

- Se pueden definir en el propio archivo o en una librería separada.
- Los tipos de datos de los parámetros son los mismos definidos para XML Schema
- El cuerpo de la función debe estar delimitado por paréntesis
- Las funciones en Xquery no utilizan la palabra reservada return, siempre devuelve el último (y único) valor.
- La sintaxis es:

```
declare function prefijo:nombre_función($parámetros_como_tipos_de_datos)  
as Tipo_de_datos_de_retorno  
{  
  ...código de la función...  
};
```

# Funciones definidas por el usuario

---

Ejemplo: función que pasa de minutos a segundos

```
ejemplo20.xq ✕  
1  (: declaración de la función :)  
2  declare function local:MinutosASegundos($tiempo as xs:decimal?) as xs:decimal?  
3  {  
4  ($tiempo*60)  
5  };  
6  
7  (: llamada a la función :)  
8  local:MinutosASegundos(14)  
9
```

```
jorge@Sirius: ~/Documentos/xquery  
jorge@Sirius:~/Documentos/xquery$ xqilla ejemplo20.xq  
840  
jorge@Sirius:~/Documentos/xquery$
```

# Referencias

---

<https://www.w3schools.com/>