

XPATH: LENGUAJE DE MARCAS

1º Usando la base de datos de productos. Buscar todos los elementos "producto" que tengan un precio mayor a 1000:

`/productos/producto[precio>1000]`

The screenshot shows a database query tool interface with the following components:

- Database:** C:/Users/...
- Editor:** Contains the XQuery `/productos/producto[precio>1000]`.
- Visualizer:** Displays the XML structure of the database. The root is `productos`, which contains `producto` elements. The selected query results are highlighted in red, showing products with prices greater than 1000: `Lenovo` (1200), `Sam...` (900), and `Apple` (1100).
- Results:** Shows 2 Results, 29... Result. The results are displayed in a table format, showing the XML structure of the selected products.
- Compiling:** Shows the compilation of the query, including the rewrite rule: `- rewrite > comparison: (precio > 1000) -> precio >= 1000.`
- Optimizing:** Shows the optimization of the query, including the rewrite rule: `- rewrite context value: . -> db: get-pre("producto", 0)`
- Optimized Query:** Shows the optimized query: `db: get-pre("producto", 0)/ productos/ producto[precio >= 1000.00000000000001]`
- Diagram:** A tree diagram showing the XML structure of the database, with the root `productos` and its children `producto`.

2º Buscar el nombre del fabricante del primer producto de la lista:

NO SALE NADA,

3º Buscar todos los productos que sean fabricados por Apple o Samsung:

/productos/producto[fabricante='Apple' or fabricante='Samsung']

The screenshot shows the XQuery IDE interface. The top bar displays the query: `/productos/producto[fabricante='Apple' or fabricante='Samsung']`. The left pane shows the file explorer with `producto.xml` selected. The middle pane shows the XML structure of `producto.xml`, with the `producto` elements highlighted in red. The right pane shows the results of the query, which are two XML fragments representing products from Apple and Samsung.

Query: `/productos/producto[fabricante='Apple' or fabricante='Samsung']`

XML Structure (producto.xml):

```
<productos>
  <producto>
    <nombreproducto>
      Portátil Lenovo
      ThinkPad X1 Carbon</nombreproducto>
    <fabricante>Lenovo</fabricante>
    <precio>1200</precio>
  </producto>
  <producto>
    <nombreproducto>
      Smartphone Samsung
    </nombreproducto>
  </producto>
</productos>
```

Results: 2 Results, 284 b

Result 1:

```
<producto>
  <nombreproducto>Smartphone Samsung Galaxy
  S21</nombreproducto>
  <fabricante>Samsung</fabricante>
  <precio>900</precio>
</producto>
```

Result 2:

```
<producto>
  <nombreproducto>Tablet Apple iPad Pro 12.9</nombreproducto>
  <fabricante>Apple</fabricante>
  <precio>1100</precio>
</producto>
```

Info:

Compiling:

- rewrite list to xs:string sequence: ("Apple", "Samsung")
- simplify or: ((fabricante = "Apple") or (fabricante = "Samsung"))
- rewrite fn:boolean(input): boolean((fabricante = ("Apple", "Samsung"))) -> (fabricante = ("Apple", "Samsung"))

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)
- apply text index for ("Apple", "Samsung")

Optimized Query:

```
db:text("producto". ("Apple". "Samsung"))/parent::
```

4) Buscar el precio de todos los productos que tengan un nombre de producto que contenga la palabra "Tablet":

/productos/producto[contains(nombreproducto, 'Tablet')]/precio

XQuery `/productos/producto[contains(nombreproducto, 'Tablet')]/precio`

C:/Users/ ... 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

```

OK 3 : 65

producto.xml

producto	nombre	fabricante	precio
Portátil Lenovo	Lenovo	1200	
Smartphone Samsung	Samsung	900	
Tablet Apple	Apple	1100	
Tablet Samsung	Samsung	400	
Tablet LG	LG	1000	
Tablet Sonos	Sonos		

1 Result, 21 b

Result

`<precio>1100</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)

Optimized Query:

`db:get-pre("producto", 0)/productos/producto[contains(nombreproducto, "Tablet")]/precio`

Query:

`/productos/producto[contains(nombreproducto, 'Tablet')]/precio`

Result:

- Hit(s): 1 Item
- Updated: 0 Items
- Printed: 21 b

5° Buscar todos los nombres de los productos que tengan un precio menor a 500:

`/productos/producto[precio<500]/nombreproducto`

XQuery `/productos/producto[precio<500]/nombreproducto`

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
  
```

OK 3 : 65

producto.xml

productos

producto	nombre	precio	fabricante
Portátil Lenovo	ThinkPad X1 Carbon	1200	Lenovo
Smartphone Samsung		900	Sam.
		1100	Apple
		400	Sonos
		1000	LG

1 Result, 51 b

Result

```
<nombreproducto>Altavoz Sonos Beam</
nombreproducto>
```

Time required: 3.79 ms

26 MB

Info

Total Time: 3.79 ms

Compiling:

- rewrite < comparison: (precio < 500) -> precio <= 499.99999999999994

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)

Optimized Query:

```
db:get-pre("producto", 0)/productos/producto[
precio <= 499.99999999999994]/nombreproducto
```

Query:

```
/productos/producto[precio<500]/nom breproducto
```

Result:

- Hit(s): 1 Item

6º Fabricante más caro.

`/productos/producto[precio = max(//producto/precio)]/fabricante`

XQuery `/productos/producto[precio = max(//producto/precio)]/fabricante`

producto.xml

```

19 <fabricante>Apple</fabricante>
20 <precio>1100</precio>
21 </producto>
22 <producto>
23 <nombreproducto>Altavoz
   Sonos Beam</nombreproducto>
24 <fabricante>Sonos</fabricante>
25 <precio>400</precio>
26 </producto>
27 </productos>

```

producto.xml

producto	nombreproducto	fabricante	precio
producto	Len...	fabric...	1100
producto	Sam...	fabric...	900
producto	Apple	fabric...	1100
producto	Altavoz	Sonos	400
producto	LG	fabric...	1000

Result

1 Result, 31 b

```
<fabricante>Lenovo</fabricante>
```

Info

Total Time: 5.99 ms

Compiling:

- merge: descendant::producto

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
- rewrite fn:max(values[,collation]): max(util:root(.)/*: productos/*:producto/precio) -> 1200

Optimized Query:

```
db:get-pre("producto", 0)/productos/producto[precio = 1200]/fabricante
```

Query:

```
/productos/producto[precio = max(//producto/precio)]/fabricante
```

db:get("producto", "producto.xml") 29 MB

7º FABRICANTE MENOS CARO

`/productos/producto[precio = min(//producto/precio)]/fabricante`

XQuery `/productos/producto[precio = min(//producto/precio)]/fabricante`

Editor

producto.xml

```

19 <fabricante>Apple</fabricante>
20 <precio>1100</precio>
21 </producto>
22 <producto>
23   <nombreproducto>Altavoz
     Sonos Beam</nombreproducto>
24   <fabricante>Sonos</fabricante>
25   <precio>400</precio>
26 </producto>
27 </productos>

```

OK 27 : 13

producto.xml

producto	producto	producto
nomb.. fabric..	nomb.. fabric..	nomb.. precio
Len.. Sam..		1100
precio	precio	fabric.. Apple
1200	900	
producto		producto
nomb.. precio		nomb.. precio
LG 1000		400
		fabric.. Sonos

Result

1 Result, 30 b

```
<fabricante>Sonos</fabricante>
```

Info

Total Time: 3.13 ms

Compiling:

- merge: descendant::producto

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
- rewrite fn:min(values[.collation]): min(util:root(.)/*: productos/*:producto/precio) -> 400

Optimized Query:

```
db:get-pre("producto", 0)/productos/producto[precio = 400]/fabricante
```

Query:

```
/productos/producto[precio = min(//producto/precio)]/fabricante
```

8º El nombre del producto con un precio exacto de 900?

`/productos/producto[precio=900]/nombreproducto`

XQuery `/productos/producto[precio=900]/nombreproducto`

C:/Users/

*.xml, *.xq*

Find contents...

BASES

producto.xml (7)

producto.xml

```

19 <fabricante>Apple</fabricante>
20 <precio>1100</precio>
21 </producto>
22 <producto>
23 <nombreproducto>Altavoz
  Sonos Beam</nombreproducto>
24 <fabricante>Sonos</fabricante>
25 <precio>400</precio>
26 </producto>
27 </productos>

```

OK 27 : 13

productos.xml

producto	producto	producto
nomb.. fabric..	nomb.. fabric..	nomb.. precio
Len..	Sam..	fabric.. precio 1100
precio 1200	precio 900	producto
nomb.. fabric..	nomb.. precio 400	nomb.. precio
LG	1000	fabric.. Sonos

1 Result, 62 b

Result

<nombreproducto>Smartphone Samsung Galaxy S21</nombreproducto>

All Total Time: 2.0 ms Info

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)

Optimized Query:

db:get-pre("producto", 0)/productos/producto[precio = 900]/nombreproducto

Query:

/productos/producto[precio=900]/nombreproducto

Result:

- Hit(s): 1 Item
- Updated: 0 Items
- Printed: 62 b
- Read Locking: producto

9º El precio del producto fabricado por Samsung?

`/productos/producto[fabricante = 'Samsung']/precio`

XQuery `/productos/producto[fabricante = 'Samsung']/precio`

C:/Users/... Editor

producto.xml

```

19 <fabricante>Apple</fabricante>
20 <precio>1100</precio>
21 </producto>
22 <producto>
23   <nombreproducto>Altavoz
     Sonos Beam</nombreproducto>
24   <fabricante>Sonos</fabricante>
25   <precio>400</precio>
26 </producto>
27 </productos>

```

OK 27 : 13

productos.xml

producto	producto	producto
nomb.. fabric..	nomb.. fabric..	nomb.. fabric..
Len..	Sam..	Apple
precio 1200	precio 900	precio 1100
producto	producto	producto
nomb.. fabric..	nomb.. fabric..	nomb.. fabric..
LG	1000	Sonos
precio	precio	precio
		400

1 Result, 20 b

Result

`<precio>900</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)
- apply text index for "Samsung"

Optimized Query:

`db:text("producto", "Samsung")/parent::fabricante/parent::producto/precio`

Query:

`/productos/producto[fabricante = 'Samsung']/precio`

Result:

- Hit(s): 1 Item
- Updated: 0 Items
- Printed: 20 b

Time required: 3.3 ms 33 MB

10º El precio del primer producto fabricado por LG?

`/productos/producto[fabricante = 'LG'][1]/precio`

XQuery `/productos/producto[fabricante = 'LG'][1]/precio`

C:/Users/ ... Editor

producto.xml

```

19 <fabricante>Apple</fabricante>
20 <precio>1100</precio>
21 </producto>
22 </producto>
23 <nombreproducto>Altavoz
    Sonos Beam</nombreproducto>
24 <fabricante>Sonos</fabricante>
25 <precio>400</precio>
26 </producto>
27 </productos>

```

OK 27 : 13

productos.xml

producto	producto	producto
nomb... fabric... Len...	nomb... fabric... Sam...	nomb... precio 1100 fabric... Apple
precio 1200	precio 900	producto nomb... precio 400 fabric... Sonos
producto nomb... fabrica... LG	precio 1000	

1 Result, 21 b

Result

`<precio>1000</precio>`

Info

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)

Optimized Query:

db:get-pre("producto", 0)/productos/producto[(fabricante = "LG")][1]/precio

Query:

/productos/producto[fabricante = 'LG'][1]/precio

Result:

- Hit(s): 1 Item
- Updated: 0 Items
- Printed: 21 b
- Read Locking: producto

11º El producto más caro fabricado por Apple?

`/productos/producto[fabricante = 'Apple' and precio = max(//producto[fabricante = 'Apple']/precio)]/nombreproducto`

XQuery

is/producto[fabricante = 'Apple' and precio = max(//producto[fabricante = 'Apple']/precio)]/nombreproducto

C:/Users/

*.xml, *.xq*

Find contents...

BASES

producto.xml (7)

producto.xml

19 <fabricante>Apple</fabricante>

20 <precio>1100</precio>

21 </producto>

22 <producto>

23 <nombreproducto>Altavoz

24 <fabricante>Sonos</fabricante>

25 <precio>400</precio>

26 </producto>

27 </productos>

OK

24 : 31

producto.xml

productos

producto	producto	producto
nomb.. fabric..	nomb.. fabric..	nomb.. fabric..
Len..	Sam..	Apple
precio	precio	precio
1200	900	1100
producto		producto
nomb.. fabric..		nomb.. fabric..
LG		Sonos
precio		precio
1000		400

1 Result, 59 b

Result

<nombreproducto>Tablet Apple iPad Pro 12.9</nombreproducto>

Q All

Total Time: 5.6 ms

Info

Compiling:

- merge: descendant::producto[(fabricante = "Apple")]
- rewrite to predicate: ((fabricante = "Apple") and (precio = max(util:root(./)descendant::producto[(fabricante = "Apple")]/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[(fabricante = "Apple")]
- apply text index for "Apple"

Optimized Query:
db:text("producto". "Apple")/parent::fabricante/

38 MB