International Journal of Information, Control and Computer Sciences
ISSN: 2517-9942
Vol:2, No:10, 2008

Query Algebra for Semistuctured Data

Ei Ei Myat, and Ni Lar Thein

***Abstract*—**With the tremendous growth of World Wide Web (WWW) data, there is an emerging need for effective information retrieval at the document level. Several query languages such as XML-QL, XPath, XQL, Quilt and XQuery are proposed in recent years to provide faster way of querying XML data, but they still lack of generality and efficiency. Our approach towards evolving a framework for querying semistructured documents is based on formal query algebra. Two elements are introduced in the proposed framework: first, a generic and flexible data model for logical representation of semistructured data and second, a set of operators for the manipulation of objects defined in the data model. In additional to accommodating several peculiarities of semistructured data, our model offers novel features such as bidirectional paths for navigational querying and partitions for data transformation that are not available in other proposals.

***Keywords*—**Algebra, Semistructured data, Query Algebra.

I. Introduction

BEFORE considering the techniques for manipulation of semistructured data, it is necessary to develop a formal abstraction for mapping real-world entities to logical ones. This requirement is fulfilled by a data model. A data model is the core of database system. In paper, a general tree-structured data model is used for representing semistructured documents. This data model serves as the foundation for the set of operations, the objects of manipulation and their attributes. The choice of data representation has a significant impact on the capabilities of the algebra. This data model is structurally similar to the Document Object Model (DOM) proposed by the World Wide Web Consortium (W3C) [4]. Unlike DOM, it has additional features for facilitating query operations. It is not specific to XML and it does not model all objects defined in XML. In this paper, the structural aspect of this model is introduced.

II. Market and Tag

The most common form of specifying semistructured data is using a markup format such as XML. The use of markup tags clearly delineates the logical entities of the document. Therefore, these tags can be used as metadata to define the structure of a document. This form of specification entails two types of data in a document—markup and text such as M and T respectively. Fig. 1 shows a simple markup document of world cup. In the specification, data units delimited by start- and end-tags such as $\langle \texttt{club} \rangle \cdots \langle \texttt{club} \rangle$ are called elements. Each element comprises markup tags, an optional set of attributes and content information.

```
<premiership>
    <club name="Manchester United">
        <category> Good </category>
        <player>
            <nationality> Italian </nationality>
            <position> Midfielder </position>
        </player>
        <player>
            <nationality> French </nationality>
            <position>  Defender </position>
        </player>
    </club>
    <club name="Newcastle">
        <category> Good </category>
        <category> Strong </category>
        <player>
            <nationality>  French </nationality>
            <position> Defender </position>
        </player>
    </club>
</premiership>
```

Fig. 1 A Simple Markup Document

III. Node

The objects of manipulation in proposed algebra are nodes belonging to a document tree (to be defined later). Let DN be the domain of all nodes. For any node n, $n \in DN$ and consequently for any set of nodes N, $N \subseteq DN$. Nodes are used to represent elements and scalar values. As an element may either have a scalar value or a set of subelements as its content, a node is used to represent both forms of content. There are three types of nodes namely, internal nodes, external nodes and root nodes.

A. Internal Node

An internal node represents an element that is a subelement of some other element and itself has element content. The domain of internal nodes is represented as $DN_I \subseteq DN$. The function parent(n) returns a singleton set containing the node representing the enclosing parent element for an internal node $n \in DN_I$.

B. External Node

An external node represents a non-root element that does not enclose any subelements, i.e., contain scalar data only (text content). External nodes belong to the domain $n \in DN_E$.

C. Root Node

A root node n_0 represents the first, outermost lexical element (element named "premiership" in the example of Fig. 3) of the document. A root element cannot be contained in other elements, therefore parent(n_0) = $\varnothing$. The domain of root

nodes is denoted $N_0 \subseteq DN$. Therefore, the set of node set is $N= = DN_I \cup DN_E \cup DN_0$.

IV. Node Set

A set of nodes V comprising three types of tree noses: the root node n_0, the set of internal nodes N_I and the set of external (or leaf) node N_E. Hence, for any given node in the document tree, n, $n \in V = \{n_0\} \cup N_I \cup N_E$.

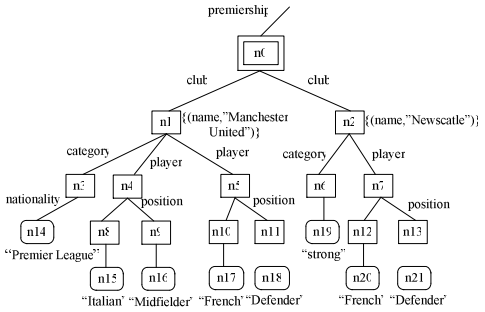

Fig. 2 Tree Representation of the Simple Markup language

V. Document

A document D may be represented by an ordered, rooted, labeled tree given as D = (V, E) where V is the set of nodes and E the set of labeled edges. This tree representation of a document is also referred to as a document tree. Every document D contains exactly one root node $n_0 \in DN_0$, a set of internal nodes $N_i \subseteq DN_I$ and a set of external nodes $N_E \subseteq DN_E$. Then, $V = \{n_0\} \cup N_I \cup N_E$.

VI. Ordering Node Set

Algebraic operators seldom deal with singular data entities. In general, they are defined on collections of primitive or complex data types. However, operands in our algebra are sets of nodes. Therefore, a node set is defined as an ordered collection of nodes, both internal and external, not necessarily from the same document tree. There are no duplicate nodes in a node set. As noted earlier, internal nodes can be compared by object equality and external nodes by value equality. Based on these notions of node equality, standard definitions of set membership $(\in)$, subset relationships $(\subseteq)$ and set equality (=) apply to node sets.

There are two notions of relative ordering among nodes in a node set as follows:

1) If nodes in a node set belong to the same document tree, their ordering in the node set is the same as the lexical order of the elements they represent.
2) If nodes in a node set belong to different documents trees, some pre-defined order among documents may be used to determine the order of nodes in the node set.

VII. Edge

The set of edges *E* in a document tree is a binary relation defined on the set $(\{n_0\} \cup N_I) \times (N_I \cup N_E)$ and is given as $E = \{\langle n_p, n_c \rangle | n_c \in \text{children}(n_p)\}$. The incoming edge to an internal node $n \in N_I$ (i.e., the edge between the parent of n and itself) is labeled by n's tag, (i.e., label(n)). External nodes of the tree, being containers of scalar values rather than representative of document structure like elements, do not have any tags. Hence, edges leading to them are unlabeled.

VIII. Tag Path

A tag path p is a sequence of labels $l_1.l_2.l_3...l_k$ of length k where $l_i \in M(1 \le i \le k)$. The symbol ϵ denotes a null path of length 0. The tag paths (or simply paths) is used to specify the reachable (target) nodes from a given (reference) node. Paths facilitate navigational querying whereby algebraic operations may be performed on nodes reachable from a given set of nodes via a certain path.

When the target nodes are descendents of the reference node(s), the path (say p) is referred to as a forward path and is denoted p. Otherwise, when p describes the location of an ancestor of the reference node; it is a reverse path and is denoted p. Where the direction of a path is not relevant or specific to the context, it shall be specified generically without the overhead arrow. Propositions in such contexts are applicable to both forward as well as reverse paths.

IX. Document Schema

Although a semistructured document is a non-rigid schema, it is useful to define loose structural descriptions using regular expression syntax as done in an XML Document Type Definition (DTD). It is possible for several differently structured documents to conform to the same structural description. An example of a tree-based graphical means of structural description is presented in Fig. 3, and called a document schema that is useful in portraying the structure of a document prior to querying. A document schema displays all the information conveyed by the markup tags (M) of a document.

A document schema is an ordered, labeled tree structured schema whose nodes correspond to element types. The edges carry labels specifying element names in terms of regular expressions.

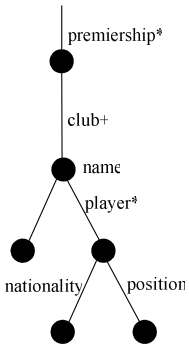

Fig. 3 Document Schema for the Document Tree of Fig. 2

X. Algebra

An algebra is defined formally in group theory as a pair (S,Ω) where S is a set of objects and Ω is a set of operators, each of which is closed with respect to S. The set of objects S is universe of node sets $P(DN)$. Hence, the objects of manipulation for the operators in Ω are node sets. Since operators that are closed with respect to $P(DN)$ accept and return node sets, it is possible to compose algebraic operators to form complex queries on node sets. A node set or a series of compositions on a node set is referred to as an algebraic expression. Operators have properties that can be applied to obtain equivalent expressions through algebraic transformations. Two expressions are equivalent if they can be applied to the same node set to obtain identical results.

TABLE I
Summary of Algebraic Operators

Operator	Description
Extend	The set of nodes reachable from node set N via path p.
Select	The subset of node set n comprising nodes that satisfy the condition $\Theta \in C$.
Product	The node set (virtual nodes) comprising all orderings among nodes in node set M and N.
Partition	A collection of node sets (partition) indexed by the node set $\Phi p(N)$.
Match	The node set of all possible matchings between partitions N and M.
Union	Set union of node sets M and N.
Intersect	Set intersection of node sets M and N.
Difference	Set difference between node sets M and N.

XI. Comparison with Other Data Model

Data models proposed elsewhere for semistructured data can be categorized into graph-structured models such as OEM [1, 2, 3] and tree-structured ones such as the XML DOM [4], all of which have been introduced earlier. In this portion, the similarities and differences between these models and the proposed data model are discussed.

A. OEM

In OEM, there is two types of node objects; atomic and complex which correspond to external and internal nodes respectively. Data is self-describing and presented by labels on edges. However, since there may be several edges leading to a node, each labeled differently, node labels need not be unique. Hence, unlike the proposed data model where node tags are unique, labels on edges may not be inferred as type descriptors of nodes connected to them.

A named set of objects (entry set) is employed like in OEM. In OEM, the named objects serve as starting points for database navigation since all objects are accessible from them. In proposed data model, the entry set coincides with the set of root nodes of document trees.

In OEM, labels serve as an exclusive form of metadata. In addition, element attributes for internal nodes are incorporated. Moreover, encoding links between internal nodes is implemented as attributes referring to the identifier of the target node. Thus, unlike OEM, links in the proposed model are implicit and differ from explicit edges that signify parent-child connections.

Finally, as in OEM, paths are defined to facilitate the specification of reachable nodes from a set of reference nodes in query contexts. However, paths in OEM are always unidirectional. Through reverse paths, the ancestors of nodes are allowed to be located and manipulated as easily as their descendents. Moreover, the notion of ordering is defined among nodes in node sets.

B. XML DOM

The XML Document Object Model (DOM) is most closely resembles our data model. However, there is a number of data storage units defined in XML that are not represented in the proposed model for simplicity. In particular, only equivalents of the data types Document, Element, Attribute and Character Data are represented. Additionally, new data objects such as paths and node sets are introduced that facilitate query operations of the algebra.

While DOM is being specific to XML, models features peculiar to the latter, the proposed model is a general purpose data model and not geared towards any particular form of syntactic specification. Instead, the proposed model aspects the characteristic of semistructured data (including XML) such as elements, markup, and so on. Therefore, several XML features such as Entities, Processing Instructions and Comments are excluded from the scope of this model.

XML provides for an optional Document Type Definition (DTD) for the specification of document structure as a regular grammar. But the proposed model treats the existence of a schema (document schema) as a descriptive aid for the user in formulating queries. It does not serve any purpose in evaluating queries or deciding their satisfiability, as in structured database systems.

This model adopts the XML approach of representing cross links as attribute references. However, XML provides the built-in attribute types ID and IDREF. XML permits entity references within attributes values. But, this model assumes attribute values to be atomic. XML provides a variety of options for setting attribute types and defaults. Unlike XML, the proposed model disallows mixed element content [5]. As mentioned earlier, internal nodes may have child node sets comprising internal or external nodes but not both. Mixed element content introduces structural inconsistencies in the tree representation, making it difficult to model.

XII. Conclusion

With an increasing amount of information which are stored, exchanged, retrieved and presented using XML data, XML query language become necessary to provide quick and efficient data retrieval from various sources. The data model proposed emphasizes on generality in representing semistructured data. The data model presented here offers simplicity in the form of a tree representation of documents broadly covers the features characteristic ordering. In this aspect, the most of the characteristics of semistructured data is modeled and yet maintains flexibility by ignoring format specific conventions. Moreover, the proposed query algebra achieves several improvements over available query models for semistructured data.

Acknowledgment

The author also would like to express her acknowledge to people who help directly or indirectly for this paper. And then, the author would like to thank her parents. Their understanding, love and care give unwavering confidence in her. Next, the author is deeply grateful to her sisters and brother for their support, encouragement to attain her destination without any trouble throughout her life, the strength and grace their provided her throughout the entire program.

References

[1] S. Abiteboul, D. Quass, J. Mchugh, J. Widom, J. Wiener. The Lorel Query Language for Semistructured Data. Journal of Digital Libraries, l(l):68-88, April 1997..

[2] J. Mchugh, S. Abiteboul, R. Goldman, D. Quass, J. Widom. Lore: A Database Management System for Semistructured Data. SIGMOD Record, 26(3):54-66, September 1997.H. Poor, *An Introduction to Signal Detection and Estimation*. New York: Springer-Verlag, 1985, ch. 4.

[3] D. Quass, A. Rajaraman, Y. Sagiv, J. Ullman, J. Widom. Querying Semistructured Heterogeneous Information. Journal of Systems Integration, pp. 7(3/4):381-407, September 1997.

[4] World Wide Web Consortium. Document Object Model (DOM) Level 1 Specification. http://www. w3. org/TR/REC-DOM-Level-1.

[5] World Wide Web Consortium. Extensible Markup Language (XML) 1.0, 1998. http://www. w3. org/TR/REC-xml.

[6] vidmar