AMCM043 XML and Structured Information

Course Work 2

Student ID:089583717

Contents

Introduction

This document forms part of the documentation for the AMCM043 Course Work 2. The objective is to provide a brief description on how the given XQUERY works. An inventory of files submitted as part of the coursework (found in Appendix A – File Inventory), along with brief notes on the XML Schemas will also be given.

Schema Definitions

Examiner Schema

The potential examiner CVs are held in the XML document examiners.xml. This is aligned to schema examiners.xsd. This has five major sections:

- Personal Details (personDetails)
- Subject area of expertise of the examiner (subjectArea)
- A list of academic posts held, including the current post (academicPost)
- Details of where the examiner was awarded their PhD, along with supervisors and examiners (PhDDetail)
- Previous experience in examining PhDs (previousExperience)

One particular point to note within this schema is the method of identification. In general the unique and distinct identification of people is not easy. Most computer systems will have their own unique identifier for a person, but there is no clear cut way to guarantee that it relates to the correct person, or that duplicate identifiers exist for the same person. Instead they rely on the two pillars of trust and verification.

The concept of trust is to rely on some other body that can state that the person is who they say they are. Common examples are Birth Certificates, Passports, National Insurance Numbers, etc. So a possible method of identification is to create a composite identifier that includes the name of the organization that issues the identification number along with that number.

The concept of verification is to somehow check that the identification given is genuine and not a forgery. The concept of verification is probably outside the scope of this course work assignment.

The actual method used within the schema is one commonly used within commercial applications. I assign a unique (at least within the confines of this course work) identifier to each person identified, regardless of the role they play, along with the more common identifier and the issuing authority.

The previous experience of the examiner is held within the previousExperience node. This holds skeletal details required for the query as well as the examination date (held in case there is a cut off point after which the experience is not considered current).

Candidate Schema

The candidate schema (candidate.xsd) is simpler than the examiner schema and holds only the information required to provide a skeletal record of the PhD candidate and the area of the PhD submission. Although different namespaces have been used between the examiner and candidate details, I have used the same type definitions where appropriate. A possible modification to this course work might have been to create a common repository of types that both the examiner and

candidate could reference in their schemas, but this seemed a little overkill for what is ostensibly an assignment of XQUERY functionality.

A candidate document (testCandidate.xml) was created from this schema to test the XQUERY.

XQUERY Overview

The query has a *prolog*, which is used to identify the two namespaces for the candidate and examiner respectively. I have used my college web space URL to uniquely assign namespaces to this particular coursework. Next, the entire body of the query is wrapped in a <examinerChoice> element to create a well formed XML document from the query.

The first lines in the *body* of the query declare the following variables:

$collegeList Used to hold the current list of colleges that represent the University of London. This list is held in an external document for ease of maintenance (i.e., can be maintained independent to the XQUERY) and so that it could be used by other applications. This is particular important as a major source of data quality issues can arise through inconsistent usage of reference data.

$candidate The candidate details are held within a single variable. This is possible since the candidate XML schema only allows one candidate per XML document.

$priorityNames This is used to hold a set of reference data for the examiner pair priority. The use of an internal reference table does contradict my earlier statement within the college list with respect to the consistent usage of reference data. Since it is declared internal to the XQUERY it is not (at least easily) available for use by other applications and would require maintenance to the XQUERY file if the table needed to be modified. However, I wished to demonstrate the technique of internal reference table declaration for the course work and would appeal to the examiner on this logic.

The next line binds the variable $localExaminar to the list of examiners in the examiners.xml document in a *for loop* that will process each examiner node. In addition, the predicate uses the college list to only select those examiners that hold a current position in one of the University of London colleges. This is analogous to a database view, selecting local prospective examiners.

The next line binds the variable $localExaminarDetails to the current context examiner personal details child node for use in the *return* statement. The line after creates a sequence of institutions that the examiner has previous marked PhDs in. This is used to later satisfy the coursework requirement "*At least one of the examiners for each candidate shall, whenever practicable, have had experience in examining for the PhD degree of this University*".

The next three lines repeat the above procedure for external examiners by negating the college list predicate to select examiners whose current post is not within one of the University of London colleges.

As mentioned above, there is a requirement to ensure that at least one of the examiners has experience of examining for a University of London PhD. The next line establishes a priority to those

examiner pairs that have this experience. This uses the previously created variables $localExaminarExperience and $externalExaminarExperience, which hold a sequence of institutions that the examiner has previously examined PhDs in. If both candidates have such experience, then the priority is set to 1 (Ideal pairing), else if either the local or external examiner has such experience, then this is set to 2 (Possible pairing), otherwise it is set to 3 (Fallback pairing). This variable is used both in the *order by* clause and as part of a look up expression within the *return* clause.

The next part of the query is the *where* clause. This checks, for both local and external examiners, that their area of expertise is the same as the candidate PhD and that they are not the supervisor of the candidate.

The query uses the *order by* clause to order the results by the previously calculated priority (held in the variable $examinarPairPriority). The idea is to place those examiner pairs with experience in examining PhDs at the University of London towards the top of the resultant examiner pair document.

The final section is the *return* clause. This uses the fact that we have two *for* clauses to combine them to create a Cartesian join to get all possible pairs of examiners. No further filtering of results is required since this has been done previously within the query. The *return* clause will be run for each Cartesian pair of results from the *for* statements. Therefore a <examinerPair> element is used to wrap each pair for output to the resultant XML document. The local and external examiners are identified by the use of different elements; <localExaminar> and <externalExaminar>.

The actual examiner details output are held within the variables $localExaminarDetails and $externalExaminarDetails, for the local examiner and external examiner respectively. These are currently bound to the personalDetails child element of the examiner. If additional, or less, detail is required these variables can be amended in their binding statement without impacting any code within the *return* statement.

The only other point worth noting is the use of the $examinarPairPriority and $priorityNames in the lookup expression. This is used to convert the priority codes to a meaningful priority description. The *data()* function is used to ensure just the tag content is used for the final presentation.

Appendix A – File Inventory

The course work consists of several files. The inventory list of these is listed below.

File Name	Description
AMCM043_CourseWork2_ID089583717.pdf	The course work documentation.
candidate.xsd	The XML Schema for the candidate details.
examiners.xml	The set of test CVs for the potential examiners.
examiners.xsd	The XML Schema for the examiners details.
examiners.xquery	The XQUERY file that produces the list of potential examiners.
examinerPairs.xml	Result of the XQUERY.
testCandidate.xml	The document giving the candidate details.
UniversityOfLondonColleges.xml	The document giving the list of colleges that form the University of London.