Specifying and Optimising XML Views

Mark Roantree, Colm Noonan, and John Murphy

Interoperable Systems Group, Dublin City University, Ireland
{mark,cnoonan,jmurphy}@computing.dcu.ie

Abstract. Many of today's middleware solutions employ XML to resolve heterogeneities and to create an interoperable layer between sources and systems. However, XML performs poorly when queried in its native format, and local and global views are not supported by current XML products. In this paper, we support the concept of data everywhere by providing a view mechanism for XML together with a highly-optimised query processing strategy.

1 Introduction

In today's computing environment, it is often the case that database information supports a more globally connected computing infrastructure and is accessed through a variety of network architectures. This distribution or multi-sourced data requires an interoperable infrastructure, while the representation of data has changed through the emergence of XML to describe both stored and transmitted data. Subsequently, many of the solutions proposed for the necessary integration in the ubiquitous computing environment involve XML. The impact of this approach is the construction of large XML stores and often there is a need to query XML databases. The work presented in this paper supports the ubiquitous computing environment by ensuring that this canonical representation (XML) of data can be efficiently processed regardless of potentially complex structures. In this paper, we provide a metamodel to support the definition and manipulation of XML views together with a supporting optimisation strategy.

In this paper, we present a view mechanism for XML databases supported by an XML metamodel to describe both the database and view constructs. This metadata is then exploited to provide a powerful query optimisation engine to materialise views. For a more complete version of both metamodel and metadata structures, please refer to [8]. The paper is structured as follows: in §2, we provide some background, concepts and terminology used in our approach; in §3, we describe our view definitions and operators; a brief overview of the query optimisation method is presented in §4; in §5, we describe the results of our experiments while in §6 we discuss similar approaches; before concluding in §7.

2 System Terminology

We refer to the XML data tree as the *database*; the schema tree as the *schema*; and later in the paper, we describe a higher level of abstraction: the *meta-schema*.

R. Cooper and J. Kennedy (Eds.): BNCOD 2007, LNCS 4587, pp. 138–146, 2007.

XML *schemas* are tree structures containing paths and nodes, with the *root node* at the top of the tree. A tree-path that begins at the root node, continuing to some *context* node is called a *FullPath* in our model. A tree-path that begins at the root node and continues to some *leaf* node is called a *LeafPath*. This distinction is necessary because a FullPath may not be unique in an XML schema, whereas the LeafPath is unique.

Property 1. *A Schema S contains a set of FullPaths* $S = \{P_1, P_1, P_2, .., P_n\}$.

The schema is divided into levels with the root at the topmost level (level 0). Each node has 0 or more child nodes, with child nodes also having 0 or more children (with the level incrementing). As the term sub-tree is rather abstract, we use the term *Family* to refer to a context node and all its descendants.

Property 2. *A Family F is a sub-tree of a Schema S.*

Furthermore, every node within the Family F represents a sub-family f of F.

Property 3. *Where node n is a member of family F, f(n) is contained in F.*

Thus, the *Family* for the root is the entire database, with all other nodes representing sub-families.

Property 4. *Each Node N at level(x) is the context node for a Family F of connected nodes at levels (x+1),(x+2),...(x+n).*

The term *Twig* is used to refer to a Family with some members missing (a pruned Family). This is useful where a Family is very large and one wishes to reduce the size of the Family sub-tree (perhaps for querying performance reasons).

Property 5. *A Twig T is a Family F with 0 or more sub-families removed.*

An XML *View* is a set of Twigs. XML databases are instances of XML schemas. Specifically, they have one or more instances of the FullPath construct, and by extension contain Families and Twigs.

3 View Definitions

In this section, we describe the specification, storage and processing of views. If we follow traditional database theory, then a view is stored query. The XPath language maintains closure by ensuring that the output of a query on an XML document is another XML document. Thus, a View is a new (virtual) XML document comprising some elements of the base document. Furthermore, we provide flexibility in specifying the new document by allowing heterogeneous sub-trees (referred to as Twigs) form part of the new document.

3.1 Specifying Views

A View is a set of families, each with zero or more sub-families removed (clipped). Where a View contains a single Twig (a clipped family), it is referred to as a *Homogeneous View*, and where the View contains multiple Twigs, it is a *Heterogeneous View*. Thus a View comprises a `Name` and one or more `Twig Descriptions`.