XSLT Ingest Example: Create New Persons and Organizations

Create New Persons and Organizations

The transform for organizations, **makeUROs.xsl**, is much simpler than that of persons and so we shall describe **makeURPs.xsl** in this section. As mentioned previously, in the case of persons, we have name parts and a uniquely assigned, but possibly missing, source **NETID** for matching. We also want that URIs be assigned uniquely even when we have an exact name part match for a set of records but **NETID** is missing in one (or more) records but not in others. Indeed it quite possible for there to be several URPs in our source that have exactly the same name parts, character for character, but different netids. In our example there are at least three (and possibly four) distinct people named 'Arthur R Fuller' and multiple **EduRecords** that have to be assigned to the correct person. Misattribution should be rare although it is unavoidable in an automated system when there is no sure way to distinguish between different people. Thus when there is no netid associated with an **EduRecord** we will add a 'weak attribution' triple in the RDF that we generate. Our main purpose here is to create a file of new person records, in the **PER0.xml** style, that specifies name parts, **uri** and **netid** if possible. These will be used in a later step to fill in missing URIs in **ED0.xml**.

```
<?xml version="1.0"?>
<xsl:stylesheet version='2.0'
        xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
        xmlns:vfx='http://vivoweb.org/ext/functions'
        xmlns:xs='http://www.w3.org/2001/XMLSchema'
        exclude-result-prefixes='xs vfx xsl'
>

<xsl:param name='unoMapFile'   required='yes'/>                    0

<xsl:output method='xml' indent='yes' normalization-form='NFC'/>

<xsl:variable name='localNamespace'>                               1
<xsl:text>http://vivo.cornell.edu/individual/</xsl:text>
</xsl:variable>

<xsl:variable name='MYNL'><xsl:text>
</xsl:text></xsl:variable>

<xsl:variable name='unomap'                                        2
        select="document($unoMapFile)/Mapping"/>
```

makeURPs.xsl Fragment 1 - Figure 10

- [F10H0] Declare the parameter that contains the name of the UNO file for Persons.
- [F10H1] Identify the namespace used to fully qualify a URI local name.
- [F10H2] Make the UNO file accessible as a set of nodes.

Figure 11 details how **Per0.xml** style person records are created. This part of the transform is the most complex in our example. This is because of the fact that we must index correctly through the set of 11 URIs constructed above as we assign them to the URPs.

```
<xsl:variable name='cgCounts' as='xs:integer*'>                    0
<xsl:for-each-group select='EduRecord[personUri=""]'
       group-by='lower-case(concat(normalize-space(ln),"|",
                                   normalize-space(fn),"|",
                                   normalize-space(mn)))'>

<xsl:variable name='cg' select='current-group()' />

<xsl:variable name='gp'>
<xsl:for-each-group select='$cg' group-by='vfx:adjust(nid)'>
<xsl:copy-of select='.'/>
</xsl:for-each-group>
</xsl:variable>

<xsl:value-of select='count($gp/EduRecord)'/>
</xsl:for-each-group>
</xsl:variable>
```

```
<xsl:variable name='cumulativeCgCounts' as='xs:integer*'>     1
<xsl:call-template name='cumulativeSum'>
<xsl:with-param name='vals' select='$cgCounts'/>
<xsl:with-param name='seq' select='()'/>
</xsl:call-template>
</xsl:variable>
```

makeURPs.xsl Fragment 2 - Figure 11

- [F11H0] This is the same grouping method employed in the Count Step. The variable **cgCounts** refers to a set of counts each of which is the number of distinct people sharing a name.
- [F11H1] The call to the recursive named template **cumulativeSum** produces a sequence containing the 0 based cumulative sums for the counts in **cgCounts**. These are the offsets into the list of 11 URIs that we need to do URI assignment to the new people. The variable **cumulativeCgCounts** contains the cumulative sum sequence. The recursive named template can be found in the Appendix D and source file **makeURPs.xsl**.

Figure 12 summarizes the counting, grouping and indexing for our result set. There are 7 names, 11 people and 3 people have no netid.

Name	Netid	Count	Offset	Index
David Augustus Green	dag065	1	0	1
Don A Horsham	dah3507	2	1	2
	dah256			3
Denise Hortense Valencia		1	3	4
Cathrine A Dale	cad2616	1	4	5
Andera S Killian		1	5	6
Arthur R Fuller		4	6	7
	arf27			8
	arf33			9
	arf72			10
Suzy Beltaine	sb016	1	10	11

Indexing Details - Figure 12

With these counts and sums in hand we can now iterate through the groups and subgroups to collect names and assign them URIs. Figure 13 displays the XSL code to finish this step. We begin as before by grouping **EduRecords** with URPs by name parts.

```
<xsl:for-each-group select='EduRecord[personUri=""]'
        group-by='lower-case(concat(normalize-space(ln),"|",
                                    normalize-space(fn),"|",
                                    normalize-space(mn)))'>

<xsl:variable name='posA' select='position()'/>                          0
<xsl:variable name='cg' select='current-group()'/>

<xsl:for-each-group select='$cg' group-by='vfx:adjust(nid)'>             1

<xsl:variable name='posB' select='position()'/>                          1a
<xsl:variable name='index'
        select='$posB + xs:integer($cumulativeCgCounts[$posA])'/>

<xsl:variable name='u'                                                   1b
        select='$unomap/map[@n = $index]/@nuno'/>

<xsl:call-template name='makePerson'>                                    1c
<xsl:with-param name='fn' select='fn'/>
<xsl:with-param name='mn' select='mn'/>
<xsl:with-param name='ln' select='ln'/>
<xsl:with-param name='uri'
        select='concat($localNamespace,$u)'/>
<xsl:with-param name='nid' select='./nid'/>
</xsl:call-template>

</xsl:for-each-group>

</xsl:for-each-group>
```

makeURPs.xsl Fragment 3 - Figure 13

- [F13H0] The variable **posA** is a counter that ranges over the 7 name groups and **cg** is the variable that contains the current group.
- [F13H1] This code groups the current group by **nid** (i.e. netid).
    - [F13H1a] The variable **posB** ranges over the number of records in the current group (by netid) of the current group (by name parts) which varies in size (see Figure 12 just above).
    - [F13H1b] The variable **index** combines **posB** and the offsets in the sequence **cumulativeCgCounts** which is then used to choose a URI for assignment to the variable **u**.
    - [F13H1c] A named template, **makePerson**, is called to construct the person XML with all the name parts, netid (if possible) and the fully qualified URI assigned.

The files **NewPers.xml** and **NewOrgs.xml** are created by applying the transforms **makeURPs.xsl** and **makeUROs.xsl** successively.

Start Previous Next