# A Collection of XML Documents and Query Tasks

Joris Graaumans

August 2005,
institute of information and computing sciences, utrecht university
technical report UU-CS-2005-038
www.cs.uu.nl

# Table of Contents

# 1. Introduction

This technical report is an external appendix of "Usability of XML Query Languages", Chapter 6 [Gra05]. This chapter describes an experiment that is conducted to explain the causes of usability differences between XSLT and XQuery. The following possible reasons for performance differences are mentioned: the complexity of query tasks, the structure of XML documents, the verboseness of language expressions, the different methods of embedding Xpath in XSLT and XQuery, user experience, and the number of expressions that are necessary for a particular query task. These factors are quantified for a large set of query tasks and the influence is determined of these factors on the performance with a query language.

This report contains the large set of query tasks used in the experiment. In addition, the complexity of each query task is determined. Although the collection presented here is developed for the usability experiment in [Gra05], we hope that the XML documents, query tasks, and the method of determining the complexity of a query task may be useful for other research as well. We provide a short introduction to the queries and their related properties.

For the experiment, 64 query tasks were selected from multiple sources. Our criterion for selection was that the query task must be suitable for novice users of XSLT and XQuery. This means that queries that we assessed were too complex for the current level of expertise of the subjects, were not selected for our experiment. The queries were applied to two types of documents, namely a document-oriented XML document and a data-oriented XML document. The document-oriented XML document is based on sgml.xml from the W3C XML Query Use Cases: Use Case "SGML": Standard Generalized Markup Language. [W3C-QU]. The data-oriented document is based on prices.xml from the W3C XML Query Use Cases: Use Case "XMP": Experiences and Exemplars, Query 10. [W3C-QU]. We selected queries from the following sources:

- W3C XML Query Use Cases [W3C-QU].
- W3 School XQuery tutorial [W3 School].
- Xmark Benchmark [SWK+02].
- Xbench family of benchmarks, text-centric multi-document collection and the data-centric single-document collection [YOK02].
- The queries from our previous experiment.

After we selected the queries, we established for each query the relevant properties. This was done as follows. First, we created for each query one or more correct solutions in both XSLT and XQuery. Of course, for each query a large number of different solutions can be made, but we tried to find only natural and likely solutions and we checked the likelihood of a particular solution both with the original solution given in the source, as well as with the result queries from our previous experiment. Second, we established for each query the properties of the solutions and this was done as follows:

1. *Complexity:* we counted the number of elements, attributes and the values that should be returned with the query, as well as the number of conditions that were used in the query. In the remainder of this document, we refer to this as the report items and the condition items. If the value of an element or attribute of a subquery was used in a condition, than these values were counted twice to represent the complexity of subqueries in a query. The complexity of a query is independent of language and document schema.
2. *XML Document type:* we used two types of XML documents:
   a. A document-oriented XML document, see Section 2.65.1, “Document-oriented”.
   b. A data-oriented XML document, see Section 2.65.2, “Data-oriented”.

   We created a solution for each type of document, so each query was applied to both types of documents.
3. *Number of expressions:* we counted for each query and each language the number of XQuery keywords and the number of XSLT elements respectively. For XSLT queries we excluded the element xsl:transform (or xsl:stylesheet) because this is a standard element for each XSLT query. Most queries had multiple solutions, and in that case we took the mean number of expressions.
4. *Verboseness:* we counted the number of characters in each query for each language after removing the tabs, line ends and double spaces from the expressions. We used the mean values when multiple solutions were made.
5. *Xpath embedding:* we counted for each solution and for each language:
   a. Number of Xpath expressions: that is, the total number of Xpath expressions embedded in an XQuery or XSLT query;
   b. Xpath length: that is, the total number of characters in Xpath expressions embedded in an XQuery or XSLT query.

The number of XSLT elements, XPath expressions and the length of Xpath expressions were determined automatically. For XQuery, the number of expressions was determined manually, the number of Xpath expressions were determined automatically by counting the character '$' in each query. The Xpath expressions were manually selected from the queries, the number of characters was determined automatically.

We provide one extended example of how the query properties were assessed in Section 2.1.2, “Query 1t: Text XML”. For the other queries in this document, we provide:

1. The source of the query task;
2. Query task descriptions for a text-oriented and data-oriented XML document;
3. Correct solutions for both XSLT and XQuery;
4. The complexity of the query task and how this complexity was assessed.

# 2. Query tasks

## 2.1. Query 1

### 2.1.1. Source

W3C Use Case "XMP": Experiences and Exemplars, Query 1 [http://www.w3.org/TR/xquery-use-cases/#xmp-queries-results-q1]:

*List books published by Addison-Wesley after 1991, including their year and title.*

Solution in XQuery:

```
<bib>
 {
  for $b in /bib/book
  where $b/publisher = "Addison-Wesley" and $b/@year > 1991
  return
    <book year="{ $b/@year }">
     { $b/title }
    </book>
 }
</bib>
```

## 2.1.2. Query 1t: Text XML

*Select the chapter that has a number higher than 1 and with title "Getting to know SGML", include the number of the chapter as an attribute and the intro of this chapter.*

The result of this query should be:

```
<chapter number="2">
<intro>
 <para>While SGML is a fairly recent technology, the use of
 <emph>markup</emph> in computer-generated documents has existed for a
 while.</para>
</intro>
</chapter>
```

### 2.1.2.1. XQuery solution

```
for $t in //chapter
where $t/@number > 1 and $t/title="Getting to know SGML"
return
<chapter number="{$t/@number}">
 {$t/intro}
</chapter>
```

```
for $t in //chapter[title="Getting to know SGML" and @number > 1 ]
return
<chapter number="{$t/@number}">
 {$t/intro}
</chapter>
```

### 2.1.2.2. XSLT solution

```
<xsl:transform xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
version="2.0">
<xsl:template match="text()" />
<xsl:output method="xml" indent="yes" />

<xsl:template match="chapter[@number &gt; 1][title='Getting to know SGML']">
   <chapter number="{@number}">
    <xsl:copy-of select="intro" />
   </chapter>
</xsl:template>
</xsl:transform>
```

```
<xsl:transform xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
version="2.0">
<xsl:output method="xml" indent="yes" />
<xsl:template match="report">
   <xsl:for-each select="chapter">
 <xsl:if test="@number &gt; 1 and title='Getting to know SGML'">
      <chapter number="{@number}">
   <xsl:copy-of select="intro" />
    </chapter>
   </xsl:if>
</xsl:for-each>
</xsl:template>
</xsl:transform>
```

```
<xsl:transform xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
version="2.0">
<xsl:output method="xml" indent="yes" />
<xsl:template match="//chapter[title = 'Getting to know SGML']">
<xsl:if test="@number > 1">
<chapter>
<xsl:attribute name="number">
<xsl:value-of select="@number"/>
</xsl:attribute>
<xsl:copy-of select="intro" />
</chapter>
</xsl:if>
</xsl:template>
</xsl:transform>
```

### 2.1.2.3. Relevant properties

The relevant properties for this query task are established as follows:

1. *Complexity*: the complexity of this query task is the sum of the report and the condition items. Items are elements, attributes and their values. Values from subqueries are counted twice to model the complexity of subqueries in a query. The following elements and attributes are counted:
   - The number of report items is 3, namely 1 chapter element, 1 number attribute and 1 intro element.
   - The number of condition items is 4, namely 1 title with 1 value "Getting to know SGML" , 1 number attribute with 1 value higher than 1.

   The complexity of this query task is therefore 7.
2. *Data structure:* this query is defined for the document-oriented XML document.
3. *Number of expressions:*
   - XQuery: the first XQuery solution has 4 keywords (for where and return), the second solution has 3 keywords (for and return). Therefore, the number of expressions for XQuery on this task is 3,5.
   - XSLT: the first solution consists of 4 XSLT elements (2 x xsl:template, 1 x xsl:output, 1 x xsl:copy-of), the second solution of 5 XSLT elements and the third solution of 6 XSLT elements. Therefore, the number of expressions for XSLT is: (4 + 5 + 6) / 3 = 5.
4. *Verboseness:*
   - XQuery: the first solution consists of 138 characters, the second solution consists of 127 characters. Therefore, the verboseness (the mean length) of this query for XQuery is 132,5.
   - XSLT: the XSLT solutions have 329, 373, and 388 characters respectively. Therefore, the verboseness (mean length) of this query for XSLT is 363,3.
5. *Xpath embedding:*
   - XQuery: the first solution has 5 Xpath expressions with a total length of 72 characters:
     - //chapter
     - $t/@number > 1
     - $t/title="Getting to know SGML"
     - $t/@number
     - $t/intro

     The second solution has 3 Xpath expressions with a total length of 68 characters:
     - //chapter[title="Getting to know SGML" and @number > 1 ]
     - $t/@number
     - $t/intro

     Therefore, the XQuery solutions have a mean number of XPaths of 4 and a mean length of 70.
   - XSLT: the XSLT solutions have 4, 5, and 4 Xpath expressions respectively, with a total length of 64, 65, and 67. Therefore, the XSLT solutions have a mean number of XPaths of 4,33 and a mean length of 65,33.

## 2.1.3. Query 1d: Data XML

*List books sold by www.amazon.com and published after 1995, including their year and title*

The result of this query should be:

```
<results>
   <book year="1999">
      <title>The Economics of Technology and Content for Digital TV</title>
   </book>
   <book year="2003">
      <title>XQuery from the experts</title>
   </book>
   <book year="1999">
      <title>XSLT: programmers reference</title>
   </book>
</results>
```

### 2.1.3.1. Complexity

Complexity = 7

- Report items: 3 [book, year, title]
- Condition items: 4 [source (1) value www.amazon.com (1), year (1) value after 1995 (1)]

### 2.1.3.2. XQuery solution

```
<results>
{
for $t in //book
where $t//source="www.amazon.com" and $t/@year > 1995
return
<book year="{$t/@year}">
 {$t/title}
</book>
}
</results>
```

```
<results>
{
for $t in //book[//source="www.amazon.com" and @year > 1995]
return
<book year="{$t/@year}">
 {$t/title}
</book>
}
</results>
```

### 2.1.3.3. XSLT solution

```
<xsl:transform xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
version="2.0">
<xsl:template match="text()" />
<xsl:output method="xml" indent="yes" />

<xsl:template match="prices">
<results>
    <xsl:apply-templates />
</results>
</xsl:template>

<xsl:template match="book[@year &gt; 1995][.//source='www.amazon.com']">
  <book year="{@year}">
    <xsl:copy-of select="title" />
</book>
</xsl:template>
</xsl:transform>
```

```
<xsl:transform xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
version="2.0">
<xsl:output method="xml" indent="yes" />
<xsl:template match="prices">
<xsl:for-each select="book">
 <xsl:if test="@year &gt; 1995 and .//source='www.amazon.com'">
      <book year="{@year}">
        <xsl:copy-of select="title" />
       </book>
   </xsl:if>
</xsl:for-each>
</xsl:template>
</xsl:transform>
```

```
<xsl:transform
xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
version="2.0">
<xsl:output method="xml" indent="yes"/>
<xsl:template match="/">
<results>
<xsl:for-each select="//book[.//source='www.amazon.com'][@year > 1995]">
      <book>
          <xsl:copy-of select="@year" />
          <xsl:copy-of select="title" />
      </book>
</xsl:for-each>
</results>
</xsl:template>
</xsl:transform>
```

# 2.2. Query 2

## 2.2.1. Source

W3C Use Case "XMP": Experiences and Exemplars, Query 2 [http://www.w3.org/TR/xquery-use-cases/#xmp-queries-results-q2]:

*Create a flat list of all the title-author pairs, with each pair enclosed in a "result" element.*

Solution in XQuery:

```
<results>
  {
    for $b in doc("http://bstore1.example.com/bib.xml")/bib/book,
        $t in $b/title,
        $a in $b/author
    return
        <result>
            { $t }
            { $a }
        </result>
  }
</results>
```

## 2.2.2. Query 2t: Text XML

*Create a flat list of all the title-author pairs of the chapters, with each pair enclosed in a "result" element.*

The result of this query should be:

```
<results>
```

```
   <result>
      <title>The business challenge</title>
      <author>
         <last>Stevens</last>
         <first>W.</first>
      </author>
   </result>
   <result>
      <title>Getting to know SGML</title>
      <author>
         <last>Stevens</last>
         <first>W.</first>
      </author>
   </result>
   <result>
      <title>Resources</title>
      <author>
         <last>Abiteboul</last>
         <first>Serge</first>
      </author>
   </result>
   <result>
      <title>Resources</title>
      <author>
         <last>Buneman</last>
         <first>Peter</first>
      </author>
   </result>
   <result>
      <title>Resources</title>
      <author>
         <last>Suciu</last>
         <first>Dan</first>
      </author>
   </result>
</results>
```

### 2.2.2.1. Complexity

Complexity = 6

- Report items: 3 [result element, title, author]
- Condition items: 3 [parent node chapter (1), author element (1), title element (1).

### 2.2.2.2. XQuery solution

```
<results>
  {
    for $b in //chapter,
        $t in $b/title,
        $a in $b/author
    return
        <result>
            { $t }
            { $a }
        </result>
  }
</results>
```

```
<results>
{
for $t in //chapter/author
return
<result>
{ $t/../title, $t }
</result>
}
```

```
</results>
```

### 2.2.2.3. XSLT solution

```
<xsl:transform xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
version="2.0">
<xsl:template match="text()" />
<xsl:output method="xml" indent="yes" />

<xsl:template match="report">
<results>
    <xsl:apply-templates />
</results>
</xsl:template>

<xsl:template match="author">
    <result>
        <xsl:copy-of select="parent::chapter/title" />
 <xsl:copy-of select="." />
    </result>
</xsl:template>
</xsl:transform>
```

```
<xsl:transform xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
version="2.0">
<xsl:output method="xml" indent="yes" />
<xsl:template match="text()" />

<xsl:template match="/">
<results>
<xsl:for-each select="//chapter//author">
<result>
    <xsl:copy-of select="preceding-sibling::title" />
    <xsl:copy-of select="." />
</result>
</xsl:for-each>
</results>
</xsl:template>
</xsl:transform>
```

```
<xsl:transform xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
version="2.0">
<xsl:output method="xml" indent="yes" />
<xsl:template match="report">
<results>
  <xsl:for-each select="chapter/author">
 <result>
  <xsl:copy-of select="parent::chapter/title" />
  <xsl:copy-of select="." />
 </result>
  </xsl:for-each>
</results>
</xsl:template>
</xsl:transform>
```

```
<xsl:transform xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
version="2.0">
<xsl:output method="xml" indent="yes" />
<xsl:template match="text()" />

<xsl:template match="/">
<results>
<xsl:apply-templates />
</results>
</xsl:template>
```

```
<xsl:template match="author">
<result>
    <xsl:copy-of select="preceding-sibling::title" />
    <xsl:copy-of select="." />
</result>
</xsl:template>
</xsl:transform>
```

## 2.2.3. Query 2d: Data XML

*For each book in pricelist, list the title and sources, grouped inside a "result" element.*

The result of this query should be [note: only the first four and the last result elements are shown]:

```
<results>
 <result>
  <title>Advanced Programming in the Unix environment</title>
  <source>bstore2.example.com</source>
  <source>bstore1.example.com</source>
  <source>www.bol.com</source>
 </result>
<!-- SNIP  -->
<result>
  <title>XSLT: programmers reference</title>
  <source>bstore2.example.com</source>
  <source>www.amazon.com</source>
 </result>
 </results>
```

### 2.2.3.1. Complexity

Complexity = 6

- Report items: 3 [result element, title, author]
- Condition items: 3 [parent node book (1), price (1), title (1).

### 2.2.3.2. XQuery solution

```
<results>
  {
    for $b in //book,
        $t in $b/title,
        $a in $b//price
    return
        <result>
            { $t }
            { $a }
        </result>
  }
</results>
```

### 2.2.3.3. XSLT solution

```
<xsl:transform
xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
version="2.0">
<xsl:output method="xml" indent="yes"/>

<xsl:template match="/">
<results>
<xsl:for-each select="//book">
<result>
```

```
    <xsl:copy-of select="title"/>
    <xsl:copy-of select=".//source"/>
</result>
</xsl:for-each>
</results>
</xsl:template>
</xsl:transform>
```

# 2.3. Query 3

## 2.3.1. Source

W3C Use Case "XMP": Experiences and Exemplars, Query 3 [http://www.w3.org/TR/xquery-use-cases/#xmp-queries-results-q3]:

*For each book in the bibliography, list the title and authors, grouped inside a "result" element.*

Solution in XQuery:

```
<results>
{
    for $b in doc("http://bstore1.example.com/bib.xml")/bib/book
    return
        <result>
            { $b/title }
            { $b/author  }
        </result>
}
</results>
```

## 2.3.2. Query 3t: Text XML

*For each chapter in the document "report.xml", list the title and authors, grouped inside a "result" element*

The result of this query should be:

```
<results>
   <result>
      <title>The business challenge</title>
      <author>
         <last>Stevens</last>
         <first>W.</first>
      </author>
   </result>
   <result>
      <title>Getting to know SGML</title>
      <author>
         <last>Stevens</last>
         <first>W.</first>
      </author>
   </result>
   <result>
      <title>Resources</title>
      <author>
         <last>Abiteboul</last>
         <first>Serge</first>
      </author>
      <author>
         <last>Buneman</last>
         <first>Peter</first>
      </author>
      <author>
         <last>Suciu</last>
         <first>Dan</first>
```

```
      </author>
   </result>
</results>
```

### 2.3.2.1. Complexity

Complexity = 4

- Report items: 3 [result, title, author]
- Condition items: 1 [for each parent chapter]

### 2.3.2.2. XQuery solution

```
<results>
{
    for $b in /report/chapter
    return
        <result>
            { $b/title }
            { $b/author  }
        </result>
}
</results>
```

```
<results>
{
for $b in //chapter
return
<result>
{
$b/title, $b/author
}
</result>
}
</results>
```

### 2.3.2.3. XSLT solution

```
<xsl:transform xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
version="2.0">
<xsl:template match="text()" />
<xsl:output method="xml" indent="yes" />

<xsl:template match="report">
<results>
    <xsl:apply-templates />
</results>
</xsl:template>

<xsl:template match="chapter/title">
    <result>
        <xsl:copy-of select="." />
 <xsl:copy-of select="parent::*/author" />
    </result>
</xsl:template>
</xsl:transform>
```

```
<xsl:transform xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
version="2.0">
<xsl:output method="xml" indent="yes" />
<xsl:template match="report">
<results>
```

```
  <xsl:for-each select="chapter">
       <result>
   <xsl:copy-of select="title" />
   <xsl:copy-of select="author" />
  </result>
  </xsl:for-each>
</results>
</xsl:template>
</xsl:transform>
```

## 2.3.3. Query 3d: Data XML

*For each book in pricelist, list the title and sources, grouped inside a "result" element*

The result of this query should be:

```
<results>
   <result>
      <title>Advanced Programming in the Unix environment</title>
      <source>bstore2.example.com</source>
      <source>bstore1.example.com</source>
      <source>www.bol.com</source>
   </result>
   <result>
      <title>TCP/IP Illustrated</title>
      <source>bstore2.example.com</source>
      <source>bstore1.example.com</source>
   </result>
   <result>
      <title>Data on the Web</title>
      <source>bstore2.example.com</source>
      <source>bstore1.example.com</source>
   </result>
   <result>
      <title>The Economics of Technology and Content for Digital TV</title>
      <source>www.amazon.com</source>
      <source>www.bol.com</source>
   </result>
   <result>
      <title>Getting started with SGML</title>
      <source>www.mystore.com</source>
      <source>www.bol.com</source>
   </result>
   <result>
      <title>Drawbacks of procedural markup</title>
      <source>bstore2.example.com</source>
   </result>
   <result>
      <title>Lords of the Ring</title>
      <source>bstore1.example.com</source>
      <source>bstore2.example.com</source>
      <source>bstore3.example.com</source>
      <source>www.bol.com</source>
      <source>www.amazon.com</source>
   </result>
   <result>
      <title>XQuery from the experts</title>
      <source>bstore2.example.com</source>
      <source>www.amazon.com</source>
   </result>
   <result>
      <title>XSLT: programmers reference</title>
      <source>bstore2.example.com</source>
      <source>www.amazon.com</source>
   </result>
</results>
```

### 2.3.3.1. Complexity

Complexity = 4

- Report items: 3 [result, title, source]
- Condition items: 1 [for each parent book]

### 2.3.3.2. XQuery solution

```
<results>
{
    for $b in /prices/book
    return
        <result>
            { $b/title }
            { $b//source  }
        </result>
}
</results>
```

```
<results>
{
for $b in //book
return
<result>
{
$b/title, $b//source
}
</result>
}
</results>
```

### 2.3.3.3. XSLT solution

```
<xsl:transform xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
version="2.0">
<xsl:output method="xml" indent="yes" />
<xsl:template match="prices">
<results>
  <xsl:for-each select="book/title">
       <result>
   <xsl:copy-of select="." />
   <xsl:copy-of select="parent::*//source" />
  </result>
  </xsl:for-each>
</results>
</xsl:template>
</xsl:transform>
```

```
<xsl:transform xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
version="2.0">
<xsl:template match="text()" />
<xsl:output method="xml" indent="yes" />

<xsl:template match="prices">
<results>
    <xsl:apply-templates />
</results>
</xsl:template>

<xsl:template match="book/title">
    <result>
        <xsl:copy-of select="." />
 <xsl:copy-of select="parent::*//source" />
    </result>
</xsl:template>
</xsl:transform>
```

# 2.4. Query 4

## 2.4.1. Source

W3C Use Case "XMP": Experiences and Exemplars, Query 5 [http://www.w3.org/TR/xquery-use-cases/#xmp-queries-results-q5]:

Description: *For each book found at both bstore1.example.com and bstore2.example.com, list the title of the book and its price from each source.*

Solution in XQuery:

```
<books-with-prices>
{
for $b in doc("http://bstore1.example.com/bib.xml")//book,
    $a in doc("http://bstore2.example.com/reviews.xml")//entry
   where $b/title = $a/title
   return
  <book-with-prices>
  { $b/title }
      <price-bstore2>{ $a/price/text() }</price-bstore2>
      <price-bstore1>{ $b/price/text() }</price-bstore1>
 </book-with-prices>
  }
</books-with-prices>
```

## 2.4.2. Query 4t: Text XML

Description: *For each title found in both "report.xml" and "prices.xml", list the title and its price together in a result element*

The result of this query should be:

```
<results>
   <result>
      <title>Getting started with SGML</title>
      <price>14.95</price>
      <price>45.95</price>
   </result>
   <result>
      <title>Getting started with SGML</title>
      <price>25.95</price>
   </result>
</results>
```

### 2.4.2.1. Complexity

Query skipped

### 2.4.2.2. XQuery solution

```
<results>
{
for $b in //report,
    $a in //book
   where $b//title= $a/title
   return
  <result>
  {$b/title,
  $a//price}
  </result>
}
```

```
</results>
```

### 2.4.2.3. XSLT solution

## 2.4.3. Query 4d: Data XML

Description: *For each title found in both prices.xml and report.xml, list the title and its authors together in a result element.*

The result of this query should be:

```
<results>
   <result>
      <title>Getting started with SGML</title>
      <author>
         <last>Stevens</last>
         <first>W.</first>
      </author>
      <author>
         <last>Stevens</last>
         <first>W.</first>
      </author>
      <author>
         <last>Abiteboul</last>
         <first>Serge</first>
      </author>
      <author>
         <last>Buneman</last>
         <first>Peter</first>
      </author>
      <author>
         <last>Suciu</last>
         <first>Dan</first>
      </author>
   </result>
</results>
```

### 2.4.3.1. Complexity

This query was excluded from the experiment.

### 2.4.3.2. XQuery solution

```
<results>
{
for $b in //report,
    $a in //book
   where $b//title= $a/title
   return
  <result>
  {$b/title,
  $a//price}
  </result>
}
</results>
```

#### 2.4.3.3. XSLT solution

# 2.5. Query 5

## 2.5.1. Source

W3C Use Case "XMP": Experiences and Exemplars, Query 7 [http://www.w3.org/TR/xquery-use-cases/#xmp-queries-results-q7]:

Description: *List the titles and years of all books published by Addison-Wesley after 1991, in alphabetic order.*

Solution in XQuery:

```
<bib>
  {
    for $b in doc("http://bstore1.example.com/bib.xml")//book
    where $b/publisher = "Addison-Wesley" and $b/@year > 1991
    order by $b/title
    return
        <book>
            { $b/@year }
            { $b/title }
        </book>
  }
</bib>
```

## 2.5.2. Query 5t: Text XML

Description: *List the titles and number of the topics that have a number lower than 5 and keyword "Markup", order by title*

The result of this query should be:

```
<results>
   <topic number="3">
      <title>Drawbacks of procedural markup</title>
   </topic>
   <topic number="2">
      <title>Generic markup</title>
   </topic>
   <topic number="1">
      <title>Procedural markup</title>
   </topic>
</results>
```

#### 2.5.2.1. Complexity

Complexity = 8

- Report items: 3 [topic, number, title]
- Condition items: 5 [number attribute (1) value lower than 5 (1), keyword element (1) value "Markup (1)", sorted by title (1)]

### 2.5.2.2. XQuery solution

```
<results>
{
for $t in //topic
where $t/@number < 5 and $t/keyword="Markup"
order by $t/title
return
<topic number="{$t/@number}">
 {$t/title}
</topic>}
</results>
```

### 2.5.2.3. XSLT solution

```
<xsl:transform
xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
version="2.0">
<xsl:output method="xml" indent="yes"/>
<xsl:template match="/">
<result>
<xsl:for-each select="//topic[@number &lt; 5][keyword='Markup']">
<xsl:sort select="title" />
<topic number="{@number}">
<xsl:copy-of select="title" />
</topic>
</xsl:for-each>
</result>
</xsl:template>
</xsl:transform>
```

```
<xsl:transform
xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
version="2.0">
<xsl:output method="xml" indent="yes"/>
<xsl:template match="/">
<result>
<xsl:for-each select="//topic[@number &lt; 5][keyword='Markup']">
<xsl:sort select="title" />
<topic>
<xsl:copy-of select="@number" />
<xsl:copy-of select="title" />
</topic>
</xsl:for-each>
</result>
</xsl:template>
</xsl:transform>
```

## 2.5.3. Query 5d: Data XML

Description: *List the titles and years of all books that are published after 1991 and sold by bstore1.example.com, in alphabetic order*

The result of this query should be:

```
<result>
   <book number="1992">
      <title>Advanced Programming in the Unix environment</title>
   </book>
   <book number="2000">
      <title>Data on the Web</title>
   </book>
   <book number="1994">
```

```
      <title>TCP/IP Illustrated</title>
   </book>
</result>
```

### 2.5.3.1. Complexity

Complexity = 8

- Report items: 3 [book, year, title]
- Condition items: 5 [year attribute (1) value lower than 1991 (1), store element (1) value "bstore1.example.com" (1), sorted by title (1)]

### 2.5.3.2. XQuery solution

```
<results>
{
for $t in //book
where $t/@year > 1991 and $t//source="bstore1.example.com"
order by $t/title
return
<book year="{$t/@year}">
 {$t/title}
</book>}
</results>
```

### 2.5.3.3. XSLT solution

```
<xsl:transform
xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
version="2.0">
<xsl:output method="xml" indent="yes"/>
<xsl:template match="/">
<results>
<xsl:for-each select="//book[@year > 1991][store/source='bstore1.example.com']">
<xsl:sort select="title"/>
 <book>
          <xsl:copy-of select="@year" />
          <xsl:copy-of select="title" />
        </book>
</xsl:for-each>
</results>
</xsl:template>
</xsl:transform>
```

```
<xsl:transform
xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
version="2.0">
<xsl:output method="xml" indent="yes"/>
<xsl:template match="/">
<results>
<xsl:for-each select="//book[@year > 1991][store/source='bstore1.example.com']">
<xsl:sort select="title"/>
 <book year="{@year}">
          <xsl:copy-of select="title" />
        </book>
</xsl:for-each>
</results>
</xsl:template>
</xsl:transform>
```

```
<xsl:transform
```

```
xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
version="2.0">
<xsl:output method="xml" indent="yes"/>
<xsl:template match="/">
<results>
<xsl:for-each select="//book[@year > 1991][store/source='bstore1.example.com']">
<xsl:sort select="title"/>
 <book>
          <xsl:copy-of select="@year" />
          <xsl:copy-of select="title" />
        </book>
</xsl:for-each>
</results>
</xsl:template>
</xsl:transform>
```

# 2.6. Query 6

## 2.6.1. Source

W3C Use Case "XMP": Experiences and Exemplars, Query 9 [http://www.w3.org/TR/xquery-use-cases/#xmp-queries-results-q9]:

Description: *In the document "books.xml", find all section or chapter titles that contain the word "XML", regardless of the level of nesting.*

Solution in XQuery:

```
<results>
  {
    for $t in doc("books.xml")//(chapter | section)/title
    where contains($t/text(), "XML")
    return $t
  }
</results>
```

## 2.6.2. Query 6t: Text XML

Description: *In the document "report.xml", find all topic or chapter titles that contain the word "SGML", regardless of the level of nesting.*

The result of this query should be:

```
<results>
   <title>Getting to know SGML</title>
</results>
```

### 2.6.2.1. Complexity

Complexity = 5

- Report items: 1 [title]
- Condition items: 3 [parent is topic (1) or chapter (1), report item contains (1) the word "SGML" (1)]

### 2.6.2.2. XQuery solution

```
<results>
  {
    for $t in //(chapter | topic)/title
```

```
    where contains($t/text(), "SGML")
    return $t
  }
</results>
```

### 2.6.2.3. XSLT solution

```
<xsl:transform
xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
version="2.0">
<xsl:template match="text()" />
<xsl:template match="//chapter/title[contains(.,'SGML')] |//topic/title[contains(.,'SGML')]">
<xsl:copy-of select="."/>
</xsl:template>
</xsl:transform>
```

```
 <xsl:transform xmlns:xsl="http://www.w3.org/1999/XSL/Transform" version="2.0">
<xsl:template match="text()" />
<xsl:output method="xml" indent="yes" />
<xsl:template match="//chapter/title" >
  <xsl:if test="contains(., 'SGML')" >
    <xsl:copy-of select="." />
  </xsl:if>
</xsl:template>
<xsl:template match="//topic/title" >
  <xsl:if test="contains(., 'SGML')" >
    <xsl:copy-of select="." />
  </xsl:if>
</xsl:template>
</xsl:transform>
```

```
<xsl:transform
xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
version="2.0">
<xsl:template match="/">
<results>
<xsl:for-each select="//title[(parent::topic or parent::chapter) and contains(.,'SGML')]">
<xsl:copy-of select="."/>
</xsl:for-each>
</results>
</xsl:template>
</xsl:transform>
```

## 2.6.3. Query 6d: Data XML

Description: *In the document "prices.xml", find all book titles that contain the word "XSLT".*

The result of this query should be:

```
<results>
   <title>XSLT: programmers reference</title>
</results>
```

### 2.6.3.1. Complexity

Complexity = 4

- Report items: 1 [title]
- Condition items: 2 [parent is book (1), report item contains (1) the word "XSLT" (1)]

#### 2.6.3.2. XQuery solution

```
<results>
  {
    for $t in //book/title
    where contains($t/text(), "XSLT")
    return $t
  }
</results>
```

#### 2.6.3.3. XSLT solution

```
<xsl:transform
xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
version="2.0">
<xsl:template match="text()" />
<xsl:template match="//book/title[contains(.,'XSLT')]">
<xsl:copy-of select="."/>
</xsl:template>
</xsl:transform>
```

# 2.7. Query 7

## 2.7.1. Source

W3C Use Case "TREE": Queries that preserve hierarchy, Query 2 [http://www.w3.org/TR/xquery-use-cases/#tree-queries-results-q2]:

Description: *Prepare a (flat) figure list for Book1, listing all the figures and their titles. Preserve the original attributes of each <figure> element, if any.*

Solution in XQuery:

```
<figlist>
  {
    for $f in //figure
    return
        <figure>
            { $f/@* }
            { $f/title }
        </figure>
  }
</figlist>
```

## 2.7.2. Query 7t: Text XML

Description: *Prepare a (flat) topic list, listing all the topics and their titles. Preserve the original attributes of each <topic> element, if any.*

The result of this query should be:

```
<topiclist>
   <topic id="top1" number="1">
      <title>Procedural markup</title>
   </topic>
   <topic id="top2" number="2">
      <title>Generic markup</title>
   </topic>
```

```
   <topic id="top3" number="3">
      <title>Drawbacks of procedural markup</title>
   </topic>
   <topic id="top4" number="4">
      <title>Structure</title>
   </topic>
   <topic id="top5" number="5">
      <title>Content</title>
   </topic>
   <topic id="top6" number="6">
      <title>Style</title>
   </topic>
</topiclist>
```

### 2.7.2.1. Complexity

Complexity = 3

- Report items: 3 [topic elements,attributes, titles]
- Condition items: 0

### 2.7.2.2. XQuery solution

```
<topiclist>
  {
    for $c in //topic
    return
        <topic>
            { $c/@* }
            { $c/title }
        </topic>
  }
</topiclist>
```

### 2.7.2.3. XSLT solution

```
<xsl:transform xmlns:xsl="http://www.w3.org/1999/XSL/Transform" version="2.0">
<xsl:output method="xml" indent="yes"/>
<xsl:template match="/">
<booklist>
 <xsl:for-each select="//topic">
  <topic>
   <xsl:copy-of select="@*"/>
   <xsl:copy-of select="title"/>
  </topic>
 </xsl:for-each>
</booklist>
</xsl:template>
</xsl:transform>
```

```
<xsl:transform xmlns:xsl="http://www.w3.org/1999/XSL/Transform" version="2.0">
<xsl:output method="xml" indent="yes"/>
<xsl:template match="text()"/>

<xsl:template match="/">
<topiclist>
<xsl:apply-templates />
</topiclist>
</xsl:template>

<xsl:template match="topic">
<topic>
   <xsl:copy-of select="@*"/>
   <xsl:copy-of select="title"/>
</topic>
```

```
</xsl:template>
</xsl:transform>
```

## 2.7.3. Query 7d: Data XML

Description: *Prepare a (flat) topic list, listing all the books and their titles. Preserve the original attributes of each <book> element, if any.*

The result of this query should be:

```
<booklist>
   <book year="1992">
      <title>Advanced Programming in the Unix environment</title>
   </book>
   <book year="1994">
      <title>TCP/IP Illustrated</title>
   </book>
   <book year="2000">
      <title>Data on the Web</title>
   </book>
   <book year="1999">
      <title>The Economics of Technology and Content for Digital TV</title>
   </book>
   <book year="1982" available="no">
      <title>Getting started with SGML</title>
   </book>
   <book year="1980" available="no">
      <title>Drawbacks of procedural markup</title>
   </book>
   <book year="1950" available="no">
      <title>Lords of the Ring</title>
   </book>
   <book year="2003">
      <title>XQuery from the experts</title>
   </book>
   <book year="1999">
      <title>XSLT: programmers reference</title>
   </book>
</booklist>
```

### 2.7.3.1. Complexity

Complexity = 3

- Report items: 3 [book elements,attributes, titles]
- Condition items: 0

### 2.7.3.2. XQuery solution

```
<booklist>
  {
    for $c in //book
    return
        <book>
            { $c/@* }
            { $c/title }
        </book>
  }
</booklist>
```

### 2.7.3.3. XSLT solution

```
 <xsl:transform
xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
version="2.0">
<xsl:output method="xml" indent="yes"/>
<xsl:template match="/">
<results>
<xsl:for-each select="//book">
<book>
  <xsl:copy-of select="@*" />
  <xsl:copy-of select="title" />

</book>
</xsl:for-each>
</results>
</xsl:template>
</xsl:transform>
```

```
<xsl:transform xmlns:xsl="http://www.w3.org/1999/XSL/Transform" version="2.0">
<xsl:output method="xml" indent="yes"/>
<xsl:template match="text()"/>

<xsl:template match="/">
<booklist>
<xsl:apply-templates />
</booklist>
</xsl:template>

<xsl:template match="book">
<book>
   <xsl:copy-of select="@*"/>
   <xsl:copy-of select="title"/>
</book>
</xsl:template>
</xsl:transform>
```

# 2.8. Query 8

## 2.8.1. Source

W3C Use Case "TREE": Queries that preserve hierarchy, Query 3 [http://www.w3.org/TR/xquery-use-cases/#tree-queries-results-q3]:

Description: *How many sections are in Book1, and how many figures?*

Solution in XQuery:

```
<section_count>{ count(//section) }</section_count>,
<figure_count>{ count(//figure) }</figure_count>
```

## 2.8.2. Query 8t: Text XML

Description: *How many para elements are in the document "report.xml", and how many graphics?*

The result of this query should be:

```
<para_count>16</para_count>
<graphic_count>2</graphic_count>
```

### 2.8.2.1. Complexity

Complexity = 4

- Report items: 2 [number of elements para, number of elements graphic]
- Condition items: 2 (count (2x))

### 2.8.2.2. XQuery solution

```
<para_count>{ count(//para) }</para_count>,
<graphic_count>{ count(//graphic) }</graphic_count>
```

```
<results>
<para_count>
{
count(for $a in //para
return $a
)}
</para_count>
<graphic_count>
{
count(for $b in //graphic
return $b
)}
</graphic_count>
</results>
```

### 2.8.2.3. XSLT solution

```
<xsl:transform
xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
version="2.0">
<xsl:output method="xml" indent="yes"/>
<xsl:variable name="p" select="//para" />
<xsl:variable name="g" select="//graphic" />

<xsl:template match="/">
<para_count><xsl:value-of select="count($p)"/></para_count>
<graphic_count><xsl:value-of select="count($g)"/></graphic_count>
</xsl:template>
</xsl:transform>
```

```
<xsl:transform
xmlns:xsl="http://www.w3.org/1999/XSL/Transform" version="2.0">
<xsl:output method="xml" indent="yes"/>

<xsl:template match="/">
<para_count><xsl:value-of select="count(//para)"/></para_count>
<graphic_count><xsl:value-of select="count(//graphic)"/></graphic_count>
</xsl:template>
</xsl:transform>
```

## 2.8.3. Query 8d: Data XML

Description: *How many books are in the document prices.xml, and how many prices?*

The result of this query should be:

```
<book_count>9</book_count>
<price_count>21</price_count>
```

### 2.8.3.1. Complexity

Complexity = 4

- Report items: 2 [number of elements books, number of elements prices]
- Condition items: 2 (2x count)

### 2.8.3.2. XQuery solution

```
<book_count>
{count(//book)}
</book_count>,
<price_count>
{count(//price)}
</price_count>
```

```
<results>
<book_count>
{
count(for $a in //book
return $a
)}
</book_count>
<price_count>
{
count(for $b in //price
return $b
)}
</price_count>
</results>
```

### 2.8.3.3. XSLT solution

```
<xsl:transform
xmlns:xsl="http://www.w3.org/1999/XSL/Transform" version="2.0">
<xsl:output method="xml" indent="yes"/>
<xsl:template match="/">
<book_count><xsl:value-of select="count(//book)"/></book_count>
<price_count><xsl:value-of select="count(//price)"/></price_count>
</xsl:template>
</xsl:transform>
```

```
 <xsl:transform
xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
version="2.0">
<xsl:output method="xml" indent="yes"/>

<xsl:variable name="b" select="//book" />
<xsl:variable name="p" select="//price" />


<xsl:template match="/">
<book_count><xsl:value-of select="count($b)"/></book_count>
<price_count><xsl:value-of select="count($p)"/></price_count>
</xsl:template>
</xsl:transform>
```

# 2.9. Query 9

## 2.9.1. Source

W3C Use Case "TREE": Queries that preserve hierarchy, Query 4 [http://www.w3.org/TR/xquery-use-cases/#tree-queries-results-q4]:

Description: *How many top-level sections are in Book1?*

Solution in XQuery:

```
<top_section_count>
 {
   count(/book/section)
 }
</top_section_count>
```

## 2.9.2. Query 9t: Text XML

Description: *How many top-level sections are the document "report.xml"? With top level is meant: sections that are direct childs of a chapter element*

The result of this query should be:

```
<top_section_count>4</top_section_count>
```

### 2.9.2.1. Complexity

Complexity = 3

- Report items: 1 [number of elements sections]
- Condition items: 2 [parent is chapter (1), count (1)]

### 2.9.2.2. XQuery solution

```
<top_section_count>
 {
   count(//chapter/section)
 }
</top_section_count>
```

```
<top_section_count>
{
for $c in count(//chapter/section)
return $c
}
</top_section_count>
```

### 2.9.2.3. XSLT solution

```
<xsl:transform xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
version="2.0">
<xsl:output method="xml" indent="yes" />
<xsl:template match="text()" />
```

```
<xsl:variable name="top" select="//chapter/section" />

<xsl:template match="/">
<top_section_count>
    <xsl:copy-of select="count($top)"/>
</top_section_count>
</xsl:template>
</xsl:transform>
```

```
<xsl:transform xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
version="2.0">
<xsl:output method="xml" indent="yes" />
<xsl:template match="text()" />

<xsl:template match="/">
<top_section_count>
    <xsl:copy-of select="count(//chapter/section)"/>
</top_section_count>
</xsl:template>
</xsl:transform>
```

## 2.9.3. Query 9d: Data XML

Description: *How many top-level titles are in the pricelist? With top level is meant: titles that are direct childs of a book element*

The result of this query should be:

```
<top_section_count>9</top_section_count>
```

### 2.9.3.1. Complexity

Complexity = 3

- Report items: 1 [number of elements sections]
- Condition items: 2 [parent is chapter (1), count (1)]

### 2.9.3.2. XQuery solution

```
<top_title_count>
 {
   count(//book/title)
 }
</top_title_count>
```

```
<top_title_count>
{
for $c in count(//book/title)
return $c
}
</top_title_count>
```

### 2.9.3.3. XSLT solution

```
<xsl:transform xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
version="2.0">
```

```
<xsl:output method="xml" indent="yes" />
<xsl:template match="text()" />
<xsl:variable name="top" select="//book/title" />
<xsl:template match="/">
<top_title_count>
    <xsl:copy-of select="count($top)"/>
</top_title_count>
</xsl:template>
</xsl:transform>
```

```
<xsl:transform xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
version="2.0">
<xsl:output method="xml" indent="yes" />
<xsl:template match="text()" />

<xsl:template match="/">
<top_title_count>
    <xsl:copy-of select="count(//book/title)"/>
</top_title_count>
</xsl:template>
</xsl:transform>
```

# 2.10. Query 10

## 2.10.1. Source

W3C Use Case "TREE": Queries that preserve hierarchy, Query 5 [http://www.w3.org/TR/xquery-use-cases/#tree-queries-results-q5]:

Description: *Make a flat list of the section elements in Book1. Instead of its original attributes, each section element should have two attributes, containing the title of the section and the number of figures immediately contained in the section.*

Solution in XQuery:

```
<section_list>
  {
    for $s in //section
    let $f := $s/figure
    return
        <section title="{ $s/title/text() }" figcount="{ count($f) }"/>
  }
</section_list>
```

## 2.10.2. Query 10t: Text XML

Description: *Make a flat list of the section elements the document "report.xml". Instead of its original attributes, each section element should have two attributes, containing the title of the section and the number of figures immediately contained in the section.*

The result of this query should be:

```
<section_list>
   <section title="What is markup, or everything you always wanted to
   know about document preparation but were afraid to ask?" graphcount="0"/>
   <section title="What SGML in the grand scheme of the universe, anyway?" graphcount="0"/>
   <section title="How is SGML and would you recommend it to your grandmother?" graphcount="1"/>
   <section title="Conferences, tutorials, and training" graphcount="0"/>
</section_list>
```

### 2.10.2.1. Complexity

Complexity = 4

- Report items: 3 [section element (1), attribute title (1), attribute graphcount (1)]
- Condition items: 1 (count)

### 2.10.2.2. XQuery solution

```
<section_list>
  {
    for $s in //section
    let $f := $s//graphic
    return
        <section title="{ $s/title/text() }" graphcount="{ count($f) }"/>
  }
</section_list>
```

```
<section_list>
{
for $s in //section
return <section title="{$s/title}" graphcount="{count($s//graphic)}"/>
}
</section_list>
```

### 2.10.2.3. XSLT solution

```
<xsl:transform xmlns:xsl=
"http://www.w3.org/1999/XSL/Transform"
version="2.0">
<xsl:output method="xml" indent="yes" />
<xsl:template match="/">
<sectionlist>
 <xsl:for-each select="//section">
  <section title="{title}" numgraphs="{count(.//graphic)}" />
 </xsl:for-each>
</sectionlist>
</xsl:template>
</xsl:transform>
```

```
<xsl:transform xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
version="2.0">
<xsl:output method="xml" indent="yes" />
<xsl:template match="text()" />

<xsl:template match="/">
<section_list>
 <xsl:apply-templates />
</section_list>
</xsl:template>

<xsl:template match="section">
<section title="{title}" graphcount="{count(.//graphic)}"/>
</xsl:template>

</xsl:transform>
```

## 2.10.3. Query 10d: Data XML

Description: *Make a flat list of the book elements the document "prices.xml". Instead of its original attributes, each book element should have two attributes, containing the title of the book and the number of stores immediately contained in the book.*

The result of this query should be:

```
<book_list>
   <section storecount="3" title="Advanced Programming in the Unix environment"/>
   <section storecount="2" title="TCP/IP Illustrated"/>
   <section storecount="2" title="Data on the Web"/>
   <section storecount="2" title="The Economics of Technology and Content for Digital TV"/>
   <section storecount="2" title="Getting started with SGML"/>
   <section storecount="1" title="Drawbacks of procedural markup"/>
   <section storecount="5" title="Lords of the Ring"/>
   <section storecount="2" title="XQuery from the experts"/>
   <section storecount="2" title="XSLT: programmers reference"/>
</book_list>
```

### 2.10.3.1. Complexity

Complexity = 4

- Report items: 3 [section element (1), attribute title (1), attribute storecount (1)]
- Condition items: 1 (count)

### 2.10.3.2. XQuery solution

```
<book_list>
  {
    for $s in //book
    let $f := $s//store
    return
        <section title="{ $s/title/text() }" storecount="{ count($f) }"/>
  }
</book_list>
```

```
<book_list>
{
for $s in //book
return <book title="{$s/title}" storecount="{count($s//store)}"/>
}
</book_list>
```

### 2.10.3.3. XSLT solution

```
<xsl:transform
xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
version="2.0">
<xsl:output method="xml" indent="yes"/>
<xsl:template match="/">
<book_list>
<xsl:for-each select="//book">
<book storecount="{count(store)}" title="{title}"/>
</xsl:for-each>
</book_list>
</xsl:template>
</xsl:transform>
```

```
<xsl:transform xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
version="2.0">
<xsl:output method="xml" indent="yes" />
<xsl:template match="text()" />
<xsl:template match="/">
<book_list>
 <xsl:apply-templates />
</book_list>
</xsl:template>
<xsl:template match="book">
<book title="{title}" graphcount="{count(.//store)}"/>
</xsl:template>
</xsl:transform>
```

# 2.11. Query 11

## 2.11.1. Source

W3C Use Case "SEQ" - Queries based on Sequence: Query 1 [http://www.w3.org/TR/xquery-use-cases/#seq-queries-results-q1]:

Description: *In the Procedure section of Report1, what Instruments were used in the second Incision?*

Solution in XQuery:

```
for $s in doc("report1.xml")//section[section.title = "Procedure"]
return ($s//incision)[2]/instrument
```

## 2.11.2. Query 11t: Text XML

Description: *In the section with shorttitle "How does SGML work?", what is the title of the second topic in this section?*

The result of this query should be:

```
<title>Content</title>
```

### 2.11.2.1. Complexity

Complexity = 6

- Report items: 1 [title]
- Condition items: [section (1) with attribute shorttitle (1) with certain value (1), topic (1), second position (1)]

### 2.11.2.2. XQuery solution

```
for $s in //section[@shorttitle = "How does SGML work?"]
return $s//topic[2]/title
```

```
for $s in //section
where $s/@shorttitle= "How does SGML work?"
return $s/topic[position() = 2]/title
```

### 2.11.2.3. XSLT solution

```
<xsl:transform xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
version="2.0">
<xsl:output method="xml" indent="yes" />
<xsl:template match="text()" />
<xsl:template match="section[@shorttitle='How does SGML work?']">
   <xsl:copy-of select="topic[2]/title" />
</xsl:template>
</xsl:transform>
```

```
<xsl:transform xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
version="2.0">
<xsl:template match="text()" />
<xsl:output method="xml" indent="yes" />

<xsl:template match="//section[@shorttitle = 'How does SGML work?']/topic[2]/title">
    <xsl:copy-of select="." />
</xsl:template>
</xsl:transform>
```

## 2.11.3. Query 11d: Data XML

Description: *In the book with title "Lords of the Ring", what is the price in the second store for this book*

The result of this query should be:

```
<price>65.95</price>
```

### 2.11.3.1. Complexity

Complexity = 6

- Report items: 1 [price]
- Condition items: [book (1) with title (1) with certain value (1), store (1), second position (1)]

### 2.11.3.2. XQuery solution

```
for $s in //book[title = "Lords of the Ring"]
return $s//store[2]/price
```

```
for $s in //book
where $s/title= "Lords of the Ring"
return $s//store[position() = 2]/price
```

### 2.11.3.3. XSLT solution

```
<xsl:transform xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
version="2.0">
<xsl:output method="xml" indent="yes" />
<xsl:template match="text()" />

<xsl:template match="//book[title='Lords of the Ring']">
    <xsl:copy-of select="store[2]/price"/>
```

```
</xsl:template>
</xsl:transform>
```

```
<xsl:transform xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
version="2.0">
<xsl:template match="text()" />
<xsl:output method="xml" indent="yes" />

<xsl:template match="//book[title = 'Lords of the Ring']/store[2]/price">
    <xsl:copy-of select="." />
</xsl:template>
</xsl:transform>
```

# 2.12. Query 12

## 2.12.1. Source

W3C Use Case "SEQ" - Queries based on Sequence: Query 2 [http://www.w3.org/TR/xquery-use-cases/#seq-queries-results-q2]:

Description: *In the Procedure section of Report1, what are the first two Instruments to be used?*

Solution in XQuery:

```
for $s in doc("report1.xml")//section[section.title = "Procedure"]
return ($s//instrument)[position()<=2]
```

## 2.12.2. Query 12t: Text XML

Description: *In the section with shorttitle "How does SGML work?" of the report, what are the first two topic titles?*

The result of this query should be:

```
<title>Structure</title>
<title>Content</title>
```

### 2.12.2.1. Complexity

Complexity = 6

- Report items: 1 [title]
- Condition items: [section (1) with attribute (1) with certain value (1), topic (1), position smaller than two (1)]

### 2.12.2.2. XQuery solution

```
for $s in //section[@shorttitle = "How does SGML work?"]
return $s//topic[position()<=2]/title
```

```
for $s in //section[@shorttitle="How does SGML work?"]
return
(
for $a at $i in $s/topic/title
where $i <= 2
return $a
```

```
)
```

### 2.12.2.3. XSLT solution

```
<xsl:transform xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
version="2.0">
<xsl:template match="text()" />
<xsl:output method="xml" indent="yes" />

<xsl:template match="section[@shorttitle='How does SGML work?']">
<xsl:for-each select="topic[position() < 3]/title">
    <xsl:copy-of select="." />
</xsl:for-each>
</xsl:template>
</xsl:transform>
```

```
<xsl:transform xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
version="2.0">
<xsl:template match="text()" />
<xsl:output method="xml" indent="yes" />

<xsl:template match="section[@shorttitle='How does SGML work?']">
  <xsl:copy-of select="topic[1]/title"/>
  <xsl:copy-of select="topic[2]/title"/>
</xsl:template>
</xsl:transform>
```

## 2.12.3. Query 12d: Data XML

Description: *In the book with title "Lords of the Ring", what are the prices of the first two stores for this book*

The result of this query should be:

```
<price>55.95</price>
<price>65.95</price>
```

### 2.12.3.1. Complexity

Complexity = 6

- Report items: 1 [title]
- Condition items: [book (1) with title (1) with certain value (1), store (1), position smaller than two (1)]

### 2.12.3.2. XQuery solution

```
for $s in //book[title = "Lords of the Ring"]
return $s//store[position()<=2]/price
```

```
for $s in //book[title="Lords of the Ring"]
return
(
for $a at $i in $s/store/price
where $i <= 2
return $a
)
```

#### 2.12.3.3. XSLT solution

```
 <xsl:transform xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
version="2.0">
<xsl:output method="xml" indent="yes" />
<xsl:template match="text()" />
<xsl:template match="book[title='Lords of the Ring']">
<xsl:for-each select="./store[position() < 3]">
    <xsl:copy-of select="price" />
</xsl:for-each>
</xsl:template>
</xsl:transform>
```

```
<xsl:transform xmlns:xsl="http://www.w3.org/1999/XSL/Transform" version="2.0">
<xsl:template match="text()"/>
<xsl:output method="xml" indent="yes"/>

<xsl:template match="book[title[contains(.,'Lords of the Ring')]]">
  <xsl:copy-of select="store[1]/price"/>
  <xsl:copy-of select="store[2]/price"/>
</xsl:template>

</xsl:transform>
```

# 2.13. Query 13

## 2.13.1. Source

W3C Use Case "SGML": Standard Generalized Markup Language: Query 1 [http://www.w3.org/TR/xquery-use-cases/#sgml-queries-results-q1]:

Description: *Locate all paragraphs the document "report.xml" (all "para" elements occurring anywhere within the "report" element).*

Solution in XQuery:

```
//report//para
```

## 2.13.2. Query 13t: Text XML

Description: *Locate all paragraphs the document "report.xml" (all "para" elements occurring anywhere within the "report" element).*

The result of this query should return 16 para elements. We show only the first and the last:

```
<result>
<para>With the ever-changing and growing global market, companies and
large organizations are searching for ways to become more viable and
competitive. Downsizing and other cost-cutting measures demand more
efficient use of corporate resources. One very important resource is
an organization's information.</para>
<!-- SNIP  -->
<para security="c">Exiled members of the former Soviet Union's secret
police, the KGB, have infiltrated the upper ranks of the GCA and are
planning the Final Revolution as soon as DSSSL is completed.</para>
</result>
```

### 2.13.2.1. Complexity

Complexity = 1

- Report items: 1 [para]
- Condition items: 0

### 2.13.2.2. XQuery solution

```
report//para
```

```
<result>
{
for $p in report//para
return $p
}
</result>
```

### 2.13.2.3. XSLT solution

```
<xsl:transform
xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
version="2.0">
<xsl:output method="xml" indent="yes"/>
<xsl:template match="text()" />

<xsl:template match="report">
 <xsl:copy-of select="//para" />
</xsl:template>
</xsl:transform>
```

## 2.13.3. Query 13d: Data XML

Description: *Locate all prices in the pricelist (all "price" elements occurring anywhere within the "prices" element)*

The result of this query should be:

```
<result>
   <price>65.95</price>
   <price>65.95</price>
   <price>45.95</price>
   <price>65.95</price>
   <price>65.95</price>
   <price>34.95</price>
   <price>39.95</price>
   <price>34.95</price>
   <price>69.95</price>
   <price>14.95</price>
   <price>45.95</price>
   <price>25.95</price>
   <price>55.95</price>
   <price>65.95</price>
   <price>25.95</price>
   <price>45.95</price>
   <price>55.95</price>
   <price>55.95</price>
   <price>50.95</price>
   <price>35.95</price>
   <price>40.95</price>
</result>
```

### 2.13.3.1. Complexity

Complexity = 1

- Report items: 1 [price]
- Condition items: 0

### 2.13.3.2. XQuery solution

```
//prices//price
```

```
<result>
{
for $p in //prices//price
return $p
}
</result>
```

### 2.13.3.3. XSLT solution

```
<xsl:transform xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
version="2.0">
<xsl:template match="text()" />
<xsl:output method="xml" indent="yes" />

<xsl:template match="prices//price">
    <xsl:copy-of select="." />
</xsl:template>
</xsl:transform>
```

# 2.14. Query 14

## 2.14.1. Source

W3C Use Case "SGML": Standard Generalized Markup Language: Query 2 [http://www.w3.org/TR/xquery-use-cases/#sgml-queries-results-q2]:

Description: *Locate all paragraph elements in an introduction (all "para" elements directly contained within an "intro" element).*

Solution in XQuery:

```
<result>
  {
    //intro/para
  }
</result>
```

## 2.14.2. Query 14t: Text XML

Description: *Locate all paragraph elements in an introduction (all "para" elements directly contained within an "intro" element).*

The result of this query should contain 9 para elements, we provide the first and the last para of the result:

```
<result>
   <para>With the ever-changing and growing global market, companies and
large organizations are searching for ways to become more viable and
competitive. Downsizing and other cost-cutting measures demand more
efficient use of corporate resources. One very important resource is
an organization's information.</para>
<!-- SNIP  -->
   <para security="c">Exiled members of the former Soviet Union's secret
police, the KGB, have infiltrated the upper ranks of the GCA and are
planning the Final Revolution as soon as DSSSL is completed.</para>
</result>
```

### 2.14.2.1. Complexity

Complexity = 2

- Report items: 1 [para]
- Condition items: 1 [parent = intro]

### 2.14.2.2. XQuery solution

```
<result>
  {
    //intro/para
  }
</result>
```

```
<result>
{
for $p in //intro/para
return $p
}
</result>
```

### 2.14.2.3. XSLT solution

```
<xsl:transform xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
version="2.0">
<xsl:template match="text()" />
<xsl:output method="xml" indent="yes" />

<xsl:template match="intro/para">
    <xsl:copy-of select="." />
</xsl:template>
</xsl:transform>
```

## 2.14.3. Query 14d: Data XML

Description: *Locate all xref elements in a tip (all "xref" elements directly contained within an "tip" element).*

The result of this query should be:

```
<result>
   <xref idref="b2"/>
   <xref idref="b1"/>
   <xref idref="b5"/>
   <xref idref="b9"/>
   <xref idref="b8"/>
```

```
   <xref idref="b5"/>
</result>
```

### 2.14.3.1. Complexity

Complexity = 2

- Report items: 1 [xref]
- Condition items: 1 [parent = tip]

### 2.14.3.2. XQuery solution

```
<result>
  {
    //tip/xref
  }
</result>
```

```
<result>
{
for $p in //tip/xref
return $p
}
</result>
```

### 2.14.3.3. XSLT solution

```
<xsl:transform xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
version="2.0">
<xsl:template match="text()" />
<xsl:output method="xml" indent="yes" />

<xsl:template match="tip/xref">
    <xsl:copy-of select="." />
</xsl:template>
</xsl:transform>
```

# 2.15. Query 15

## 2.15.1. Source

W3C Use Case "SGML": Standard Generalized Markup Language: Query 4 [http://www.w3.org/TR/xquery-use-cases/#sgml-queries-results-q4]:

Description: *Locate the second paragraph in the third section in the second chapter (the second "para" element occurring in the third "section" element occurring in the second "chapter" element occurring in the "report").*

Solution in XQuery:

```
<result>
  {
    ((((//chapter)[2]//section)[3]//para)[2]
  }
</result>
```

## 2.15.2. Query 15t: Text XML

Description: *Locate the second paragraph in the third section in the second chapter (the second "para" element occurring in the third "section" element occurring in the second "chapter" element occurring in the "report").*

The result of this query should be:

```
<result>
<para>At the heart of an SGML application is a file called the DTD, or
Document Type Definition. The DTD sets up the structure of a document,
much like a database schema describes the types of information it
handles.</para>
</result>
```

### 2.15.2.1. Complexity

Complexity = 5

- Report items: 1 [para]
- Condition items: 4 [parent = section (1), position =3 (1), ancestor chapter (1), position = 2 (1)]

### 2.15.2.2. XQuery solution

```
<result>
  {
    (((//chapter)[2]//section)[3]//para)[2]
  }
</result>
```

```
<result>
{
for $p in (((//chapter[2])//section[3])//para[2])
return $p
}
</result>
```

### 2.15.2.3. XSLT solution

```
<xsl:transform xmlns:xsl=
"http://www.w3.org/1999/XSL/Transform"
version="2.0">
<xsl:output method="xml" indent="yes" />

<xsl:template match="text()"/>

<xsl:template match="//chapter[2]//section[3]//para[2]">
     <xsl:copy-of select="." />
</xsl:template>
</xsl:transform>
```

## 2.15.3. Query 15d: Data XML

Description: *Locate the price in the third store of the seventh book (the "price" element occurring in the third "store" element occurring in the seventh "book" element occurring in "prices").*

The result of this query should be:

```
<result>
   <price>25.95</price>
</result>
```

### 2.15.3.1. Complexity

Complexity = 5

- Report items: 1 [price]
- Condition items: 4 [parent = store (1), position =3 (1), ancestor book (1), position = 7 (1)]

### 2.15.3.2. XQuery solution

```
<result>
  {
    ((((//book)[7]//store)[3]//price)
  }
</result>
```

```
<result>
{
for $p in ((((//book)[7]//store)[3]//price)
return $p
}
</result>
```

### 2.15.3.3. XSLT solution

```
<xsl:transform xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
version="2.0">
<xsl:output method="xml" indent="yes" />
<xsl:template match="text()" />

<xsl:template match="book[7]/store[3]/price">
   <result><xsl:copy-of select="."/></result>
</xsl:template>
</xsl:transform>
```

# 2.16. Query 16

## 2.16.1. Source

W3C Use Case "SGML": Standard Generalized Markup Language: Query 5 [http://www.w3.org/TR/xquery-use-cases/#sgml-queries-results-q5]:

Description: *Locate all classified paragraphs (all "para" elements whose "security" attribute has the value "c").*

Solution in XQuery:

```
<result>
  {
    //para[@security = "c"]
  }
</result>
```

## 2.16.2. Query 16t: Text XML

Description: *Locate all classified paragraphs (all "para" elements whose "security" attribute has the value "c").*

The result of this query should be:

```
<result>
<para security="c">Exiled members of the former Soviet Union's secret
police, the KGB, have infiltrated the upper ranks of the GCA and are
planning the Final Revolution as soon as DSSSL is completed.</para>
</result>
```

### 2.16.2.1. Complexity

Complexity = 3

- Report items: 1 [para]
- Condition items: attribute security (1), with a certain value (1).

### 2.16.2.2. XQuery solution

```
<result>
  {
    //para[@security = "c"]
  }
</result>
```

```
for $p in //para
where $p/@security = "c"
return $p
```

### 2.16.2.3. XSLT solution

```
<xsl:transform xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
version="2.0">
<xsl:output method="xml" indent="yes" />
<xsl:template match="text()" />

<xsl:template match="/">
<result>
<xsl:for-each select="//para[@security='c']">
    <xsl:copy-of select="." />
</xsl:for-each>
</result>
</xsl:template>
</xsl:transform>
```

```
<xsl:transform xmlns:xsl="http://www.w3.org/1999/XSL/Transform" version="2.0">
<xsl:template match="text()" />
<xsl:output method="xml" indent="yes" />
<xsl:template match="//para[@security='c']">
<result>
<xsl:copy-of select="."/>
</result>
</xsl:template>
</xsl:transform>
```

## 2.16.3. Query 16d: Data XML

Description: *Locate all books from 1982 (all "book" elements whose "year" attribute has the value "1982")*

The result of this query should be:

```
<result>
   <book year="1982" available="no">
 <title>Getting started with SGML</title>
        <store>
        <source>www.mystore.com</source>
         <price>14.95</price>
        </store>
     <store>
        <source>www.bol.com</source>
        <price>45.95</price>
      </store>
    </book>
</result>
```

### 2.16.3.1. Complexity

Complexity = 3

- Report items: 1 [book]
- Condition items: attribute year (1), with a certain value (1).

### 2.16.3.2. XQuery solution

```
<result>
  {
    //book[@year = "1982"]
  }
</result>
```

```
for $b in //book
where $b/@year = "1982"
return $b
```

### 2.16.3.3. XSLT solution

```
<xsl:transform
xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
version="2.0">
<xsl:output method="xml" indent="yes"/>
<xsl:template match="/">
<xsl:for-each select="//book[@year='1982']">
 <xsl:copy-of select="." />
</xsl:for-each>
</xsl:template>
</xsl:transform>
```

```
<xsl:transform xmlns:xsl="http://www.w3.org/1999/XSL/Transform" version="2.0">
<xsl:template match="text()" />
<xsl:output method="xml" indent="yes" />
<xsl:template match="//book[@year='1982']">
<result>
<xsl:copy-of select="."/>
```

```
</result>
</xsl:template>
</xsl:transform>
```

# 2.17. Query 17

## 2.17.1. Source

W3C Use Case "SGML": Standard Generalized Markup Language: Query 6 [http://www.w3.org/TR/xquery-use-cases/#sgml-queries-results-q6]:

Description: *List the short titles of all sections (the values of the "shorttitle" attributes of all "section" elements, expressing each short title as the value of a new element.)*

Solution in XQuery:

```
<result>
  {
    for $s in //section/@shorttitle
    return <stitle>{ $s }</stitle>
  }
</result>
```

## 2.17.2. Query 17t: Text XML

Description: *List the short titles of all sections (the values of the "shorttitle" attributes of all "section" elements, expressing each short title as the value of a new element.)*

The result of this query should be:

```
<result>
   <stitle shorttitle="What is markup?"/>
   <stitle shorttitle="What is SGML?"/>
   <stitle shorttitle="How does SGML work?"/>
</result>
```

### 2.17.2.1. Complexity

Complexity = 3

- Report items: 2 [stitle element, with an attribute shorttitle]
- Condition items: 1 parent is a section (1).

### 2.17.2.2. XQuery solution

```
<result>
  {
    for $s in //section/@shorttitle
    return <stitle>{ $s }</stitle>
  }
</result>
```

### 2.17.2.3. XSLT solution

```
<xsl:transform xmlns:xsl=
```

```
"http://www.w3.org/1999/XSL/Transform"
version="2.0">
<xsl:output method="xml" indent="yes" />
<xsl:template match="text()" />

<xsl:template match="section[@shorttitle]">
<stitle shorttitle="{@shorttitle}"/>
</xsl:template>
</xsl:transform>
```

```
<xsl:transform xmlns:xsl=
"http://www.w3.org/1999/XSL/Transform"
version="2.0">
<xsl:output method="xml" indent="yes" />
<xsl:template match="/">
<result>
    <xsl:for-each select="//section[@shorttitle]">
<stitle shorttitle="{@shorttitle}"/>
</xsl:for-each>
</result>
</xsl:template>
</xsl:transform>
```

## 2.17.3. Query 17d: Data XML

Description: *List the values of the available attributes of all book*

The result of this query should be:

```
<result>
   <available available="no"/>
   <available available="no"/>
   <available available="no"/>
</result>
```

### 2.17.3.1. Complexity

Complexity = 3

- Report items: 2 [available element, with an attribute available]
- Condition items: 1 parent is a book (1).

### 2.17.3.2. XQuery solution

```
<result>
  {
    for $a in //book/@available
    return <available>{ $a }</available>
  }
</result>
```

### 2.17.3.3. XSLT solution

```
<xsl:transform xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
version="2.0">
<xsl:output method="xml" indent="yes" />
<xsl:template match="text()" />

<xsl:template match="//book[@available]">
```

```
   <available><xsl:copy-of select="@available"/></available>
</xsl:template>
</xsl:transform>
```

```
<xsl:transform xmlns:xsl=
"http://www.w3.org/1999/XSL/Transform"
version="2.0">
<xsl:output method="xml" indent="yes" />
<xsl:template match="text()" />

<xsl:template match="/">
<result>
<xsl:for-each select="//book[@available]">
<available available="{@available}"/>
</xsl:for-each>
</result>
</xsl:template>
</xsl:transform>
```

# 2.18. Query 18

## 2.18.1. Source

W3C Use Case "SGML": Standard Generalized Markup Language: Query 9 [http://www.w3.org/TR/xquery-use-cases/#sgml-queries-results-q9]:

Description: *Locate all the topics referenced by a cross-reference anywhere the document "report.xml" (all the "topic" elements whose "id" attribute value is the same as an "idref" attribute value of any "xref" element).*

Solution in XQuery:

```
<result>
  {
    for $id in //xref/@idref
    return //topic[@id = $id]
  }
</result>
```

```
for $t in //topic
for $x in //xref
where $t/@id = $x/@idref
return $t
```

## 2.18.2. Query 18t: Text XML

Description: *Locate all the topics referenced by a cross-reference anywhere in the document "report.xml" (all the "topic" elements whose "id" attribute value is the same as an "idref" attribute value of any "xref" element).*

The result of this query should be:

```
<result>
<topic id="top4" number="4">
 <title>Structure</title>
    <keyword>DTD</keyword>
 <para>At the heart of an SGML application is a file called the DTD, or
 Document Type Definition. The DTD sets up the structure of a document,
 much like a database schema describes the types of information it
 handles.</para>
```

```
 <para security="u">A database schema also defines the relationships between the
 various types of data. Similarly, a DTD specifies <emph>rules</emph>
 to help ensure documents have a consistent, logical structure.</para>
</topic>
</result>
```

### 2.18.2.1. Complexity

Complexity = 5

- Subquery 1 (Get topics which id is equal to one of the values of subquery 2):
  - Report items: [1], topic element.
  - Condition items: [3], id attribute (1) and a reference value equal to [report subquery 2] (2).

  Complexity Subquery 1 = 4
- Subquery 2 (get the values of all xref elements):
  - Report items: 1 [values of xref elements]
  - Condition items: [0]

  Complexity subquery 2 = 1

### 2.18.2.2. XQuery solution

```
<result>
  {
    for $id in //xref/@xref
    return //topic[@id = $id]
  }
</result>
```

```
for $t in //topic
for $x in //xref
where $t/@id = $x/@idref
return $t
```

### 2.18.2.3. XSLT solution

```
<xsl:transform xmlns:xsl=
"http://www.w3.org/1999/XSL/Transform"
version="2.0">
<xsl:output method="xml" indent="yes" />
<xsl:variable select="//xref/@idref" name="refs" />

<xsl:template match="/">
<result>
<xsl:for-each select="//topic[@id=$refs]">
     <xsl:copy-of select="." />
</xsl:for-each>
</result>
</xsl:template>
</xsl:transform>
```

## 2.18.3. Query 18d: Data XML

Description: *Locate all the books referenced by a cross-reference anywhere in the pricelist (all the "book" elements whose "id" attribute value is the same as an "idref" attribute value of any "xref" element).*

The result of this query should be:

```
<result>
```

```
<book year="1994" id="b2">
  <title>TCP/IP Illustrated</title>
  <store>
    <source>bstore2.example.com</source>
    <price>65.95</price>
  </store>
  <store>
    <source>bstore1.example.com</source>
    <price>65.95</price>
  </store>
  <tip>
    <xref idref="b1" />
  </tip>
</book>
<book year="1992" id="b1">
  <title>Advanced Programming in the Unix environment</title>
  <store>
    <source>bstore2.example.com</source>
    <price>65.95</price>
  </store>
  <store>
    <source>bstore1.example.com</source>
    <price>65.95</price>
  </store>
  <store>
    <source>www.bol.com</source>
    <price>45.95</price>
  </store>
  <tip>
    <xref idref="b2" />
  </tip>
</book>
<book year="1982" available="no" id="b5">
  <title>Getting started with SGML</title>
  <store>
    <source>www.mystore.com</source>
    <price>14.95</price>
  </store>
  <store>
    <source>www.bol.com</source>
    <price>45.95</price>
  </store>
</book>
<book year="1999" id="b9">
  <title>XSLT: programmers reference</title>
  <store>
    <source>bstore2.example.com</source>
    <price>35.95</price>
  </store>
  <store>
    <source>www.amazon.com</source>
    <price>40.95</price>
  </store>
  <tip>
    <xref idref="b8" />
    <xref idref="b5" />
  </tip>
</book>
<book year="2003" id="b8">
  <title>XQuery from the experts</title>
  <store>
    <source>bstore2.example.com</source>
    <price>55.95</price>
  </store>
  <store>
    <source>www.amazon.com</source>
    <price>50.95</price>
  </store>
  <tip>
    <xref idref="b9" />
  </tip>
</book>
<book year="1982" available="no" id="b5">
  <title>Getting started with SGML</title>
  <store>
    <source>www.mystore.com</source>
    <price>14.95</price>
```

```
    </store>
    <store>
      <source>www.bol.com</source>
      <price>45.95</price>
    </store>
  </book>
</result>
```

### 2.18.3.1. Complexity

Complexity = 5

- Subquery 1 (Get books which id is equal to one of the values of subquery 2):
  - Report items: [1], book element.
  - Condition items: [3], id attribute (1) and a reference value equal to [report subquery 2] (2).

  Complexity Subquery 1 = 4
- Subquery 2 (get the values of all xref elements):
  - Report items: 1 [values of xref elements]
  - Condition items: [0]

  Complexity subquery 2 = 1

### 2.18.3.2. XQuery solution

```
<result>
  {
    for $id in //xref/@idref
    return //book[@id = $id]
  }
</result>
```

```
for $b in //book
for $x in //xref
where $b/@id = $x/@idref
return $b
```

### 2.18.3.3. XSLT solution

```
<xsl:transform xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
version="2.0">
<xsl:template match="text()" />
<xsl:output method="xml" indent="yes" />

<xsl:variable name="xrefs" select="//xref/@idref"/>

<xsl:template match="book[@id=$xrefs]">
    <xsl:copy-of select="." />
</xsl:template>
</xsl:transform>
```

# 2.19. Query 19

## 2.19.1. Source

W3C Use Case "STRING": String SearchQ1 [http://www.w3.org/TR/xquery-use-cases/#text-queries-results-q1]

Description: *Find the titles of all news items where the string "Foobar Corporation" appears in the title.*

Solution in XQuery:

```
doc("string.xml")//news_item/title[contains(., "Foobar Corporation")]
```

## 2.19.2. Query 19t: Text XML

Description: *Find the titles of all sections where the string "grand scheme" appears in the title.*

The result of this query should be:

```
<title>What <emph>is</emph> SGML in the grand scheme of the universe, anyway?</title>
```

### 2.19.2.1. Complexity

Complexity = 5

- Report items: 1 [title elements]
- Condition items: 3 [parent is section (1), report item contains (1) the string "grand scheme" (1)]

### 2.19.2.2. XQuery solution

```
//section/title[contains(., "grand scheme")]
```

```
for $t in //section/title
where contains($t, "grand scheme")
return $t
```

### 2.19.2.3. XSLT solution

```
<xsl:transform
xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
version="2.0">
<xsl:output method="xml" indent="yes"/>
<xsl:template match="/">
<results>
<xsl:for-each select="//section/title[contains(.,'grand scheme')]">
<xsl:copy-of select="." />
</xsl:for-each>
</results>
</xsl:template>
</xsl:transform>
```

```
<xsl:transform
xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
version="2.0">
<xsl:output method="xml" indent="yes"/>

<xsl:template match="/">
<xsl:copy-of select="//section/title[contains(., 'grand scheme')]"/>
</xsl:template>
</xsl:transform>
```

## 2.19.3. Query 19d: Data XML

Description: *Find the titles of all books where the string "Advanced Programming" appears in the title.*

The result of this query should be:

```
<title>Advanced Programming in the Unix environment</title>
```

### 2.19.3.1. Complexity

Complexity = 5

- Report items: 1 [title elements]
- Condition items: 3 [parent is book (1), report item contains (1) the string "Advance Programming" (1)]

### 2.19.3.2. XQuery solution

```
//book/title[contains(., "Advanced Programming")]
```

```
for $t in //book/title
where contains($t, "Advanced Programming")
return $t
```

### 2.19.3.3. XSLT solution

```
<xsl:transform
xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
version="2.0">
<xsl:output method="xml" indent="yes"/>
<xsl:template match="/">
<results>
<xsl:for-each select="//book[contains(., 'Advanced Programming')]">
<xsl:copy-of select="." />
</xsl:for-each>
</results>
</xsl:template>
</xsl:transform>
```

```
<xsl:transform xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
version="2.0">
<xsl:output method="xml" indent="yes" />
<xsl:template match="text()" />

<xsl:template match="book[contains(., 'Advanced Programming')]">
   <xsl:copy-of select="title"/>
</xsl:template>
</xsl:transform>
```

# 2.20. Query 20

## 2.20.1. Source

W3C Use Case "STRING": String Search Query 5 [http://www.w3.org/TR/xquery-use-cases/#text-queries-results-q5].

Description: *For each news item that is relevant to the Gorilla Corporation, create an "item summary" element. The content of the item summary is the content of the title, date, and first paragraph of the news item, separated*

*by periods. A news item is relevant if the name of the company is mentioned anywhere within the content of the news item.*

Solution in XQuery:

```
for $item in doc("string.xml")//news_item
where contains(string($item/content), "Gorilla Corporation")
return
    <item_summary>
        { $item/title/text() }
        { $item/date/text() }
        { string(($item//par)[1]) }
    </item_summary>
```

## 2.20.2. Query q20t: Text XML

Description: *For each topic that is about an SGML Application, create an "topic summary" element. The content of the topic summary is the content of the title, keyword, and first para of the topic. A topic is about an SGML Application when the string "SGML application" occurs anywhere within a para of the topic.*

The result of this query should be (with whitespace reformatted for readability):

```
<topic_summary>Structure DTD At the heart of an SGML application is a file called the DTD, or
Document Type Definition. The DTD sets up the structure of a document,
much like a database schema describes the types of information it
handles.</topic_summary>
```

### 2.20.2.1. Complexity

Complexity = 9

- Report items: 4 [topic_summary element (1), values of title (1), keyword (1) and para (1)]
- Condition items: 5 [para child (1), that contains (1) a string "SGML application" (1), report para element has position 1 (1), values of report elements (1).]

### 2.20.2.2. XQuery solution

```
for $topic in //topic
where contains(string($topic/para), "SGML application")
return
<topic_summary>
   { $topic/title/text() }
   { $topic/keyword/text() }
   { string(($topic/para)[1])}
</topic_summary>
```

```
for $t in //topic
where contains(string($t/para), "SGML application")
return
<topic_summary>
 {string($t/title)}{string($t/keyword)}{string($t/para[1])}
</topic_summary>
```

### 2.20.2.3. XSLT solution

```
<xsl:transform xmlns:xsl=
```

```
"http://www.w3.org/1999/XSL/Transform"
version="2.0">
<xsl:output method="xml" indent="yes" />
<xsl:template match="/">
<summary>
    <xsl:for-each select="//topic[para[contains(., 'SGML application')]]">
     <xsl:value-of select="title" />
     <xsl:value-of select="keyword" />
     <xsl:value-of select=".//para[1]" />
    </xsl:for-each>
</summary>
</xsl:template>
</xsl:transform>
```

```
<xsl:transform xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
version="2.0">
<xsl:output method="xml" indent="yes" />
<xsl:template match="text()" />
<xsl:template match="//topic[para[contains(text(), 'SGML application')]]">
<topic_summary>
<xsl:value-of select="title" />
<xsl:value-of select="keyword" />
<xsl:value-of select="para[1]" />
</topic_summary>
</xsl:template>
</xsl:transform>
```

## 2.20.3. Query 20d: Data XML

Description: *For each book that is about Digital TV, create an "book summary" element. The content of the book summary is the content of the title, year, and first store where the book is sold. A book is about an Digital TV when the string "for Digital" occurs anywhere within the title of the book.*

The result of this query should be (with whitespace reformatted for readability):

```
<book_summary>
The Economics of Technology and Content for Digital TV 1999
www.amazon.com
34.95
</book_summary>
```

### 2.20.3.1. Complexity

Complexity = 9

- Report items: 4 [book_summary element (1), values of title (1), year (1) and store (1)]
- Condition items: 5 [title child (1), that contains (1) a string "Digital TV" (1), report store element has position 1 (1), values of report elements (1).]

### 2.20.3.2. XQuery solution

```
for $book in //book
where contains(string($book/title), "Digital TV")
return
<book_summary>
   { $book/title/text() }
   { string($book/@year) }
   { string($book/store[1])}
</book_summary>
```

```
for $t in //book
where contains($t/title,"for Digital")
return
      <book_summary>
      {string($t/title)}{string($t/@year)}{string($t/store[1])}
      </book_summary>
```

#### 2.20.3.3. XSLT solution

```
<xsl:transform xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
version="2.0">
<xsl:template match="text()" />
<xsl:output method="xml" indent="yes" />
<xsl:template match="book[title[contains(.,'for Digital')]]">
<book_summary>
    <xsl:value-of select="title"/>
    <xsl:value-of select="@year"/>
    <xsl:value-of select="store[1]/source" />
</book_summary>
</xsl:template>
</xsl:transform>
```

```
<xsl:transform xmlns:xsl=
"http://www.w3.org/1999/XSL/Transform"
version="2.0">
<xsl:output method="xml" indent="yes" />
<xsl:template match="/">
<book_summary>
    <xsl:for-each select="//book[title[contains(.,'for Digital')]]">
      <xsl:value-of select="title"/>
      <xsl:value-of select="@year"/>
      <xsl:value-of select="store[1]/source" />
    </xsl:for-each>
</book_summary>
</xsl:template>
</xsl:transform>
```

# 2.21. Query 21

## 2.21.1. Source

W3Schools XQuery tutorial [http://www.w3schools.com/xquery/xquery_example.asp].

Description: *Select all titles of books.*

Solution in XQuery:

```
doc("books.xml")/bib/book/title
```

## 2.21.2. Query q21t: Text XML

Description: *Select all titles of chapters*

The result of this query should be:

```
<title>The business challenge</title>
<title>Getting to know SGML</title>
<title>Resources</title>
```

### 2.21.2.1. Complexity

Complexity = 2

- Report items: 1 [title elements]
- Condition items: 1 [parent = chapter]

### 2.21.2.2. XQuery solution

```
/report/chapter/title
```

```
for $t in //chapter/title
return $t
```

### 2.21.2.3. XSLT solution

```
<xsl:transform
xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
version="2.0">
<xsl:output method="xml" indent="yes"/>
<xsl:template match="/">
<xsl:for-each select="//chapter/title">
 <xsl:copy-of select="." />
</xsl:for-each>
</xsl:template>
</xsl:transform>
```

```
<xsl:transform xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
version="2.0">
<xsl:output method="xml" indent="yes" />
<xsl:template match="text()" />
<xsl:template match="//chapter/title">
    <xsl:copy-of select="." />
</xsl:template>
</xsl:transform>
```

## 2.21.3. Query 21d: Data XML

Description: *Select all titles of books*

The result of this query should be:

```
<title>Advanced Programming in the Unix environment</title>
<title>TCP/IP Illustrated</title>
<title>Data on the Web</title>
<title>The Economics of Technology and Content for Digital TV</title>
<title>Getting started with SGML</title>
<title>Drawbacks of procedural markup</title>
<title>Lords of the Ring</title>
<title>XQuery from the experts</title>
<title>XSLT: programmers reference</title>
```

### 2.21.3.1. Complexity

Complexity = 2

- Report items: 1 [title elements]
- Condition items: 1 [parent = book]

### 2.21.3.2. XQuery solution

```
//book/title
```

```
for $t in //book/title
return $t
```

### 2.21.3.3. XSLT solution

```
<xsl:transform xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
version="2.0">
<xsl:template match="text()" />
<xsl:output method="xml" indent="yes" />

<xsl:template match="book">
    <xsl:copy-of select="title" />
</xsl:template>
</xsl:transform>
```

```
<xsl:transform xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
version="2.0">
<xsl:output method="xml" indent="yes" />
<xsl:template match="/">
<xsl:for-each select="//book/title">
    <xsl:copy-of select="." />
</xsl:for-each>
</xsl:template>
</xsl:transform>
```

# 2.22. Query 22

## 2.22.1. Source

W3Schools XQuery tutorial [http://www.w3schools.com/xquery/xquery_example.asp].

Description: *Select all titles*

Solution in XQuery:

```
doc("books.xml")//title
```

## 2.22.2. Query 22t: Text XML

Description: *Select all titles the document "report.xml"*

The result of this query should be:

```
<result>
   <title>Getting started with SGML</title>
   <title>The business challenge</title>
```

```
   <title>Getting to know SGML</title>
   <title>What is markup, or everything you always wanted to know about document preparation but were afraid to a
   <title>Procedural markup</title>
   <title>Generic markup</title>
   <title>Drawbacks of procedural markup</title>
   <title>What <emph>is</emph> SGML in the grand scheme of the universe, anyway?</title>
   <title>How is SGML and would you recommend it to your grandmother?</title>
   <title>Structure</title>
   <title>Content</title>
   <title>Style</title>
   <title>Resources</title>
   <title>Conferences, tutorials, and training</title>
</result>
```

### 2.22.2.1. Complexity

Complexity = 1

- Report items: 1 [title elements]
- Condition items: 0

### 2.22.2.2. XQuery solution

```
//title
```

```
<result>
{
for $t in //title
return $t
}
</result>
```

### 2.22.2.3. XSLT solution

```
<xsl:transform xmlns:xsl=
"http://www.w3.org/1999/XSL/Transform"
version="2.0">
<xsl:output method="xml" indent="yes" />
<xsl:template match="/">
<result>
    <xsl:for-each select="//title">
     <xsl:copy-of select="." />
</xsl:for-each>
</result>
</xsl:template>
</xsl:transform>
```

```
<xsl:transform xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
version="2.0">
<xsl:template match="text()" />
<xsl:output method="xml" indent="yes" />

<xsl:template match="title">
    <xsl:copy-of select="." />
</xsl:template>
</xsl:transform>
```

## 2.22.3. Query 22d: Data XML

Description: *Select all titles in prices.xml*

The result of this query should be:

```
<result>
   <title>Advanced Programming in the Unix environment</title>
   <title>TCP/IP Illustrated</title>
   <title>TCP/IP Illustrated</title>
   <title>Advanced Programming in the Unix environment</title>
   <title>Data on the Web</title>
   <title>The Economics of Technology and Content for Digital TV</title>
   <title>Getting started with SGML</title>
   <title>Drawbacks of procedural markup</title>
   <title>Getting started with SGML</title>
   <title>Lords of the Ring</title>
   <title>XQuery from the experts</title>
   <title>XSLT: programmers reference</title>
   <title>XSLT: programmers reference</title>
   <title>XQuery from the experts</title>
</result>
```

### 2.22.3.1. Complexity

Complexity = 1

- Report items: 1 [title elements]
- Condition items: 0

### 2.22.3.2. XQuery solution

```
//title
```

```
<result>
{
for $t in //title
return $t
}
</result>
```

### 2.22.3.3. XSLT solution

```
<xsl:transform xmlns:xsl=
"http://www.w3.org/1999/XSL/Transform"
version="2.0">
<xsl:output method="xml" indent="yes" />
<xsl:template match="/">
<result>
    <xsl:for-each select="//title">
     <xsl:copy-of select="." />
</xsl:for-each>
</result>
</xsl:template>
</xsl:transform>
```

```
<xsl:transform xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
version="2.0">
```

```
<xsl:template match="text()" />
<xsl:output method="xml" indent="yes" />

<xsl:template match="title">
    <xsl:copy-of select="." />
</xsl:template>
</xsl:transform>
```

# 2.23. Query 23

## 2.23.1. Source

W3Schools XQuery tutorial [http://www.w3schools.com/xquery/xquery_example.asp].

Description: *Extracting Nodes With an Expression*

Solution in XQuery:

```
doc("books.xml")/bib/book[price<50]
```

## 2.23.2. Query 23t: Text XML

Description: *Select topics with a number lower than 3*

The result of this query should be:

```
<result>
  <topic id="top1" number="1">
    <title>Procedural markup</title>
    <keyword>Markup</keyword>
    <para security="u">Most electronic publishing systems today use
    some form of procedural markup. Procedural markup codes are
    good for one presentation of the information.</para>
  </topic>
  <topic id="top2" number="2">
    <title>Generic markup</title>
    <keyword>Markup</keyword>
    <para>Generic markup (also known as descriptive markup)
    describes the
    <emph>purpose</emph>of the text in a document. A basic concept
    of generic markup is that the content of a document must be
    separate from the style. Generic markup allows for multiple
    presentations of the information.</para>
  </topic>
</result>
```

### 2.23.2.1. Complexity

Complexity = 3

- Report items: 1 [topics]
- Condition items: [attribute number (1), with a particular value (1)]

### 2.23.2.2. XQuery solution

```
<result>
{
//topic[@number<3]
}
```

```
</result>
```

```
<result>
{
for $t in //topic
where $t/@number<3
return $t
}
</result>
```

#### 2.23.2.3. XSLT solution

```
 <xsl:transform xmlns:xsl="http://www.w3.org/1999/XSL/Transform" version="2.0">
<xsl:output method="xml" indent="yes"/>
<xsl:template match="/">
<result>
<xsl:copy-of select="//topic[@number &lt; 3]"/>
</result>
</xsl:template>
</xsl:transform>
```

```
<xsl:transform xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
version="2.0">
<xsl:template match="text()" />
<xsl:output method="xml" indent="yes" />

<xsl:template match="//topic[@number &lt; 3]">
    <xsl:copy-of select="." />
</xsl:template>
</xsl:transform>
```

### 2.23.3. Query 23d: Data XML

Description: *Select books that have a year attribute smaller than 1994.*

The result of this query should be:

```
<result>
  <book year="1992">
    <title>Advanced Programming in the Unix environment</title>
    <store>
      <source>bstore2.example.com</source>
      <price>65.95</price>
    </store>
    <store>
      <source>bstore1.example.com</source>
      <price>65.95</price>
    </store>
    <store>
      <source>www.bol.com</source>
      <price>45.95</price>
    </store>
    <tip>
      <title>TCP/IP Illustrated</title>
    </tip>
  </book>
  <book year="1982" available="no">
    <title>Getting started with SGML</title>
    <store>
      <source>www.mystore.com</source>
      <price>14.95</price>
    </store>
    <store>
```

```
      <source>www.bol.com</source>
      <price>45.95</price>
    </store>
  </book>
  <book year="1980" available="no">
    <title>Drawbacks of procedural markup</title>
    <store>
      <source>bstore2.example.com</source>
      <price>25.95</price>
    </store>
    <tip>
      <title>Getting started with SGML</title>
    </tip>
  </book>
  <book year="1950" available="no">
    <title>Lords of the Ring</title>
    <store>
      <source>bstore1.example.com</source>
      <price>55.95</price>
    </store>
    <store>
      <source>bstore2.example.com</source>
      <price>65.95</price>
    </store>
    <store>
      <source>bstore3.example.com</source>
      <price>25.95</price>
    </store>
    <store>
      <source>www.bol.com</source>
      <price>45.95</price>
    </store>
    <store>
      <source>www.amazon.com</source>
      <price>55.95</price>
    </store>
  </book>
</result>
```

### 2.23.3.1. Complexity

Complexity = 3

- Report items: 1 [book elements]
- Condition items: [attribute year (1), with a particular value (1)]

### 2.23.3.2. XQuery solution

```
//book[@year<1994]
```

```
<result>
{
for $b in //book
where $b/@year<1994
return $b
}
</result>
```

### 2.23.3.3. XSLT solution

```
<xsl:transform xmlns:xsl="http://www.w3.org/1999/XSL/Transform" version="2.0">
<xsl:output method="xml" indent="yes"/>
<xsl:template match="/">
<result>
<xsl:copy-of select="//book[@year &lt; 1994]"/>
```

```
</result>
</xsl:template>
</xsl:transform>
```

```
<xsl:transform xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
version="2.0">
<xsl:template match="text()" />
<xsl:output method="xml" indent="yes" />

<xsl:template match="//book[@year &lt; 1994]">
    <xsl:copy-of select="." />
</xsl:template>
</xsl:transform>
```

# 2.24. Query 24

## 2.24.1. Source

W3Schools XQuery tutorial [http://www.w3schools.com/xquery/xquery_flwor.asp].

Description: *Extracting Nodes With FLWOR*

Solution in XQuery:

```
for $x in doc("books.xml")/bib/book
where $x/price>50
order by $x/title
return $x/title
```

## 2.24.2. Query 24t: Text XML

Description: *Select the titles of topics with a number higher than 4, ordered by title*

The result of this query should be:

```
<result>
<title>Content</title>
<title>Style</title>
</result>
```

### 2.24.2.1. Complexity

Complexity = 5

- Report items: 1 [title elements]
- Condition items: 4 [parent = topic (1), attribute number (1), with a particular value (1), sort on report item (1) ]

### 2.24.2.2. XQuery solution

```
<result>
{
for $t in //topic
where $t/@number>4
order by $t/title
return $t/title
}
```

```
</result>
```

```
<result>
{
for $t in //topic[@number>4]
order by $t/title
return $t/title
}
</result>
```

### 2.24.2.3. XSLT solution

```
<xsl:transform xmlns:xsl=
"http://www.w3.org/1999/XSL/Transform"
version="2.0">
<xsl:output method="xml" indent="yes" />
<xsl:template match="/">
<result>
    <xsl:for-each select="//topic[@number > 4]">
    <xsl:sort select="title" />
     <xsl:copy-of select="title" />
</xsl:for-each>
</result>
</xsl:template>
</xsl:transform>
```

## 2.24.3. Query 24d: Data XML

Description: *Select the titles of books with a year higher than 1999, ordered by title*

The result of this query should be:

```
<result>
   <title>Data on the Web</title>
   <title>XQuery from the experts</title>
</result>
```

### 2.24.3.1. Complexity

Complexity = 5

- Report items: 1 [title elements]
- Condition items: 4 [parent = book (1), attribute year (1), with a particular value (1), sort on report item (1) ]

### 2.24.3.2. XQuery solution

```
<result>
{
for $b in //book
where $b/@year>1999
order by $b/title
return $b/title
}
</result>
```

```
<result>
{
```

```
for $b in //book[@year>1999]
order by $b/title
return $b/title
}
</result>
```

#### 2.24.3.3. XSLT solution

```
<xsl:transform xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
version="2.0">
<xsl:output method="xml" indent="yes" />
<xsl:template match="text()" />

<xsl:template match="/">
<result>
  <xsl:for-each select="//book[@year > 1999]">
     <xsl:sort select="title" />
     <xsl:copy-of select="title" />
   </xsl:for-each>
</result>
</xsl:template>

</xsl:transform>
```

# 2.25. Query 25

## 2.25.1. Source

Xmark benchmark [http://monetdb.cwi.nl/xml/] Query 1

Description: *Q1.Return the name of the person with ID `person0' registered in North America.*

Note that the "registered in North America" is not reflected in the query and therefore, we leave such a restriction aside too.

Solution in XQuery:

```
FOR $b IN doc("auction.xml")/site/people/person[@id="person0"]
RETURN $b/name/text()
```

## 2.25.2. Query 25t: Text XML

Description: *Return the text of the title of the topic with ID 'top3'*

The result of this query should be:

```
Drawbacks of procedural markup
```

#### 2.25.2.1. Complexity

Complexity = 5

- Report items: 1 [text values]
- Condition items: 4 [parent = title (1), ancestor = topic (1), attribute id (1), with a certain value (1)]

### 2.25.2.2. XQuery solution

```
for $b in //topic[@id="top3"]
return $b/title/text()
```

```
//topic[@id="top3"]/title/text()
```

### 2.25.2.3. XSLT solution

```
<xsl:transform xmlns:xsl=
"http://www.w3.org/1999/XSL/Transform"
version="2.0">
<xsl:template match="/">
    <xsl:for-each select="//topic[@id='top3']">
     <xsl:value-of select="title" />
</xsl:for-each>
</xsl:template>
</xsl:transform>
```

```
<xsl:transform xmlns:xsl=
"http://www.w3.org/1999/XSL/Transform"
version="2.0">
<xsl:template match="text()"/>
<xsl:output method="xml" indent="yes"/>

<xsl:template match="//topic[@id='top3']">
<xsl:value-of select="title"/>
</xsl:template>
</xsl:transform>
```

## 2.25.3. Query 25d: Data XML

Description: *Return the text of the title of the book with year '1999'*

The result of this query should be:

```
The Economics of Technology and Content for Digital TV
XSLT: programmers reference
```

### 2.25.3.1. Complexity

Complexity = 5

- Report items: 1 [text values]
- Condition items: 4 [parent = title (1), ancestor = book (1), attribute year (1), with a certain value (1)]

### 2.25.3.2. XQuery solution

```
for $b in //book[@year="1999"]
return $b/title/text()
```

```
//book[@year="1999"]/title/text()
```

#### 2.25.3.3. XSLT solution

```
<xsl:transform xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
version="2.0">
<xsl:template match="/">
<xsl:for-each select="//book[@year='1999']">
    <xsl:value-of select="title" />
</xsl:for-each>
</xsl:template>
</xsl:transform>
```

```
<xsl:transform xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
version="2.0">
<xsl:template match="text()"/>
<xsl:output method="xml" indent="yes"/>

<xsl:template match="//book[@year='1999']">
    <xsl:value-of select="title" />
</xsl:template>
</xsl:transform>
```

# 2.26. Query 26

## 2.26.1. Source

Xmark benchmark [http://monetdb.cwi.nl/xml/] Query 2

Description: *Return the initial increases of all open auctions.*

Solution in XQuery:

```
FOR $b IN doc("auction.xml")/site/open_auctions/open_auction
RETURN <increase> $b/bidder[1]/increase/text() </increase>
```

## 2.26.2. Query 26t: Text XML

Description: *Return the title of the first topic for all sections.*

The result of this query should be:

```
<result>
   <title>Procedural markup</title>
   <title>Structure</title>
</result>
```

#### 2.26.2.1. Complexity

Complexity = 4

- Report items: 1 [title elements]
- Condition items: 3 [parent = topic (1), position parent = first (1), ancestor = section (1)]

### 2.26.2.2. XQuery solution

```
<result>
{
for $b in //section
return
$b/topic[1]/title
}
</result>
```

```
//section/topic[1]/title
```

### 2.26.2.3. XSLT solution

```
<xsl:transform xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
version="2.0">
<xsl:output method="xml" indent="yes" />
<xsl:template match="text()"/>

<xsl:template match="/">
   <result>
      <xsl:apply-templates/>
   </result>
</xsl:template>

<xsl:template match="//section/topic[position()=1]">
   <xsl:copy-of select="title"/>
</xsl:template>
</xsl:transform>
```

```
<xsl:transform xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
version="2.0">
<xsl:output method="xml" indent="yes" />
<xsl:template match="text()"/>

<xsl:template match="//section/topic[position()=1]">
   <xsl:copy-of select="title"/>
</xsl:template>
</xsl:transform>
```

```
<xsl:transform xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
version="2.0">
<xsl:output method="xml" indent="yes" />
<xsl:template match="text()" />
<xsl:template match="/">
<xsl:for-each select="//section/topic[1]">
    <xsl:copy-of select="title" />
</xsl:for-each>
</xsl:template>
</xsl:transform>
```

## 2.26.3. Query 26d: Data XML

Description: *Return the source of the first store for all books.*

The result of this query should be:

```
<result>
   <source>bstore2.example.com</source>
   <source>bstore2.example.com</source>
   <source>bstore2.example.com</source>
   <source>www.amazon.com</source>
   <source>www.mystore.com</source>
   <source>bstore2.example.com</source>
   <source>bstore1.example.com</source>
   <source>bstore2.example.com</source>
   <source>www.amazon.com</source>
   <source>bstore2.example.com</source>
</result>
```

### 2.26.3.1. Complexity

Complexity = 4

- Report items: 1 [source elements]
- Condition items: 3 [parent = store (1), position parent = first (1), ancestor = book (1)]

### 2.26.3.2. XQuery solution

```
<result>
{
for $b in //book
return
$b/store[1]/source
}
</result>
```

```
//book/store[1]/source
```

### 2.26.3.3. XSLT solution

```
<xsl:transform xmlns:xsl=
"http://www.w3.org/1999/XSL/Transform"
version="2.0">
<xsl:output method="xml" indent="yes" />
<xsl:template match="/">
<xsl:for-each select="//book">
<xsl:copy-of select="store[1]/source" />
</xsl:for-each>
</xsl:template>
</xsl:transform>
```

```
<xsl:transform xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
version="2.0">
<xsl:output method="xml" indent="yes" />
<xsl:template match="text()"/>

<xsl:template match="/">
   <result>
      <xsl:apply-templates/>
   </result>
</xsl:template>

<xsl:template match="book/store[1]/source">
   <xsl:copy-of select="."/>
</xsl:template>
</xsl:transform>
```

```
<xsl:transform xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
version="2.0">
<xsl:output method="xml" indent="yes" />
<xsl:template match="text()"/>

<xsl:template match="book/store[1]/source">
   <xsl:copy-of select="."/>
</xsl:template>
</xsl:transform>
```

# 2.27. Query 27

## 2.27.1. Source

Xmark benchmark [http://monetdb.cwi.nl/xml/] Query 5

Description: *How many sold items cost more than 40?*

Solution in XQuery:

```
count(for $i in doc("auction.xml")/site/closed_auctions/closed_auction
        where  $i/price/text() >= 40
        return $i/price)
```

## 2.27.2. Query 27t: Text XML

Description: *How many topics have a number higher than or equal to 4?*

The result of this query should be:

```
3
```

### 2.27.2.1. Complexity

Complexity = 4

- Report items: 1 [number]
- Condition items: 3 [count (1) topic element (1) number attribute (1) with a certain value (1).]

### 2.27.2.2. XQuery solution

```
count(for $i in //topic
where  $i/@number >= 4
return $i)
```

```
count(for $i in //topic[@number >= 4]
return $i)
```

### 2.27.2.3. XSLT solution

```
<xsl:transform xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
```

```
version="2.0">
<xsl:output method="xml" indent="yes" />
<xsl:template match="text()" />
<xsl:template match="/">
    <xsl:value-of select="count(//topic[@number >= 4])" />
</xsl:template>
</xsl:transform>
```

```
<xsl:transform xmlns:xsl="http://www.w3.org/1999/XSL/Transform" version="2.0">
<xsl:template match="text()" />
<xsl:variable name="c" select="//topic[@number>=4]" />
<xsl:template match="/">
    <xsl:copy-of select="count($c)" />
</xsl:template>
</xsl:transform>
```

## 2.27.3. Query 27d: Data XML

Description: *How many books have a price higher than or equal to 40?*

The result of this query should be:

```
7
```

### 2.27.3.1. Complexity

Complexity = 4

- Report items: 1 [number]
- Condition items: 3 [count (1) book element (1) price element (1) with a certain value (1).]

### 2.27.3.2. XQuery solution

```
count(for $i in //book
        where  $i//price >= 40
        return $i)
```

```
count(for $i in //book[//price >= 40]
        return $i)
```

### 2.27.3.3. XSLT solution

```
<xsl:transform xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
version="2.0">
<xsl:template match="text()" />
<xsl:output method="xml" indent="yes" />

<xsl:variable name="price" select="//book[store/price >= 40]"/>

<xsl:template match="/">
   <result>
   <xsl:copy-of select="count($price)" />
   </result>
</xsl:template>
</xsl:transform>
```

```
<xsl:transform xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
version="2.0">
<xsl:output method="xml" indent="yes" />
<xsl:template match="text()" />
<xsl:template match="/">
    <xsl:value-of select="count(//book[store/price >= 40])" />
</xsl:template>
</xsl:transform>
```

# 2.28. Query 28

## 2.28.1. Source

Xmark benchmark [http://monetdb.cwi.nl/xml/] Query 6

Description: *How many items are listed on all continents?*

Solution in XQuery:

```
for    $b in doc("auction.xml")/site/regions
return count ($b//item)
```

## 2.28.2. Query 28t: Text XML

Description: *How many para elements are the document "report.xml"?*

The result of this query should be:

```
16
```

### 2.28.2.1. Complexity

Complexity = 3

- Report items: 1 [number]
- Condition items: 2 [para elements (1), count (1)]

### 2.28.2.2. XQuery solution

```
count(//para)
```

```
let $p := //para
return count($p)
```

### 2.28.2.3. XSLT solution

```
<xsl:transform xmlns:xsl=
"http://www.w3.org/1999/XSL/Transform"
version="2.0">
<xsl:output method="xml" indent="yes" />
<xsl:variable name="p" select="//para" />
```

```
<xsl:template match="/">
     <xsl:copy-of select="count($p)" />
</xsl:template>
</xsl:transform>
```

```
<xsl:transform xmlns:xsl=
"http://www.w3.org/1999/XSL/Transform"
version="2.0">
<xsl:template match="/">
     <xsl:copy-of select="count(//para)" />
</xsl:template>
</xsl:transform>
```

## 2.28.3. Query 28d: Data XML

Description: *How many source elements are in the pricelist?*

The result of this query should be:

```
21
```

### 2.28.3.1. Complexity

Complexity = 3

- Report items: 1 [number]
- Condition items: 2 [source elements (1), count (1)]

### 2.28.3.2. XQuery solution

```
count(//source)
```

```
let $s := //source
return count($s)
```

### 2.28.3.3. XSLT solution

```
<xsl:transform xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
version="2.0">
<xsl:template match="text()" />
<xsl:output method="xml" indent="yes" />

<xsl:variable name="sources" select="//source"/>

<xsl:template match="/">
    <xsl:copy-of select="count($sources)" />
</xsl:template>
</xsl:transform>
```

```
<xsl:transform xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
version="2.0">
<xsl:template match="/">
    <xsl:copy-of select="count(//source)" />
</xsl:template>
```

```
</xsl:transform>
```

# 2.29. Query 29

## 2.29.1. Source

Xmark benchmark [http://monetdb.cwi.nl/xml/] Query 7

Description: *How many pieces of prose are in our database?*

Solution in XQuery:

```
for $p in doc("auction.xml")/site
return count($p//description) + count($p//annotation) + count($p//email)
```

## 2.29.2. Query 29t: Text XML

Description: *How many titles, emph and keyword elements are in the document "report.xml"?*

The result of this query should be:

```
27
```

### 2.29.2.1. Complexity

Complexity = 5

- Report items: 1 [number]
- Condition items: 4 [title element (1), emph element (1), keyword element (1) count (1)]

### 2.29.2.2. XQuery solution

```
for $p in /report
return count($p//title) + count($p//emph) + count($p//keyword)
```

```
count(//title) + count(//emph) + count(//keyword)
```

### 2.29.2.3. XSLT solution

```
<xsl:transform xmlns:xsl=
"http://www.w3.org/1999/XSL/Transform"
version="2.0">

<xsl:variable name="elem" select="//title | //emph | //keyword" />

<xsl:template match="/">
    <xsl:value-of select="count($elem)" />
</xsl:template>
</xsl:transform>
```

```
<xsl:transform xmlns:xsl=
"http://www.w3.org/1999/XSL/Transform"
version="2.0">

<xsl:template match="/">
<xsl:copy-of select="count(//title | //emph | //keyword)"/>
</xsl:template>
</xsl:transform>
```

## 2.29.3. Query 29d: Data XML

Description: *How many titles, price and source elements are in our pricelist?*

The result of this query should be:

```
51
```

### 2.29.3.1. Complexity

Complexity = 5

- Report items: 1 [number]
- Condition items: 4 [title element (1), price element (1), source element (1) count (1)]

### 2.29.3.2. XQuery solution

```
for $p in /prices
return count($p//title) + count($p//price) + count($p//source)
```

```
count(//title) + count(//price) + count(//source)
```

### 2.29.3.3. XSLT solution

```
<xsl:transform xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
version="2.0">
<xsl:variable name="elem" select="//source | //title | //price"/>

<xsl:template match="/">
    <xsl:copy-of select="count($elem)" />
</xsl:template>
</xsl:transform>
```

```
<xsl:transform xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
version="2.0">

<xsl:template match="/">
    <xsl:copy-of select="count(//source | //title | //price)" />
</xsl:template>
</xsl:transform>
```

# 2.30. Query 30

## 2.30.1. Source

Xmark benchmark [http://monetdb.cwi.nl/xml/] Q8

Description: *List the names of persons and the number of items they bought. (joins person, closed_auction)}*

Solution in XQuery:

## 2.30.2. Query 30t: Text XML

Description: *List the title of all topics and the number of para elements in the topic(s) they refer to. A topic refers to another topic if an xref element occurs in the topic.*

The result of this query should be:

```
<results>
   <result>
      <title>Procedural markup</title>
      <count>0</count>
   </result>
   <result>
      <title>Generic markup</title>
      <count>0</count>
   </result>
   <result>
      <title>Drawbacks of procedural markup</title>
      <count>0</count>
   </result>
   <result>
      <title>Structure</title>
      <count>0</count>
   </result>
   <result>
      <title>Content</title>
      <count>2</count>
   </result>
   <result>
      <title>Style</title>
      <count>0</count>
   </result>
</results>
```

### 2.30.2.1. Complexity

Complexity = 16

- Subquery 1 (get the title, idref attributes (implicit), and result of subquery 2):
  - Report items: 5 (title element (1), result subquery 2 (2), implicit attribute value (2))
  - Condition items: 5 (parent = topic (1) with an id attribute (1) with a reference value equal to [report subquery 1] (2), parent idref = xref (1)).

  Complexity Subquery 1 = 10
- Subquery 2 (select the number of para's from topics with id equal to the implicit report of subquery 1):
  - Report items: 2 (number (1) of para's (1))
  - Condition items: 4 (parent = topic (1), with id attribute (1), with a certain reference value (2).)

  Complexity subquery 2 = 6

### 2.30.2.2. XQuery solution

```
<results>
{
for $p in //topic
let $a := for $t in //topic
             where $t/@id = $p//xref/@idref
             return $t//para
return
<result>
{$p/title}
<count>{count($a)}</count>
</result>
}
</results>
```

```
for $t in //topic
let $r := //topic[@id=$t//xref/@idref]
return <result>
        {$t/title}
        <count>{count($r//para)}</count>
       </result>
```

### 2.30.2.3. XSLT solution

```
<xsl:transform xmlns:xsl=
"http://www.w3.org/1999/XSL/Transform"
version="2.0">
<xsl:output method="xml" indent="yes" />
<xsl:variable name="root" select="/" />

<xsl:template match="/">
<result>
<xsl:for-each select="//topic">
      <xsl:variable name="refs" select=".//xref/@idref" />
      <xsl:copy-of select="title" />
      <count>
         <xsl:copy-of select="count($root//topic[@id=$refs]//para)" />
      </count>
</xsl:for-each>
</result>
</xsl:template>
</xsl:transform>
```

```
<xsl:transform xmlns:xsl="http://www.w3.org/1999/XSL/Transform" version="2.0">
<xsl:output method="xml" indent="yes" />
<xsl:template match="text()" />

<xsl:template match="topic">
<xsl:variable name="refs" select=".//xref/@idref" />
 <xsl:copy-of select="title" />
 <count><xsl:copy-of select="count(//topic[@id=$refs]//para)" /></count>
</xsl:template>
</xsl:transform>
```

## 2.30.3. Query 30d: Data XML

Description: *List the title of all books. Include the number of store elements of the books each book refers to.*

The result of this query should be:

```
<results>
   <result>
      <title>Advanced Programming in the Unix environment</title>
      <count>2</count>
   </result>
   <result>
      <title>TCP/IP Illustrated</title>
      <count>3</count>
   </result>
   <result>
      <title>Data on the Web</title>
      <count>0</count>
   </result>
   <result>
      <title>The Economics of Technology and Content for Digital TV</title>
      <count>0</count>
   </result>
   <result>
      <title>Getting started with SGML</title>
      <count>0</count>
   </result>
   <result>
      <title>Drawbacks of procedural markup</title>
      <count>2</count>
   </result>
   <result>
      <title>Lords of the Ring</title>
      <count>0</count>
   </result>
   <result>
      <title>XQuery from the experts</title>
      <count>2</count>
   </result>
   <result>
      <title>XSLT: programmers reference</title>
      <count>4</count>
   </result>
</results>
```

### 2.30.3.1. Complexity

Complexity = 16

- Subquery 1 (get the title, idref attributes (implicit), and result of subquery 2):
  - Report items: 5 (title element (1), result subquery 2 (2), implicit attribute value (2))
  - Condition items: 5 (parent = book (1) with an id attribute (1) with a reference value equal to [report subquery 1] (2), parent idref = xref (1)).

  Complexity Subquery 1 = 10
- Subquery 2 (select the number of store's from books with id equal to the implicit report of subquery 1):
  - Report items: 2 (number (1) of store's (1))
  - Condition items: 4 (parent = book (1), with id attribute (1), with a certain reference value (2).)

  Complexity subquery 2 = 6

### 2.30.3.2. XQuery solution

```
<results>
{
for $p in //book
let $a := for $t in //book
             where $t/@id = $p//xref/@idref
             return $t//store
return
<result>
{$p/title}
<count>{count($a)}</count>
</result>
}
```

```
</results>
```

```
for $t in //book
let $r := //book[@id=$t//xref/@idref]
return <result>
        {$t/title}
        <count>{count($r//store)}</count>
       </result>
```

### 2.30.3.3. XSLT solution

```
<xsl:transform xmlns:xsl="http://www.w3.org/1999/XSL/Transform" version="2.0">
<xsl:output method="xml" indent="yes" />
<xsl:template match="text()" />

<xsl:template match="book">
<xsl:variable name="refs" select=".//xref/@idref" />
 <xsl:copy-of select="title" />
 <count><xsl:copy-of select="count(//book[@id=$refs]//store)" /></count>
</xsl:template>
</xsl:transform>
```

```
<xsl:transform xmlns:xsl=
"http://www.w3.org/1999/XSL/Transform"
version="2.0">
<xsl:output method="xml" indent="yes" />
<xsl:variable name="root" select="/" />

<xsl:template match="/">
<result>
<xsl:for-each select="//book">
      <xsl:variable name="refs" select=".//xref/@idref" />
      <xsl:copy-of select="title" />
      <count>
         <xsl:copy-of select="count($root//book[@id=$refs]//store)" />
      </count>
</xsl:for-each>
</result>
</xsl:template>
</xsl:transform>
```

# 2.31. Query 31

## 2.31.1. Source

Xmark benchmark [http://monetdb.cwi.nl/xml/] Query 13

Description: *Q13. List the names of items registered in Australia along with their descriptions.*

Solution in XQuery:

```
for $i in doc("auction.xml")/site/regions/australia/item
return <item name="{$i/name/text()}"> {$i/description }</item>
```

## 2.31.2. Query 31t: Text XML

Description: *List the titles of chapters along with their intro, if present*

The result of this query should be:

```
<results>
  <chapter>
    <title>The business challenge</title>
    <intro>
      <para>With the ever-changing and growing global market,
      companies and large organizations are searching for ways to
      become more viable and competitive. Downsizing and other
      cost-cutting measures demand more efficient use of corporate
      resources. One very important resource is an organization's
      information.</para>
      <para>As part of the move toward integrated information
      management, whole industries are developing and implementing
      standards for exchanging technical information. This report
      describes how one such standard, the Standard Generalized
      Markup Language (SGML), works as part of an overall
      information management strategy.</para>
      <graphic graphname="infoflow" />
    </intro>
  </chapter>
  <chapter>
    <title>Getting to know SGML</title>
    <intro>
      <para>While SGML is a fairly recent technology, the use of
      <emph>markup</emph>in computer-generated documents has
      existed for a while.</para>
    </intro>
  </chapter>
  <chapter>
    <title>Resources</title>
  </chapter>
</results>
```

### 2.31.2.1. Complexity

Complexity = 4

- Report items: 3 [chapter element, title element, intro element]
- Condition items: 1 [report items title and intro have the same parent chapter (1)]

### 2.31.2.2. XQuery solution

```
<results>
{
for $i in //chapter
return
<chapter>
{$i/title}
{$i/intro}
</chapter>
}
</results>
```

### 2.31.2.3. XSLT solution

```
<xsl:transform xmlns:xsl=
"http://www.w3.org/1999/XSL/Transform"
version="2.0">
<xsl:output method="xml" indent="yes" />
<xsl:template match="/">
<results>
    <xsl:for-each select="//chapter">
    <chapter>
```

```
      <xsl:copy-of select="title" />
      <xsl:copy-of select="intro" />
   </chapter>
</xsl:for-each>
</results>
</xsl:template>
</xsl:transform>
```

```
<xsl:transform xmlns:xsl=
"http://www.w3.org/1999/XSL/Transform"
version="2.0">
<xsl:output method="xml" indent="yes" />
<xsl:template match="text()" />

<xsl:template match="chapter">
<chapter>
      <xsl:copy-of select="title" />
      <xsl:copy-of select="intro" />
</chapter>
</xsl:template>
</xsl:transform>
```

## 2.31.3. Query 31d: Data XML

Description: *List the titles of books along with their tips, if present*

The result of this query should be:

```
<results>
  <book>
    <title>Advanced Programming in the Unix environment</title>
    <tip>
      <xref idref="b2" />
    </tip>
  </book>
  <book>
    <title>TCP/IP Illustrated</title>
    <tip>
      <xref idref="b1" />
    </tip>
  </book>
  <book>
    <title>Data on the Web</title>
  </book>
  <book>
    <title>The Economics of Technology and Content for Digital
    TV</title>
  </book>
  <book>
    <title>Getting started with SGML</title>
  </book>
  <book>
    <title>Drawbacks of procedural markup</title>
    <tip>
      <xref idref="b5" />
    </tip>
  </book>
  <book>
    <title>Lords of the Ring</title>
  </book>
  <book>
    <title>XQuery from the experts</title>
    <tip>
      <xref idref="b9" />
    </tip>
  </book>
  <book>
    <title>XSLT: programmers reference</title>
    <tip>
```

```
      <xref idref="b8" />
      <xref idref="b5" />
    </tip>
  </book>
</results>
```

### 2.31.3.1. Complexity

Complexity = 4

- Report items: 3 [book element, title element, tip element]
- Condition items: 1 [report items title and tip have the same parent chapter (1)]

### 2.31.3.2. XQuery solution

```
<results>
{
for $i in //book
return
<book>
{$i/title}
{$i/tip}
</book>
}
</results>
```

### 2.31.3.3. XSLT solution

```
<xsl:transform
xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
version="2.0">
<xsl:output method="xml" indent="yes"/>
<xsl:template match="/">
<result>
<xsl:for-each select="//book">
 <book>
   <xsl:copy-of select="title" />
   <xsl:copy-of select=".//tip" />
 </book>
</xsl:for-each>
</result>
</xsl:template>
</xsl:transform>
```

```
<xsl:transform xmlns:xsl=
"http://www.w3.org/1999/XSL/Transform"
version="2.0">
<xsl:output method="xml" indent="yes" />
<xsl:template match="text()" />

<xsl:template match="book">
<book>
      <xsl:copy-of select="title" />
      <xsl:copy-of select=".//tip" />
</book>
</xsl:template>
</xsl:transform>
```

# 2.32. Query 32

## 2.32.1. Source

Xmark benchmark [http://monetdb.cwi.nl/xml/] Query 14

Description: *Return the names of all items whose description contains the word "gold".*

Solution in XQuery:

```
for $i in doc("auction.xml")/site//item
where contains ($i/description,"gold")
return $i/name/text()
```

## 2.32.2. Query 32t: Text XML

Description: *Return the titles of topics whose first para element contains the word "markup".*

The result of this query should be:

```
<result>
   <title>Procedural markup</title>
   <title>Generic markup</title>
   <title>Drawbacks of procedural markup</title>
</result>
```

### 2.32.2.1. Complexity

Complexity = 6

- Report items: 1 [title elements]
- Condition items: 5 [parent topic (1), para element (1), position =1 (1), contains (1) the word "Markup" (1)]

### 2.32.2.2. XQuery solution

```
<result>
{
for $i in //topic
where contains($i//para[1],"markup")
return $i/title
}
</result>
```

```
for $i in //topic[contains(para[1],"markup")]
return $i/title
```

### 2.32.2.3. XSLT solution

```
<xsl:transform xmlns:xsl=
"http://www.w3.org/1999/XSL/Transform"
version="2.0">
<xsl:output method="xml" indent="yes" />
<xsl:template match="/">
<result>
```

```
    <xsl:for-each select="//topic[para[1][contains(., 'markup')]]">
     <xsl:copy-of select="title" />
</xsl:for-each>
</result>
</xsl:template>
</xsl:transform>
```

```
<xsl:transform xmlns:xsl="http://www.w3.org/1999/XSL/Transform" version="2.0">
<xsl:output method="xml" indent="yes" />
<xsl:template match="text()"/>

<xsl:template match="topic[para[1][contains(., 'markup')]]/title">
  <xsl:copy-of select="." />
</xsl:template>
</xsl:transform>
```

## 2.32.3. Query 32d: Data XML

Description: *Return the titles of books when the title element contains the word "Unix"*

The result of this query should be:

```
<result>
   <title>Advanced Programming in the Unix environment</title>
</result>
```

### 2.32.3.1. Complexity

Complexity = 5

- Report items: 1 [title elements]
- Condition items: 4 [parent book (1), report item (1), contains (1) the word "Unix" (1)]

### 2.32.3.2. XQuery solution

```
<result>
{
for $i in //book
where contains($i/title,"Unix")
return $i/title
}
</result>
```

```
for $i in //book[contains(title,"Unix")]
return $i/title
```

### 2.32.3.3. XSLT solution

```
<xsl:transform
xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
version="2.0">
<xsl:output method="xml" indent="yes"/>

<xsl:template match="text()"/>

<xsl:template match="book/title[contains(.,'Unix')]">
<result><xsl:copy-of select="." /></result>
```

```
</xsl:template>
</xsl:transform>
```

```
<xsl:transform xmlns:xsl="http://www.w3.org/1999/XSL/Transform" version="2.0">
<xsl:output method="xml" indent="yes" />
<xsl:template match="text()"/>

<xsl:template match="book/title[contains(.,'Unix')]">
  <xsl:copy-of select="." />
</xsl:template>
</xsl:transform>
```

# 2.33. Query 33

## 2.33.1. Source

Xmark benchmark [http://monetdb.cwi.nl/xml/] Q15

Description: *Print the keywords in emphasis in annotations of closed auctions.*

Solution in XQuery:

```
for $a in doc("auction.xml")/site/closed_auctions/closed_auction/annotation/\
          description/parlist/listitem/parlist/listitem/text/emph/keyword/text()
return <text> $a <text>
```

## 2.33.2. Query 33t: Text XML

Description: *Print the emph elements in para elements with security attribute value "u".*

The result of this query should be:

```
<text>
   <emph>procedural</emph>
   <emph>rules</emph>
</text>
```

### 2.33.2.1. Complexity

Complexity = 5

- Report items: 1 [emph elements]
- Condition items: 4 [parent = para (1), attribute security (1) with a certain value (1)]

### 2.33.2.2. XQuery solution

```
let $a := //para[@security="u"]
return <text>{$a/emph}</text>
```

```
<result>
{
for $p in //para
where $p/@security = "u"
return $p/emph
```

```
}
</result>
```

### 2.33.2.3. XSLT solution

```
<xsl:transform
xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
version="2.0">
<xsl:output method="xml" indent="yes"/>
<xsl:template match="/">
<xsl:for-each select="//para[@security='u']">
 <xsl:copy-of select="emph" />
</xsl:for-each>
</xsl:template>
</xsl:transform>
```

```
<xsl:transform xmlns:xsl="http://www.w3.org/1999/XSL/Transform" version="2.0">
<xsl:template match="text()" />
<xsl:output method="xml" indent="yes" />

<xsl:template match="//para[@security = 'u']">
  <xsl:copy-of select="emph" />
</xsl:template>

</xsl:transform>
```

## 2.33.3. Query 33d: Data XML

Description: *Print the title elements in books elements with year attribute value 1982*

The result of this query should be:

```
<result>
   <title>Getting started with SGML</title>
</result>
```

### 2.33.3.1. Complexity

Complexity = 5

- Report items: 1 [title elements]
- Condition items: 4 [parent = book (1), attribute year (1) with a certain value (1)]

### 2.33.3.2. XQuery solution

```
let $a := //book[@year="1982"]
return $a/title
```

```
for $b in //book
where $b/@year = "1982"
return $b/title
```

#### 2.33.3.3. XSLT solution

```
<xsl:transform xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
version="2.0">
<xsl:template match="text()" />
<xsl:output method="xml" indent="yes" />

<xsl:template match="book[@year = 1982]/title">
    <xsl:copy-of select="." />
</xsl:template>
</xsl:transform>
```

```
<xsl:transform
xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
version="2.0">
<xsl:output method="xml" indent="yes"/>
<xsl:template match="/">
<result>
<xsl:for-each select="//book[@year = 1982]/title">
 <xsl:copy-of select="." />
</xsl:for-each>
</result>
</xsl:template>
</xsl:transform>
```

# 2.34. Query 34

## 2.34.1. Source

Xmark benchmark [http://monetdb.cwi.nl/xml/] Query 19

Description: *Q19. Give an alphabetically ordered list of all items along with their location.*

Solution in XQuery:

```
FOR    $b IN doc("auction.xml")/site/regions//item
LET    $k := $b/name/text()
RETURN <item name=$k> $b/location/text() </item>
SORTBY (.)
[note: query does not provide the correct results, description of our queries is orbably simplified]
```

## 2.34.2. Query 34t: Text XML

Description: *Make a list of all authors, order by last name*

The result of this query should be:

```
<result>
   <author>
      <last>Abiteboul</last>
      <first>Serge</first>
   </author>
   <author>
      <last>Buneman</last>
      <first>Peter</first>
   </author>
   <author>
      <last>Stevens</last>
      <first>W.</first>
```

```
   </author>
   <author>
      <last>Stevens</last>
      <first>W.</first>
   </author>
   <author>
      <last>Suciu</last>
      <first>Dan</first>
   </author>
</result>
```

### 2.34.2.1. Complexity

Complexity = 3

- Report items: 1 [author elements]
- Condition items: 4 [last element (1), sorted by last (1)]

### 2.34.2.2. XQuery solution

```
<result>
{
for $a in //author
order by $a/last
return $a
}
</result>
```

### 2.34.2.3. XSLT solution

```
<xsl:transform
xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
version="2.0">
<xsl:output method="xml" indent="yes"/>
<xsl:template match="/">
<listings>
<xsl:for-each select="//author">
<xsl:sort select="last" />
<xsl:copy-of select="." />
</xsl:for-each>
</listings>
</xsl:template>
</xsl:transform>
```

## 2.34.3. Query 34d: Data XML

Description: *Give a list of all books, ordered by title*

The result of this query should be [only the first and last book are shown:]:

```
<result>
  <book year="1992" id="b1">
    <title>Advanced Programming in the Unix environment</title>
    <store>
      <source>bstore2.example.com</source>
      <price>65.95</price>
    </store>
    <store>
      <source>bstore1.example.com</source>
      <price>65.95</price>
    </store>
    <store>
      <source>www.bol.com</source>
```

```
      <price>45.95</price>
    </store>
    <tip>
      <xref idref="b2" />
    </tip>
  </book>
<!-- SNIP  -->
  <book year="1999" id="b9">
    <title>XSLT: programmers reference</title>
    <store>
      <source>bstore2.example.com</source>
      <price>35.95</price>
    </store>
    <store>
      <source>www.amazon.com</source>
      <price>40.95</price>
    </store>
    <tip>
      <xref idref="b8" />
      <xref idref="b5" />
    </tip>
  </book>
</result>
```

### 2.34.3.1. Complexity

Complexity = 3

- Report items: 1 [book elements]
- Condition items: 4 [title element (1), sorted by title (1)]

### 2.34.3.2. XQuery solution

```
<result>
{
for $a in //book
order by $a/title
return $a
}
</result>
```

### 2.34.3.3. XSLT solution

```
<xsl:transform
xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
version="2.0">
<xsl:output method="xml" indent="yes"/>
<xsl:template match="/">
<result>
<xsl:for-each select="//book">
  <xsl:sort select="title" />
  <xsl:copy-of select="." />
</xsl:for-each>
</result>
</xsl:template>
</xsl:transform>
```

# 2.35. Query 35

## 2.35.1. Source

X-mach benchmark [http://dbs.uni-leipzig.de/en/projekte/XML/XMach-1_queries.html] Query 4.

Description: *Return a flat list of head elements which are children of section elements. Parameter: doc_id = "d1", suffix = "10".*

Solution in XQuery:

```
for $a in /document10[@doc_id="d1"]//section10/head10
return $a
```

## 2.35.2. Query 35t: Text XML

Description: *Return flat list of last elements which are children of author elements. Select only authors from chapter with number "3"*

The result of this query should be:

```
<result>
   <last>Abiteboul</last>
   <last>Buneman</last>
   <last>Suciu</last>
</result>
```

### 2.35.2.1. Complexity

Complexity = 5

- Report items: 1 [last elements]
- Condition items: 4 [parent = author (1), ancester = chapter (1), attribute number (1), with a certain value (1)]

### 2.35.2.2. XQuery solution

```
<result>
{
for $a in //chapter
where $a/@number="3"
return $a//author/last
}
</result>
```

```
//chapter[@number="3"]//author/last
```

### 2.35.2.3. XSLT solution

```
<xsl:transform xmlns:xsl=
"http://www.w3.org/1999/XSL/Transform"
version="2.0">
<xsl:output method="xml" indent="yes" />
<xsl:template match="text()" />
<xsl:template match="chapter[@number = 3]">
<result>
    <xsl:for-each select="author/last">
     <xsl:copy-of select="." />
</xsl:for-each>
</result>
</xsl:template>
</xsl:transform>
```

```
<xsl:transform xmlns:xsl="http://www.w3.org/1999/XSL/Transform" version="2.0">
<xsl:output method="xml" indent="yes" />
<xsl:template match="text()" />

<xsl:template match="chapter[@number=3]">
  <xsl:copy-of select="author/last"/>
</xsl:template>

</xsl:transform>
```

## 2.35.3. Query 35d: Data XML

Description: *Return flat list of source elements which are children of store elements. Select only source elements from book with year "2003"*

The result of this query should be:

```
<result>
   <source>bstore2.example.com</source>
   <source>www.amazon.com</source>
</result>
```

### 2.35.3.1. Complexity

Complexity = 4

- Report items: 1 [source elements]
- Condition items: 3 [parent = book (1), attribute year (1), with a certain value (1)]

### 2.35.3.2. XQuery solution

```
<result>
{
for $a in //book
where $a/@year="2003"
return $a//store/source
}
</result>
```

```
//book[@year="2003"]//store/source
```

### 2.35.3.3. XSLT solution

```
<xsl:transform
xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
version="2.0">
<xsl:output method="xml" indent="yes"/>
<xsl:template match="text()"/>

<xsl:template match="book[@year=2003]">
<result>
<xsl:for-each select="store">
 <xsl:copy-of select="source" />
</xsl:for-each>
</result>
</xsl:template>
</xsl:transform>
```

```
<xsl:transform xmlns:xsl="http://www.w3.org/1999/XSL/Transform" version="2.0">
<xsl:output method="xml" indent="yes" />
<xsl:template match="text()" />

<xsl:template match="book[@year=2003]">
  <xsl:copy-of select="store/source"/>
</xsl:template>

</xsl:transform>
```

# 2.36. Query 36

## 2.36.1. Source

Xbench family of benchmarks [http://db.uwaterloo.ca/~ddbms/projects/xbench/] TC/MD Query 1

Description: *TC/MD_Q1 Return the title of the article that has matching id attribute value (1).*

Solution in XQuery:

```
for $art in input()/article[@id="1"]
return
    $art/prolog/title
```

## 2.36.2. Query 36t: Text XML

Description: *Return the title of the chapter that has matching number attribute value (1).*

The result of this query should be:

```
<title>The business challenge</title>
```

### 2.36.2.1. Complexity

Complexity = 4

- Report items: 1 [title elements]
- Condition items: 3 [parent = chapter (1), attribute number (1), with a certain value (1)]

### 2.36.2.2. XQuery solution

```
for $a in //chapter[@number="1"]
return
    $a/title
```

```
<result>
{
for $a in //chapter
where $a/@number = "1"
return $a/title
}
</result>
```

### 2.36.2.3. XSLT solution

```
<xsl:transform xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
version="2.0">
<xsl:output method="xml" indent="yes" />
<xsl:template match="text()" />
<xsl:template match="chapter[@number=1]">
    <xsl:copy-of select="title" />
</xsl:template>
</xsl:transform>
```

## 2.36.3. Query 36d: Data XML

Description: *Return the title of the book that has matching id attribute value (b3).*

The result of this query should be:

```
<title>Data on the Web</title>
```

### 2.36.3.1. Complexity

Complexity = 4

- Report items: 1 [title elements]
- Condition items: 3 [parent = book (1), attribute id (1), with a certain value (1)]

### 2.36.3.2. XQuery solution

```
<result>
{
for $a in //book
where $a/@id = "b3"
return $a/title
}
</result>
```

```
for $a in //book[@id="b3"]
return
    $a/title
```

### 2.36.3.3. XSLT solution

```
<xsl:transform xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
version="2.0">
<xsl:output method="xml" indent="yes" />
<xsl:template match="text()" />
<xsl:template match="book[@id='b3']">
    <xsl:copy-of select="title" />
</xsl:template>
</xsl:transform>
```

# 2.37. Query 37

## 2.37.1. Source

Xbench family of benchmarks [http://db.uwaterloo.ca/~ddbms/projects/xbench/] Query 2

Description: *Find the title of the article authored by (Ben Yang).*

Solution in XQuery:

```
for $prolog in doc("collection.xml")/article/prolog
where $prolog/authors/author/name="Ben Yang"
return $prolog/title
```

## 2.37.2. Query 37t: Text XML

Description: *Find the title of the chapter authored by an author with last name "Suciu".*

The result of this query should be:

```
<title>Resources</title>
```

### 2.37.2.1. Complexity

Complexity = 5

- Report items: 1 [title elements]
- Condition items: 4 [parent = chapter (1), author (1), last element (1) with a certain value (1)]

### 2.37.2.2. XQuery solution

```
for $a in //chapter[author/last="Suciu"]
return
    $a/title
```

```
<result>
{
for $a in //chapter
where $a/author/last = "Suciu"
return $a/title
}
</result>
```

### 2.37.2.3. XSLT solution

```
<xsl:transform xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
version="2.0">
<xsl:output method="xml" indent="yes" />
<xsl:template match="text()" />
<xsl:template match="chapter[author/last='Suciu']">
    <xsl:copy-of select="title" />
</xsl:template>
</xsl:transform>
```

```
<xsl:transform xmlns:xsl="http://www.w3.org/1999/XSL/Transform" version="2.0">
<xsl:template match="text()"/>
<xsl:output method="xml" indent="yes"/>

<xsl:template match="/">
 <xsl:for-each select="//chapter[author/last='Suciu']">
  <xsl:copy-of select="title"/>
 </xsl:for-each>
</xsl:template>

</xsl:transform>
```

## 2.37.3. Query 37d: Data XML

Description: *Return the title of the book that is sold in a store with source "www.bol.com"*

The result of this query should be:

```
<title>Advanced Programming in the Unix environment</title>
<title>The Economics of Technology and Content for Digital TV</title>
<title>Getting started with SGML</title>
<title>Lords of the Ring</title>
```

### 2.37.3.1. Complexity

Complexity = 5

- Report items: 1 [title elements]
- Condition items: 4 [parent = book (1), store (1), source element (1) with a certain value (1)]

### 2.37.3.2. XQuery solution

```
for $b in //book[store/source="www.bol.com"]
return
    $b/title
```

```
<result>
{
for $b in //book
where $b/store/source="www.bol.com"
return $b/title
}
</result>
```

### 2.37.3.3. XSLT solution

```
<xsl:transform xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
version="2.0">
<xsl:template match="text()" />
<xsl:output method="xml" indent="yes" />

<xsl:variable name="book" select="//book[store/source='www.bol.com']" />

<xsl:template match="/">
    <xsl:copy-of select="$book/title" />
</xsl:template>
```

```
</xsl:transform>
```

```
<xsl:transform xmlns:xsl="http://www.w3.org/1999/XSL/Transform" version="2.0">
<xsl:template match="text()"/>
<xsl:output method="xml" indent="yes"/>

<xsl:template match="/">
 <xsl:for-each select="//book[store/source='www.bol.com']">
  <xsl:copy-of select="title"/>
 </xsl:for-each>
</xsl:template>

</xsl:transform>
```

# 2.38. Query 38

## 2.38.1. Source

Xbench family of benchmarks [http://db.uwaterloo.ca/~ddbms/projects/xbench/] Query 3.

Description: *Group articles by date and calculate the total number of articles in each group.*

Solution in XQuery:

```
for $a in distinct-values (doc("collection.xml")/article/prolog/dateline/date)
let $b := doc("collection.xml")/article/prolog/dateline[date=$a]
return
    <Output>
        <Date>{$a/text()}</Date>
        <NumberOfArticles>{count($b)}</NumberOfArticles>
    </Output>
```

## 2.38.2. Query 38t: Text XML

Description: *Group topics by keyword and calculate the total number of topics in each group.*

The result of this query should be:

```
<results>
   <result>
      <keyword>Markup</keyword>
      <number_of_topics>4</number_of_topics>
   </result>
   <result>
      <keyword>DTD</keyword>
      <number_of_topics>1</number_of_topics>
   </result>
</results>
```

### 2.38.2.1. Complexity

Complexity = 15.

- Subquery 1:
  - Report: 2 [report items subquery 2 (1) and subquery 3. (1)]
  - Condition: 4 [two reference items: [report item subquery 2] = [the condition value subquery 2]]

  Complexity subquery 1 = 6
- Subquery 2:
  - Report: 1 [keyword elements]

    - Condition: 1 [unique values]

  Complexity: 2
- Subquery 3:
    - Report: 1 [a number]
    - Condition: 6 [parent topic (1) has keyword (1) that = [the report of subquery 2] (2), count (1) para elements (1)]

  Complexity subquery 3 = 7.

### 2.38.2.2. XQuery solution

```
<results>
{
for $a in distinct-values (//keyword)
let $b := //topic[keyword=$a]
return
    <result>
        <keyword>{$a}</keyword>
        <number_of_topics>{count($b)}</number_of_topics>
    </result>
}
</results>
```

```
<results>
{
for $k in distinct-values(//keyword)
return
<result>
<keyword>{$k}</keyword>
<number_of_topics>
{count(//topic[keyword=$k])}
</number_of_topics>
</result>
}
</results>
```

### 2.38.2.3. XSLT solution

```
<xsl:transform
xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
version="2.0">
<xsl:output method="xml" indent="yes"/>

<xsl:template match="/">
<results>
<xsl:for-each-group select="//topic" group-by="keyword">
 <result>
        <keyword><xsl:copy-of select="current-grouping-key()"/></keyword>
        <count><xsl:copy-of select="count(current-group())"/></count>
        </result>
</xsl:for-each-group>
</results>
</xsl:template>
</xsl:transform>
```

## 2.38.3. Query 38d: Data XML

Description: *Group books by source and calculate the total number of books in each group.*

The result of this query should be:

```
<results>
   <result>
      <source>bstore2.example.com</source>
      <number_of_books>7</number_of_books>
   </result>
   <result>
      <source>bstore1.example.com</source>
      <number_of_books>4</number_of_books>
   </result>
   <result>
      <source>www.bol.com</source>
      <number_of_books>4</number_of_books>
   </result>
   <result>
      <source>www.amazon.com</source>
      <number_of_books>4</number_of_books>
   </result>
   <result>
      <source>www.mystore.com</source>
      <number_of_books>1</number_of_books>
   </result>
   <result>
      <source>bstore3.example.com</source>
      <number_of_books>1</number_of_books>
   </result>
</results>
```

### 2.38.3.1. Complexity

Complexity = 15.

- Subquery 1:
  - Report: 2 [report items subquery 2 (1) and subquery 3. (1)]
  - Condition: 4 [two reference items: [report item subquery 2] = [the condition value subquery 2]]

  Complexity subquery 1 = 6
- Subquery 2:
  - Report: 1 [source elements]
  - Condition: 1 [unique values]

  Complexity: 2
- Subquery 3:
  - Report: 1 [a number]
  - Condition: 6 [book element (1) has source (1) that = [the report of subquery 2] (2), count (1) book elements (1)]

  Complexity subquery 3 = 7.

### 2.38.3.2. XQuery solution

```
<results>
{
for $a in distinct-values (//source)
let $b := //book[store/source=$a]
return
    <result>
        <source>{$a}</source>
        <number_of_books>{count($b)}</number_of_books>
    </result>
}
</results>
```

```
<results>
{
for $s in distinct-values(//source)
return
<result>
```

```
<source>{$s}</source>
<number_of_books>
{count(//book[.//source=$s])}
</number_of_books>
</result>
}
</results>
```

#### 2.38.3.3. XSLT solution

```
<xsl:transform
xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
version="2.0">
<xsl:output method="xml" indent="yes"/>

<xsl:template match="/">
<results>
<xsl:for-each-group select="//book" group-by="store/source">
<result>
<source><xsl:copy-of select="current-grouping-key()"/></source>
<count><xsl:copy-of select="count(current-group())"/></count>

</result>
</xsl:for-each-group>
</results>
</xsl:template>
</xsl:transform>
```

# 2.39. Query 39

## 2.39.1. Source

Xbench family of benchmarks [http://db.uwaterloo.ca/~ddbms/projects/xbench/] Query 5.

Description: *Return the headings of the first section of a certain article with id attribute value (9).*

Solution in XQuery:

```
for $a in doc("collection.xml")/article[@id="9"]
return
    <HeadingOfSection>
        {$a/body/section[1]/@heading}
    </HeadingOfSection>
```

## 2.39.2. Query 39t: Text XML

Description: *Return the shorttitle of the first section of a certain chapter with number attribute value (2).*

The result of this query should be:

```
<section shorttitle="What is markup?"/>
```

#### 2.39.2.1. Complexity

Complexity = 6

- Report items: 2 [section elements, shortttitle attributes]
- Condition items: 4 [parent = chapter (1), attribute number (1), with a certain value (1), position report item = 1 (1)]

### 2.39.2.2. XQuery solution

```
for $a in //chapter
where $a/@number=2
return
    <section>
        {$a/section[1]/@shorttitle}
    </section>
```

```
<section>
{//chapter[@number=2]/section[1]/@shorttitle}
</section>
```

### 2.39.2.3. XSLT solution

```
<xsl:transform
xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
version="2.0">
<xsl:output method="xml" indent="yes"/>

<xsl:template match="text()" />

<xsl:template match="chapter[@number=2]">
<xsl:for-each select="section[1]">
<section><xsl:copy-of select="@shorttitle" /></section>
</xsl:for-each>
</xsl:template>
</xsl:transform>
```

```
<xsl:transform xmlns:xsl="http://www.w3.org/1999/XSL/Transform" version="2.0">
<xsl:output method="xml" indent="yes" />
<xsl:template match="text()"/>

<xsl:template match="//chapter[@number=2]/section[position()=1]">
   <section>
      <xsl:copy-of select="@shorttitle"/>
   </section>
</xsl:template>
</xsl:transform>
```

## 2.39.3. Query 39d: Data XML

Description: *Return the source of the first store of a certain book with id attribute value (b4).*

The result of this query should be:

```
<book>
   <source>www.amazon.com</source>
</book>
```

### 2.39.3.1. Complexity

Complexity = 6

- Report items: 1 [source elements]
- Condition items: 5 [parent = store (1) at position = 1 (1), ancestor = book (1), attribute id (1), with a certain value (1)]

### 2.39.3.2. XQuery solution

```
for $b in //book
where $b/@id="b4"
return
    <book>
        {$b/store[1]/source}
    </book>
```

```
<book>
{//book[@id='b4']/store[1]/source}
</book>
```

### 2.39.3.3. XSLT solution

```
<xsl:transform
xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
version="2.0">
<xsl:output method="xml" indent="yes"/>

<xsl:template match="text()" />

<xsl:template match="book[@id='b4']">
<xsl:for-each select="store[1]">
<section><xsl:copy-of select="source" /></section>
</xsl:for-each>
</xsl:template>
</xsl:transform>
```

```
<xsl:transform
xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
version="2.0">
<xsl:output method="xml" indent="yes"/>
<xsl:template match="text()" />

<xsl:template match="book[@id='b4']">
<result>
<xsl:copy-of select="store[1]/source" />
</result>
</xsl:template>
</xsl:transform>
```

# 2.40. Query 40

## 2.40.1. Source

Xbench family of benchmarks [http://db.uwaterloo.ca/~ddbms/projects/xbench/] Query 8 [simplified]

Description: *Return the names of all authors (one element name unknown) of the article with matching id attribute value (2).*

Solution in XQuery:

```
for $art in doc("collection.xml")/article[@id="2"]
return
    $art/prolog/*/author/name
```

## 2.40.2. Query 40t: Text XML

Description: *Return the graphic elements of the topic with title "Content".*

The result of this query should be:

```
<graphic graphname="tagexamp"/>
```

### 2.40.2.1. Complexity

Complexity = 4

- Report items: 1 [graphic elements]
- Condition items: 3 [ancestor = topic (1) with title (1), with a certain value (1)]

### 2.40.2.2. XQuery solution

```
for $t in //topic[title="Content"]
return
    $t//graphic
```

```
for $t in //topic
where $t/title="Content"
return $t//graphic
```

### 2.40.2.3. XSLT solution

```
<xsl:transform
xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
version="2.0">
<xsl:template match="text()" />
<xsl:template match="//topic[title='Content']">

<xsl:copy-of select="graphic"/>

</xsl:template>
</xsl:transform>
```

```
<xsl:transform
xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
version="2.0">
<xsl:output method="xml" indent="yes"/>
<xsl:template match="/">
<xsl:for-each select="//topic[title='Content']">
 <xsl:copy-of select="graphic" />
</xsl:for-each>
</xsl:template>
</xsl:transform>
```

## 2.40.3. Query 40d: Data XML

Description: *Return the xref elements of the book with title "Drawbacks of procedural markup".*

The result of this query should be:

```
<xref idref="b5"/>
```

### 2.40.3.1. Complexity

Complexity = 4

- Report items: 1 [xref elements]
- Condition items: 3 [ancestor = book (1) with title (1), with a certain value (1)]

### 2.40.3.2. XQuery solution

```
for $t in //book[title="Drawbacks of procedural markup"]
return
    $t//xref
```

```
for $t in //book
where $t/title="Drawbacks of procedural markup"
return $t//xref
```

### 2.40.3.3. XSLT solution

```
<xsl:transform
xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
version="2.0">
<xsl:template match="text()" />
<xsl:template match="//book[title='Drawbacks of procedural markup']">

<xsl:copy-of select=".//xref"/>

</xsl:template>
</xsl:transform>
```

```
<xsl:transform
xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
version="2.0">
<xsl:output method="xml" indent="yes"/>
<xsl:template match="/">
<xsl:for-each select="//book[title='Drawbacks of procedural markup']">
 <xsl:copy-of select=".//xref" />
</xsl:for-each>
</xsl:template>
</xsl:transform>
```

# 2.41. Query 41

## 2.41.1. Source

Xbench family of benchmarks [http://db.uwaterloo.ca/~ddbms/projects/xbench/] Query 9

Description: *Return all author names (several consecutive element unknown) of the article with matching id attribute value (3).*

Solution in XQuery:

```
for $art in doc("collection.xml")/article[@id="3"]
return
    $art//author/name
```

## 2.41.2. Query 41t: Text XML

Description: *Return the xref elements of the topic with keyword "Markup".*

The result of this query should be:

```
<xref idref="top4"/>
```

### 2.41.2.1. Complexity

Complexity = 4

- Report items: 1 [xref elements]
- Condition items: 3 [ancestor = topic (1) with keyword (1), with a certain value (1)]

### 2.41.2.2. XQuery solution

```
for $t in //topic[keyword="Markup"]
return
    $t//xref
```

```
//topic[keyword="Markup"]//xref
```

### 2.41.2.3. XSLT solution

```
<xsl:transform xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
version="2.0">
<xsl:output method="xml" indent="yes" />
<xsl:template match="text()" />
<xsl:template match="topic[keyword='Markup']">
    <xsl:copy-of select=".//xref" />
</xsl:template>
</xsl:transform>
```

## 2.41.3. Query 41d: Data XML

Description: *Return the price elements of the book with year attribute value (2000).*

The result of this query should be:

```
<price>34.95</price>
<price>39.95</price>
```

### 2.41.3.1. Complexity

Complexity = 4

- Report items: 1 [price elements]
- Condition items: 3 [ancestor = book (1) with attribute year (1), with a certain value (1)]

#### 2.41.3.2. XQuery solution

```
for $t in //book[@year="2000"]
return
    $t//price
```

```
//book[@year="2000"]//price
```

#### 2.41.3.3. XSLT solution

```
<xsl:transform
xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
version="2.0">
<xsl:output method="xml" indent="yes"/>
<xsl:template match="text()"/>
<xsl:template match="book[@year=2000]">
<xsl:copy-of select=".//price" />
</xsl:template>
</xsl:transform>
```

# 2.42. Query 42

## 2.42.1. Source

Xbench family of benchmarks [http://db.uwaterloo.ca/~ddbms/projects/xbench/]

Description: *List the titles of articles sorted by country.*

Solution in XQuery:

```
for $a in doc("collection.xml")/article/prolog
order by $a/dateline/country
return
    <Output>
        {$a/title}
        {$a/dateline/country}
    </Output>
```

## 2.42.2. Query 42t: Text XML

Description: *List the titles of chapters sorted by the first's author first name. Provide also this author first element.*

The result of this query should be:

```
<result>
   <result>
      <title>Resources</title>
      <first>Serge</first>
   </result>
   <result>
      <title>The business challenge</title>
      <first>W.</first>
   </result>
```

```
   <result>
      <title>Getting to know SGML</title>
      <first>W.</first>
   </result>
</result>
```

### 2.42.2.1. Complexity

Complexity = 6

- Report items: 2 [title and first elements]
- Condition items: 4 [parent = chapter (1), author (1), position = 1 (1), sort by report item (1)]

### 2.42.2.2. XQuery solution

```
<result>
{
for $a in //chapter
order by $a/author[1]/first
return
    <result>
        {$a/title}
        {$a/author[1]/first}
    </result>
}
</result>
```

### 2.42.2.3. XSLT solution

```
<xsl:transform
xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
version="2.0">
<xsl:output method="xml" indent="yes"/>
<xsl:template match="/">
<results>
<xsl:for-each select="//chapter">
<xsl:sort select="author[1]/first" />
<result>
<xsl:copy-of select="title" />
<xsl:copy-of select="author[1]/first" />
</result>
</xsl:for-each>
</results>
</xsl:template>
</xsl:transform>
```

## 2.42.3. Query 42d: Data XML

Description: *List the titles of books sorted by the first's store source. Provide also this source element.*

The result of this query should be:

```
<result>
   <result>
      <title>Lords of the Ring</title>
      <source>bstore1.example.com</source>
   </result>
   <result>
      <title>Advanced Programming in the Unix environment</title>
      <source>bstore2.example.com</source>
   </result>
   <result>
      <title>TCP/IP Illustrated</title>
```

```
      <source>bstore2.example.com</source>
   </result>
   <result>
      <title>Data on the Web</title>
      <source>bstore2.example.com</source>
   </result>
   <result>
      <title>Drawbacks of procedural markup</title>
      <source>bstore2.example.com</source>
   </result>
   <result>
      <title>XQuery from the experts</title>
      <source>bstore2.example.com</source>
   </result>
   <result>
      <title>XSLT: programmers reference</title>
      <source>bstore2.example.com</source>
   </result>
   <result>
      <title>The Economics of Technology and Content for Digital TV</title>
      <source>www.amazon.com</source>
   </result>
   <result>
      <title>Getting started with SGML</title>
      <source>www.mystore.com</source>
   </result>
</result>
```

### 2.42.3.1. Complexity

Complexity = 6

- Report items: 2 [title and source elements]
- Condition items: 4 [parent = book (1), store (1), position = 1 (1), sort by report item (1)]

### 2.42.3.2. XQuery solution

```
<result>
{
for $a in //book
order by $a/store[1]/source
return
    <result>
        {$a/title}
        {$a/store[1]/source}
    </result>
}
</result>
```

### 2.42.3.3. XSLT solution

```
<xsl:transform
xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
version="2.0">
<xsl:output method="xml" indent="yes"/>
<xsl:template match="/">
<listings>
<xsl:for-each select="//book">
<xsl:sort select="./store[1]/source" />
<result>
<xsl:copy-of select="title" />
<xsl:copy-of select="./store[1]/source" />
</result>
</xsl:for-each>
</listings>
</xsl:template>
```

```
</xsl:transform>
```

# 2.43. Query 43

## 2.43.1. Source

Xbench family of benchmarks [http://db.uwaterloo.ca/~ddbms/projects/xbench/] Query 11

Description: *List the titles of articles that have a matching country element type (Canada), sorted by date.*

Solution in XQuery:

```
for $a in doc("collection.xml")/article/prolog
where $a/dateline/country="Canada"
order by $a/dateline/date
return
    <Output>
        {$a/title}
        {$a/dateline/date}
    </Output>
```

## 2.43.2. Query 43t: Text XML

Description: *List the title and para elements of topics that have a matching keyword element "Markup", order by title*

The result of this query should be:

```
<result>
  <result>
    <title>Content</title>
    <para>Content is the information itself. The method for
    identifying the information and its meaning within this
    framework is called
    <emph>tagging</emph>. Tagging must conform to the rules
    established in the DTD (see
    <xref idref="top4" />).</para>
  </result>
  <result>
    <title>Drawbacks of procedural markup</title>
    <para security="u">Industries involved in technical
    documentation increasingly prefer generic over
    <emph>procedural</emph>markup schemes. When a company changes
    software or hardware systems, enormous data translation tasks
    arise, often resulting in errors.</para>
  </result>
  <result>
    <title>Generic markup</title>
    <para>Generic markup (also known as descriptive markup)
    describes the
    <emph>purpose</emph>of the text in a document. A basic concept
    of generic markup is that the content of a document must be
    separate from the style. Generic markup allows for multiple
    presentations of the information.</para>
  </result>
  <result>
    <title>Procedural markup</title>
    <para security="u">Most electronic publishing systems today use
    some form of procedural markup. Procedural markup codes are
    good for one presentation of the information.</para>
  </result>
</result>
```

### 2.43.2.1. Complexity

Complexity = 6

- Report items: 2 [title and para elements]
- Condition items: 4 [parent = topic (1), keyword (1), value "Markup" (1), sort by report item (1)]

### 2.43.2.2. XQuery solution

```
<result>
{
for $a in //topic
where $a/keyword="Markup"
order by $a/title
return
    <result>
        {$a/title}
        {$a/para}
    </result>
}
</result>
```

```
for $t in //topic[keyword = "Markup"]
order by $t/title
return
<result>
  {$t/title}
  {$t/para}
</result>
```

### 2.43.2.3. XSLT solution

```
<xsl:transform xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
version="2.0">
<xsl:template match="text()" />
<xsl:output method="xml" indent="yes" />

<xsl:template match="/">
<results>
<xsl:for-each select="//topic[keyword='Markup']">
<xsl:sort select="title" />
<xsl:copy-of select="title" />
<xsl:copy-of select="para"/>
</xsl:for-each>
</results>
</xsl:template>
</xsl:transform>
```

## 2.43.3. Query 43d: Data XML

Description: *List the title and tip elements of books that have a price element "65.95", order by title*

The result of this query should be:

```
<results>
  <result>
    <title>Advanced Programming in the Unix environment</title>
    <tip>
      <xref idref="b2" />
    </tip>
  </result>
```

```
  <result>
    <title>Lords of the Ring</title>
  </result>
  <result>
    <title>TCP/IP Illustrated</title>
    <tip>
      <xref idref="b1" />
    </tip>
  </result>
</results>
```

### 2.43.3.1. Complexity

Complexity = 6

- Report items: 2 [title and tip elements]
- Condition items: 4 [parent = books (1), price (1), value 65.95 (1), sort by report item (1)]

### 2.43.3.2. XQuery solution

```
<results>
{
for $a in //book
where $a//price="65.95"
order by $a/title
return
    <result>
        {$a/title}
        {$a/tip}
    </result>
}
</results>
```

```
<results>
{
for $a in //book[//price="65.95"]
order by $a/title
return
    <result>
        {$a/title}
        {$a/tip}
    </result>
}
</results>
```

### 2.43.3.3. XSLT solution

```
<xsl:transform
xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
version="2.0">
<xsl:output method="xml" indent="yes"/>
<xsl:template match="text()"/>

<xsl:template match="/">
<results>
<xsl:for-each select="//book[.//price='65.95']">
<xsl:sort select="title" />
<book>
<xsl:copy-of select="title" />
<xsl:copy-of select="tip" />
</book>
</xsl:for-each>
</results>
</xsl:template>
```

```
</xsl:transform>
```

# 2.44. Query 44

## 2.44.1. Source

Xbench family of benchmarks [http://db.uwaterloo.ca/~ddbms/projects/xbench/]

Description: *Retrieve the body of the article that has a matching id attribute value (4).*

Solution in XQuery:

```
for $a in doc("collection.xml")/article[@id="4"]
return
    <Article>
        {$a/body}
    </Article>
```

## 2.44.2. Query 44t: Text XML

Description: *Retrieve the intro of the chapter that has a matching number attribute value (3).*

The result of this query should be:

```
<book>
  <intro>
    <para>The Graphic Communications Association has been
    instrumental in the development of SGML. GCA provides
    conferences, tutorials, newsletters, and publication sales for
    both members and non-members.</para>
    <para security="c">Exiled members of the former Soviet Union's
    secret police, the KGB, have infiltrated the upper ranks of the
    GCA and are planning the Final Revolution as soon as DSSSL is
    completed.</para>
  </intro>
</book>
```

### 2.44.2.1. Complexity

Complexity = 4

- Report items: 1 [intro elements]
- Condition items: 3 [parent = chapter (1), number attribute (1) with a certain value (1)]

### 2.44.2.2. XQuery solution

```
for $a in //chapter[@number="3"]
return
    <book>
        {$a//intro}
    </book>
```

```
for $a in //chapter
where $a/@number="3"
return
    <book>
        {$a//intro}
```

```
    </book>
```

### 2.44.2.3. XSLT solution

```
<xsl:transform
xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
version="2.0">
<xsl:output method="xml" indent="yes"/>
<xsl:template match="text()" />
<xsl:template match="chapter[@number='3']">
<xsl:copy-of select=".//intro"/>
</xsl:template>
</xsl:transform>
```

## 2.44.3. Query 44d: Data XML

Description: *Retrieve the title of the book that has a matching id attribute value (b6).*

The result of this query should be:

```
<book>
   <title>Drawbacks of procedural markup</title>
</book>
```

### 2.44.3.1. Complexity

Complexity = 4

- Report items: 1 [title elements]
- Condition items: 3 [parent = book (1), id attribute (1) with a certain value (1)]

### 2.44.3.2. XQuery solution

```
for $a in //book[@id="b6"]
return
    <book>
        {$a/title}
    </book>
```

```
for $b in //book
where $b/@id = "b6"
return $b/title
```

### 2.44.3.3. XSLT solution

```
<xsl:transform xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
version="2.0">
<xsl:template match="text()" />
<xsl:output method="xml" indent="yes" />

<xsl:template match="book[@id ='b6']">
    <xsl:copy-of select="title" />
</xsl:template>
</xsl:transform>
```

# 2.45. Query 45

## 2.45.1. Source

Xbench family of benchmarks [http://db.uwaterloo.ca/~ddbms/projects/xbench/] Query 13

Description: *Construct a brief information on the article that has a matching id attribute value (5), including title, the name of first author, date and abstract.*

Solution in XQuery:

```
for $a in doc("collection.xml")/article[@id="5"]
return
    <Output>
        {$a/prolog/title}
        {$a/prolog/authors/author[1]/name}
        {$a/prolog/dateline/date}
        {$a/body/abstract}
    </Output>
```

## 2.45.2. Query 45t: Text XML

Description: *Construct a brief information on the chapter that has a title "The business challenge", including title, the first author, keyword and intro.*

The result of this query should be:

```
<chapter_summary>
  <title>The business challenge</title>
  <author>
    <last>Stevens</last>
    <first>W.</first>
  </author>
  <intro>
    <para>With the ever-changing and growing global market,
    companies and large organizations are searching for ways to
    become more viable and competitive. Downsizing and other
    cost-cutting measures demand more efficient use of corporate
    resources. One very important resource is an organization's
    information.</para>
    <para>As part of the move toward integrated information
    management, whole industries are developing and implementing
    standards for exchanging technical information. This report
    describes how one such standard, the Standard Generalized
    Markup Language (SGML), works as part of an overall information
    management strategy.</para>
    <graphic graphname="infoflow" />
  </intro>
</chapter_summary>
```

### 2.45.2.1. Complexity

Complexity = 9

- Report items: 5 [chapter_summary element (1), title (1), author (1), keyword (1), intro (1).]
- Condition items: 4 [common parent = chapter (1), with a title element (1) with a certain value (1). Report item author has position 1 (1)]

### 2.45.2.2. XQuery solution

```
for $a in //chapter[title="The business challenge"]
return
    <chapter_summary>
        {$a/title}
        {$a/author[1]}
        {$a/keyword}
        {$a/intro}
    </chapter_summary>
```

```
for $t in //chapter
where $t/title = "The business challenge"
return <chapter_summary>{$t/title,$t/author[1],$t/keyword,$t/intro}</chapter_summary>
```

### 2.45.2.3. XSLT solution

```
<xsl:transform xmlns:xsl="http://www.w3.org/1999/XSL/Transform" version="2.0">
<xsl:output method="xml" indent="yes" />
<xsl:template match="text()" />
<xsl:template match="chapter[title='The business challenge']">
<chapter_summary>
   <xsl:copy-of select="title"/>
   <xsl:copy-of select="author[position()=1]"/>
   <xsl:copy-of select="keyword"/>
   <xsl:copy-of select="intro"/>
</chapter_summary>
</xsl:template>
</xsl:transform>
```

```
<xsl:transform xmlns:xsl="http://www.w3.org/1999/XSL/Transform" version="2.0">
<xsl:output method="xml" indent="yes" />

<xsl:template match="/">
<xsl:for-each select="//chapter[title='The business challenge']">
<chapter_summary>
   <xsl:copy-of select="title"/>
   <xsl:copy-of select="author[position()=1]"/>
   <xsl:copy-of select="keyword"/>
   <xsl:copy-of select="intro"/>
</chapter_summary>
</xsl:for-each>
</xsl:template>
</xsl:transform>
```

## 2.45.3. Query 45d: Data XML

Description: *Construct a brief information on the book that has a title "Lords of the Ring", including year, availability, title, and the first store.*

The result of this query should be:

```
<book_summary year="1950" available="no">
  <title>Lords of the Ring</title>
  <store>
    <source>bstore1.example.com</source>
    <price>55.95</price>
  </store>
</book_summary>
```

### 2.45.3.1. Complexity

Complexity = 9

- Report items: 5 [book_summary element (1), attribute year (1), attribute available (1), title (1), store (1).]
- Condition items: 4 [common parent = book (1), with a title element (1) with a certain value (1). Report item store has position 1 (1)]

### 2.45.3.2. XQuery solution

```
for $a in //book[title="Lords of the Ring"]
return
    <book_summary>
 {$a/@year}
 {$a/@available}
 {$a/title}
    {$a/store[1]}
    </book_summary>
```

```
for $a in //book
where $a/title="Lords of the Ring"
return
    <book_summary>
 {$a/@year}
 {$a/@available}
 {$a/title}
    {$a/store[1]}
    </book_summary>
```

### 2.45.3.3. XSLT solution

```
<xsl:transform xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
version="2.0">
<xsl:template match="text()" />
<xsl:output method="xml" indent="yes" />
<xsl:template match="/">
<xsl:for-each select="//book[title ='Lords of the Ring']">
<book_summary>
   <xsl:copy-of select="@year" />
   <xsl:copy-of select="@available" />
   <xsl:copy-of select="title" />
   <xsl:copy-of select="store[1]" />
</book_summary>
</xsl:for-each>
</xsl:template>
</xsl:transform>
```

```
<xsl:transform xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
version="2.0">
<xsl:template match="text()" />
<xsl:output method="xml" indent="yes" />

<xsl:template match="book[title ='Lords of the Ring']">
<book_summary>
   <xsl:copy-of select="@year" />
   <xsl:copy-of select="@available" />
   <xsl:copy-of select="title" />
   <xsl:copy-of select="store[1]" />
</book_summary>
</xsl:template>
</xsl:transform>
```

# 2.46. Query 46

## 2.46.1. Source

Xbench family of benchmarks [http://db.uwaterloo.ca/~ddbms/projects/xbench/] Query 16

Description: *Get the article by its id attribute value (6).*

Solution in XQuery:

```
for $a in doc("collection.xml")/article[@id="6"]
return $a
```

## 2.46.2. Query 46t: Text XML

Description: *Get the topic by its id attribute value (top6).*

The result of this query should be:

```
<topic id="top6" number="6">
  <title>Style</title>
  <para>SGML does not standardize style or other processing methods
  for information stored in SGML.</para>
</topic>
```

### 2.46.2.1. Complexity

Complexity = 3

- Report items: 1 [topic element]
- Condition items: 2 [id attribute (1) with a certain value (1)]

### 2.46.2.2. XQuery solution

```
for $a in //topic[@id="top6"]
return $a
```

```
for $a in //topic
where $a/@id = "top6"
return $a
```

### 2.46.2.3. XSLT solution

```
<xsl:transform xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
version="2.0">
<xsl:output method="xml" indent="yes" />
<xsl:template match="text()" />
<xsl:template match="topic[@id='top6']">
    <xsl:copy-of select="." />
</xsl:template>
</xsl:transform>
```

```
<xsl:transform xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
version="2.0">
<xsl:output method="xml" indent="yes" />
<xsl:template match="text()" />
<xsl:template match="topic">
<xsl:if test="@id='top6'">
    <xsl:copy-of select="." />
</xsl:if>
</xsl:template>
</xsl:transform>
```

## 2.46.3. Query 46d: Data XML

Description: *Get the book by its id attribute value (b1).*

The result of this query should be:

```
<book year="1992" id="b1">
  <title>Advanced Programming in the Unix environment</title>
  <store>
    <source>bstore2.example.com</source>
    <price>65.95</price>
  </store>
  <store>
    <source>bstore1.example.com</source>
    <price>65.95</price>
  </store>
  <store>
    <source>www.bol.com</source>
    <price>45.95</price>
  </store>
  <tip>
    <xref idref="b2" />
  </tip>
</book>
```

### 2.46.3.1. Complexity

Complexity = 3

- Report items: 1 [book element]
- Condition items: 2 [id attribute (1) with a certain value (1)]

### 2.46.3.2. XQuery solution

```
for $a in //book[@id="b1"]
return $a
```

```
for $a in //book
where $a/@id="b1"
return $a
```

### 2.46.3.3. XSLT solution

```
 <xsl:transform
xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
version="2.0">
<xsl:output method="xml" indent="yes"/>
```

```
<xsl:template match="text()" />
<xsl:template match="book[@id='b1']">
<xsl:copy-of select="." />
</xsl:template>
</xsl:transform>
```

```
<xsl:transform
xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
version="2.0">
<xsl:output method="xml" indent="yes"/>
<xsl:template match="text()" />
<xsl:template match="book">
<xsl:if test="@id='b1'">
<xsl:copy-of select="." />
</xsl:if>
</xsl:template>
</xsl:transform>
```

# 2.47. Query 47

## 2.47.1. Source

Xbench family of benchmarks [http://db.uwaterloo.ca/~ddbms/projects/xbench/] Query 17.

Description: *Return the titles of articles which contain a certain word ("hockey").*

Solution in XQuery:

```
for $a in doc("collection.xml")/article
where contains ($a//p, "hockey")
return
    $a/prolog/title
```

## 2.47.2. Query 47t: Text XML

Description: *Return the titles of topics which contain the word "documents".*

The result of this query should be:

```
<title>Structure</title>
```

### 2.47.2.1. Complexity

Complexity = 4

- Report items: 1 [title element]
- Condition items: 3 [parent = topic (1), contain (1) a certain word (1)]

### 2.47.2.2. XQuery solution

```
<result>
{
for $a in //topic
where contains ($a, "documents")
return
    $a/title
}
```

```
</result>
```

```
for $a in //topic[contains (., "documents")]
return $a/title
```

### 2.47.2.3. XSLT solution

```
<xsl:transform xmlns:xsl=
"http://www.w3.org/1999/XSL/Transform"
version="2.0">
<xsl:output method="xml" indent="yes" />
<xsl:template match="/">
    <xsl:for-each select="//topic[contains(.,'documents')]">
     <xsl:copy-of select="title" />
</xsl:for-each>
</xsl:template>
</xsl:transform>
```

```
<xsl:transform xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
version="2.0">
<xsl:template match="text()" />
<xsl:output method="xml" indent="yes" />
<xsl:template match="topic[contains(.,'documents')]">
    <xsl:copy-of select="title" />
</xsl:template>
</xsl:transform>
```

## 2.47.3. Query 47d: Data XML

Description: *Return the titles of books which contain the word "www.amazon.com"*

The result of this query should be:

```
<result>
   <title>The Economics of Technology and Content for Digital TV</title>
   <title>Lords of the Ring</title>
   <title>XQuery from the experts</title>
   <title>XSLT: programmers reference</title>
</result>
```

### 2.47.3.1. Complexity

Complexity = 4

- Report items: 1 [title element]
- Condition items: 3 [parent = topic (1), contain (1) a certain word (1)]

### 2.47.3.2. XQuery solution

```
<result>
{
for $a in //book
where contains ($a, "www.amazon.com")
return $a/title
}
</result>
```

```
for $a in //book[contains(., "www.amazon.com")]
return $a/title
```

#### 2.47.3.3. XSLT solution

```
<xsl:transform xmlns:xsl="http://www.w3.org/1999/XSL/Transform" version="2.0">
<xsl:output method="xml" indent="yes" />

<xsl:template match="/">
<xsl:for-each select="//book[store[contains(., 'www.amazon.com')]]">
  <xsl:copy-of select="title" />
</xsl:for-each>
</xsl:template>
</xsl:transform>
```

```
<xsl:transform xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
version="2.0">
<xsl:template match="text()" />
<xsl:output method="xml" indent="yes" />

<xsl:template match="book[store[contains(., 'www.amazon.com')]]">
    <xsl:copy-of select="title" />
</xsl:template>
</xsl:transform>
```

# 2.48. Query 48

## 2.48.1. Source

Xbench family of benchmarks [http://db.uwaterloo.ca/~ddbms/projects/xbench/] Query 18.

Description: *List the titles and abstracts of articles which contain a given phrase ("the hockey").*

Solution in XQuery:

```
for $a in doc("collection.xml")/article
where contains ($a//p, "the hockey")
return
    <Output>
        {$a/prolog/title}
        {$a/body/abstract}
    </Output>
```

## 2.48.2. Query 48t: Text XML

Description: *Return the title and keywords of topics which contain the phrase "A database schema".*

The result of this query should be:

```
<result>
   <title>Structure</title>
   <keyword>DTD</keyword>
</result>
```

### 2.48.2.1. Complexity

Complexity = 5

- Report items: 2 [title and keyword elements]
- Condition items: 3 [parent = topic (1), contain (1) a certain phrase (1)]

### 2.48.2.2. XQuery solution

```
for $a in //topic
where contains ($a, "A database schema")
return
<result>
    {$a/title}
    {$a/keyword}
</result>
```

```
for $a in //topic[contains(., "A database schema")]
return
<result>
    {$a/title}
    {$a/keyword}
</result>
```

### 2.48.2.3. XSLT solution

```
<xsl:transform
xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
version="2.0">
<xsl:output method="xml" indent="yes"/>
<xsl:template match="text()" />

<xsl:template match="topic[contains(.,'A database schema')]">
  <xsl:copy-of select="title" />
  <xsl:copy-of select="keyword" />
</xsl:template>
</xsl:transform>
```

```
<xsl:transform
xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
version="2.0">
<xsl:output method="xml" indent="yes"/>
<xsl:template match="/">
<results>
<xsl:for-each select="//topic[contains(.,'A database schema')]">
  <xsl:copy-of select="title" />
  <xsl:copy-of select="keyword" />
</xsl:for-each>
</results>
</xsl:template>
</xsl:transform>
```

## 2.48.3. Query 48d: Data XML

Description: *Return the years and titles of books which contain the phrase "in the".*

The result of this query should be:

```
<result year="1992">
   <title>Advanced Programming in the Unix environment</title>
</result>
```

### 2.48.3.1. Complexity

Complexity = 5

- Report items: 2 [year attributes and titles elements]
- Condition items: 3 [parent = books (1), contain (1) a certain phrase (1)]

### 2.48.3.2. XQuery solution

```
for $a in //book
where contains ($a, "in the")
return
    <result>
    {$a/@year}
    {$a/title}
    </result>
```

```
for $a in //book[contains (., "in the")]
return
    <result>
    {$a/@year}
    {$a/title}
    </result>
```

### 2.48.3.3. XSLT solution

```
<xsl:transform
xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
version="2.0">
<xsl:output method="xml" indent="yes"/>
<xsl:template match="/">
<results>
<xsl:for-each select="//book[title[contains(., 'in the')]]">
  <xsl:copy-of select="@year" />
  <xsl:copy-of select="title" />
</xsl:for-each>
</results>
</xsl:template>
</xsl:transform>
```

```
<xsl:transform xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
version="2.0">
<xsl:template match="text()" />
<xsl:output method="xml" indent="yes" />

<xsl:template match="book[title[contains(., 'in the')]]">
<book>
    <xsl:copy-of select="@year" />
    <xsl:copy-of select="title" />
</book>
</xsl:template>
</xsl:transform>
```

# 2.49. Query 49

## 2.49.1. Source

Xbench family of benchmarks [http://db.uwaterloo.ca/~ddbms/projects/xbench/] Query 19

Description: *List the names of articles cited by an article with a certain id attribute value (7).*

Solution in XQuery:

```
for $a in doc("collection.xml")/article[@id='7']/epilog/references/a_id,
    $b in doc("collection.xml")/article
where $a = $b/@id
return
    <Output>
        {$b/prolog/title}
    </Output>
```

## 2.49.2. Query 49t: Text XML

Description: *List the title and id attribute of topics referred to by a topic with an id attribute value "top5".*

The result of this query should be:

```
<topic id="top4">
   <title>Structure</title>
</topic>
```

### 2.49.2.1. Complexity

Complexity = 11.

- Subquery 1 (get the 'title' and 'id' of the topic where the id is equal to the report item of subquery2):
  - Report items: 2 (title element and id attribute)
  - Condition items: 4 (parent = topic (1) with an id attribute (1) with a reference value equal to [report subquery 2]) (2).

  Complexity Subquery 1 = 6
- Subquery 2 (Get the id value of the xref elements of a topic with id value "top5"):
  - Report items: 1 (value of idref attribute)
  - Condition items: 4 (parent = xref (1), ancestor = topic (1) with id attribute (1), with a certain value (1).)

  Complexity subquery 2 = 5

### 2.49.2.2. XQuery solution

```
<result>
{
for $a in //topic[@id='top5']//xref/@idref,
    $b in //topic
where $a = $b/@id
return
<topic>
 {$b/@id}
        {$b/title}
</topic>
}
</result>
```

```
for $a in //topic[@id='top5']//xref/@idref,
    $b in //topic[@id=$a]
return
    <topic>
 {$b/@id}
        {$b/title}
    </topic>
```

### 2.49.2.3. XSLT solution

```
<xsl:transform xmlns:xsl=
"http://www.w3.org/1999/XSL/Transform"
version="2.0">
<xsl:output method="xml" indent="yes" />
<xsl:template match="text()" />

<xsl:variable name="ref" select="//topic[@id='top5']//xref/@idref" />

<xsl:template match="topic[@id= $ref]">
<topic id="{$ref}">
<xsl:copy-of select="title" />
</topic>
</xsl:template>
</xsl:transform>
```

## 2.49.3. Query 49d: Data XML

Description: *List the title and id attribute of books tipped by a book with an id attribute value "b9".*

The result of this query should be:

```
<results>
   <book id="b8">
      <title>XQuery from the experts</title>
   </book>
   <book id="b5">
      <title>Getting started with SGML</title>
   </book>
</results>
```

### 2.49.3.1. Complexity

Complexity = 11.

- Subquery 1 (get the 'title' and 'id' of the book where the id is equal to the report item of subquery2):
  - Report items: 2 (title element and id attribute)
  - Condition items: 4 (parent = book (1) with an id attribute (1) with a reference value equal to [report subquery 2]) (2).

  Complexity Subquery 1 = 6
- Subquery 2 (Get the id value of the xref elements of a book with id value "top5"):
  - Report items: 1 (value of idref attribute)
  - Condition items: 4 (parent = xref (1), ancestor = book (1) with id attribute (1), with a certain value (1).)

  Complexity subquery 2 = 5

### 2.49.3.2. XQuery solution

```
<result>
{
```

```
for $a in //book[@id='b9']//xref/@idref,
    $b in //book
where $a = $b/@id
return
<book>
 {$b/@id}
        {$b/title}
</book>
}
</result>
```

```
for $a in //book[@id='b9']//tip/xref/@idref,
    $b in //book[@id=$a]
return
    <book>
 {$b/@id}
        {$b/title}
    </book>
```

### 2.49.3.3. XSLT solution

```
<xsl:transform xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
version="2.0">
<xsl:template match="text()" />
<xsl:output method="xml" indent="yes" />

<xsl:variable name="tipid" select="//book[@id='b9']//xref/@idref" />

<xsl:template match="book[@id=$tipid]">
<book>
<xsl:copy-of select="@id" />
<xsl:copy-of select="title" />
</book>
</xsl:template>
</xsl:transform>
```

# 2.50. Query 50

## 2.50.1. Source

The following queries are used in a previous experiment, we will therefore skip the original descriptions and sources.

## 2.50.2. Query 50t: Text XML

Description: *Find the author with last name "Buneman"*

The result of this query should be:

```
<author>
  <last>Buneman</last>
  <first>Peter</first>
</author>
```

### 2.50.2.1. Complexity

Complexity = 3

- Report items: 1 [author elements]

- Condition items: 2 [child element last (1) with a certain value (1)]

### 2.50.2.2. XQuery solution

```
for $author in //author[last="Buneman"]
return $author
```

```
for $author in //author
where $author[last="Buneman"]
return $author
```

### 2.50.2.3. XSLT solution

```
<xsl:transform xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
version="2.0">
<xsl:output method="xml" indent="yes" />
<xsl:template match="text()" />
<xsl:template match="author[last='Buneman']">
    <xsl:copy-of select="." />
</xsl:template>
</xsl:transform>
```

```
<xsl:transform xmlns:xsl="http://www.w3.org/1999/XSL/Transform" version="2.0">
<xsl:template match="/">
<results>
<xsl:for-each select="//author[last='Buneman']">
<xsl:copy-of select="."/>
</xsl:for-each>
</results>
</xsl:template>
</xsl:transform>
```

## 2.50.3. Query 50d: Data XML

Description: *Find the store with source "bstore3.example.com".*

The result of this query should be:

```
<store>
  <source>bstore3.example.com</source>
  <price>25.95</price>
</store>
```

### 2.50.3.1. Complexity

Complexity = 3

- Report items: 1 [store elements]
- Condition items: 2 [child element source (1) with a certain value (1)]

### 2.50.3.2. XQuery solution

```
for $store in //store[source="bstore3.example.com"]
```

```
return $store



for $store in //store
where $store/source="bstore3.example.com"
return $store
```

### 2.50.3.3. XSLT solution

```
<xsl:transform xmlns:xsl="http://www.w3.org/1999/XSL/Transform" version="2.0">
<xsl:template match="/">
<results>
<xsl:for-each select="//store[source='bstore3.example.com']">
<xsl:copy-of select="."/>
</xsl:for-each>
</results>
</xsl:template>
</xsl:transform>



<xsl:transform xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
version="2.0">
<xsl:output method="xml" indent="yes" />
<xsl:template match="text()" />
<xsl:template match="store[source='bstore3.example.com']">
   <xsl:copy-of select="." />
</xsl:template>
</xsl:transform>
```

# 2.51. Query 51

## 2.51.1. Query 51t: Text XML

Description: *Create a result element that contains the number, title, and keyword of topics that have:*

1. *a number that is not 5.*
2. *a number that is not 4.*
3. *a keyword element with value "Markup"*

The result of this query should be:

```
<results>
   <result number="1">
      <title>Procedural markup</title>
      <keyword>Markup</keyword>
   </result>
   <result number="2">
      <title>Generic markup</title>
      <keyword>Markup</keyword>
   </result>
   <result number="3">
      <title>Drawbacks of procedural markup</title>
      <keyword>Markup</keyword>
   </result>
</results>
```

### 2.51.1.1. Complexity

Complexity = 10

- Report items: 4 (result element (1) with title (1), number attribute (1) and keyword (1))
- Condition items: (parent = topic (1), with attribute number (1) with two possible values (2), keyword element (1) with a certain value (1))

### 2.51.1.2. XQuery solution

```
<results>
{
for $b in //topic
where $b/@number != 5 and $b/@number != 4 and $b/keyword="Markup"
return <result> {$b/@number, $b/title, $b/keyword } </result>
}
</results>
```

### 2.51.1.3. XSLT solution

```
<xsl:transform xmlns:xsl=
"http://www.w3.org/1999/XSL/Transform"
version="2.0">
<xsl:output method="xml" indent="yes" />
<xsl:template match="/">
<results>
<xsl:for-each select="//topic[@number != 5][@number != 4][keyword= 'Markup']">
<result number="{@number}">
 <xsl:copy-of select="title" />
 <xsl:copy-of select="keyword" />
</result>
</xsl:for-each>
</results>
</xsl:template>
</xsl:transform>
```

```
<xsl:transform xmlns:xsl= "http://www.w3.org/1999/XSL/Transform" version="2.0">
<xsl:output method="xml" indent="yes" />
<xsl:template match="text()" />

<xsl:template match="/">
<results>
<xsl:apply-templates />
</results>
</xsl:template>

<xsl:template match="topic[@number != 5][@number != 4][keyword= 'Markup']">
<result>
<xsl:copy-of select="@number" />
<xsl:copy-of select="title" />
<xsl:copy-of select="keyword" />
</result>
</xsl:template>
</xsl:transform>
```

## 2.51.2. Query 51d: Data XML

Description: *Create a result element that contains the year, id, and title of books that have:*

1. *a year that is not 1980.*
2. *a year that is not 1950.*
3. *a source element with value "www.amazon.com"*

The result of this query should be:

```
<results>
   <result year="1999" id="b4">
      <title>The Economics of Technology and Content for Digital TV</title>
   </result>
   <result year="2003" id="b8">
      <title>XQuery from the experts</title>
   </result>
   <result year="1999" id="b9">
      <title>XSLT: programmers reference</title>
   </result>
</results>
```

### 2.51.2.1. Complexity

Complexity = 10

- Report items: 4 (result element (1) with title (1), year attribute (1) and id (1))
- Condition items: (parent = book (1), with year number (1) with two possible values (2), source element (1) with a certain value (1))

### 2.51.2.2. XQuery solution

```
<results>
{
for $b in //book
where $b/@year != 1980 and $b/@year != 1950 and $b//source="www.amazon.com"
return <result> {$b/@year, $b/@id, $b/title} </result>
}
</results>
```

### 2.51.2.3. XSLT solution

```
<xsl:transform
xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
version="2.0">
<xsl:output method="xml" indent="yes"/>
<xsl:template match="text()" />
<xsl:template match="/">
<results>
<xsl:for-each select="//book[@year != 1980][@year != 1950][.//source = 'www.amazon.com']">
<result>
<xsl:copy-of select="@year" />
<xsl:copy-of select="@id" />
<xsl:copy-of select="title" />
</result>
</xsl:for-each>
</results>
</xsl:template>
</xsl:transform>
```

```
<xsl:transform xmlns:xsl= "http://www.w3.org/1999/XSL/Transform" version="2.0">
<xsl:output method="xml" indent="yes" />
<xsl:template match="text()" />

<xsl:template match="/">
<results>
<xsl:apply-templates />
</results>
</xsl:template>

<xsl:template match="book[@year != 1980][@year != 1950][.//source = 'www.amazon.com']">
<result>
<xsl:copy-of select="@year" />
<xsl:copy-of select="@id" />
```

```
<xsl:copy-of select="title" />
</result>
</xsl:template>
</xsl:transform>
```

# 2.52. Query 52

## 2.52.1. Query 52t: Text XML

Description: *Make a unique list of keywords*

The result of this query should be:

```
<results>
   <keyword>Markup</keyword>
   <keyword>DTD</keyword>
</results>
```

### 2.52.1.1. Complexity

Complexity = 2

- Report items: 1 (keywords)
- Condition items: 1 (unique values)

### 2.52.1.2. XQuery solution

```
<results>
{
for $n in distinct-values(//keyword)
return <keyword>{$n}</keyword>
}
</results>
```

### 2.52.1.3. XSLT solution

```
<xsl:transform xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
version="2.0">
<xsl:output method="xml" indent="yes" />
<xsl:template match="text()" />
<xsl:template match="/">
<results>
<xsl:for-each select="distinct-values(//keyword)">
    <keyword><xsl:copy-of select="." /></keyword>
</xsl:for-each>
</results>
</xsl:template>
</xsl:transform>
```

## 2.52.2. Query 52d: Data XML

Description: *Make a unique list of sources*

The result of this query should be:

```
<results>
   <sources>bstore2.example.com</sources>
```

```
   <sources>bstore1.example.com</sources>
   <sources>www.bol.com</sources>
   <sources>www.amazon.com</sources>
   <sources>www.mystore.com</sources>
   <sources>bstore3.example.com</sources>
</results>
```

#### 2.52.2.1. Complexity

Complexity = 2

- Report items: 1 (source)
- Condition items: 1 (unique values)

#### 2.52.2.2. XQuery solution

```
<results>
{
for $n in distinct-values(//source)
return <sources>{$n}</sources>
}
</results>
```

#### 2.52.2.3. XSLT solution

```
<xsl:transform
xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
version="2.0">
<xsl:output method="xml" indent="yes"/>
<xsl:template match="/">
<results>
<xsl:for-each select="distinct-values(//source)">
 <source><xsl:copy-of select="." /></source>
</xsl:for-each>
</results>
</xsl:template>
</xsl:transform>
```

# 2.53. Query 53

## 2.53.1. Query 53t: Text XML

Description: *Find the titles of topics(s) that have a keyword that is equal to the keyword from the topic with id "top3".*

The result of this query should be:

```
<result>
   <title>Procedural markup</title>
   <title>Generic markup</title>
   <title>Drawbacks of procedural markup</title>
   <title>Content</title>
</result>
```

#### 2.53.1.1. Complexity

Complexity = 9

- Subquery 1 (Get the title of topics where a keyword is equal to the report of subquery 2):
  - Report items: 1, [title element].
  - Condition items: 4, [parent topic (1), with keyword element (1) and a reference value equal to [report subquery 2] (2)].

  Complexity Subquery 1 = 5
- Subquery 2 (get the keyword of the topic with id attribute value "top3"):
  - Report items: 1 [keyword]
  - Condition items: 3 [parent = topic (1), with an id attribute (1), with a certain value (1)]

  Complexity subquery 2 = 4

### 2.53.1.2. XQuery solution

```
<result>
{
for $n in //topic
where $n/keyword = //topic[@id="top3"]/keyword
return $n/title
}
</result>
```

```
for $k in //topic[@id="top3"]/keyword
return
<result>
{
for $t in //topic
where $t/keyword = $k
return $t/title
}
</result>
```

### 2.53.1.3. XSLT solution

```
<xsl:transform xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
version="2.0">
<xsl:template match="text()" />
<xsl:output method="xml" indent="yes" />

<xsl:variable name="key" select="//topic[@id='top3']/keyword"/>

<xsl:template match="topic[keyword=$key]">
    <xsl:copy-of select="title" />
</xsl:template>
</xsl:transform>
```

```
<xsl:transform xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
version="2.0">
<xsl:template match="text()" />
<xsl:output method="xml" indent="yes" />

<xsl:variable name="key" select="//topic[@id='top3']/keyword"/>

<xsl:template match="/">
<result>
<xsl:for-each select="//topic[keyword=$key]">
    <xsl:copy-of select="title" />
</xsl:for-each>
</result>
</xsl:template>
</xsl:transform>
```

## 2.53.2. Query 53d: Data XML

Description: *Find the titles of book(s) that have a year that is equal to the year from the book with title "XSLT: programmers reference".*

The result of this query should be:

```
<result>
   <title>The Economics of Technology and Content for Digital TV</title>
   <title>XSLT: programmers reference</title>
</result>
```

### 2.53.2.1. Complexity

Complexity = 9

- Subquery 1 (Get the title of books where a year attribute is equal to the report of subquery 2):
  - Report items: 1, [title element].
  - Condition items: 4, [parent book (1), with attribute year (1) and a reference value equal to [report subquery 2] (2)].

  Complexity Subquery 1 = 5
- Subquery 2 (get the year from the book with title value "XSLT: Programmer's reference"):
  - Report items: 1 [year attribute]
  - Condition items: 3 [parent = book (1), with a title (1), with a certain value (1)]

  Complexity subquery 2 = 4

### 2.53.2.2. XQuery solution

```
<result>
{
for $n in //book
where $n/@year = //book[title="XSLT: programmers reference"]/@year
return $n/title
}
</result>
```

```
for $y in //book[title="XSLT: programmers reference"]/@year
return
<result>
{
for $b in //book
where $b/@year = $y
return $b/title
}
</result>
```

### 2.53.2.3. XSLT solution

```
<xsl:transform xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
version="2.0">
<xsl:template match="text()" />
<xsl:output method="xml" indent="yes" />

<xsl:variable name="year" select="//book[title='XSLT: programmers reference']/@year"/>

<xsl:template match="/">
<result>
```

```
<xsl:for-each select="//book[@year=$year]">
    <xsl:copy-of select="title" />
</xsl:for-each>
</result>
</xsl:template>
</xsl:transform>
```

```
<xsl:transform xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
version="2.0">
<xsl:template match="text()" />
<xsl:output method="xml" indent="yes" />

<xsl:variable name="year" select="//book[title='XSLT: programmers reference']/@year"/>

<xsl:template match="book[@year=$year]">
    <xsl:copy-of select="title" />
</xsl:template>
</xsl:transform>
```

# 2.54. Query 54

## 2.54.1. Query 54t: Text XML

Description: *Group topic elements by keyword and give the keyword elements with all the titles of topics that have this keyword.*

The result of this query should be:

```
<result>
   <group>
      <keyword>Markup</keyword>
      <title>Procedural markup</title>
      <title>Generic markup</title>
      <title>Drawbacks of procedural markup</title>
      <title>Content</title>
   </group>
   <group>
      <keyword>DTD</keyword>
      <title>Structure</title>
   </group>
</result>
```

### 2.54.1.1. Complexity

Complexity = 13.

- Subquery 1:
  - Report: 2 [report items subquery 2 (1) and subquery 3. (1)]
  - Condition: 4 [two reference items: [report item subquery 2] = [the condition value subquery 2]]

  Complexity subquery 1 = 6
- Subquery 2:
  - Report: 1 [keyword elements]
  - Condition: 1 [unique values]

  Complexity subquery 2 = 2
- Subquery 3:
  - Report: 1 [title elements]
  - Condition: 4 [parent topic (1) has keyword (1) that = [the report of subquery 2] (2)]

  Complexity subquery 3 = 5.

### 2.54.1.2. XQuery solution

```
<result>
{
for $a in distinct-values(//keyword)
let $b := //topic[keyword=$a]
return
<group>
<keyword>{$a}</keyword>
{$b/title}
</group>
}
</result>
```

```
<result>
{
for $key in distinct-values(//keyword)
return
<group>
<keyword>
{$key}
</keyword>
{for $title in //topic[keyword=$key]/title
return $title}
</group>
}
</result>
```

### 2.54.1.3. XSLT solution

```
<xsl:transform
xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
version="2.0">
<xsl:output method="xml" indent="yes"/>

<xsl:template match="/">
<results>
<xsl:for-each-group select="//topic" group-by="keyword">
<result>
<keyword><xsl:copy-of select="current-grouping-key()"/></keyword>
<xsl:copy-of select="current-group()/title"/>
</result>
</xsl:for-each-group>
</results>
</xsl:template>
</xsl:transform>
```

## 2.54.2. Query 54d: Data XML

Description: *Group book elements by year and give the year with all the titles of books from that year.*

The result of this query should be:

```
<result>
   <group>
      <year>1992</year>
      <title>Advanced Programming in the Unix environment</title>
   </group>
   <group>
      <year>1994</year>
      <title>TCP/IP Illustrated</title>
   </group>
   <group>
```

```
      <year>2000</year>
      <title>Data on the Web</title>
   </group>
   <group>
      <year>1999</year>
      <title>The Economics of Technology and Content for Digital TV</title>
      <title>XSLT: programmers reference</title>
   </group>
   <group>
      <year>1982</year>
      <title>Getting started with SGML</title>
   </group>
   <group>
      <year>1980</year>
      <title>Drawbacks of procedural markup</title>
   </group>
   <group>
      <year>1950</year>
      <title>Lords of the Ring</title>
   </group>
   <group>
      <year>2003</year>
      <title>XQuery from the experts</title>
   </group>
</result>
```

### 2.54.2.1. Complexity

Complexity = 13.

- Subquery 1:
  - Report: 2 [report items subquery 2 (1) and subquery 3. (1)]
  - Condition: 4 [two reference items: [report item subquery 2] = [the condition value subquery 2]]

  Complexity subquery 1 = 6
- Subquery 2:
  - Report: 1 [year attributes]
  - Condition: 1 [unique values]

  Complexity subquery 2 = 2
- Subquery 3:
  - Report: 1 [title elements]
  - Condition: 4 [parent book (1) has year attribute (1) that = [the report of subquery 2] (2)]

  Complexity subquery 3 = 5.

### 2.54.2.2. XQuery solution

```
<result>
{
for $a in distinct-values(//@year)
let $b := //book[@year=$a]
return
<group>
<year>{$a}</year>
{$b/title}
</group>
}
</result>
```

```
<result>
{
for $year in distinct-values(//@year)
return
<group>
<year>
{$year}
</year>
```

```
{for $title in //book[@year=$year]/title
return $title}
</group>
}
</result>
```

#### 2.54.2.3. XSLT solution

```
<xsl:transform
xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
version="2.0">
<xsl:output method="xml" indent="yes"/>
<xsl:template match="/">
<listings>
<xsl:for-each-group select="//book" group-by="@year">
<book>
<year><xsl:value-of select="current-grouping-key()"/></year>
<xsl:copy-of select="current-group()/title"/>
</book>
</xsl:for-each-group>
</listings>
</xsl:template>
</xsl:transform>
```

# 2.55. Query 55

## 2.55.1. Query 55t: Text XML

Description: *Find all emph elements*

The result of this query should be:

```
<result>
   <emph>markup</emph>
   <emph>marking</emph>
   <emph>procedural markup</emph>
   <emph>purpose</emph>
   <emph>procedural</emph>
   <emph>is</emph>
   <emph>rules</emph>
   <emph>tagging</emph>
</result>
```

#### 2.55.1.1. Complexity

Complexity = 1

- Report items: 1 [emph]
- Condition items: 0

#### 2.55.1.2. XQuery solution

```
<result>
{
//emph
}
</result>
```

```
for $e in //emph
return $e
```

### 2.55.1.3. XSLT solution

```
<xsl:transform xmlns:xsl=
"http://www.w3.org/1999/XSL/Transform"
version="2.0">
<xsl:output method="xml" indent="yes" />
<xsl:template match="/">
<result>
<xsl:for-each select="//emph">
     <xsl:copy-of select="." />
</xsl:for-each>
</result>
</xsl:template>
</xsl:transform>
```

```
<xsl:transform xmlns:xsl="http://www.w3.org/1999/XSL/Transform" version="2.0">
<xsl:template match="text()"/>
<xsl:output method="xml" indent="yes"/>

<xsl:template match="/">
<result>
 <xsl:copy-of select="//emph"/>
</result>
</xsl:template>
</xsl:transform>
```

## 2.55.2. Query 55d: Data XML

Description: *Find all xref elements*

The result of this query should be:

```
<result>
   <xref idref="b2"/>
   <xref idref="b1"/>
   <xref idref="b5"/>
   <xref idref="b9"/>
   <xref idref="b8"/>
   <xref idref="b5"/>
</result>
```

### 2.55.2.1. Complexity

Complexity = 1

- Report items: 1 [xref]
- Condition items: 0

### 2.55.2.2. XQuery solution

```
<result>
{
//xref
}
</result>
```

```
for $x in //xref
return $x
```

### 2.55.2.3. XSLT solution

```
<xsl:transform
xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
version="2.0">
<xsl:output method="xml" indent="yes"/>

<xsl:template match="/">
<result>
<xsl:for-each select="//xref">
 <xsl:copy-of select="."/>
</xsl:for-each>
</result>
</xsl:template>
</xsl:transform>
```

```
<xsl:transform xmlns:xsl="http://www.w3.org/1999/XSL/Transform" version="2.0">
<xsl:template match="text()"/>
<xsl:output method="xml" indent="yes"/>

<xsl:template match="/">
<result>
 <xsl:copy-of select="//xref"/>
</result>
</xsl:template>
</xsl:transform>
```

# 2.56. Query 56

## 2.56.1. Query 56t: Text XML

Description: *Group chapters by last name of the authors and provide the last names of the authors and the chapter titles in each group.*

The result of this query should be:

```
<result>
   <group>
      <last>Stevens</last>
      <title>The business challenge</title>
      <title>Getting to know SGML</title>
   </group>
   <group>
      <last>Abiteboul</last>
      <title>Resources</title>
   </group>
   <group>
      <last>Buneman</last>
      <title>Resources</title>
   </group>
   <group>
      <last>Suciu</last>
      <title>Resources</title>
   </group>
</result>
```

### 2.56.1.1. Complexity

Complexity = 15

- Subquery 1:
  - Report: 2 [report items subquery 2 (1) and subquery 3. (1)]
  - Condition: 4 [two reference items: [report item subquery 2] = [the condition value subquery 2]]

  Complexity subquery 1 = 6
- Subquery 2:
  - Report: 1 [last elements]
  - Condition: 2 [unique values, parent = author]

  Complexity subquery 2 = 3
- Subquery 3:
  - Report: 1 [title elements]
  - Condition: 5 [parent chapter (1), child author (1), child last (1) that = [the report of subquery 2] (2)]

  Complexity subquery 3 = 6.

### 2.56.1.2. XQuery solution

```
<result>
{
for $a in distinct-values(//last)
let $b := //chapter[author/last=$a]
return
<group>
<last>{$a}</last>
{$b/title}
</group>
}
</result>
```

```
<result>
{
for $a in distinct-values(//last)
return
<group>
<last>{$a}</last>
{for $c in //chapter
where $c/author/last=$a
return $c/title}
</group>
}
</result>
```

### 2.56.1.3. XSLT solution

```
<xsl:transform
xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
version="2.0">
<xsl:output method="xml" indent="yes"/>

<xsl:template match="/">
<results>
<xsl:for-each-group select="//chapter" group-by="author/last">
<book>
 <last>
        <xsl:copy-of select="current-grouping-key()"/>
        </last>
 <xsl:copy-of select="current-group()/title"/>
</book>
</xsl:for-each-group>
</results>
```

```
</xsl:template>
</xsl:transform>
```

## 2.56.2. Query 56d: Data XML

Description: *For the book with id is "b1", group the store elements by price list for each price the source elements.*

The result of this query should be:

```
<result>
   <group>
      <price>65.95</price>
      <source>bstore2.example.com</source>
      <source>bstore1.example.com</source>
   </group>
   <group>
      <price>45.95</price>
      <source>www.bol.com</source>
   </group>
</result>
```

### 2.56.2.1. Complexity

Complexity = 18

- Subquery 1:
  - Report: 2 [report items subquery 2 (1) and subquery 3. (1)]
  - Condition: 4 [two reference items: [report item subquery 2] = [the condition value subquery 2]]

  Complexity subquery 1 = 6
- Subquery 2:
  - Report: 1 [price elements]
  - Condition: 3 [unique values, ancestor book (1) with id (1) with a certain value b1 (1)]

  Complexity: 4
- Subquery 3:
  - Report: 1 [store elements]
  - Condition: [parent store (1) has price (1) that = [the report of subquery 2] (2), ancestor book (1) with id (1) with a certain value b1 (1)]

  Complexity subquery 3 = 8.

### 2.56.2.2. XQuery solution

```
<result>
{
for $book in //book[@id="b1"]
for $a in distinct-values($book//price)
let $b := $book/store[price=$a]
return
<group>
<price>{$a}</price>
{$b/source}
</group>
}
</result>
```

```
<result>
{
for $b in //book[@id="b1"]
for $p in distinct-values($b//price)
return
<group>
```

```
<price>{$p}</price>
{for $store in $b//store
where $store/price=$p
return $store/source}
</group>
}
</result>
```

### 2.56.2.3. XSLT solution

```
<xsl:transform
xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
version="2.0">
<xsl:output method="xml" indent="yes"/>
<xsl:template match="text()" />

<xsl:template match="book[@id='b1']">
<result>
<xsl:for-each-group select="store" group-by="price">
<group>
   <price><xsl:copy-of select="current-grouping-key()"/></price>
   <xsl:copy-of select="current-group()/source"/>
</group>
</xsl:for-each-group>
</result>
</xsl:template>
</xsl:transform>
```

# 2.57. Query 57

## 2.57.1. Query 57t: Text XML

Description: *Select the last names of author elements that have a first name "W."*

The result of this query should be:

```
<result>
   <last>Stevens</last>
   <last>Stevens</last>
</result>
```

### 2.57.1.1. Complexity

Complexity = 4

- Report items: 1 [last elements]
- Condition items: [parent = author (1), first element (1) with a certain value (1)]

### 2.57.1.2. XQuery solution

```
<result>
{
for $a in //author
where $a/first = "W."
return $a/last
}
</result>
```

### 2.57.1.3. XSLT solution

```
<xsl:transform
xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
version="2.0">
<xsl:output method="xml" indent="yes"/>
<xsl:template match="/">
<xsl:for-each select="//author[first='W.']/last">
 <xsl:copy-of select="." />
</xsl:for-each>
</xsl:template>
</xsl:transform>
```

```
<xsl:transform
xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
version="2.0">
<xsl:output method="xml" indent="yes"/>
<xsl:template match="text()" />
<xsl:template match="author[first='W.']/last">
 <xsl:copy-of select="." />
</xsl:template>
</xsl:transform>
```

## 2.57.2. Query 57d: Data XML

Description: *Select the year attributes of book elements that have an attribute available value "no".*

The result of this query should be:

```
<result>
   <year year="1982"/>
   <year year="1980"/>
   <year year="1950"/>
</result>
```

### 2.57.2.1. Complexity

Complexity = 4

- Report items: 1 [year attributes]
- Condition items: [parent = book (1), attribute available (1) with a certain value (1)]

### 2.57.2.2. XQuery solution

```
<result>
{
for $a in //book
where $a/@available = "no"
return
<year>
{$a/@year}
</year>
}
</result>
```

### 2.57.2.3. XSLT solution

```
<xsl:transform xmlns:xsl=
"http://www.w3.org/1999/XSL/Transform"
version="2.0">
<xsl:output method="xml" indent="yes" />
<xsl:template match="/">
    <xsl:for-each select="//book[@available='no']">
    <year><xsl:copy-of select="@year" /></year>
    </xsl:for-each>
</xsl:template>
</xsl:transform>
```

```
<xsl:transform
xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
version="2.0">
<xsl:output method="xml" indent="yes"/>
<xsl:template match="text()" />
<xsl:template match="book[@available='no']">
  <year><xsl:copy-of select="@year" /></year>
</xsl:template>
</xsl:transform>
```

# 2.58. Query 58

## 2.58.1. Query 58t: Text XML

Description: *Find the titles of chapters that have a number lower than 3. Sort the list alphabetically by title.*

The result of this query should be:

```
<result>
   <title>Getting to know SGML</title>
   <title>The business challenge</title>
</result>
```

### 2.58.1.1. Complexity

Complexity = 5

- Report items: 1 [title elements]
- Condition items: [parent = chapter (1), number attribute (1) lower than a certain value (1), sort by report item (1)]

### 2.58.1.2. XQuery solution

```
<result>
{
for $a in //chapter
where $a/@number < 3
order by $a/title
return $a/title
}
</result>
```

```
for $c in //chapter[@number < 3]
order by $c/title
return $c/title
```

### 2.58.1.3. XSLT solution

```
<xsl:transform xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
version="2.0">
<xsl:template match="text()" />
<xsl:output method="xml" indent="yes" />

<xsl:template match="/">
<xsl:for-each select="//chapter[@number < 3]">
<xsl:sort select="title" />
  <xsl:copy-of select="title" />
</xsl:for-each>
</xsl:template>
</xsl:transform>
```

## 2.58.2. Query 58d: Data XML

Description: *Find the titles of books that have a year lower than 1999. Sort the list alphabetically by title.*

The result of this query should be:

```
<result>
   <title>Drawbacks of procedural markup</title>
   <title>Getting started with SGML</title>
   <title>Lords of the Ring</title>
</result>
```

### 2.58.2.1. Complexity

Complexity = 5

- Report items: 1 [title elements]
- Condition items: [parent = book (1), year attribute (1) lower than a certain value (1), sort by report item (1)]

### 2.58.2.2. XQuery solution

```
<result>
{
for $a in //book
where $a/@year < 1990
order by $a/title
return $a/title
}
</result>
```

```
for $a in //book[@year < 1990]
order by $a/title
return $a/title
```

### 2.58.2.3. XSLT solution

```
<xsl:transform
xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
version="2.0">
<xsl:output method="xml" indent="yes"/>
<xsl:template match="/">
<results>
```

```
<xsl:for-each select="//book[@year &lt; 1999]">
<xsl:sort select="title" />
<xsl:copy-of select="title" />
</xsl:for-each>
</results>
</xsl:template>
</xsl:transform>
```

# 2.59. Query 59

## 2.59.1. Query 59t: Text XML

Description: *Find the title and keyword of the topic that has id top4 or top5*

The result of this query should be:

```
<results>
   <result>
      <title>Structure</title>
      <keyword>DTD</keyword>
   </result>
   <result>
      <title>Content</title>
      <keyword>Markup</keyword>
   </result>
</results>
```

### 2.59.1.1. Complexity

Complexity = 6

- Report items: 2 [title and keyword elements]
- Condition items: [parent = topic (1), id attribute (1) with two possible certain value (2)]

### 2.59.1.2. XQuery solution

```
<results>
{
for $a in //topic
where $a/@id ="top4" or $a/@id ="top5"
return <result>
{
$a/title, $a/keyword
}
</result>
}
</results>
```

### 2.59.1.3. XSLT solution

```
<xsl:transform xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
version="2.0">
<xsl:output method="xml" indent="yes" />
<xsl:template match="text()" />

<xsl:template match="topic[@id='top4'] | topic[@id='top5']">
<result>
    <xsl:copy-of select="title" />
    <xsl:copy-of select="keyword" />
</result>
</xsl:template>
```

```
</xsl:transform>
```

```
<xsl:transform xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
version="2.0">
<xsl:output method="xml" indent="yes" />
<xsl:template match="text()" />

<xsl:template match="/">
<results>
<xsl:for-each select="//topic[@id='top4'] | //topic[@id='top5']">
<result>
    <xsl:copy-of select="title" />
    <xsl:copy-of select="keyword" />
</result>
</xsl:for-each>
</results>
</xsl:template>
</xsl:transform>
```

## 2.59.2. Query 59d: Data XML

Description: *Find the title and store of the book that has year 1980 or 2003.*

The result of this query should be:

```
<results>
  <result>
    <title>Drawbacks of procedural markup</title>
    <store>
      <source>bstore2.example.com</source>
      <price>25.95</price>
    </store>
  </result>
  <result>
    <title>XQuery from the experts</title>
    <store>
      <source>bstore2.example.com</source>
      <price>55.95</price>
    </store>
    <store>
      <source>www.amazon.com</source>
      <price>50.95</price>
    </store>
  </result>
</results>
```

### 2.59.2.1. Complexity

Complexity = 6

- Report items: 2 [title and store elements]
- Condition items: [parent = book (1), year attribute (1) with two possible certain value (2)]

### 2.59.2.2. XQuery solution

```
<results>
{
for $a in //book
where $a/@year =1980 or $a/@year =2003
return
<result>
{
$a/title, $a/store
}
```

```
</result>
}
</results>
```

### 2.59.2.3. XSLT solution

```
<xsl:transform xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
version="2.0">
<xsl:output method="xml" indent="yes" />
<xsl:template match="text()" />

<xsl:template match="book[@year=1980]|book[@year=2003]">
<book>
    <xsl:copy-of select="title" />
    <xsl:copy-of select="store" />
</book>
</xsl:template>
</xsl:transform>
```

```
<xsl:transform xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
version="2.0">
<xsl:output method="xml" indent="yes" />
<xsl:template match="text()" />

<xsl:template match="/">
<results>
<xsl:for-each select="//book[@year=1980]| //book[@year=2003]">
<book>
    <xsl:copy-of select="title" />
    <xsl:copy-of select="store" />
</book>
</xsl:for-each>
</results>
</xsl:template>
</xsl:transform>
```

# 2.60. Query 60

## 2.60.1. Query 60t: Text XML

Description: *Find all xref elements that refer to the id of the topic element with attribute number value "4".*

The result of this query should be:

```
<results>
   <xref idref="top4"/>
</results>
```

### 2.60.1.1. Complexity

Complexity = 8.

- Subquery 1 (get xref element that have the same idref as the [report - subquery2]):
  - Report items: 1 (xref elements)
  - Condition items: (idref attribute (1) that has a reference value equal to [report subquery 2] (2))

  Complexity Subquery 1 = 4
- Subquery 2 (get the id of the topic with attribute number value "4"):
  - Report items: 1 (id attribute)
  - Condition items: (parent = topic (1), attribute number (1) with a certain value (1))

Complexity subquery 2 = 4

### 2.60.1.2. XQuery solution

```
<results>
{
for $b in //topic[@number=4]/@id
for $a in //xref
where $a/@idref = $b
return $a
}
</results>
```

### 2.60.1.3. XSLT solution

```
<xsl:transform xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
version="2.0">
<xsl:template match="text()" />
<xsl:output method="xml" indent="yes" />

<xsl:variable name="id" select="//topic[@number = 4]/@id" />

<xsl:template match="xref[@idref = $id]">
    <results><xsl:copy-of select="." /></results>
</xsl:template>
</xsl:transform>
```

```
<xsl:transform xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
version="2.0">
<xsl:output method="xml" indent="yes" />

<xsl:template match="/">
<results>
<xsl:for-each select="//xref[@idref = //topic[@number = 4]/@id]">
    <xsl:copy-of select="." />
</xsl:for-each>
</results>
</xsl:template>
</xsl:transform>
```

## 2.60.2. Query 60d: Data XML

Description: *Find all books that have a year attribute that is higher than the year attribute of the third book in the document.*

The result of this query should be:

```
<results>
  <book year="2003" id="b8">
    <title>XQuery from the experts</title>
    <store>
      <source>bstore2.example.com</source>
      <price>55.95</price>
    </store>
    <store>
      <source>www.amazon.com</source>
      <price>50.95</price>
    </store>
    <tip>
      <xref idref="b9" />
    </tip>
  </book>
```

```
</results>
```

### 2.60.2.1. Complexity

Complexity = 7.

- Subquery 1 (get book element that have a year attribute that is higher than the [report - subquery2]):
  - Report items: 1 (book elements)
  - Condition items: (year attribute (1) that has a reference value higher than [report subquery 2] (2))

  Complexity Subquery 1 = 4
- Subquery 2 (get the year attribute of the third book):
  - Report items: 1 (year attribute)
  - Condition items: (parent = book (1), with a certain position (1))

  Complexity subquery 2 = 3

### 2.60.2.2. XQuery solution

```
<results>
{
for $b in //book[3]/@year
for $a in //book
where $a/@year > $b
return $a
}
</results>
```

### 2.60.2.3. XSLT solution

```
<xsl:transform xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
version="2.0">
<xsl:template match="text()" />
<xsl:output method="xml" indent="yes" />

<xsl:variable name="year" select="//book[3]/@year"/>

<xsl:template match="book[@year > $year]">
    <xsl:copy-of select="." />
</xsl:template>
</xsl:transform>
```

```
<xsl:transform xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
version="2.0">
<xsl:template match="text()" />
<xsl:output method="xml" indent="yes" />

<xsl:template match="/">
<results>
<xsl:for-each select="//book[@year > //book[3]/@year]">
    <xsl:copy-of select="." />
</xsl:for-each>
</results>
</xsl:template>
</xsl:transform>
```

# 2.61. Query 61

## 2.61.1. Query 61t: Text XML

Description: *Count the number of title elements the document "report.xml"*

The result of this query should be:

```
14
```

### 2.61.1.1. Complexity

Complexity = 2

- Report items: 1 [number of title element]
- Condition items: 1 [count (1)]

### 2.61.1.2. XQuery solution

```
<results>
{
count(//title)
}
</results>
```

```
count(for $b in //title
return $b)
```

### 2.61.1.3. XSLT solution

```
<xsl:transform xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
version="2.0">
<xsl:template match="text()" />
<xsl:output method="xml" indent="yes" />

<xsl:variable name="titles" select="//title"/>

<xsl:template match="/">
    <results><xsl:value-of select="count($titles)" /></results>
</xsl:template>
</xsl:transform>
```

```
<xsl:transform xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
version="2.0">
<xsl:template match="/">
    <results><xsl:value-of select="count(//title)" /></results>
</xsl:template>
</xsl:transform>
```

## 2.61.2. Query 61d: Data XML

Description: *Count the number of title elements in the pricelist*

The result of this query should be:

```
<results>9</results>
```

### 2.61.2.1. Complexity

Complexity = 2

- Report items: 1 [number of title element]
- Condition items: 1 [count (1)]

### 2.61.2.2. XQuery solution

```
<results>
{
count(//title)
}
</results>
```

```
count(for $b in //title
return $b)
```

### 2.61.2.3. XSLT solution

```
<xsl:transform xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
version="2.0">
<xsl:template match="text()" />
<xsl:output method="xml" indent="yes" />

<xsl:variable name="titles" select="//title"/>

<xsl:template match="/">
    <results><xsl:value-of select="count($titles)" /></results>
</xsl:template>
</xsl:transform>
```

```
<xsl:transform xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
version="2.0">
<xsl:template match="/">
    <results><xsl:value-of select="count(//title)" /></results>
</xsl:template>
</xsl:transform>
```

# 2.62. Query 62

## 2.62.1. Query 62t: Text XML

Description: *Find the second para element of the first topic element the document "report.xml".*

The result of this query should be:

```
<results>
  <para security="u">A database schema also defines the
  relationships between the various types of data. Similarly, a DTD
  specifies
  <emph>rules</emph>to help ensure documents have a consistent,
  logical structure.</para>
</results>
```

### 2.62.1.1. Complexity

Complexity = 4

- Report items: 1 [para element]
- Condition items: [position report item = 2 (1), parent = topic (1), position =1 (1)]

### 2.62.1.2. XQuery solution

```
<results>
{
(//topic[1]/para[2])
}
</results>
```

```
for $t in //topic[position() = 1]
return $t/para[position() = 2]
```

### 2.62.1.3. XSLT solution

```
<xsl:transform xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
version="2.0">
<xsl:output method="xml" indent="yes" />
<xsl:template match="text()" />

<xsl:template match="/">
<xsl:for-each select="//topic">
<xsl:if test="position()=1">
<xsl:copy-of select="para[2]" />
</xsl:if>
</xsl:for-each>
</xsl:template>
</xsl:transform>
```

```
<xsl:transform xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
version="2.0">
<xsl:output method="xml" indent="yes" />
<xsl:template match="text()" />

<xsl:template match="topic[1]//para[2]">
   <xsl:copy-of select="." />
</xsl:template>
</xsl:transform>
```

## 2.62.2. Query 62d: Data XML

Description: *Find the second store element of the first book element in the pricelist.*

The result of this query should be:

```
<results>
 <store>
 <source>bstore1.example.com</source>
 <price>65.95</price>
   </store>
</results>
```

### 2.62.2.1. Complexity

Complexity = 4

- Report items: 1 [store element]
- Condition items: [position report item = 2 (1), parent = book (1), position =1 (1)]

### 2.62.2.2. XQuery solution

```
<results>
{
(//book[1]/store[2])
}
</results>
```

```
for $b in //book[position() = 1]
return $b/store[position() = 2]
```

### 2.62.2.3. XSLT solution

```
<xsl:transform xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
version="2.0">
<xsl:output method="xml" indent="yes" />
<xsl:template match="text()" />

<xsl:template match="/">
<xsl:for-each select="//topic">
<xsl:if test="position()=1">
<xsl:copy-of select="para[2]" />
</xsl:if>
</xsl:for-each>
</xsl:template>
</xsl:transform>
```

```
<xsl:transform xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
version="2.0">
<xsl:output method="xml" indent="yes" />
<xsl:template match="text()" />

<xsl:template match="book[1]//store[2]">
<results>
    <xsl:copy-of select="." />
</results>
</xsl:template>
</xsl:transform>
```

# 2.63. Query 63

## 2.63.1. Query 63t: Text XML

Description: *Count the number of para elements in the first chapter*

The result of this query should be:

```
<results>2</results>
```

### 2.63.1.1. Complexity

Complexity = 4

- Report items: 1 [number]
- Condition items: [para element (1), with parent chapter (1), at position =1 (1), count (1)]

### 2.63.1.2. XQuery solution

```
<results>
{
count((//chapter[1]//para))
}
</results>
```

### 2.63.1.3. XSLT solution

```
<xsl:transform xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
version="2.0">
<xsl:template match="text()" />
<xsl:output method="xml" indent="yes" />

<xsl:variable name="chap" select="//chapter[1]" />

<xsl:template match="/">
<results><xsl:copy-of select="count($chap//para)" /></results>
</xsl:template>
</xsl:transform>
```

```
<xsl:transform xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
version="2.0">
<xsl:template match="text()" />
<xsl:output method="xml" indent="yes" />

<xsl:template match="/">
<xsl:copy-of select="count(//chapter[1]//para)" />
</xsl:template>
</xsl:transform>
```

## 2.63.2. Query 63d: Data XML

Description: *Count the number of xref elements in the ninth book*

The result of this query should be:

```
<results>2</results>
```

### 2.63.2.1. Complexity

Complexity = 4

- Report items: 1 [number]
- Condition items: [xref element (1), with parent book (1), at position =9 (1), count (1)]

### 2.63.2.2. XQuery solution

```
<results>
{
count((//book[9]//xref))
}
</results>
```

### 2.63.2.3. XSLT solution

```
<xsl:transform xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
version="2.0">
<xsl:template match="text()" />
<xsl:output method="xml" indent="yes" />

<xsl:template match="book[9]">
<count>
<xsl:value-of select="count(.//xref)" />
</count>
</xsl:template>
</xsl:transform>
```

```
<xsl:transform xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
version="2.0">
<xsl:output method="xml" indent="yes" />
<xsl:variable select="//book[9]//xref" name="ref"/>
<xsl:template match="/">
<count>
<xsl:value-of select="count($ref)" />
</count>
</xsl:template>
</xsl:transform>
```

# 2.64. Query 64

## 2.64.1. Query 64t: Text XML

Description: *Retrieve the chapter title of the chapter that has an para with an security attribute value "c"*

The result of this query should be:

```
<title>Resources</title>
```

### 2.64.1.1. Complexity

Complexity = 4

- Report items: 1 [title]
- Condition items: [parent chapter (1), with attribute security (1), with a certain value (1)]

### 2.64.1.2. XQuery solution

```
for $c in //chapter
where $c//para/@security="c"
return $c/title
```

### 2.64.1.3. XSLT solution

```
 <xsl:transform xmlns:xsl=
"http://www.w3.org/1999/XSL/Transform"
version="2.0">
<xsl:output method="xml" indent="yes" />
<xsl:template match="text()" />

<xsl:template match="chapter[.//para/@security='c']">
     <xsl:copy-of select="title" />
</xsl:template>
</xsl:transform>
```

```
<xsl:transform xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
version="2.0">

<xsl:output method="xml" indent="yes" />
<xsl:template match="text()" />

<xsl:template match="/">
<xsl:for-each select="//chapter[.//para/@security='c']">
     <xsl:copy-of select="title" />
</xsl:for-each>
</xsl:template>
</xsl:transform>
```

## 2.64.2. Query 64d: Data XML

Description: *Retrieve the book title of the book that has three store elements*

The result of this query should be:

```
<title>Advanced Programming in the Unix environment</title>
```

### 2.64.2.1. Complexity

Complexity = 5

- Report items: 1 [title]
- Condition items: [parent book (1), with store element (1), count (1) with a certain value (1)]

### 2.64.2.2. XQuery solution

```
for $c in //book
where count($c//store) = 3
return $c/title
```

### 2.64.2.3. XSLT solution

```
<xsl:transform xmlns:xsl=
"http://www.w3.org/1999/XSL/Transform"
version="2.0">
<xsl:output method="xml" indent="yes" />
<xsl:template match="/">
    <xsl:for-each select="//book">
    <xsl:if test="count(.//store) = 3">
 <xsl:copy-of select="title" />
```

```
    </xsl:if>
    </xsl:for-each>
</xsl:template>
</xsl:transform>
```

```
<xsl:transform xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
version="2.0">
<xsl:output method="xml" indent="yes" />

<xsl:template match="text()"/>
<xsl:template match="book[count(.//store) = 3]">
 <xsl:copy-of select="title" />
</xsl:template>
</xsl:transform>
```

# 2.65. XML documents

## 2.65.1. Document-oriented

### 2.65.1.1. DTD

```
<!ELEMENT author (last, first)>
<!ELEMENT chapter (title, author+, intro?, section*)>
<!ATTLIST chapter
 number (1 | 2 | 3) #REQUIRED
>
<!ELEMENT emph (#PCDATA)>
<!ELEMENT first (#PCDATA)>
<!ELEMENT graphic EMPTY>
<!ATTLIST graphic
 graphname (infoflow | tagexamp) #REQUIRED
>
<!ELEMENT intro (para+, graphic?)>
<!ELEMENT keyword (#PCDATA)>
<!ELEMENT last (#PCDATA)>
<!ELEMENT para (#PCDATA | emph | xref)*>
<!ATTLIST para
 security (c | u) #IMPLIED
>
<!ELEMENT report (title, chapter+)>
<!ELEMENT section (title, intro, topic*)>
<!ATTLIST section
 shorttitle CDATA #IMPLIED
>
<!ELEMENT title (#PCDATA | emph)*>
<!ELEMENT topic (title, keyword?, para+, graphic?)>
<!ATTLIST topic
 id ID #IMPLIED
 number CDATA #IMPLIED
>
<!ELEMENT xref EMPTY>
<!ATTLIST xref
 idref IDREF #REQUIRED
>
```

### 2.65.1.2. Document instance

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE report SYSTEM "report.dtd">
<report>
  <title>Getting started with SGML</title>
  <chapter number="1">
    <title>The business challenge</title>
    <author>
```

```
    <last>Stevens</last>
    <first>W.</first>
  </author>
  <intro>
    <para>With the ever-changing and growing global market,
    companies and large organizations are searching for ways to
    become more viable and competitive. Downsizing and other
    cost-cutting measures demand more efficient use of corporate
    resources. One very important resource is an organization's
    information.</para>
    <para>As part of the move toward integrated information
    management, whole industries are developing and implementing
    standards for exchanging technical information. This report
    describes how one such standard, the Standard Generalized
    Markup Language (SGML), works as part of an overall
    information management strategy.</para>
    <graphic graphname="infoflow" />
  </intro>
</chapter>
<chapter number="2">
  <title>Getting to know SGML</title>
  <author>
    <last>Stevens</last>
    <first>W.</first>
  </author>
  <intro>
    <para>While SGML is a fairly recent technology, the use of
    <emph>markup</emph>in computer-generated documents has
    existed for a while.</para>
  </intro>
  <section shorttitle="What is markup?">
    <title>What is markup, or everything you always wanted to
    know about document preparation but were afraid to
    ask?</title>
    <intro>
      <para>Markup is everything in a document that is not
      content. The traditional meaning of markup is the manual
      <emph>marking</emph>up of typewritten text to give
      instructions for a typesetter or compositor about how to
      fit the text on a page and what typefaces to use. This kind
      of markup is known as
      <emph>procedural markup</emph>.</para>
    </intro>
    <topic id="top1" number="1">
      <title>Procedural markup</title>
      <keyword>Markup</keyword>
      <para security="u">Most electronic publishing systems today
      use some form of procedural markup. Procedural markup codes
      are good for one presentation of the information.</para>
    </topic>
    <topic id="top2" number="2">
      <title>Generic markup</title>
      <keyword>Markup</keyword>
      <para>Generic markup (also known as descriptive markup)
      describes the
      <emph>purpose</emph>of the text in a document. A basic
      concept of generic markup is that the content of a document
      must be separate from the style. Generic markup allows for
      multiple presentations of the information.</para>
    </topic>
    <topic id="top3" number="3">
      <title>Drawbacks of procedural markup</title>
      <keyword>Markup</keyword>
      <para security="u">Industries involved in technical
      documentation increasingly prefer generic over
      <emph>procedural</emph>markup schemes. When a company
      changes software or hardware systems, enormous data
      translation tasks arise, often resulting in errors.</para>
    </topic>
  </section>
  <section shorttitle="What is SGML?">
    <title>What
    <emph>is</emph>SGML in the grand scheme of the universe,
    anyway?</title>
    <intro>
      <para>SGML defines a strict markup scheme with a syntax for
      defining document data elements and an overall framework
```

```
        for marking up documents.</para>
        <para>SGML can describe and create documents that are not
        dependent on any hardware, software, formatter, or
        operating system. Since SGML documents conform to an
        international standard, they are portable.</para>
      </intro>
    </section>
    <section shorttitle="How does SGML work?">
      <title>How is SGML and would you recommend it to your
      grandmother?</title>
      <intro>
        <para>You can break a typical document into three layers:
        structure, content, and style. SGML works by separating
        these three aspects and deals mainly with the relationship
        between structure and content.</para>
      </intro>
      <topic id="top4" number="4">
        <title>Structure</title>
        <keyword>DTD</keyword>
        <para>At the heart of an SGML application is a file called
        the DTD, or Document Type Definition. The DTD sets up the
        structure of a document, much like a database schema
        describes the types of information it handles.</para>
        <para security="u">A database schema also defines the
        relationships between the various types of data. Similarly,
        a DTD specifies
        <emph>rules</emph>to help ensure documents have a
        consistent, logical structure.</para>
      </topic>
      <topic id="top5" number="5">
        <title>Content</title>
        <keyword>Markup</keyword>
        <para>Content is the information itself. The method for
        identifying the information and its meaning within this
        framework is called
        <emph>tagging</emph>. Tagging must conform to the rules
        established in the DTD (see
        <xref idref="top4" />).</para>
        <graphic graphname="tagexamp" />
      </topic>
      <topic id="top6" number="6">
        <title>Style</title>
        <para>SGML does not standardize style or other processing
        methods for information stored in SGML.</para>
      </topic>
    </section>
  </chapter>
  <chapter number="3">
    <title>Resources</title>
    <author>
      <last>Abiteboul</last>
      <first>Serge</first>
    </author>
    <author>
      <last>Buneman</last>
      <first>Peter</first>
    </author>
    <author>
      <last>Suciu</last>
      <first>Dan</first>
    </author>
    <section>
      <title>Conferences, tutorials, and training</title>
      <intro>
        <para>The Graphic Communications Association has been
        instrumental in the development of SGML. GCA provides
        conferences, tutorials, newsletters, and publication sales
        for both members and non-members.</para>
        <para security="c">Exiled members of the former Soviet
        Union's secret police, the KGB, have infiltrated the upper
        ranks of the GCA and are planning the Final Revolution as
        soon as DSSSL is completed.</para>
      </intro>
    </section>
  </chapter>
</report>
```

## 2.65.2. Data-oriented

### 2.65.2.1. DTD

```
<!ELEMENT book (title, store+, tip?)>
<!ATTLIST book
 year CDATA #IMPLIED
 id ID #IMPLIED
 available CDATA #IMPLIED
>
<!ELEMENT price (#PCDATA)>
<!ELEMENT prices (book+)>
<!ELEMENT source (#PCDATA)>
<!ELEMENT store (source, price)>
<!ELEMENT tip (xref+)>
<!ELEMENT title (#PCDATA)>
<!ELEMENT xref EMPTY>
<!ATTLIST xref
 idref IDREF #IMPLIED
>
```

### 2.65.2.2. Document instance

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE prices SYSTEM "prices.dtd">
<prices>
  <book year="1992" id="b1">
    <title>Advanced Programming in the Unix environment</title>
    <store>
      <source>bstore2.example.com</source>
      <price>65.95</price>
    </store>
    <store>
      <source>bstore1.example.com</source>
      <price>65.95</price>
    </store>
    <store>
      <source>www.bol.com</source>
      <price>45.95</price>
    </store>
    <tip>
      <xref idref="b2" />
    </tip>
  </book>
  <book year="1994" id="b2">
    <title>TCP/IP Illustrated</title>
    <store>
      <source>bstore2.example.com</source>
      <price>65.95</price>
    </store>
    <store>
      <source>bstore1.example.com</source>
      <price>65.95</price>
    </store>
    <tip>
      <xref idref="b1" />
    </tip>
  </book>
  <book year="2000" id="b3">
    <title>Data on the Web</title>
    <store>
      <source>bstore2.example.com</source>
      <price>34.95</price>
    </store>
    <store>
      <source>bstore1.example.com</source>
      <price>39.95</price>
    </store>
  </book>
```

```
<book year="1999" id="b4">
  <title>The Economics of Technology and Content for Digital
  TV</title>
  <store>
    <source>www.amazon.com</source>
    <price>34.95</price>
  </store>
  <store>
    <source>www.bol.com</source>
    <price>69.95</price>
  </store>
</book>
<book year="1982" available="no" id="b5">
  <title>Getting started with SGML</title>
  <store>
    <source>www.mystore.com</source>
    <price>14.95</price>
  </store>
  <store>
    <source>www.bol.com</source>
    <price>45.95</price>
  </store>
</book>
<book year="1980" available="no" id="b6">
  <title>Drawbacks of procedural markup</title>
  <store>
    <source>bstore2.example.com</source>
    <price>25.95</price>
  </store>
  <tip>
    <xref idref="b5" />
  </tip>
</book>
<book year="1950" available="no" id="b7">
  <title>Lords of the Ring</title>
  <store>
    <source>bstore1.example.com</source>
    <price>55.95</price>
  </store>
  <store>
    <source>bstore2.example.com</source>
    <price>65.95</price>
  </store>
  <store>
    <source>bstore3.example.com</source>
    <price>25.95</price>
  </store>
  <store>
    <source>www.bol.com</source>
    <price>45.95</price>
  </store>
  <store>
    <source>www.amazon.com</source>
    <price>55.95</price>
  </store>
</book>
<book year="2003" id="b8">
  <title>XQuery from the experts</title>
  <store>
    <source>bstore2.example.com</source>
    <price>55.95</price>
  </store>
  <store>
    <source>www.amazon.com</source>
    <price>50.95</price>
  </store>
  <tip>
    <xref idref="b9" />
  </tip>
</book>
<book year="1999" id="b9">
  <title>XSLT: programmers reference</title>
  <store>
    <source>bstore2.example.com</source>
    <price>35.95</price>
  </store>
  <store>
```

```
      <source>www.amazon.com</source>
      <price>40.95</price>
    </store>
    <tip>
      <xref idref="b8" />
      <xref idref="b5" />
    </tip>
  </book>
</prices>
```

# Bibliography


[Gra05] Usability of XML Query LanguagesJ. Graaumans. Ph.D. Thesis, Instituut voor Informatica en Informatiekunde, Universiteit Utrecht, 2005.

[W3C-QU] W3C XML Query Use Cases. W3C Technical Report, D. Chamberlin, P. Fankhauser, D. Florescu, M. Marchiori, J. Robie. http://www.w3.org/TR/xquery-use-cases/, 04 April 2005.

[W3 School] W3C School. http://www.w3schools.com/

[YOK02] XBench - A Family of Benchmarks for XML DBMSs. B. B. Yao, M. T. Oszu, J. Keenlyside. December 2002.

[SWK+02] XMark: A Benchmark for XML Data Management. A. R. Schmidt, F. Waas, M. L. Kersten, M. J. Carey, I. Manolescu, R. Busse. Proceedings of the International Conference on Very Large Data Bases (VLDB), Hong Kong, China, pages 974-985, August 2002.