XQuery Language

Erick Gonzalez Parada ID: 178145
Emiliano Ruiz Plancarte ID: 177478
Andre Francois Duhamel Gutierrez ID: 177315
Antonio Gutiérrez Blanco ID: 177442

March 15, 2025

Abstract

This document explores the fundamentals of the XQuery language, showcasing query examples, methodologies, and conclusions drawn from practical applications.

Keywords: XML, DTD, XPath, XQuery, BaseX.

1 Theoretical Framework

XQuery is a powerful and flexible query language designed for querying and transforming XML data. It is often referred to as "SQL for XML" due to its ability to extract and manipulate data stored in XML documents. XQuery is built on XPath expressions and provides additional features such as FLWOR (For, Let, Where, Order by, Return) expressions, which allow for complex data retrieval and transformation tasks [1].

One of the key strengths of XQuery is its ability to handle hierarchical and nested data structures, which are common in XML documents. This makes it particularly useful for applications such as web services, data integration, and content management systems. For example, XQuery can be used to extract specific elements from an XML document, transform the data into a different format, or generate reports based on the data [2].

Another important feature of XQuery is its support for strong typing and schema validation. This ensures that the data being queried adheres to a predefined structure, reducing the risk of errors and improving the reliability of the queries. Additionally, XQuery supports modularity, allowing developers to create reusable modules and functions that can be shared across different projects [3].

Goals

The goal of this lab is to have a first encounter with the XQuery language to examine XML documents.

Materials

- *BaseX*

2 Methodology

The methodology involves writing and executing XQuery scripts to perform the operations on the XML documents. Each query is designed to demonstrate a specific feature or use case of XQuery, such as filtering, sorting, grouping, and transforming data. The results of each query are analyzed to understand the underlying principles and techniques.

3 Query Results

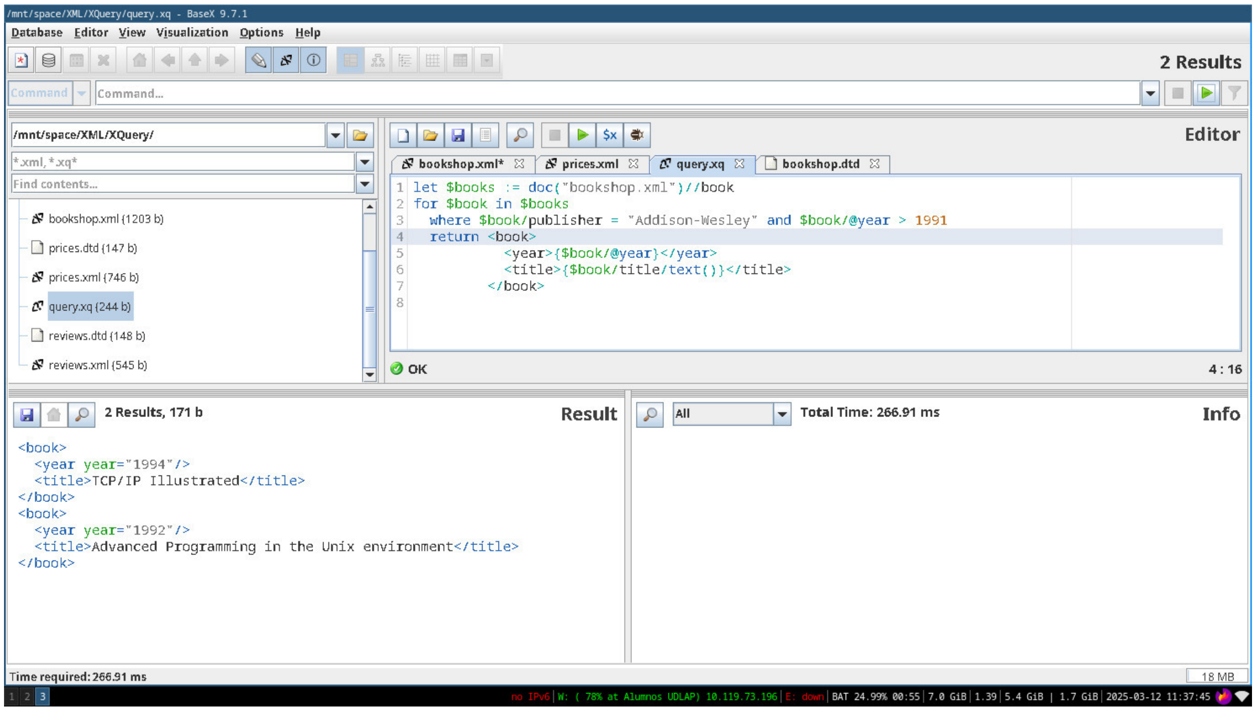
Below are the XQuery scripts and their corresponding results:

Here is the corrected chunk of LaTeX with added captions and explanations for each figure:

1. List books published by "Addison-Wesley" after 1991:

```
        let $books := doc("bookshop.xml")//book
        where $books/publisher = "Addison-Wesley" and $books/@year > 1991
        return
            <book><year>{$books/@year}</year><title>{$books/title/text()}</title></book>
```

Query Result



```
let $books := doc("bookshop.xml")//book
for $book in $books
  where $book/publisher = "Addison-Wesley" and $book/@year > 1991
  return <book>
              <year>{$book/@year}</year>
              <title>{$book/title/text()}</title>
            </book>
```

```
<book>
  <year year="1994"/>
  <title>TCP/IP Illustrated</title>
</book>
<book>
  <year year="1992"/>
  <title>Advanced Programming in the Unix environment</title>
</book>
```

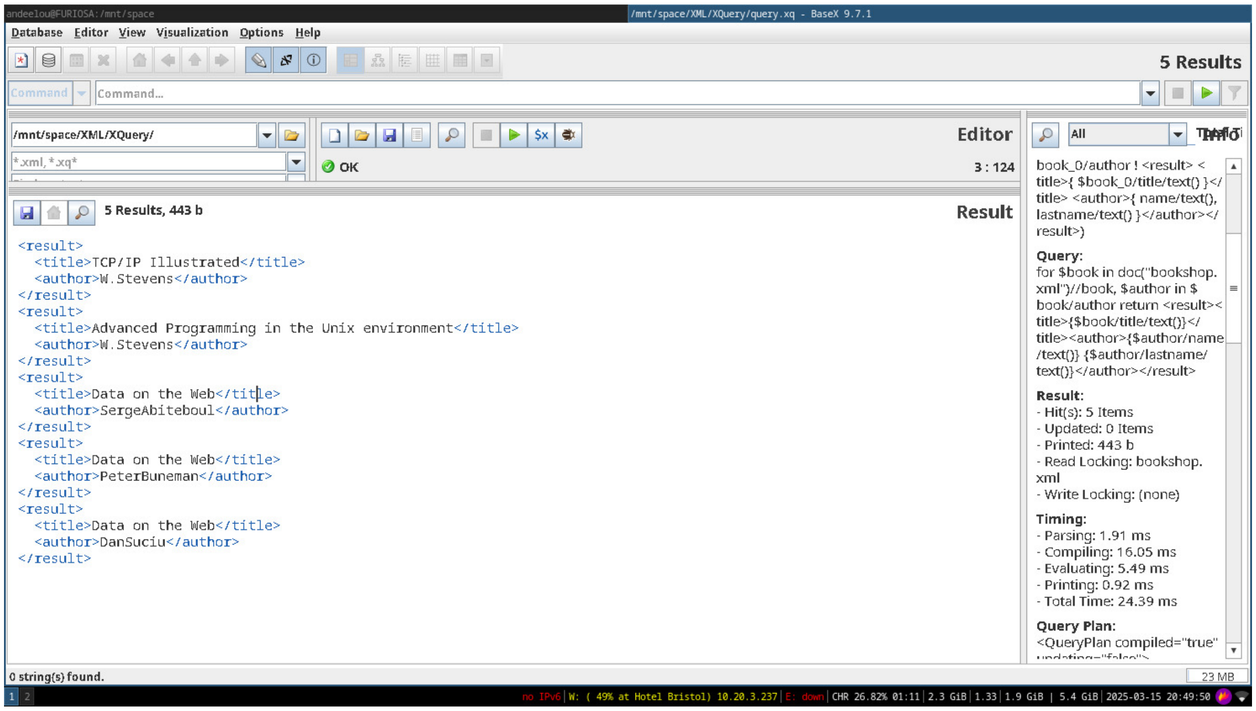
```
no IPv6|W: ( 78% at Alumnos UDLAP) 10.119.73.196|E: down|BAT 24.99% 00:55|7.0 GiB|1.39|5.4 GiB | 1.7 GiB|2025-03-12 11:37:45
```

Figure 1: Books published by Addison-Wesley after 1991.
The query filters books by publisher and year, returning the year and title of each book.

2. Create a flattened list of "result" items with title and author:

```
        for $book in doc("bookshop.xml")//book,
        $author in $book/author
        return
            <result><title>{$book/title/text()}</title><author>{$author/name/text()}
            {$author/lastname/text()}</author></result>
```

Query Result



/mnt/space/XML/XQuery/query.xq - BaseX 9.7.1
Database Editor View Visualization Options Help
5 Results
Editor
OK
3 : 124
Result

5 Results, 443 b

```
<result>
  <title>TCP/IP Illustrated</title>
  <author>W.Stevens</author>
</result>
<result>
  <title>Advanced Programming in the Unix environment</title>
  <author>W.Stevens</author>
</result>
<result>
  <title>Data on the Web</title>
  <author>SergeAbiteboul</author>
</result>
<result>
  <title>Data on the Web</title>
  <author>PeterBuneman</author>
</result>
<result>
  <title>Data on the Web</title>
  <author>DanSuciu</author>
</result>
```

```
book_0/author ! <result> <title>{ $book_0/title/text() }</title> <author>{ name/text(), lastname/text() }</author></result>)

Query:
for $book in doc("bookshop.xml")//book, $author in $book/author return <result><title>{$book/title/text()}</title><author>{$author/name/text()} {$author/lastname/text()}</author></result>

Result:
- Hit(s): 5 Items
- Updated: 0 Items
- Printed: 443 b
- Read Locking: bookshop.xml
- Write Locking: (none)

Timing:
- Parsing: 1.91 ms
- Compiling: 16.05 ms
- Evaluating: 5.49 ms
- Printing: 0.92 ms
- Total Time: 24.39 ms

Query Plan:
<QueryPlan compiled="true"
```

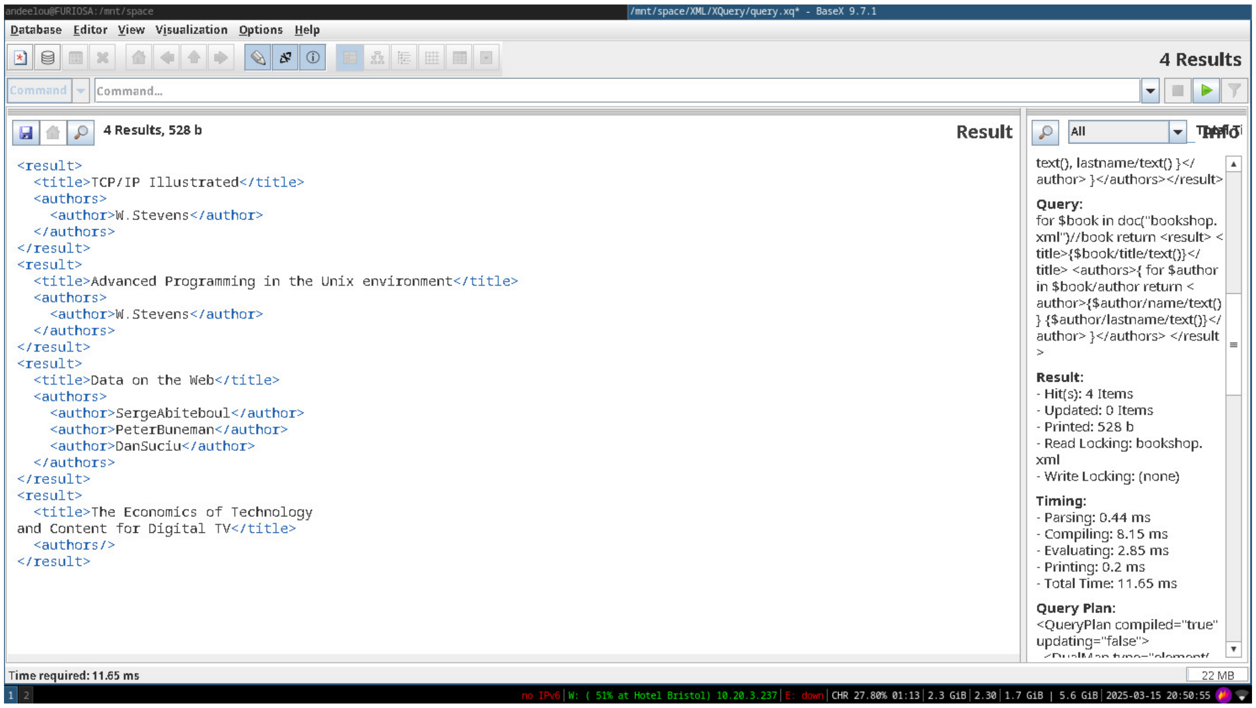
0 string(s) found.

Figure 2: Flattened list of books with titles and authors.
The query iterates over each book and author, creating a flat list of results with titles and authors.

3. List titles with grouped authors:

```
for $book in doc("bookshop.xml")//book
return <result>
<title>{$book/title/text()}</title>
<authors>{
  for $author in $book/author
  return <author>{$author/name/text()}
      {$author/lastname/text()}</author>
}</authors>
</result>
```

Query Result



4 Results, 528 b

```
<result>
  <title>TCP/IP Illustrated</title>
  <authors>
    <author>W.Stevens</author>
  </authors>
</result>
<result>
  <title>Advanced Programming in the Unix environment</title>
  <authors>
    <author>W.Stevens</author>
  </authors>
</result>
<result>
  <title>Data on the Web</title>
  <authors>
    <author>SergeAbiteboul</author>
    <author>PeterBuneman</author>
    <author>DanSuciu</author>
  </authors>
</result>
<result>
  <title>The Economics of Technology
and Content for Digital TV</title>
  <authors/>
</result>
```

```
text(), lastname/text() }</
author> }</authors></result>

Query:
for $book in doc("bookshop.
xml")//book return <result> <
title>{$book/title/text()}</
title> <authors>{ for $author
in $book/author return <
author>{$author/name/text()
} {$author/lastname/text()}</
author> }</authors> </result
>

Result:
- Hit(s): 4 Items
- Updated: 0 Items
- Printed: 528 b
- Read Locking: bookshop.
xml
- Write Locking: (none)

Timing:
- Parsing: 0.44 ms
- Compiling: 8.15 ms
- Evaluating: 2.85 ms
- Printing: 0.2 ms
- Total Time: 11.65 ms

Query Plan:
<QueryPlan compiled="true"
updating="false">
```

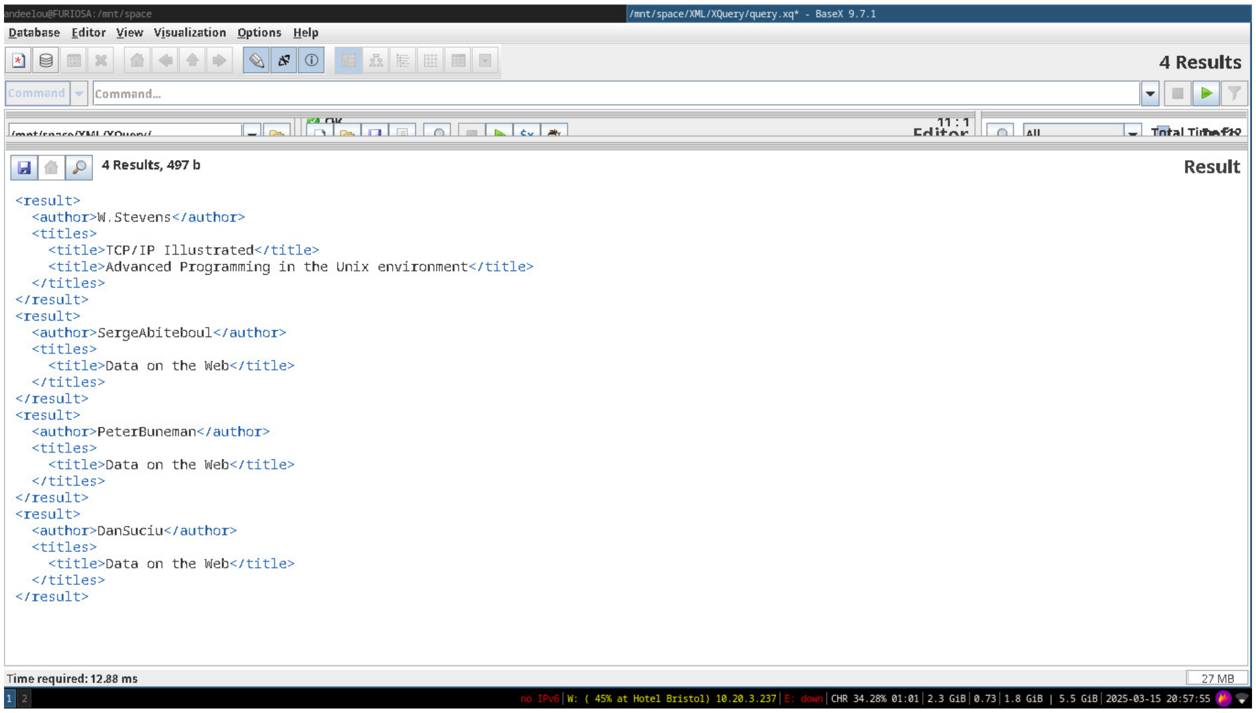
Time required: 11.65 ms

Figure 3: Book titles with grouped authors.
The query groups authors under each book title, creating a nested structure.

4. List authors with their books:

```
for $author in distinct-values(doc("bookshop.xml")//author)
let $books := doc("bookshop.xml")//book[author/name = $author/name and
    author/lastname = $author/lastname]
return <result>
<author>{$author/name/text()} {$author/lastname/text()}</author>
<titles>{
  for $book in $books
  return <title>{$book/title/text()}</title>
}</titles>
</result>
```

Query Result



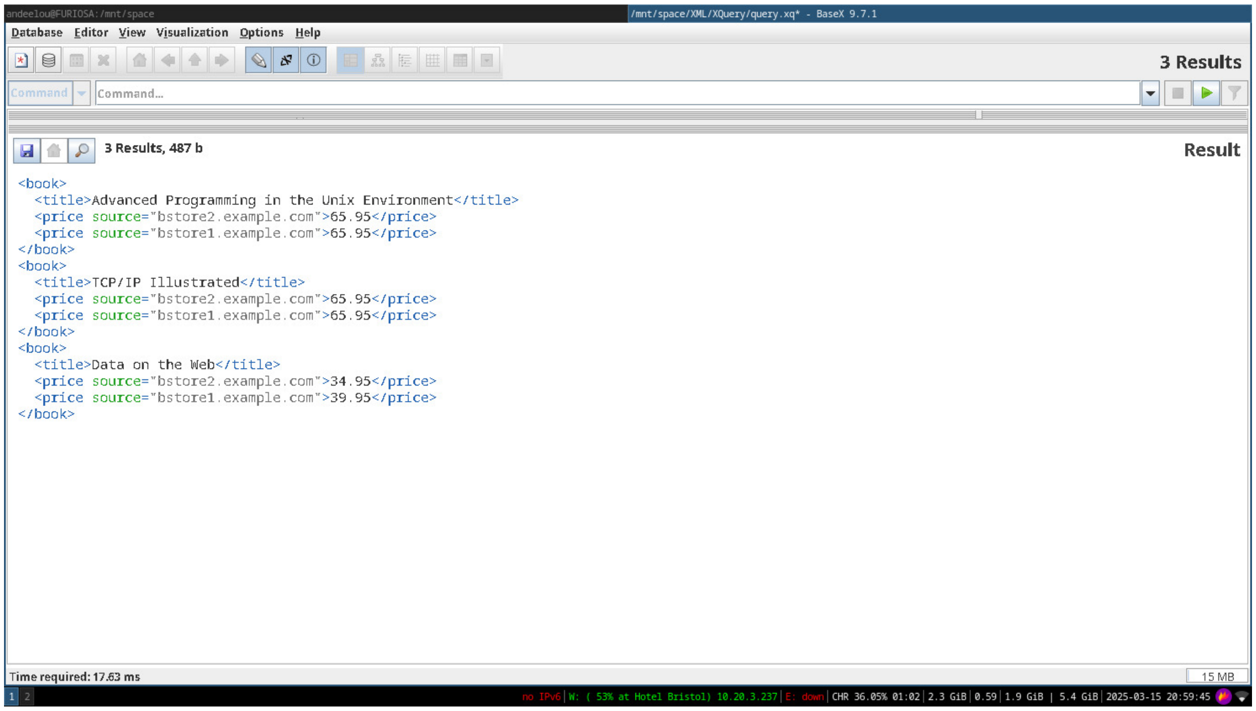
```
<result>
  <author>W.Stevens</author>
  <titles>
    <title>TCP/IP Illustrated</title>
    <title>Advanced Programming in the Unix environment</title>
  </titles>
</result>
<result>
  <author>SergeAbiteboul</author>
  <titles>
    <title>Data on the Web</title>
  </titles>
</result>
<result>
  <author>PeterBuneman</author>
  <titles>
    <title>Data on the Web</title>
  </titles>
</result>
<result>
  <author>DanSuciu</author>
  <titles>
    <title>Data on the Web</title>
  </titles>
</result>
```

Figure 4: Authors with their corresponding books.
The query lists each author along with the titles of their books.

5. Books with number of authors:

```
        for $book in doc("bookshop.xml")//book
        return <book>
        <title>{$book/title/text()}</title>
        {if ($book/author) then
            <number-of-authors>{count($book/author)}</number-of-authors> else
            ()}
        </book>
```

Query Result


/mnt/space/XML/XQuery/query.xq* - BaseX 9.7.1
Database Editor View Visualization Options Help
3 Results
Command...
Result
15 MB

3 Results, 487 b

```
<book>
  <title>Advanced Programming in the Unix Environment</title>
  <price source="bstore2.example.com">65.95</price>
  <price source="bstore1.example.com">65.95</price>
</book>
<book>
  <title>TCP/IP Illustrated</title>
  <price source="bstore2.example.com">65.95</price>
  <price source="bstore1.example.com">65.95</price>
</book>
<book>
  <title>Data on the Web</title>
  <price source="bstore2.example.com">34.95</price>
  <price source="bstore1.example.com">39.95</price>
</book>
```

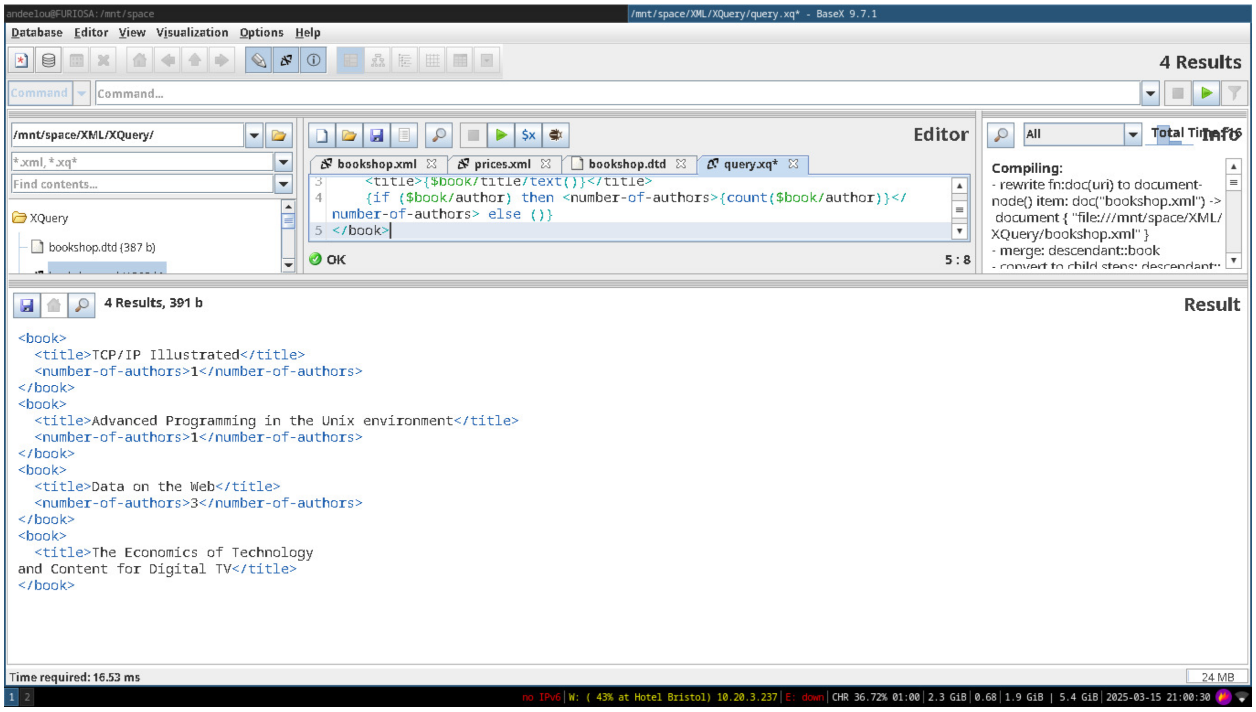
Time required: 17.63 ms

Figure 5: Books with the number of authors.
The query counts the number of authors for each book and includes it in the result.

6. Find minimum price for each book in prices.xml:

```
    for $title in distinct-values(doc("prices.xml")//book/title)
    let $minPrice := min(doc("prices.xml")//book[title = $title]/price)
    return <minimum-price title="{$title}">{$minPrice}</minimum-price>
```

Query Result



```
    <title>{$book/title/text()}</title>
    {if ($book/author) then <number-of-authors>{count($book/author)}</
number-of-authors> else ()}
</book>
```

OK 5 : 8

Compiling:
- rewrite fn:doc(uri) to document-node() item: doc("bookshop.xml") -> document { "file:///mnt/space/XML/XQuery/bookshop.xml" }
- merge: descendant::book

4 Results, 391 b

```
<book>
  <title>TCP/IP Illustrated</title>
  <number-of-authors>1</number-of-authors>
</book>
<book>
  <title>Advanced Programming in the Unix environment</title>
  <number-of-authors>1</number-of-authors>
</book>
<book>
  <title>Data on the Web</title>
  <number-of-authors>3</number-of-authors>
</book>
<book>
  <title>The Economics of Technology
and Content for Digital TV</title>
</book>
```

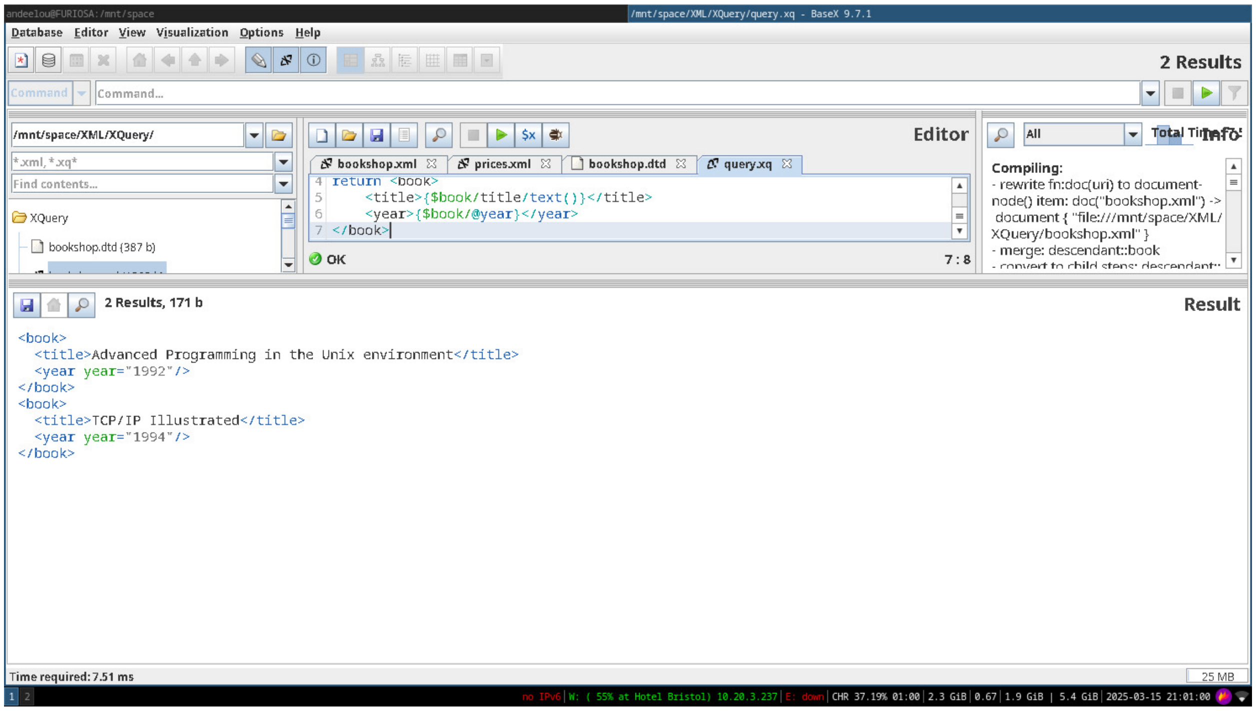
Time required: 16.53 ms

Figure 6: Minimum price for each book.
The query calculates the minimum price for each book across different stores.

7. Alphabetically list Addison-Wesley books after 1991:

```
        for $book in doc("bookshop.xml")//book
        where $book/publisher = "Addison-Wesley" and $book/@year > 1991
        order by $book/title
        return <book>
        <title>{$book/title/text()}</title>
        <year>{$book/@year}</year>
        </book>
```

Query Result



```
    return <book>
        <title>{$book/title/text()}</title>
        <year>{$book/@year}</year>
    </book>
```

OK 7 : 8

Compiling:
- rewrite fn:doc(uri) to document-node() item: doc("bookshop.xml") -> document { "file:///mnt/space/XML/XQuery/bookshop.xml" }
- merge: descendant::book

2 Results, 171 b

```
<book>
  <title>Advanced Programming in the Unix environment</title>
  <year year="1992"/>
</book>
<book>
  <title>TCP/IP Illustrated</title>
  <year year="1994"/>
</book>
```

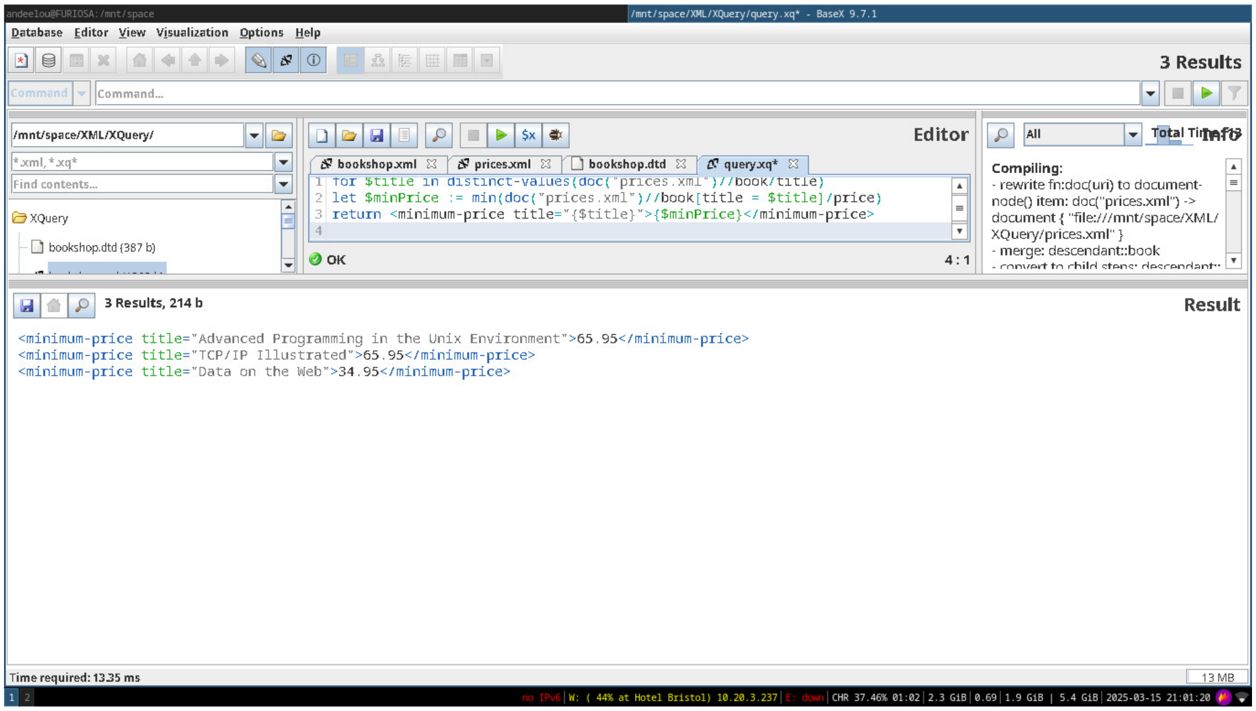
Time required: 7.51 ms

Figure 7: Alphabetically listed Addison-Wesley books after 1991.
The query filters and sorts books by title, showing only those published by Addison-Wesley after 1991.

8. Return book element for authored books, reference for published ones:

```
        for $book in doc("bookshop.xml")//book
        return
        if ($book/author)
        then <book>
        <title>{$book/title/text()}</title>
        <authors>{
          for $author in $book/author
          return <author>{$author/name/text()}
              {$author/lastname/text()}</author>
        }</authors>
        </book>
        else if ($book/editor)
        then <reference>
        <title>{$book/title/text()}</title>
        <affiliation>{$book/editor/affiliation/text()}</affiliation>
        </reference>
        else ()
```

Query Result



```
for $title in distinct-values(doc("prices.xml")//book/title)
let $minPrice := min(doc("prices.xml")//book[title = $title]/price)
return <minimum-price title="{$title}">{$minPrice}</minimum-price>
```

Compiling:
- rewrite fn:doc(uri) to document-node() item: doc("prices.xml") -> document { "file:///mnt/space/XML/XQuery/prices.xml" }
- merge: descendant::book

3 Results, 214 b

```
<minimum-price title="Advanced Programming in the Unix Environment">65.95</minimum-price>
<minimum-price title="TCP/IP Illustrated">65.95</minimum-price>
<minimum-price title="Data on the Web">34.95</minimum-price>
```

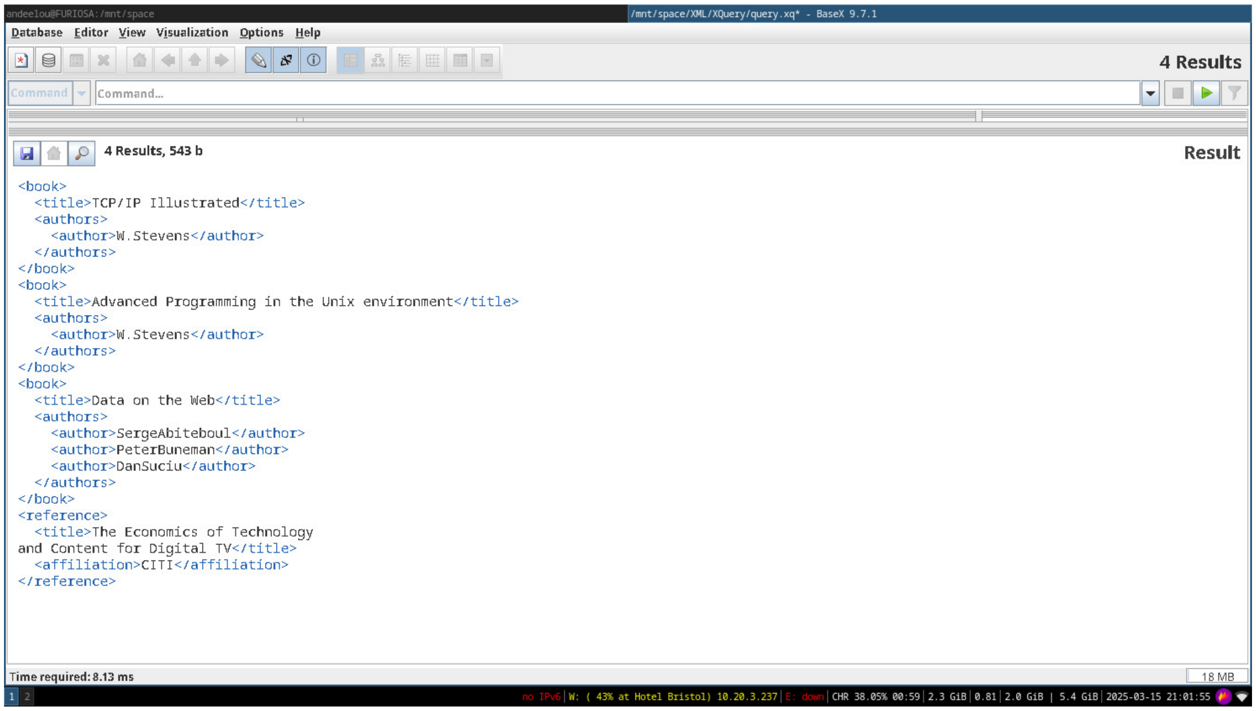
Time required: 13.35 ms

Figure 8: Books and references differentiated by author or editor.
The query differentiates between authored books and edited references, returning different XML structures for each.

9. List book titles with their prices at each bookshop:

```
for $book in distinct-values(doc("prices.xml")//book/title)
return <book>
<title>{$book/text()}</title>
{
  for $store in doc("prices.xml")//book[title = $book]
  return <price
      source="{$store/source/text()}">{$store/price/text()}</price>
}
</book>
```

Query Result


andeelou@FURIOSA:/mnt/space
/mnt/space/XML/XQuery/query.xq* - BaseX 9.7.1
Database Editor View Visualization Options Help
4 Results
Command
Command...
4 Results, 543 b
Result
Time required: 8.13 ms
18 MB

```
<book>
  <title>TCP/IP Illustrated</title>
  <authors>
    <author>W.Stevens</author>
  </authors>
</book>
<book>
  <title>Advanced Programming in the Unix environment</title>
  <authors>
    <author>W.Stevens</author>
  </authors>
</book>
<book>
  <title>Data on the Web</title>
  <authors>
    <author>SergeAbiteboul</author>
    <author>PeterBuneman</author>
    <author>DanSuciu</author>
  </authors>
</book>
<reference>
  <title>The Economics of Technology
and Content for Digital TV</title>
  <affiliation>CITI</affiliation>
</reference>
```

Figure 9: Book titles with prices at each bookshop.
The query lists each book title along with its prices from different stores.

4 Conclusions

Through this exercise, we gained a deeper understanding of the XQuery language and its capabilities in querying and transforming XML data. The practical examples demonstrated the flexibility and power of XQuery in handling complex data retrieval and manipulation tasks. Future work could explore advanced features such as XQuery functions, modules, and integration with other technologies, will ORACLE save us from a painfull dev experiencie with XML?

Our Team experiencie with AI on the activity

We as a team were unable to implement the solutions that the artificial intelligence tried to provide because they still need to mature on archaic technologies, and because this technologies are used in the laboral field, the information of methodologies of using such technologies are not in the range of ChatGPT's or any other AI database.

References

[1] XQuery 3.1: An XML query language. (n.d.). Www.w3.org. Retrieved March 15, 2025, from `https://www.w3.org/TR/xquery-31/`

[2] XQuery language reference (SQL Server). (n.d.). Microsoft.com. Retrieved March 15, 2025, from `https://learn.microsoft.com/en-us/sql/xquery/xquery-language-reference-sql-server?view=sql-server-ver16`

[3] XQuery tutorial. (n.d.). W3schools.com. Retrieved March 15, 2025, from https://www.w3schools.com/xml/xquery_intro.asp