XML and Web Technologies

INFO-H509

XQuery

14 May 2021

BAKKALI Yahya : 000445166
HAUWAERT Maxime : 000461714

Université Libre de Bruxelles (ULB)

Contents

1 Introduction

DBLP is an online bibliographical database for computer science containing around 1 million references. Its content is publicly available in XML format. The goal of this project is to write a XQuery program for each query performed on a small excerpt of this data. In this report, all the hypotheses that have been made as well as the explanations of the XQuery programs will be seen.

2 XQuery programs

In this section, each program is described, and the requested output format is indicated. In addition, the blocks of the format are highlighted to show the different tags. The highlighting is used to make the link between the format and the solution of the query. Since some solutions can be complex, highlighting allows to refer in the solution to where each block is processed.

2.1 Program 1

```
Give for each author the number of co-authors and the number of joint
    publications with each of them, using the following output format.

<authors_coauthors>
    <author>

        <name>A. B. M. Shawkat Ali</name>
        <coauthors number="4">
            <coauthor>
                <name>M. Delowar Hossain</name>
                <nb_joint_pubs>1</nb_joint_pubs>
            </coauthor>
           ...
        </coauthors>
    </author>
       ...
</authors_coauthors>
```

2.1.1 Solution

```
<authors_coauthors>
```

```
{
  for $author in distinct-values(//author)
  return element author
  {
    <name>{$author}</name>,
    let $coauthors := distinct-values(//*[author=$author]/author[not(.=$author)])
    return element coauthors
    {
      attribute number {count($coauthors)},
      for $coauthor in $coauthors
      return element coauthor
      {
        <name>{$coauthor}</name>,
        <nb_joint_pubs>{count(//*[author=$coauthor]/author[.=$author])}</nb_joint_pubs>
      }
    }
  }
}
</authors_coauthors>
```

2.1.2 Explanation

The reasoning behind this solution is as follows. First, all authors are collected. Then, for each author, all co-authors who collaborate with him are also extracted. Their number can be calculated with the count() function. Finally, for each co-author, the number of joint publications between them is counted.

2.2 Program 2

```
For each proceeding give its title and the titles of articles appearing in it,
    using the following output format.

<proceedings>
    <proc_title>6th Annual IEEE/ACIS International Conference
        (...)</proc_title>
    <title>Understanding Consumer Search Activity and Online (...)</title>
    <title>Approximate Element Computational Time for Domain (...)</title>
```

```
    <title>Towards a Table Driven XML QoS Aware Transmission
        Framework.</title>
    ...
</proceedings>
```

2.2.1 Solution

```
for $proc in /dblp/proceedings
return element proceedings
{
  <proc_title>{data($proc/title)}</proc_title>,
  for $crossref in data($proc/@key), $article in /dblp/*[crossref=$crossref]
  return
    <title>{data($article/title)}</title>
}
```

2.2.2 Explanation

Here, the procedure is as follows. First, all the proceedings are extracted and then for each of them, its title is obtained by the data() function as well as its reference which is stored in the key attribute. Finally, the reference is used to collect all the articles that have it as a cross-reference in their information.

2.3 Program 3

```
Write an XQuery program that computes, for each pair of authors x and y ≠ x the
    distance between x and y using the following output format.

<distances>
    <distance author1="Lizhu Zhou" author2="Dengfeng Zhang" distance="3"/>
    <distance author1="Lizhu Zhou" author2="Xuesong Yan" distance="2"/>
    ....
</distances>
```

2.3.1 Solution

```
declare variable $root := .;

declare function local:print($author as xs:string, $authors as xs:string*,
    $distance as xs:integer)
{
  for $coauthor in $authors
  return
      <distance author1="{$author}" author2="{$coauthor}" distance="{$distance}"/>
};

declare function local:explore($author as xs:string,$authors as xs:string*,
    $checked_authors as xs:string*, $distance as xs:integer)
{
  let $coauthors :=
      distinct-values($root//*[author=$authors]/author[not(.=($authors,$checked_authors))])
  return
    if (not(empty($coauthors))) then (
      local:print($author, $coauthors, $distance),
      local:explore($author, $coauthors, ($authors, $checked_authors),
          $distance+1)
    ) else (
      local:print($author, $coauthors, $distance)
    )
};

<distances>
{
  for $author in distinct-values(//author)
  return local:explore($author, ($author), (), 1)
}
</distances>
```

2.3.2 Explanation

For this query, it was decided to consider the file as a graph of co-authors whose distance can be calculated by exploring the paths. The breadth-first search algorithm was used to obtain for each author all the distances between him and all other authors that can be reached. Below is a

pseudo-code of how the algorithm was designed and implemented in the XQuery program.

Algorithm 1 $Explore(author, authors, checked\_authors, distance)$

$checked \leftarrow authors \cup checked\_authors$
$coauthors \leftarrow \{coauthor \in all\_authors \setminus checked \mid \exists p \in authors \wedge distance(p, coauthor) = 1\}$
if $coauthors = \emptyset$ **then**
  for all $coauthor \in coauthors$ **do**
    $Print(author, coauthor, distance)$
  end for
else
  for all $coauthor \in coauthors$ **do**
    $Print(author, coauthor, distance)$
  end for
  $Explore(author, coauthors, (checked\_authors \cup authors), distance + 1)$
end if

3 Program manual

For executing these queries above, the following command line can be used.

```
$ java -cp saxon9he.jar net.sf.saxon.Query -s:"dblp-excerpt.xml"
    -q:"query1.xq" -o:"output.xml"
```