Querying and Transformation - Xquery - Nested Queries

Dhamini Nijgal - 2047013

2020

**Querying and Transformation**

Given the increasing number of applications that use XML to exchange, mediate, and store data, tools for effective management of XML data are becoming increasingly important. In particular, tools for querying and transformation of XML data are essential to extract information from large bodies of XML data, and to convert data between different representations (schemas) in XML. Just as the output of a relational query is a relation, the output of an XML query can be an XML document. As a result, querying and transformation can be combined into a single tool.

Several languages provide increasing degrees of querying and transformation capabilities:

• XPath is a language for path expressions, and is actually a building block for the remaining two query languages.

• XSLT was designed to be a transformation language, as part of the XSL style sheet system, which is used to control the formatting of XML data into HTML or other print or display languages. Although designed for formatting, XSLT can generate XML as output, and can express many interesting queries. Furthermore, it is currently the most widely available language for manipulating XML data.

• XQuery has been proposed as a standard for querying of XML data. XQuery combines features from many of the earlier proposals for querying XML, in particular the language Quilt.

##### **XQuery**

The World Wide Web Consortium (W3C) is developing XQuery, a query language for XML. The main features we cover here will not change substantially. The XQuery language derives from an XML query language called Quilt; most of the XQuery features we outline here are part of Quilt. Quilt itself includes features from earlier languages such as XPath.

Unlike XSLT, XQuery does not represent queries in XML. Instead, they appear more like SQL queries, and are organized into “FLWR” (pronounced “ﬂower”) expressions comprising four sections: for, let, where, and return. The for section gives a series of variables that range over the results of XPath expressions. When more than one variable is speciﬁed, the results include the Cartesian product of the possible values the variables can take, making the for clause similar in spirit to the from clause of an SQL query. The let clause simply allows complicated expressions to be assigned to variable names for simplicity of representation. The where section, like the SQL where clause, performs additional tests on the joined tuples from the for section. Finally, the return section allows the construction of results in XML.

* XQuery is a general purpose query language for XML data, Currently being standardized by the World Wide Web Consortium (W3C)
* XQuery is derived from the Quilt query language, which itself borrows from

SQL, XQL and XML-QL

XQuery uses a

for … let … where … order by …result …

*syntax*

for ⬄ SQL from

where ⬄ SQL where

order by ⬄ SQL order by

result ⬄ SQL select

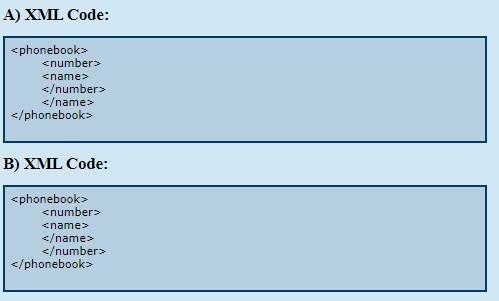
**Nested Queries**

The file contained a root element and a couple of elements that were inside the root element. When an element appears within another element, it is said that the inner element is "nested". The term nested can be related directly to the word "nest". If an element is nested within another element, then it is surrounded, protected, or encapsulated by the outer element.

***Need of nesting:***

Besides being such an easy term to understand, nesting also serves a wonderful purpose of keeping order in an XML document. Much like parentheses in a math problem, elements must be closed in the order that they are opened.

This means that an element which is nested inside another element must end itself before the outer element. Below are two example XML documents (A & B). One is properly nested and the other has a small problem.



 In example B was the properly nested document, then you are correct. All elements are closed in the order that they were opened. Those elements which were opened first are closed last.

In example A, The name element is nested within the number element, yet the number element is closed before name! Example B is what example A must look like to be a well-formed XML document.

***Nesting rules***

Nesting in plain English, with little few rules:

* Elements opened first must be closed last. That means that the root element, the first element in an XML document, must also be closed last.
* Nested elements, ones that occur in the middle of the document, must be closed before those that came before them.

***Example:***

The following query converts data from the flat structure for university information

into the nested structure used in university-1

<university-1>

{ for $d in /university/department

return <department>

{ $d/\* }

{ for $c in /university/course[dept name = $d/dept name]

return $c }

</department>

}

{ for $i in /university/instructor

return <instructor>

{ $i/\* }

{ for $c in /university/teaches[IID = $i/IID]

return $c/course id }

</instructor>

}

</university-1>

$c/\* denotes all the children of the node to which $c is bound, without the enclosing

top-level tag