

Listar todos los productos y sus fabricantes.

for \$i in //producto return concat(\$i/nombreproducto/text(), " - ", \$i/fabricante/text())

The screenshot shows the BaseX 10.5 interface. The XQuery editor contains the query: `for $i in //producto return concat($i/nombreproducto/text(), " - ", $i/fabricante/text())`. The results pane shows 5 results, which are listed in the bottom left:
Portátil Lenovo ThinkPad X1 Carbon - Lenovo
Smartphone Samsung Galaxy S21 - Samsung
Monitor LG 27GN950-B - LG
Tablet Apple iPad Pro 12.9 - Apple
Altavoz Sonos Beam - Sonos

The bottom right pane shows a tree diagram of the XML structure, with the root element 'productos' containing five 'producto' elements. The 'Info' pane shows the compilation and optimization steps for the query.

Compiling:

- merge: descendant::producto
- rewrite fn:concat(value1,value2[,...]): concat(\$i_0/nombreproducto/text(), " - ", \$i_0/fabricante/text()) -> (\$i_0/nombreproducto/text() || " - " || \$i_0/fabricante/text())
- inline for \$i_0 in util:root(.)//descendant::producto
- simplify FLWOR expression: util:root(.)//descendant::producto ! (nombreproducto/text() || " - " || fabricante/text())

Optimizing:

- rewrite context value: . -> db:get-pre("producto", 0)
- rewrite util:root(nodes): util:root(db:get-pre("producto", 0)) -> db:get-pre("producto", 0)
- convert to child steps: descendant::producto

Optimized Query:

```
db:get("producto", "producto.xml")
```

Listar todos los productos con un precio mayor a \$500.

for \$i in //producto where \$i/precio >500 return concat(\$i/nombreproducto/text(),"- €", \$i/precio)

C:/Users/Hp EliteDesk/Desktop/BASE/basex/Bases/producto.xml [producto] - BaseX 10.5

Database Editor View Visualization Options Help

XQuery `for $i in //producto where $i/precio > 50 return concat($i/nombreproducto/text(),"- €", $i/precio)` 5 Results

producto.xml

```

1 <productos>
2 <producto>
3 <nombreproducto>Portátil
  Lenovo ThinkPad X1 Carbon</
  nombreproducto>
4 <fabricante>Lenovo</fabricante
  >
5 <precio>1200</precio>
6 </producto>
7 <producto>
8 <nombreproducto>Smartphone
  Samsung Galaxy S21</
  nombreproducto>
9 <fabricante>Samsung</
  fabricante>
10 <precio>900</precio>
11 </producto>
12 <producto>
13 <nombreproducto>Monitor LG

```

producto.xml

productos

producto

nombrepro..

fabricante

precio

Lenovo

Samsung

Apple

1200

900

1000

400

LG

Sonos

1100

5 Results, 179 b

Result

Portátil Lenovo ThinkPad X1 Carbon - €1200

Smartphone Samsung Galaxy S21- € 900

Monitor LG 27GN950-B- €1000

Tablet Apple iPad Pro 12.9- €1100

Altavoz Sonos Beam- €400

Compiling:

- merge: descendant::producto
- rewrite > comparison: (\$i_0/precio > 50)
- > \$i_0/precio >= 50.00000000000001
- rewrite fn:concat(value1,value2[,...]):
- concat(\$i_0/nombreproducto/text(), "- €", \$i_0/precio) -> (\$i_0/nombreproducto/text() | "- €" || \$i_0/precio)
- rewrite to predicate: precio >= 50.00000000000001
- inline for \$i_0 in util:root(./)descendant::producto[precio >= 50.00000000000001]
- simplify FLWOR expression: util:root(./)descendant::producto[precio >= 50.00000000000001] ! (nombreproducto/text() || "- €" || precio)

Optimizing:

- rewrite context value: .-> db:get-pre("producto", 0)
- rewrite util:root(nodes): util:root(db:get-pre("producto", 0)) -> db:get-pre("producto", 0)

db:get("producto", "producto.xml")/productos/text()

55 MB

Listar todos los productos con un precio menor o igual a \$500.

for \$i in //producto where \$i/precio <= 500 return concat(\$i/nombreproducto/text(), "- €", \$i/precio)

1 Result

XQuery `for $i in //producto where $i/precio <=500 return concat($i/nombreproducto/text(),"- €", $i/precio)`

C:/Users/Hp Eli

*.xml, *.xq*

Find contents...

Bases

producto.xml (700 b)

producto.xml

```

1 <productos>
2 <producto>
3 <nombreproducto>Portátil
  Lenovo ThinkPad X1 Carbon</
  nombreproducto>
4 <fabricante>Lenovo</fabricante
  >
5 <precio>1200</precio>
6 </producto>
7 <producto>
8 <nombreproducto>Smartphone
  Samsung Galaxy S21</
  nombreproducto>
9 <fabricante>Samsung</fabri
  cante>
10 <precio>900</precio>
11 </producto>
12 <producto>
13 <nombreproducto>Monitor LG

```

OK 27 : 13

producto.xml

productos

producto

nombrepro..

fabricante

Lenovo

precio

1200

producto

nombrepro..

fabricante

Samsung

precio

900

producto

nombrepro..

fabricante

Apple

precio

1100

producto

nombrepro..

fabricante

LG

precio

1000

producto

nombrepro..

fabricante

Sonos

precio

400

1 Result, 26 b

Altavoz Sonos Beam- €400

Result

Q All Total Time: 2.18 Info

Compiling:

- merge: descendant::producto
- rewrite <= comparison: (\$i_0/precio <= 500) -> \$i_0/precio <= 500.0
- rewrite fn:concat(value1,value2[,...]): concat(\$i_0/nombreproducto/text(), "- €", \$i_0/precio) -> (\$i_0/nombreproducto/text() | "- €" || \$i_0/precio)
- rewrite to predicate: precio <= 500.0
- inline for \$i_0 in util:root().)/descendant::producto[precio <= 500.0]
- simplify FLWOR expression: util:root().)/descendant::producto[precio <= 500.0] ! (nombreproducto/text() || "- €" || precio)

Optimizing:

- rewrite context value: . -> db:get-pre("producto", 0)
- rewrite util:root(nodes): util:root(db:get-pre("producto", 0)) -> db:get-pre("producto", 0)
- convert to child steps: descendant::

producto.xml

productos

producto

nombrepro..

fabricante

Lenovo

precio

1200

producto

nombrepro..

fabricante

Samsung

precio

900

producto

nombrepro..

fabricante

Apple

precio

1100

producto

nombrepro..

fabricante

LG

precio

1000

producto

nombrepro..

fabricante

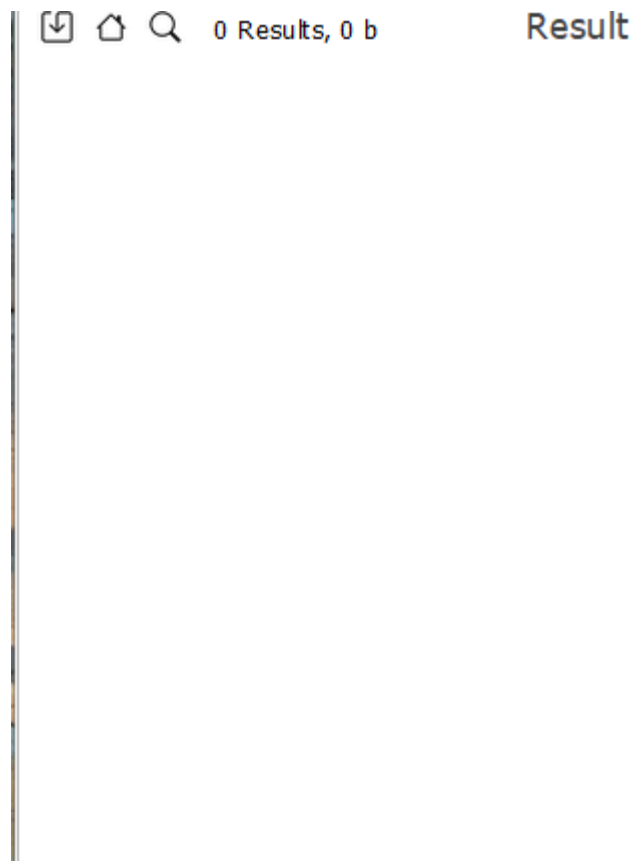
Sonos

precio

400

Listar los nombres de los productos que fabrica "Fabricante A".

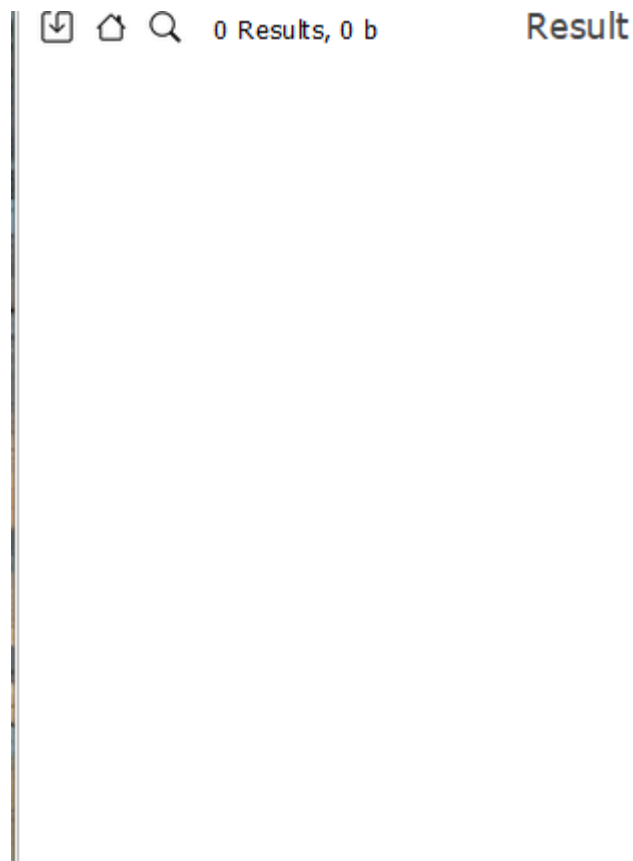
Está vacía la lista



for \$i in //producto where \$i/fabricante="FABRICANTE A" return \$i/nombreproducto/text()

Listar los nombres de los productos que tienen un precio mayor a \$100 y que son fabricados por "Fabricante B".

Está vacía la lista



for \$i in //producto where \$i/fabricante = "Fabricante B" and \$i/precio > 100 return
\$i/nombreproducto/text()

**Listar los nombres de los productos y sus precios, ordenados por precio
ascendente.**

for \$i in //producto let \$precio :=xs:integer(\$i/precio) order by \$precio ascending return
concat(\$i/nombreproducto/text(), "-€", \$precio)

```
b:get("producto", "producto.xml")/productos/producto/text()
```

Listar los nombres de los productos y sus precios, ordenados por precio descendente.

for \$i in //producto let \$precio :=xs:integer(\$i/precio) order by \$precio descending return concat(\$i/nombreproducto/text(), "-€", \$precio)

The screenshot shows the XQuery IDE interface. The top toolbar includes icons for Database, Editor, View, Visualization, Options, and Help. The XQuery editor contains the query: `producto let $precio :=xs:integer($i/precio) order by $precio descending return concat($i/nombreproducto/text(), "-€", $precio)`. The left pane shows the XML file `producto.xml` with the following content:

```
<productos>
  <producto>
    <nombreproducto>Portátil
    Lenovo ThinkPad X1 Carbon</
    nombreproducto>
    <fabricante>Lenovo</fabricante
  >
    <precio>1200</precio>
  </producto>
  <producto>
    <nombreproducto>Smartphone
    Samsung Galaxy S21</
    nombreproducto>
    <fabricante>Samsung</
    fabricante>
    <precio>900</precio>
  </producto>
  <producto>
    <nombreproducto>Monitor LG
    </nombreproducto>
    <fabricante>
    <precio>
  </producto>
```

The right pane displays the results of the query, showing a tree structure of the XML data. The bottom pane shows the results of the query, listing the products and their prices in descending order: `Portátil Lenovo ThinkPad X1 Carbon -€1200`, `Tablet Apple iPad Pro 12.9-€1100`, `Monitor LG 27GN950-B-€1000`, `Smartphone Samsung Galaxy S21-€900`, and `Altavoz Sonos Beam-€400`. The bottom right pane shows the XML tree structure.

Result

Portátil Lenovo ThinkPad X1 Carbon
-€1200
Tablet Apple iPad Pro 12.9-€1100
Monitor LG 27GN950-B-€1000
Smartphone Samsung Galaxy S21-€900
Altavoz Sonos Beam-€400

Compiling:

- merge: descendant::producto
- rewrite fn:concat(value1,value2[,...]): concat(\$i_0/nombreproducto/text(), "-€", \$precio_1) -> (\$i_0/nombreproducto/text() || "-€" || \$precio_1)

Optimizing:

- rewrite context value: . -> db:get-pre("producto", 0)
- rewrite util:root(nodes): util:root(db:get-pre("producto", 0) -> db:get-pre("producto", 0))
- convert to child steps: descendant::producto

Optimized Query:

(for \$i_0 in db:get-pre("producto", 0)/*: productos/*:producto let \$precio_1 := xs:integer(\$i_0/precio) order by \$precio_1 descending empty least return (\$i_0/nombreproducto/text() || "-€" || \$precio_1))

Listar los nombres de los productos que contienen la palabra "Cámara"

for \$p in //producto where contains(\$p/nombreproducto/text(), "Cámara") return \$p/nombreproducto/text()

Listar los nombres de los productos que contienen la letra "E" en su nombre.

```
for $i in //producto where contains($i/nombreproducto/text(), "E") return
$i/nombreproducto/text()
```

```
//no sale nada
```

Listar los nombres de los productos cuyo fabricante comienza con la letra "S"

```
for $i in //producto where starts-with($i/fabricante/text(), "S") return $i/nombreproducto/text()
```