

**CÓDIGO EN XML:**

```
<Clientes>
  <Cliente>
    <empresa>Empresa A</empresa>
    <pais>España</pais>
    <facturacion>75000</facturacion>
  </Cliente>
  <Cliente>
    <empresa>Empresa B</empresa>
    <pais>Alemania</pais>
    <facturacion>60000</facturacion>
  </Cliente>
  <Cliente>
    <empresa>Empresa C</empresa>
    <pais>Francia</pais>
    <facturacion>80000</facturacion>
  </Cliente>
  <Cliente>
    <empresa>Empresa D</empresa>
    <pais>España</pais>
    <facturacion>90000</facturacion>
  </Cliente>
  <Cliente>
    <empresa>Empresa E</empresa>
    <pais>Alemania</pais>
    <facturacion>100000</facturacion>
  </Cliente>
  <Cliente>
    <empresa>Empresa F</empresa>
    <pais>Francia</pais>
    <facturacion>75000</facturacion>
  </Cliente>
  <Cliente>
    <empresa>Empresa G</empresa>
    <pais>España</pais>
    <facturacion>55000</facturacion>
  </Cliente>
  <Cliente>
    <empresa>Empresa H</empresa>
    <pais>Alemania</pais>
    <facturacion>95000</facturacion>
  </Cliente>
  <Cliente>
    <empresa>Empresa I</empresa>
    <pais>Francia</pais>
```

```

    <facturacion>70000</facturacion>
  </Cliente>
  <Cliente>
    <empresa>Empresa J</empresa>
    <pais>España</pais>
    <facturacion>80000</facturacion>
  </Cliente>
</Clientes>

```

The screenshot displays the SQL Server Enterprise Manager interface. The 'Editor' pane shows the XML content being loaded into the 'Clientes' table. The 'Result' pane shows the loaded data, and the 'Info' pane shows the commands and results of the operation.

**Editor:**

```

40 <facturacion>95000</facturacion>
41 </Cliente>
42 <Cliente>
43 <empresa>Empresa I</empresa>
44 <pais>Francia</pais>
45 <facturacion>70000</facturacion>
46 </Cliente>
47 <Cliente>
48 <empresa>Empresa J</empresa>
49 <pais>España</pais>
50 <facturacion>80000</facturacion>
51 </Cliente>
52 </Clientes>

```

**Result:**

```

<Clientes>
  <Cliente>
    <empresa>Empresa A</empresa>
    <pais>España</pais>
    <facturacion>75000</facturacion>
  </Cliente>
  <Cliente>
    <empresa>Empresa B</empresa>
    <pais>Alemania</pais>
    <facturacion>60000</facturacion>
  </Cliente>
  <Cliente>
    <empresa>Empresa C</empresa>
    <pais>Francia</pais>
    <facturacion>80000</facturacion>
  </Cliente>

```

**Info:**

Command: SET ARCHIVENAME true

Command: SET SKIPCORRUPT true

Command: CREATE DB Trailor C:\Users\ginj\OneDrive\Desktop\BASEX\Bases\Trailor.xml

Result: Database 'Trailor' created in 122.99 ms.

**1º Seleccionar los nodos cliente que contienen el elemento facturacion con un valor mayor a 80000.**

//Cliente[facturacion>80000]

The screenshot displays an XQuery IDE interface with the following components:

- Query Editor:** Shows the XQuery `//Cliente[facturacion>80000]`.
- XML Editor:** Displays the source XML document `Traijor.xml` with the following structure:

```
100000</facturacion>
</Cliente>
<Cliente>
  <empresa>Empresa F
</empresa>
  <pais>Francia</
pais>
  <facturacion>75000
</facturacion>
</Cliente>
<Cliente>
  <empresa>Empresa G
</empresa>
```
- Result Set:** Shows 3 results (370 b) for the query. The results are XML fragments:

```
facturacion>
</Cliente>
<Cliente>
  <empresa>Empresa E</
empresa>
  <pais>Alemania</pais>
  <facturacion>100000</
facturacion>
</Cliente>
<Cliente>
  <empresa>Empresa H</
empresa>
  <pais>Alemania</pais>
  <facturacion>95000</
facturacion>
</Cliente>
```
- Execution Plan:** Shows the compilation and optimization steps:
  - Compiling:**
    - rewrite > comparison: ( facturacion > 80000 ) -> facturacion >= 80000. 00000000001
    - merge: descendant::Cliente[ facturacion >= 80000. 00000000001]
  - Optimizing:**
    - rewrite context value: . -> db: get-pre("Traijor", 0)
    - rewrite util:root(nodes): util: root(db:get-pre("Traijor", 0)) -> db:get-pre("Traijor", 0)
    - convert to child steps:
- Diagram:** A tree diagram showing the execution plan. The root node is `Traijor.xml`, which points to `Traijor.xml`, which then points to `Clientes`. The `Clientes` node branches into multiple `Cliente` nodes, which then branch into `emp..`, `fact..`, and `pais` nodes.

**2) Seleccionar los nodos cliente que contienen el elemento pais con el valor "España" y el elemento facturacion con un valor entre 60000 y 90000.**

//Cliente[pais="España" and facturacion >=60000 and facturacion <=90000]

**XQuery Editor:**

```
//Cliente[pais="España" and facturacion >=60000 and facturacion <=90000]
```

**Traijor.xml Source:**

```

25 10000</facturacion>
26 </Cliente>
27 <Cliente>
28   <empresa>Empresa F
29 </empresa>
30   <pais>Francia</
31   <facturacion>75000
32 </facturacion>
33 </Cliente>
34 <Cliente>
35   <empresa>Empresa G
36 </empresa>

```

**Results (3 Results, 367 b):**

```

<pais>España</pais>
<facturacion>75000</
facturacion>
</Cliente>
<Cliente>
  <empresa>Empresa D</
empresa>
  <pais>España</pais>
  <facturacion>90000</
facturacion>
</Cliente>
<Cliente>
  <empresa>Empresa J</
empresa>
  <pais>España</pais>
  <facturacion>80000</
facturacion>

```

**Compiling:**

- rewrite >= comparison: ( facturacion >= 60000 ) -> facturacion >= 60000.0
- rewrite <= comparison: ( facturacion <= 90000 ) -> facturacion <= 90000.0
- simplify and: ((pais = "España") and facturacion >= 60000.0 and facturacion <= 90000.0)
- rewrite to predicate: ((pais = "España") and facturacion >= 60000.0 and facturacion <= 90000.0)
- merge: descendant::Cliente[( pais = "España")\|facturacion >= 60000.0 and facturacion <= 90000.0]

**Traijor.xml Diagram:**

```

graph TD
    Traijorxml[Traijor.xml] --> Clientes[Clientes]
    Clientes --> Cliente[Cliente]
    Cliente --> emp[emp..]
    Cliente --> fact[fact..]
    Cliente --> pais[pais]

```

### 3) Sumar los valores de todos los elementos facturacion en el documento XML.

sum(//facturacion)

Query: `sum(//facturacion)`

Editor: Trajor.xml

```

30 <facturacion>75000
31 </facturacion>
32 </Cliente>
33 <Cliente>
34   <empresa>Empresa G
35   </empresa>
36   <pais>España</pais>
37 </Cliente>
38 <facturacion>55000
39 </facturacion>
40 </Cliente>
41 </Clientes>

```

Result: 780000

Compiling:

- merge: descendant: facturacion

Optimizing:

- rewrite context value: . -> db: get-pre("Trajor", 0)
- rewrite util:root(nodes): util: root(db:get-pre("Trajor", 0)) -> db:get-pre("Trajor", 0)
- convert to child steps: descendant::facturacion
- rewrite fn:sum(values[,zero]): sum(db:get-pre("Trajor", 0)/\*: Clientes/\*:Cliente/\*:facturacion) -> 780000

Trajor.xml

Clientes

Cliente

emp.. fact.. em.. fact.. em.. fact.. em.. fact..

pais pais pais pais

Cliente

emp.. fact.. fact.. e.. pais fact.. fact..

Cliente

emp.. fact.. emp.. pais fact.. emp.. pais fact..

Cliente

emp.. fact.. emp.. pais fact.. emp.. pais fact..

33 MB

#### 4) Encontrar todas las empresas cuya facturación es mayor a 750,000 euros:

`//Cliente[facturacion>75000]/empresa`

[illegible]

## 5) Encontrar todas las empresas de España:

//Cliente[pais="España"]/empresa

The screenshot displays the XQuery Studio interface. The top toolbar shows the XQuery icon. The main window is divided into three panes:

- Left Pane (File Explorer):** Shows the project structure with files like `producto.xml` and `Traijor.xml`.
- Center Pane (Editor):** Contains the XQuery code:

```
<?xml version='1.0' encoding='UTF-8'>
<facturacion>75000
</facturacion>
</Cliente>
<Cliente>
  <empresa>Empresa G
</empresa>
  <pais>España</pais>
  <facturacion>55000
</facturacion>
</Cliente>
</Traijor>
```
- Right Pane (Visualizer):** Shows a tree view of the XML document structure, with nodes like `Cientes`, `Cliente`, `emp..`, `fact..`, and `pais`.

Below the editor, the **Results** pane shows the output of the query:

```
<empresa>Empresa A</
empresa>
<empresa>Empresa D</
empresa>
<empresa>Empresa G</
empresa>
<empresa>Empresa J</
empresa>
```

The **Info** pane on the right provides details about the query execution:

- Compiling:**
  - merge: descendant::Cliente[ pais = "España"]
- Optimizing:**
  - rewrite context value: . -> db: get-pre("Traijor", 0)
  - rewrite util: root(nodes): util: root(db: get-pre("Traijor", 0)) -> db: get-pre("Traijor", 0)
  - apply text index for "España"
- Optimized Query:**

```
db: text("Traijor", "España")/
parent::pais/parent::Cliente/
empres
```

The bottom right pane shows a diagram of the XML document structure, with nodes like `Traijor.xml`, `Cientes`, and `Cliente`.

## 6) Encontrar la facturación promedio de las empresas alemanas:

avg(//Cliente[pais="Alemania"]/facturacion)

XQuery

avg(/Cliente[pais="Alemania"]/facturacion)

Result

85000

Compiling:

- merge: descendant::Cliente[ pais = "Alemania"]

Optimizing:

- rewrite context value: .-> db: get-pre("Trajor", 0)
- rewrite util:root(nodes): util: root(db:get-pre("Trajor", 0))-> db:get-pre("Trajor", 0)
- apply text index for "Alemania"

Optimized Query:

```
avg(db:text("Trajor", "Alemania")/parent::pais/parent::Cliente/facturacion)
```

Time required: 22.03 ms

36 MB

## 7º Encontrar la empresa con la facturación más alta

//Cliente[facturacion = max(/Cliente/facturacion)]/empresa

XQuery

//Cliente[facturacion = max(/Cliente/facturacion)]/empresa

Result

<empresa>Empresa E</ empresa>

Compiling:

- merge: descendant::Cliente
- merge: descendant::Cliente[ facturacion = max(util:root(.)/ descendant::Cliente/facturacion )]

Optimizing:

- rewrite context value: .-> db: get-pre("Trajor", 0)
- rewrite util:root(nodes): util: root(db:get-pre("Trajor", 0))-> db:get-pre("Trajor", 0)
- convert to child steps: descendant::Cliente
- rewrite fn:max(values[

6



## 8º Encontrar el identificador de la empresa "XYZ Corp":

[//Cliente\[empresa="XYZ Corp"\]/id no sale nada no se si será así](#)

## XQUERY

### 9) Encontrar todas las empresas de Francia en orden alfabético:

for \$c in /Clientes/Cliente[pais = 'Francia'] order by \$c/empresa ascending return \$c/empresa

The screenshot displays the XQuery IDE interface. At the top, the query editor shows the query: `for $c in /Clientes/Cliente[pais = 'Francia'] order by $c/empresa ascending return $c/empresa`. Below the editor, the 'Trajor.xml' file is open, showing XML content with elements like `<facturacion>`, `<Cliente>`, `<empresa>`, and `<pais>`. The 'Result' pane at the bottom left shows three results: `<empresa>Empresa C</empresa>`, `<empresa>Empresa F</empresa>`, and `<empresa>Empresa I</empresa>`. The 'Info' pane on the right provides details about the query execution, including the total time (17.37 ms) and the optimized query: `(for $c_0 in db:text("Trajor", "Francia")/parent::pais/parent::Cliente order by $c_0/empresa empty least return $c_0/empresa)`. The 'Result' section of the info pane shows 'Hit(s): 3 Items'.

Time required: 17.37 ms

40 MB

### **10) Encontrar el promedio de facturación de todas las empresas:**

let \$f := /Clientes/Cliente/facturacion return avg(\$f)

The screenshot shows the XQuery editor with the query: `let $f := /Clientes/Cliente/facturacion return avg($f)`. The editor displays the XML structure of `Traijor.xml`, which contains a `Clientes` element with multiple `Cliente` elements, each having `emp..`, `fact..`, and `pais` elements. The result viewer shows a single result: `78000`. The bottom panel provides detailed information about the query execution, including the total time (1.34 ms) and the optimized query.

**Compiling:**

- inline let \$f\_0 := util:root(./)Clientes/Cliente/facturacion
- simplify FLWOR expression: avg(util:root(./)Clientes/Cliente/facturacion)

**Optimizing:**

- rewrite context value: . -> db:get-pre("Traijor", 0)
- rewrite util:root(nodes): util:root(db:get-pre("Traijor", 0)) -> db:get-pre("Traijor", 0)
- rewrite fn:avg(values): avg(db:get-pre("Traijor", 0)/Clientes/Cliente/facturacion) -> 78000

**Optimized Query:**

78000

**Query:**

Time required: 1.34 ms

### **11) Encontrar todas las empresas cuya facturación es mayor a 500,000 euros y se encuentran en España o Alemania:**

for \$c in /Clientes/Cliente where \$c/pais = ('España', 'Alemania') and \$c/facturacion > 500000 return \$c/empresa

no sale nada porque todas las facturaciones no llegan ni siquiera a 100000 euros.

## 12) Encontrar la suma total de facturación de todas las empresas:

let \$facturaciones := /Clientes/Cliente/facturacion return sum(\$facturaciones)

The screenshot displays the XQuery editor and result viewer. The query is: `let $f := /Clientes/Cliente/facturacion return sum($f)`. The XML file `Traijor.xml` is open, showing a structure with `Clientes` and `Cliente` elements, each containing `facturacion` values. The result viewer shows a single result: `780000`. The bottom panel provides details on the query execution, including compilation and optimization steps.

**Query:** `let $f := /Clientes/Cliente/facturacion return sum($f)`

**Result:** 780000

**Compiling:**

- inline let \$f\_0 := util:root(./)Clientes/Cliente/facturacion
- simplify FLWOR expression: sum(util:root(./)Clientes/Cliente/facturacion)

**Optimizing:**

- rewrite context value: . -> db:get-pre("Traijor", 0)
- rewrite util:root(nodes): util:root(db:get-pre("Traijor", 0)) -> db:get-pre("Traijor", 0)
- rewrite fn:sum(values[,zero]): sum(db:get-pre("Traijor", 0)Clientes/Cliente/facturacion) -> 780000

**Optimized Query:** 780000

The screenshot displays the XQuery IDE interface with the following components:

- Top Panel (XQuery Editor):** Contains the query: `let $p := avg(/Clientes/Cliente/facturacion) for $c in /Clientes/Cliente where $c/facturacion > $p return $c/empresa`. The file `Trajor.xml` is open, showing XML content with client information and average facturation values.
- Left Panel (File Explorer):** Shows the project structure with files like `producto.xml` and `Trajor.xml`.
- Right Panel (Visualizer):** Displays a tree view of the XML document structure, highlighting the `Cliente` and `facturacion` elements.
- Bottom Panel (Results and Logs):**
  - Results:** Shows the output of the query, listing the `empresa` names for clients with facturation greater than the average.
  - Compilation/Optimization:** Displays the XQuery compiler's internal transformations, including predicate rewriting and context value optimization.