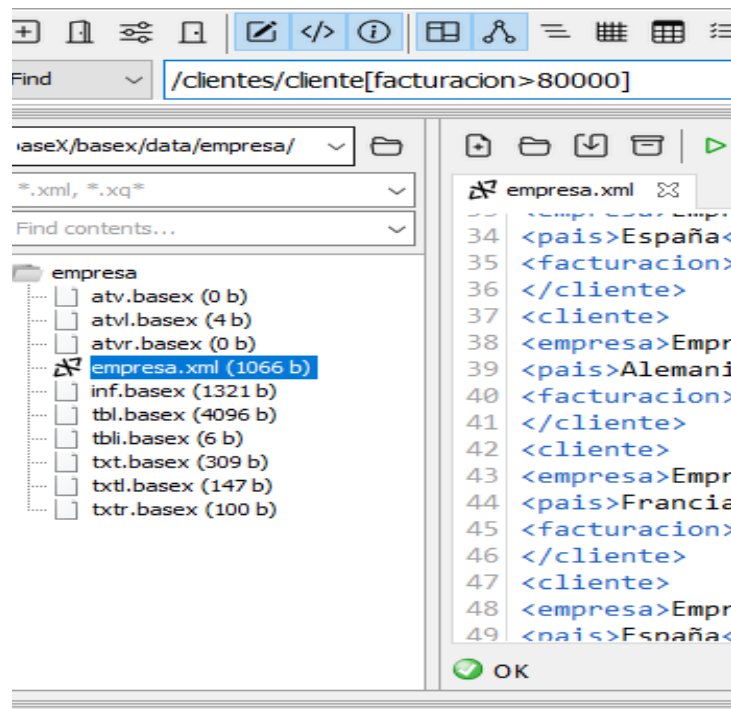


Se hace primero la base de datos.

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



3 Results, 328 b

```
<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 H</empresa>
<pais>Alemania</pais>
<facturacion>95000</facturacion>
</cliente>
```

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

Find

aseX/basex/data/empresa/

empresa

- atv.basex (0 b)
- atvl.basex (4 b)
- atvr.basex (0 b)
- empresa.xml (1066 b)**
- inf.basex (1321 b)
- tbl.basex (4096 b)
- tbli.basex (6 b)
- txt.basex (309 b)
- txtl.basex (147 b)
- txtr.basex (100 b)

empresa.xml

```

30 <facturacion>5000</facturacion>
31 </cliente>
32 <cliente>
33 <empresa>Empresa G</empresa>
34 <pais>España</pais>
35 <facturacion>55000</facturacion>
36 </cliente>
37 <cliente>
38 <empresa>Empresa H</empresa>
39 <pais>Alemania</pais>
40 <facturacion>95000</facturacion>
41 </cliente>
42 <cliente>
43 <empresa>Empresa I</empresa>
44 <pais>Francia</pais>
45 <facturacion>70000</facturacion>

```

OK

3 Results, 325 b

Result

```

<cliente>
<empresa>Empresa A</empresa>
<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>
</cliente>

```

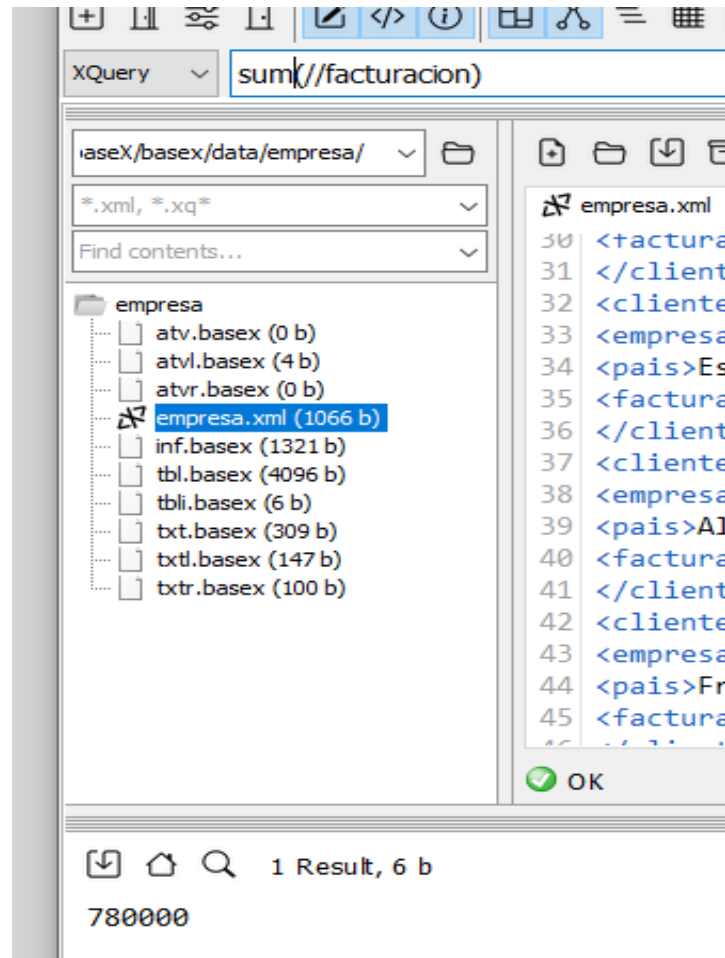
Compiling:

- rewrite >= comparis
- >= 60000.0
- rewrite <= comparis
- <= 90000.0
- simplify and: ((pais =
- and facturacion <= 90
- rewrite to predicate: (
- 60000.0 and facturac

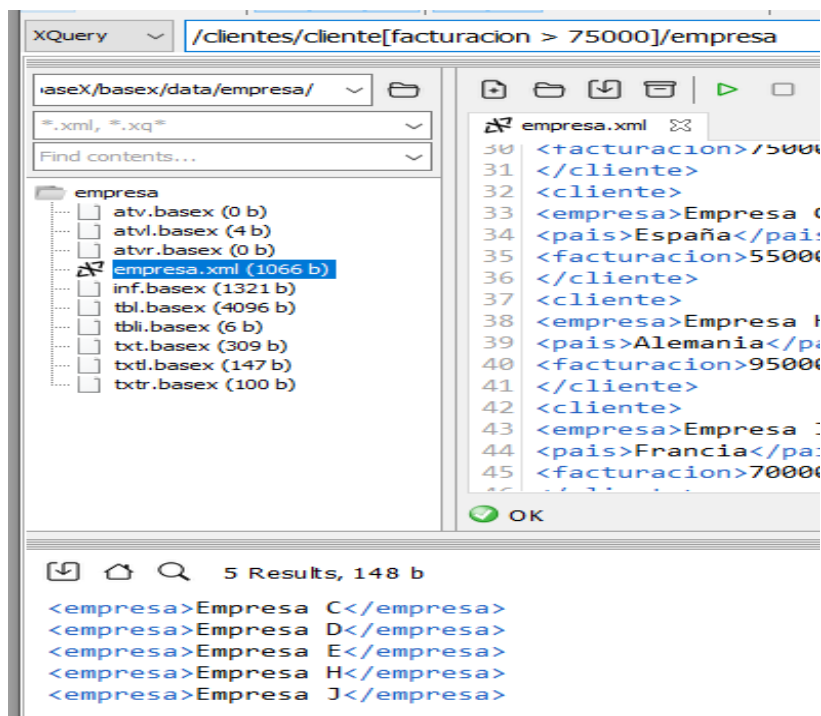
Optimizing:

- rewrite context value
- rewrite util:root(node:
- :get-pre("empresa", 0)
- flatten nested cach
- 90000)/parent::factura
- apply text index for "E

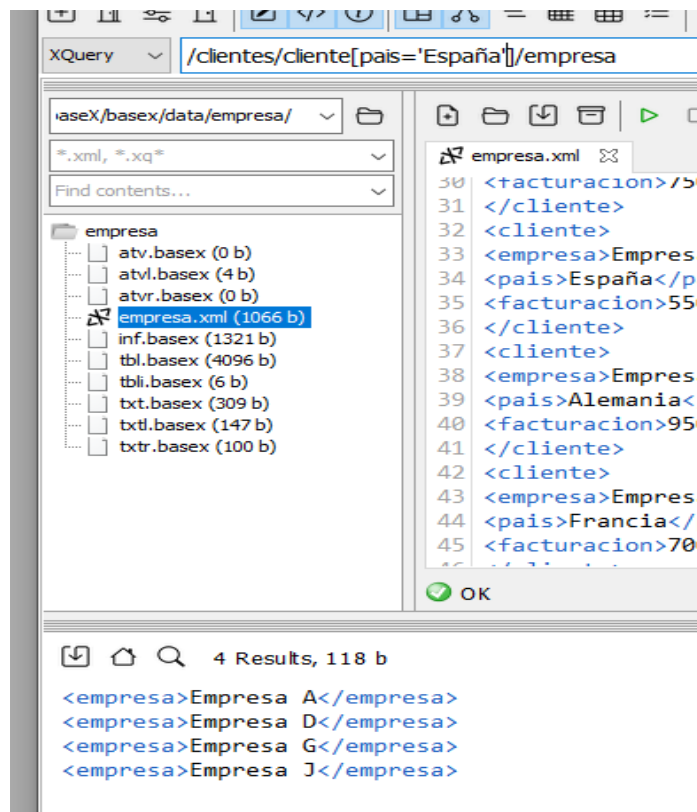
3º Sumar los valores de todos los elementos facturacion en el documento XML.



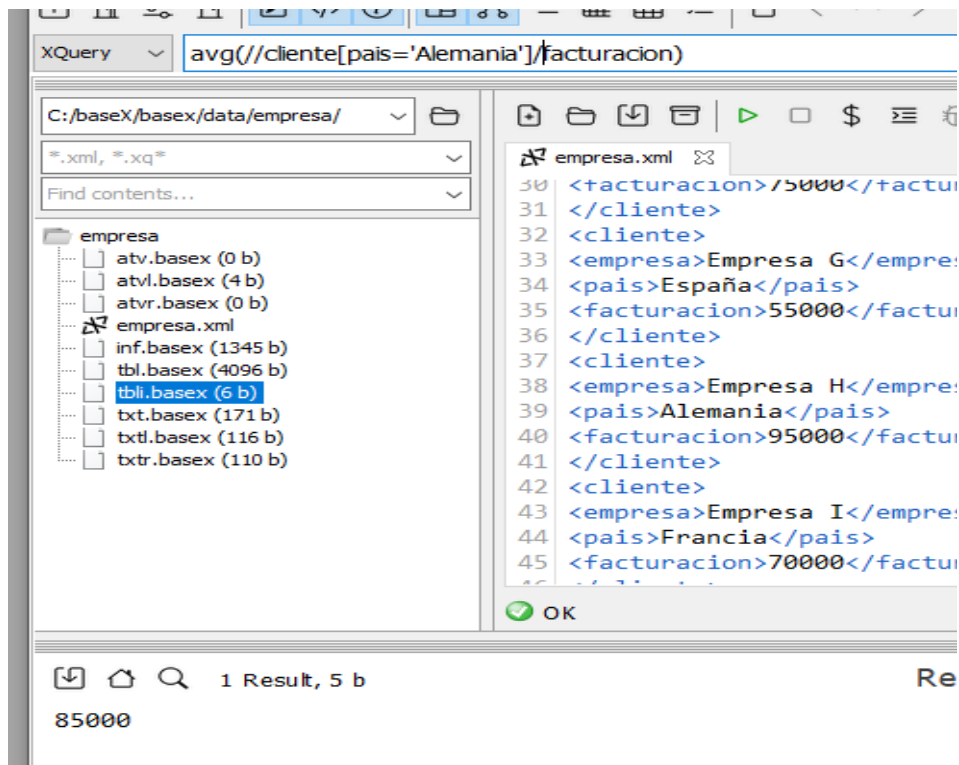
4º Encontrar todas las empresas cuya facturación es mayor a 75000 euros



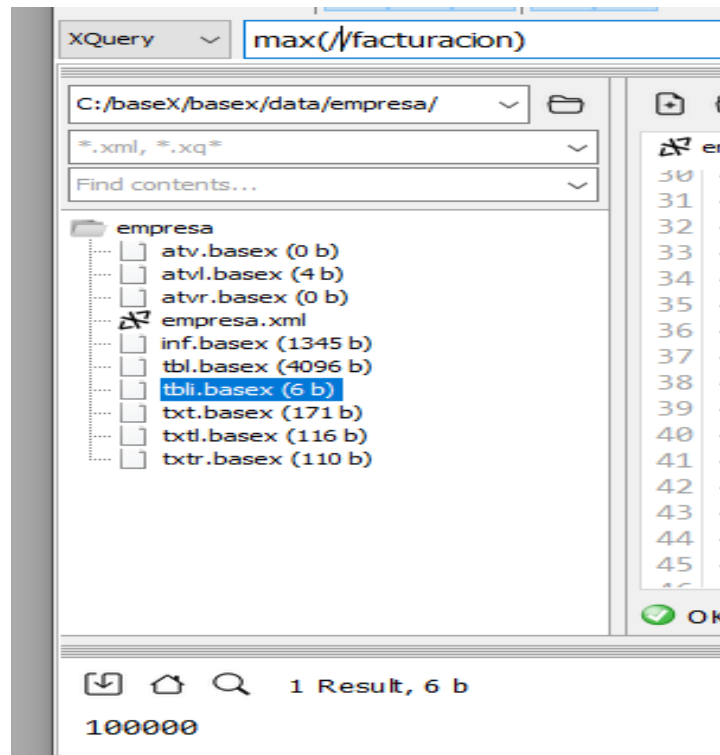
5º Encontrar todas las empresas de España:



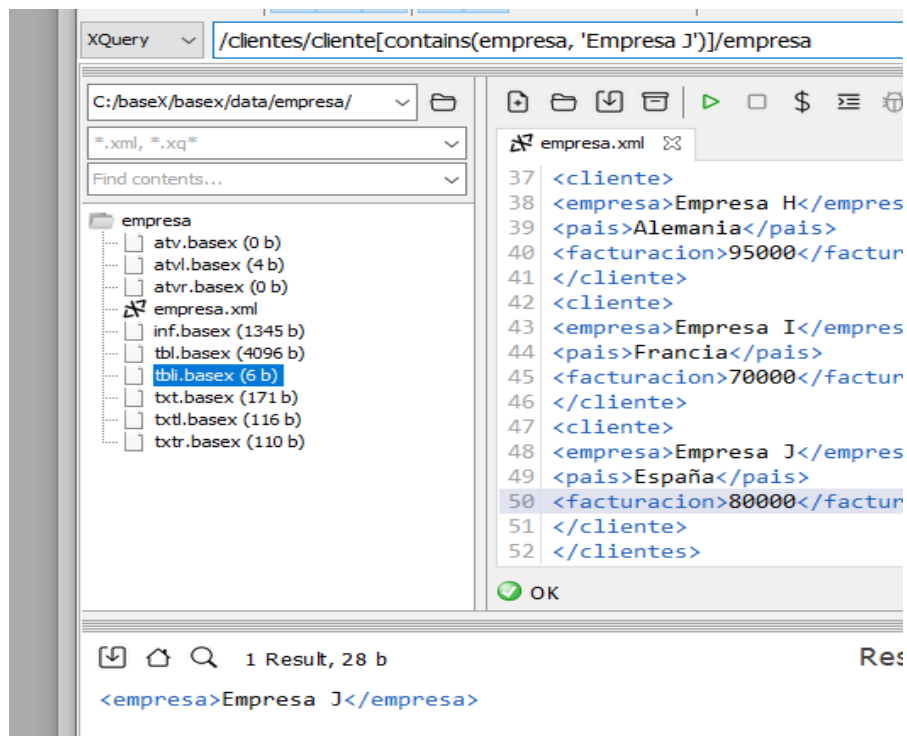
6º Encontrar la facturación promedio de las empresas alemanas:



7º Encontrar la empresa con la facturación más alta:



8º Encontrar el identificador de la empresa "Empresa J":



9º Encontrar todas las empresas de Francia en orden alfabético:

The screenshot shows an XQuery editor with the following query: `for $cliente in //cliente where $cliente/pais='Francia' order by $cliente/empresa return $cliente/empresa`. The file explorer on the left shows a directory structure with files like `atv.basex`, `atvl.basex`, `atvr.basex`, `empresa.xml`, `inf.basex`, `tbl.basex`, `tbli.basex`, `txt.basex`, `txtl.basex`, and `txtr.basex`. The main editor displays the XML content of `empresa.xml`, which lists three companies: `Empresa A` (Spain), `Empresa B` (Germany), and `Empresa C` (France). The results pane at the bottom shows three results: `<empresa>Empresa C</empresa>`, `<empresa>Empresa F</empresa>`, and `<empresa>Empresa I</empresa>`. The status bar indicates 3 Results, 88 b.

Context: db:get("empresa")

Result

3 Results, 88 b

Compiling:
- merge: descendant::cliente
- rewrite to predicate: (pais =

10º Encontrar el promedio de facturación de todas las empresas:

The screenshot shows an XQuery editor with the following query: `let $facturacion := /clientes/cliente return <media>{avg($facturacion/facturacion)}</media>`. The file explorer on the left shows the same directory structure as the previous screenshot. The main editor displays the XML content of `empresa.xml`, which lists three companies: `Empresa A` (Spain), `Empresa B` (Germany), and `Empresa C` (France). The results pane at the bottom shows one result: `<media>78000</media>`. The status bar indicates 1 Result, 20 b.

Context: db:get("emp

Result

1 Result, 20 b

Compiling:
- flatten nested cache

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

Se pone 50000 euros en lugar de 500000 no hay,

The screenshot shows the XQuery editor with the following query: `for $factura in /clientes/cliente[facturacion>=50000 and pais="España" or pais="Alemania"] return $factura/pais`. The editor displays the XML structure of the 'empresa.xml' file, which contains three clients. The result pane shows 7 results, 155 b, listing the countries: España, Alemania, España, Alemania, España, Alemania, and España. The context is set to 'db:get("empresa")'.

Query: `for $factura in /clientes/cliente[facturacion>=50000 and pais="España" or pais="Alemania"] return $factura/pais`

XML Structure (empresa.xml):

```
1 <clientes>
2 <cliente>
3   <empresa>Empresa A</empresa>
4   <pais>España</pais>
5   <facturacion>75000</facturacion>
6 </cliente>
7 <cliente>
8   <empresa>Empresa B</empresa>
9   <pais>Alemania</pais>
10  <facturacion>60000</facturacion>
11 </cliente>
12 <cliente>
13   <empresa>Empresa C</empresa>
14   <pais>Francia</pais>
15   <facturacion>80000</facturacion>
16 </cliente>
```

Result: 7 Results, 155 b

```
<pais>España</pais>
<pais>Alemania</pais>
<pais>España</pais>
<pais>Alemania</pais>
<pais>España</pais>
<pais>Alemania</pais>
<pais>España</pais>
```

Compiling:

- rewrite >= comparison: (facturacion 0
- inline for \$factura_0 in util:root(.) / cliente and (pais = "España") or (pais = "Alemania")
- simplify FLWOR expression: util:root(.) / cliente and (pais = "España") or (pais = "Alemania")

12º Encontrar la suma total de facturación de todas las empresas:

The screenshot shows the XQuery editor with the following query: `let $facturacion := /clientes/cliente return <total>{sum($facturacion/facturacion)}</total>`. The editor displays the XML structure of the 'empresa.xml' file, which contains three clients. The result pane shows 1 result, 21 b, with the total turnover: 780000. The context is set to 'db:get("empresa")'.

Query: `let $facturacion := /clientes/cliente return <total>{sum($facturacion/facturacion)}</total>`

XML Structure (empresa.xml):

```
1 <clientes>
2 <cliente>
3   <empresa>Empresa A</empresa>
4   <pais>España</pais>
5   <facturacion>75000</facturacion>
6 </cliente>
7 <cliente>
8   <empresa>Empresa B</empresa>
9   <pais>Alemania</pais>
10  <facturacion>60000</facturacion>
11 </cliente>
12 <cliente>
13   <empresa>Empresa C</empresa>
14   <pais>Francia</pais>
15   <facturacion>80000</facturacion>
16 </cliente>
17 <cliente>
18   <empresa>Empresa D</empresa>
19   <pais>España</pais>
```

Result: 1 Result, 21 b

```
<total>780000</total>
```

13º Encontrar todas las empresas cuya facturación es mayor a la facturación promedio de todas las empresas:

XQuery

```
let $precio := avg(/clientes/cliente/facturacion)for $company in /clientes/cliente where $company/facturacion >$precio return $company/empresa
```

Context: db:get("empresa")

empresa.xml

```
1 <clientes>
2 <cliente>
3 <empresa>Empresa A</empresa>
4 <pais>España</pais>
5 <facturacion>75000</facturacion>
6 </cliente>
7 <cliente>
8 <empresa>Empresa B</empresa>
9 <pais>Alemania</pais>
10 <facturacion>60000</facturacion>
11 </cliente>
12 <cliente>
13 <empresa>Empresa C</empresa>
14 <pais>Francia</pais>
15 <facturacion>80000</facturacion>
16 </cliente>
17 <cliente>
18 <empresa>Empresa D</empresa>
19 <pais>España</pais>
```

5 Results, 148 b

```
<empresa>Empresa C</empresa>
<empresa>Empresa D</empresa>
<empresa>Empresa E</empresa>
<empresa>Empresa H</empresa>
<empresa>Empresa J</empresa>
```

Result

Compiling:

- rewrite to predicate: (facturacion > \$precio_0)
- inline for \$company_1 in util:root(.) / clientes/cliente[(facturacion > \$precio_0)]

Optimizing:

- rewrite context value: . -> db:get-pre("empresa", 0)