DHARPA
Design & Consulting

- Software Architecture, Design and Development
- Web-based Database Solutions
- All Aspects of Database Design and Maintenance

ABN : 43 548 173 674

ph: 0421 416 786
7 Mahomed St,
Alice Springs NT 0870
info@dharpa.com

Instructions

National Curriculum Import to SQL database

November 2012

(released with permission of Yirara College, Alice Springs)

Overview

This tool runs as a stand alone application on a Windows PC.

The tool is designed to process the national curriculum documents in RDF format (a form of XML) that are generated by the national curriculum site, and dump the data into a blank MS SQL Server database.

Note the following points:

- the tool analyses the content of the document in order to work out what to generate as output, so it will accept future types of attribute, as long as they are simple single or multiple attributes and not structurally important (a summary of critical structural criteria is discussed in a later section)

- the tool creates tables and relationships in the SQL database based on the structure of the document, then inserts the data into the tables based on the document content

- the database diagram tool in MS SQL Server Management Studio can then be used to create database diagrams that automatically show the relationships once the tables have been displayed

Technically, the tool is written in C# as a Windows Forms application using Microsoft Visual Studio 2010. Source code is included.

Section 1: Retrieving the RDF data file

The current URL for retrieval of the RDF data is
http://rdf.australiancurriculum.edu.au/sparqltest

Given the ‘test’ in the URL, it should be expected that the URL will soon change. However, the operation is expected to stay essentially the same.

The current sample query up at http://rdf.australiancurriculum.edu.au/sparqltest is:

```
prefix asn: <http://purl.org/ASN/schema/core/>

construct
 { ?s ?p ?o }
where
{
  ?s a asn:StandardDocument .
   ?s ?p ?o .
}
```

this constrains the retrieved RDF statements to be statements about StandardDocuments (the top level curricula):

“?s a asn:StandardDocument .”   =   “?s is a Standard Document”.

Removing that clause leaves:

```
prefix asn: <http://purl.org/ASN/schema/core/>

construct
 { ?s ?p ?o }
where
{
   ?s ?p ?o .
}
```

which should return the entire curriculum document.

This should be save to the hard drive.

Section 2: Preparing the Program for the Conversion

(a) Preparing the RDF Document

At present, the document type is declared to be UTF-8 but contains some characters which violate UTF-8 encoding. This causes problems with the XML parsing library.

In the application, there is a button marked ‘Rewrite as UTF-8’. This may be used to rewrite the document in-place as true UTF-8. Note that the rewriting process also strips line breaks and makes the file much less readable, so you may want to keep a copy of the original RDF file if you need to view it.

Presumably the document will be fixed and at that point this step will no longer be necessary.

(b) Configuring the Program

Firstly a ***new blank database*** needs to be created on the database Server.
This can be achieved using MSSQL Management Studio by right clicking on the ‘Databases’ node of the server and choosing ‘New Database’.
Removing all the tables in the database is also an option, but the dependencies in the database make this a non-trivial task. It is easier to drop and re-create the database, taking care to save any diagrams or views that have been constructed on the database.

Next, copy the ‘RdfImport’ folder in the release ZIP file, to a trusted location such as the ‘documents’ folder. The ‘RdfImport.exe.config’ file in that folder needs to be edited so that a valid connection string points at the SQL Server. A sample is present, so generally the server name only needs to be modified.

(c) Running the program

Run the ‘RdfImport.exe’ file. Select the file and click ‘Scan’.
The process should take around 20 seconds, and not more than 60 seconds (depending on the speed of the server).

A log file with the data analysis, dump of the collected data, and SQL statements is generated. This log file has the same name and location as the input file, but with ‘.txt’ added to the end of the filename.
This log data also appears in the text area at the bottom of the screen.

The SQL statements generate the tables and relationships, and dump the data into them, so the database should be ready to go at the end of this step.

Section 3: Heuristics and Limitations of the Analysis Tool

(a) Input Document

The Import tool is reasonably resilient to changes in the format of the input document.

However the following structural assumptions must be true for the input document:
- it must be in RDF format
- it must be nested only one level deep
- all relationships between nodes must be established using the 'isChildOf' and 'hasChild' attributes
- the <rdf:Description> nodes are used as the units (rows) of data, with the following attributes used to determine the *node type* – the first non-empty attribute being used
    - the <statementLabel> attribute
    - the <title> attribute
    - the <rdf:type> attribute

- the <description> attribute generally contains the value of the node
- attribute types are designated 'single' if the attribute occurs at most once for a parent node
- attribute types are designated 'multiple' if the attribute ever occurs multiple times for a parent node

Major structural changes to the RDF document breaking the above assumptions will require changes to the tool. ACARA has advised that it is not anticipated that this will occur.

(b) Text Summary

The text summary generated has the following sections:

XML Namespaces and URLs
lists the XML namespaces referred to by XMLNS properties

Attribute Types
lists the attribute types found in the documents, and their properties in the following columns:
- attribute type in the format *a:b* where *a* is a 'user friendly' version of the namespace URL and *b* is the attribute name
- *single* or *multiple* as described above
- the *URL stem* used by that attribute when specifying value information

Description Node Types
lists the node types found during the document scan, using the rules specified above to determine the node type. Listed are:
- node type and the number of nodes of that type encountered
- the distinct types of child or parent nodes found
- a list of all attribute types encountered for that node type

Summary Node List
lists a summary line for each node with the key and the main value

Detailed Node List
lists each node with all detected and processed attributes, including an additional line containing the type and description of declared parent and child nodes

(c) SQL Database Generation

The SQL database generated has the following characteristics:

- a ***key table*** is generated for each entry in the Description Node Types list
  eg. Sub_strand

- each attribute designate *single* listed for the node type becomes a ***column*** in the table

- each attribute designate *multiple* listed for the node type generates a ***many-to-one table*** named as the key table with two following underscores and the name of the multiple attribute. This is the only way to represent multiple attributes in the relational model.
  eg. Sub_strand__conceptKeyword

- a foreign key relationship is generated between many-to-one tables and their related key table

- a foreign key relationship is generated between key tables related by the ‘isChildOf’ / ‘hasChild’ attributes