



VIT[®]
Vellore Institute of Technology
(Deemed to be University under section 3 of UGC Act, 1956)

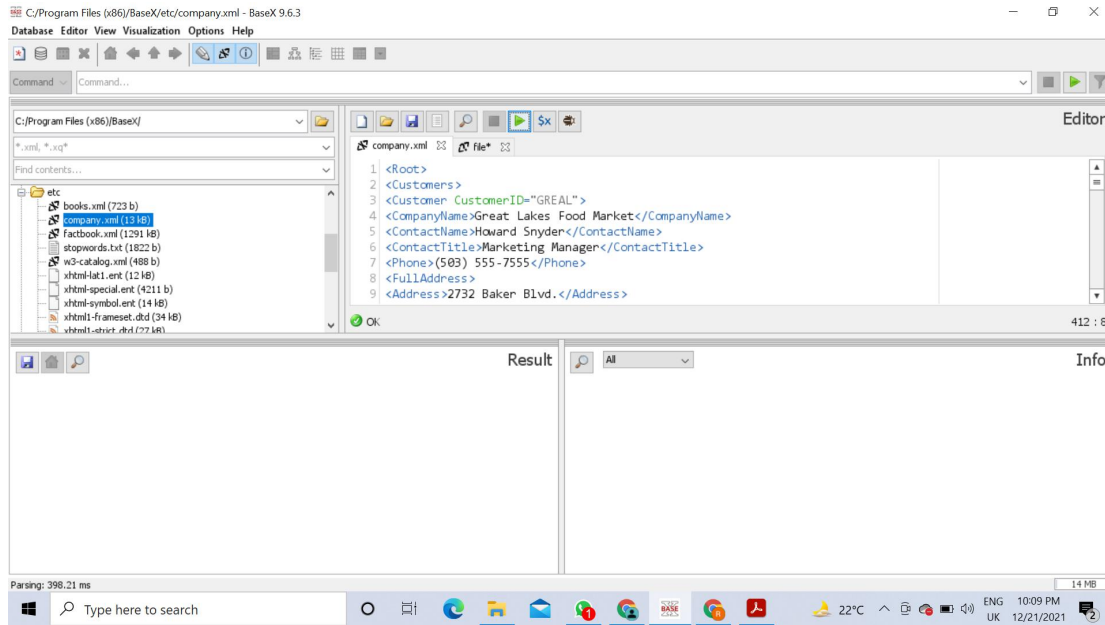
CSE5011-Database Systems and Design

EMBEDDED LAB

XML

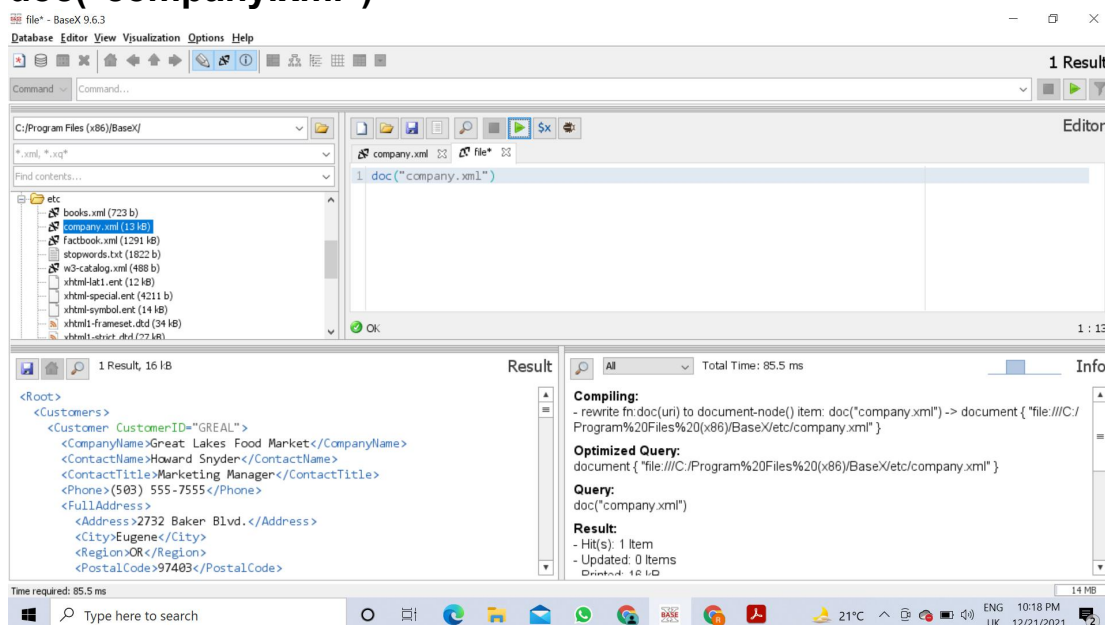
KAUSHAL BAGHEL
21MAI0003

Consider the following XML data for solving the queries:
https://docs.google.com/document/d/1y6x7q6OCdFqKfdcCjRkEPthEVL6b_gVQEAzxs4QqB7w/edit?usp=sharing



Based on the XML data given write XML Query for the following:

1. Read an XML file and its tags
doc("company.xml")



2. Apply Inline filter of an XML file and retrieve the details of 'Great Lakes Food Market'

doc("company.xml")/Root/Customers/Customer[CompanyName="Great Lakes Food Market"]

The screenshot shows the BaseX 9.6.3 application window. The 'Editor' pane displays the XPath query: `1 doc("company.xml")/Root/Customers/Customer[CompanyName="Great Lakes Food Market"]`. The 'Result' pane shows the XML output for the first result (CustomerID="GREAL"):

```
<Customer CustomerID="GREAL">
  <CompanyName>Great Lakes Food Market</CompanyName>
  <ContactName>Howard Snyder</ContactName>
  <ContactTitle>Marketing Manager</ContactTitle>
  <Phone>(583) 555-7555</Phone>
  <FullAddress>
    <Address>2732 Baker Blvd.</Address>
    <City>Eugene</City>
    <Region>OR</Region>
    <PostalCode>97403</PostalCode>
    <Country>USA</Country>
  </FullAddress>
</Customer>
```

The 'Info' pane shows the following details:

- Compiling:** - rewrite fn:doc(uri) to document-node() item: doc("company.xml") -> document { "file:///C:/Program%20Files%20(x86)/BaseX/etc/company.xml" }
- apply text index for "Great Lakes Food Market"
- Optimized Query:** db:text("company", "Great Lakes Food Market")/parent: CompanyName/parent: Customer
- Query:** doc("company.xml")/Root/Customers/Customer[CompanyName="Great Lakes Food Market"]
- Result:** - Hit(s): 1 item
- Updated: 0 items

The status bar at the bottom indicates the time required was 20.03 ms.

3. Apply Inline filter of an XML file and retrieve the details of 'LETSS'

doc("company.xml")/Root/Customers/Customer[@CustomerID = 'LETSS']

The screenshot shows the BaseX 9.6.3 application window. The 'Editor' pane displays the XPath query: `1 doc("company.xml")/Root/Customers/Customer[@CustomerID="LETSS"]`. The 'Result' pane shows the XML output for the first result (CustomerID="LETSS"):

```
<Customer CustomerID="LETSS">
  <CompanyName>Let's Stop N Shop</CompanyName>
  <ContactName>Jaime Yonnes</ContactName>
  <ContactTitle>Owner</ContactTitle>
  <Phone>(415) 555-5938</Phone>
  <FullAddress>
    <Address>87 Polk St. Suite 5</Address>
    <City>San Francisco</City>
    <Region>CA</Region>
    <PostalCode>94117</PostalCode>
    <Country>USA</Country>
  </FullAddress>
</Customer>
```

The 'Info' pane shows the following details:

- Compiling:** - rewrite fn:doc(uri) to document-node() item: doc("company.xml") -> document { "file:///C:/Program%20Files%20(x86)/BaseX/etc/company.xml" }
- apply attribute index for "LETSS"
- Optimized Query:** db:attribute("company", "LETSS")/self::attribute(CustomerID)/parent: Customer
- Query:** doc("company.xml")/Root/Customers/Customer[@CustomerID="LETSS"]
- Result:** - Hit(s): 1 item
- Updated: 0 items
- Printed: 404 b
- Read Locking: company.xml
- Write Locking: (none)

The status bar at the bottom indicates the time required was 114.86 ms.

4. Find all the orders for the CA region
doc("company.xml")/Root/Orders/Order[ShipInfo/ShipRegion="CA"]

The screenshot shows the BaseX 9.6.3 application. The Command window contains the XPath query: `doc("company.xml")/Root/Orders/Order[ShipInfo/ShipRegion="CA"]`. The Editor window shows the XML document structure. The Result window displays 4 results, 2194 bytes, showing the XML structure of the orders for the CA region. The Info window shows the Query Plan and timing information.

Command: `doc("company.xml")/Root/Orders/Order[ShipInfo/ShipRegion="CA"]`

Result: 4 Results, 2194 b

```
<Order>
  <CustomerID>LETSS</CustomerID>
  <EmployeeID>1</EmployeeID>
  <OrderDate>1997-06-25T00:00:00</OrderDate>
  <RequiredDate>1997-07-23T00:00:00</RequiredDate>
  <ShipInfo ShippedDate="1997-07-04T00:00:00">
    <ShipVia>2</ShipVia>
    <Freight>13.73</Freight>
    <ShipName>Let's Stop N Shop</ShipName>
    <ShipAddress>87 Polk St. Suite 5</ShipAddress>
    <ShipCity>San Francisco</ShipCity>
    <ShipRegion>CA</ShipRegion>
    <ShipPostalCode>94117</ShipPostalCode>
    <ShipCountry>USA</ShipCountry>
  </ShipInfo>
</Order>
```

Info: Total Time: 14.85 ms

- Evaluating: 1.43 ms
- Printing: 0.66 ms
- Total Time: 14.85 ms

Query Plan:

```
<QueryPlan compiled="true" updating="false">
  <CachedPath type="element(Order)" database="company">
    <ValueAccess index="text" name="element(ShipRegion)" type="element()" database="company">
      <IndexStaticDb type="item()" database="company">
        <Str type="xs:string" size="1">CA</Str>
      </ValueAccess>
    <IterStep axis="parent" test="ShipInfo" type="element(ShipInfo)"/>
    <IterStep axis="parent" test="Order" type="element(Order)"/>
  </CachedPath>
</QueryPlan>
```

5. Order the output by the freight(Weight of the cargo)
**for \$x in doc("company.xml")/Root/Orders/Order
 order by xs:decimal(\$x/ShipInfo/Freight)
 return \$x/ShipInfo/Freight**

The screenshot shows the BaseX 9.6.3 application. The Command window contains the XQuery query: `for $x in doc("company.xml")/Root/Orders/Order order by xs:decimal($x/ShipInfo/Freight) return $x/ShipInfo/Freight`. The Editor window shows the XML document structure. The Result window displays 22 results, 568 bytes, showing the sorted freight values. The Info window shows the Query Plan and timing information.

Command: `for $x in doc("company.xml")/Root/Orders/Order order by xs:decimal($x/ShipInfo/Freight) return $x/ShipInfo/Freight`

Result: 22 Results, 568 b

```
<Freight>0.2</Freight>
<Freight>3.35</Freight>
<Freight>4.42</Freight>
<Freight>7.48</Freight>
<Freight>11.92</Freight>
<Freight>13.73</Freight>
<Freight>14.01</Freight>
<Freight>18.53</Freight>
<Freight>18.84</Freight>
<Freight>20.12</Freight>
<Freight>25.19</Freight>
<Freight>30.34</Freight>
<Freight>33.68</Freight>
<Freight>45.13</Freight>
<Freight>45.97</Freight>
<Freight>51.44</Freight>
```

Info: Total Time: 31.45 ms

- Hit(s): 22 Items
- Updated: 0 Items
- Printed: 568 b
- Read Locking: company.xml
- Write Locking: (none)

Timing:

- Parsing: 0.8 ms
- Compiling: 27.72 ms
- Evaluating: 2.66 ms
- Printing: 0.26 ms
- Total Time: 31.45 ms

Query Plan:

```
for $x in doc("company.xml")/Root/Orders/Order order by xs:decimal($x/ShipInfo/Freight)
return $x/ShipInfo/Freight
```

6. Insert xml data in html format for presentation by using a list

```
<html>
<body>
<h1>COMAPNY</h1>
<ul>
{
for $x in doc("company.xml")/Root/Customers/Customer
order by $x/CompanyName
return <li>{data($x/CompanyName)}.CustomerID:
{data($x/@CustomerID)} </li>
}
</ul>
</body>
</html>
```

The screenshot shows the BaseX 9.6.3 application window. The 'Editor' pane contains the following XQuery:

```
1 <html>
2 <body>
3 <h1>COMAPNY</h1>
4 <ul>
5 {
6 for $x in doc("company.xml")/Root/Customers/Customer
7 order by $x/CompanyName
8 return <li>{data($x/CompanyName)}.CustomerID:
9 {data($x/@CustomerID)} </li>
10 }
11 </ul>
12 </body>
13 </html>
```

The 'Result' pane shows the output of the query:

```
<html>
<body>
<h1>COMAPNY</h1>
<ul>
<li>Great Lakes Food Market.CustomerID: GREAL</li>
<li>Hungry Coyote Import Store.CustomerID: HUNGC</li>
<li>Lazy K Kountry Store.CustomerID: LAZYK</li>
<li>Let's Stop N Shop.CustomerID: LETSS</li>
</ul>
</body>
</html>
```

The 'Info' pane displays the following information:

- Compiling:** - rewrite fn:doc(uri) to document-node() item: doc("company.xml") -> document { "file:///C:/Program%20Files%20(x86)/BaseX/etc/company.xml" }
- Optimized Query:**

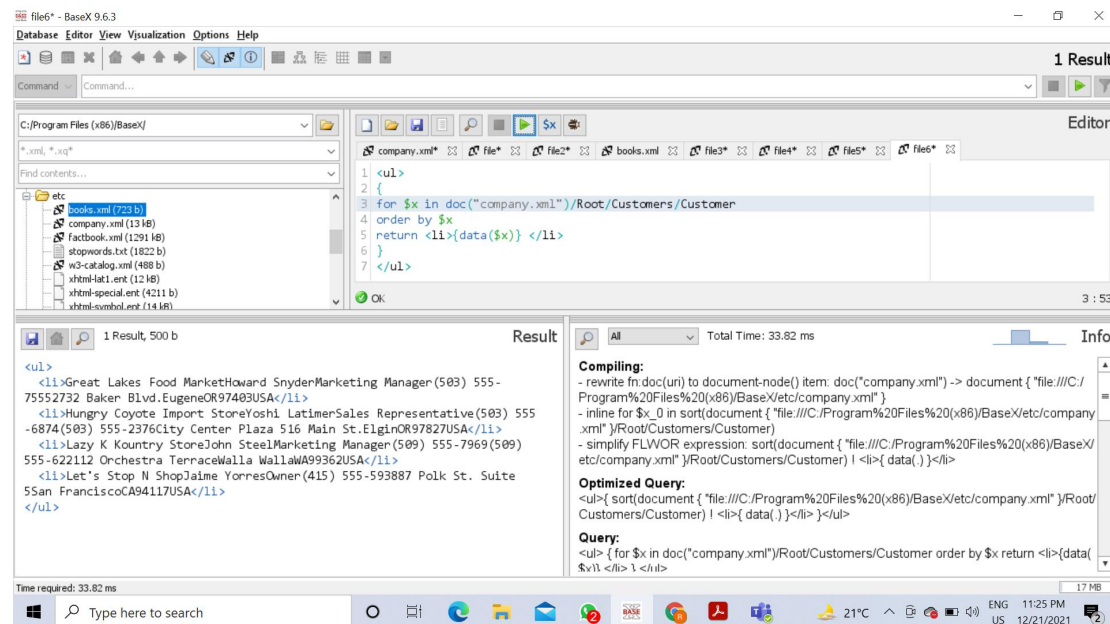
```
<html> <body> <h1>COMAPNY</h1> <ul>{ (for $x_0 in document { "file:///C:/Program%20Files%20(x86)/BaseX/etc/company.xml" }/Root/Customers/Customer order by $x_0/CompanyName empty least return <li>{ data($x_0/CompanyName), ".CustomerID: ", data($x_0/@CustomerID) }</li> }</ul></body></html>
```
- Query:**

```
<html> <body> <h1>COMAPNY</h1> <ul>{ for $x in doc("company.xml")/Root/Customers/ Customer order by $x/CompanyName return <li>{data($x/CompanyName)}.CustomerID: { data($x/@CustomerID)} </li> } </ul> </body> </html>
```

The status bar at the bottom indicates 'Time required: 27.34 ms' and '22 MB'.

7. Insert only the data from the xml ignoring the xml tags. Hine: USE data() function

```
<ul>
{
for $x in doc("company.xml")/Root/Customers/Customer
order by $x
5return <li>{data($x)} </li>
}
</ul>
```



8. Use Conditional statement and retrieve the results for if the freight weight is above 10

(10000 Metric Tons)then it is tagged as a heavy load else a light load

```
for $x in doc("company.xml")/Root/Orders/Order
return if ($x/ShipInfo/Freight>10)
then <heavyload>{data($x/ShipInfo/Freight)} </heavyload>
else <lightload>{data($x/ShipInfo/Freight)} </lightload>
```

file5* - BaseX 9.6.3

Database Editor View Visualization Options Help

Command: Command...

22 Results

Editor

```
1 for $x in doc("company.xml")/Root/Orders/Order
2 return if ($x/ShipInfo/Freight>10)
3 then <heavyload>{data($x/ShipInfo/Freight)} </heavyload>
4 else <lightload>{data($x/ShipInfo/Freight)} </lightload>
```

OK

4 : 42

22 Results, 656 b

Result

```
<lightload>3.35</lightload>
<lightload>4.42</lightload>
<heavyload>116.53</heavyload>
<heavyload>18.53</heavyload>
<heavyload>57.15</heavyload>
<heavyload>76.13</heavyload>
<heavyload>719.78</heavyload>
<heavyload>33.68</heavyload>
<heavyload>25.19</heavyload>
<heavyload>18.84</heavyload>
<heavyload>14.01</heavyload>
<heavyload>20.12</heavyload>
<heavyload>30.34</heavyload>
<lightload>0.2</lightload>
<heavyload>45.13</heavyload>
<heavyload>111.79</heavyload>
```

Info

heavyload={ data(ShipInfo/Freight) }</heavyload> else <lig...

Optimized Query:

```
document { "file:///C:/Program%20Files%20(x86)/BaseX/etc/company.xml" }/Root/Orders/
Order | (if(ShipInfo/Freight >= 10.000000000000002) then <heavyload>{ data(ShipInfo/
Freight) }</heavyload> else <lightload>{ data(ShipInfo/Freight) }</lightload>)
```

Query:

```
for $x in doc("company.xml")/Root/Orders/Order return if ($x/ShipInfo/Freight>10) then <
heavyload>{data($x/ShipInfo/Freight)} </heavyload> else <lightload>{data($x/ShipInfo/Freight
)} </lightload>
```

Result:

- Hit(s): 22 Items
- Updated: 0 Items
- Printed: 656 b
- Read Locking: company.xml
- Write Locking: none

Time required: 18.44 ms

22 MB

ENG 6:47 PM 12/22/2021