Association for Information Systems
AIS Electronic Library (AISeL)

ECIS 2003 Proceedings

European Conference on Information Systems (ECIS)

2003

Hybrid XML Data Model Architecture for Efficient Document Management

Eun-Young Kim
Ansan College of Technology, kay@ansantc.ac.kr

Jin-Ho Choi
Korea Advanced Institute of Science & Technology, jhchoi@kgsm.kaist.ac.kr

Jhung-Soo Hong
Korea Advanced Institute of Science & Technology, wilco@sktelecom.com

Tae-Hun Kim
Kyungsung University, kdbdc@star.kyungsung.ac.kr

Se-Hak Chun
Hallym University, shchun@hallym.ac.kr

Hybrid XML Data Model Architecture for Efficient Document Management

Eun-Young Kim
Ansan College of Technology
671 Chosi-dong, Ansan City, Kyunggi-do, 425-792, Korea
Phone : 82-31-490-890
key@ansantc.ac.kr

Jin-Ho Choi
Korea Advanced Institute of Science and Technology, 207-43
Cheongryangri-dong, Dongdaemoon-gu, Seoul 130-012, Korea
Phone : 82-02-958-3682
jhchoi@kgsm.kaist.ac.kr

Jhung-Soo Hong
SK Telecom, Serin-dong 99, Jongro-gu, Seoul 110-110
Korea
Phone : 82-02-2121-4183
wilco@sktelecom.com

Tae-Hun Kim
Department of MIS, Kyungsung University
110-1, Daeyeon-dong, Nam-gu, Pusan, 608-736, Korea
Phone : 82-51-620-4456, Fax : 82-51-625-4536
kdbdc@star.kyungsung.ac.kr

Se-Hak Chun
Hallym University, 1 Okchun-dong, Chunchun City, Kangwon-do, 200-702
Korea
Phone : 82-33-240-1837 Fax : 82-33-256-3424
shchun@hallym.ac.kr

Abstract

XML has been known as a document standard in representation and exchange of data on the Internet, and is also used as a standard language for the search and reuse of scattered documents on the Internet. The issues related to XML are how to model data on effective and efficient management of semi-structured data and how to actually store the modeled data when implementing a XML contents management system. Previous researches on XML have limitations in (1) reproduction of XML documents from the stored data, (2) retrieval of XML sub-graph from search, (3) supporting only top-down search, not full-search, and (4) dependency of data structure on XML documents. The purpose of this paper is to present a hybrid XML data model architecture for the storage and search of XML document data. By representing both data and structure views of XML documents, this new XML data model technique overcomes the limitations of previous researches on data model for XML documents as well as the existing database systems such as relational and object-oriented data model.

Keywords

Generalized Object Model, XML Document Management, Relational Database, Objective Database, Multi-format information retrieval

1. Introduction

XML (eXtensible Markup Language) based on SGML(Standard Generalized Markup Language) is a simple and flexible markup language. It provides a method for finding information which users want and XML's abundant data representative method enables users to do business on the web by making intellectual mechanism. Therefore, it has received a lot of attention and it has been utilized on almost all fields on the web such as on-line banking, push technology, search engine, web based control system and agent and so on. In addition, application areas are expanding rapidly, and XML documents are made possible to be reused from returned XML documents through the search engine, another existing XML documents, or transmitted XML documents users can draw out data that they want and process their own data structure and store it.

Approaches to manage XML documents data can be classified into using new database system for XML documents and using existing database system such as relational database system. Using new database system for XML documents is based on a new data model that represents semi-structured XML data. Also, when users take the existing relational database system or the object system in consideration, mapping model that fits in a relational database system or an object system is required.

Individual data structure was defined according to the XML documents type in the previous researches on data model of XML documents. This poses some problems because the new data structure should be defined for XML documents that have new structures and data; it is difficult to expand the domain and the data structure that needs to be defined in a different way every time according to the object to apply and top-down access is the only available way to search for information.

Both the data and structure view of XML's original documents are lost in the data model to map the existing relational database system or object system, so XML documents cannot be generated from the stored data. The range of the application can be expanded easily when the data model for XML documents has a unique data structure regardless of data and structures of XML documents. Also, in terms of searching for information on XML documents, bottom-up access and left-right access should be possible as well as top-down access. The search should also be possible without previous knowledge on the XML document structure.

In this paper, we suggest a data model that supports all these requirements and is applicable to new database system and existing relational database system for the XML document and represents the data and the structure view of the XML document.

2. Motivation and Research Background

2.1 Motivation of the Research

An XML document describes data and structures but it cannot be regarded as a database system simply because it describes data. Even though it contains data, it is just a general text file that cannot perform any function without additional software that manages data.

However, once an XML document, a related XML tool and XML's various functions are united, it can be regarded as a database system because it contains storage, schema, query language, programming interface (factors of database system) and so forth. But, it cannot support effective storage, index, security, transaction, data perfection, multi-user access, trigger, and query on multiple documents that the existing database system supports. Therefore, if the amount of data is not large and the number of users are small and the circumstances require just general capabilities, the XML document can be used as a database, but if there are many users and the circumstances require data perfection and advanced capabilities, the XML document can be used as a database. In this case, the database that stores the XML data and structure is necessary. At present, there are two ways to store data of the XML document; one is to store it as a new data model in the new database system only for the XML document and another is to store the data by mapping the data of the XML document into an existing relational or object system.

The new database system only for the XML document requires a new model that can store, represent and query data of the XML document effectively, while the data model for the existing relational database system or object system requires a data model that coincides with these database system models naturally.

2.2 Research Background

The representative data models considering the new database system for the XML document are Stanford University's extended OEM(Object Exchange Model) model (Cluet 1997) for the XML document and Pennsylvania University's Edge Labeled Graph model (Buneman&Davidson&Hillebrand&Suciu 1996) and AT&T's List model (Fernndez&Simon&Suciu&Wadler 1999).

The model which is an extended OEM model (Goldman&McHugh&Widom 1999) of Lore system to represent the XML document data represents the XML element as (eid, value); eid is a unique ID of the XML element and value is a text string which is an atomic value or a complicated value including an XML tag, an attribute list, a reference element list and so on. The basic information of the Edge Labeled Graph model (Buneman et al. 1996) is represented by the label and its function of encoding relational database is powerful. Data types available as label are String, Integer, Real and these types are used as value in relation database.

List model (Fernndez et al. 1999) is based on the data model for XSLT (Wadler 1999), added a reference node and combined attribute and element node. A basic type of the model is node and each node is text, element or reference. The attribute is added to the letter at(@)in front of the name and attribute is treated like element and child is represented as a list.

In the X-Ray (Kapperl&Kapsammer&Retschitzegger 2000), a relational mapping model, DTD(Document Type Definition) of XML is been mapped as a relational data base system. The element of DTD is been mapped as a relation of relational database system or as an attribute of relation according to element type, the number of repetition and whether an attribute is contained or not. The attribute of DTD is been mapped as the attribute of relation in the relational database system. In Inlining technique (Shanmugasundaram&Gang&Tufte& Zhang&DeWitt&Naughton 1999), one of the models for a relational system, one element is represented in one relation with the child element containing one element and the element of DTD corresponds to entity or the attribute of an ER-Model.

3. Generalized Object Model (GOM)

3.1 Overview of Proposed Hybrid XML Data Model

This paper proposes a data model that is needed when a new XML-native database system for only XML is designed and is applicable to an existing relational database system. Based on the basic structure of Lore's XML Data Model and Edge Labeled Graph Model, proposed hybrid xml data model is renewed and extended to support requirements shown below.

1) Document in global domain is the object.

2) Data model doesn't depend on the type of database system chosen.

3) Both structure and data of the document are represented.

4) Structure of the document and data change is applicable flexibly.

5) Mapping document from data model reversely, all element orders of document are preserved.

6) Top-down, bottom-up, left-right, right-left search about query should be possible.

3.2 Basic Model of XML

It is the most natural to represent the XML document as a graph because the element structure of the XML document can be represented as a tree and mutual reference among elements is added. When representing a basic structure of the XML document data as a graph, each element of the XML document covers a graph that takes root on the node having its own tag name. Therefore, another element included to element is represented as a sub-graph in the element graph containing itself. Also, each node is linked by edges and has level or represents reference meanings.

In the basic model, the XML document is represented as a graph that all nodes and edges are labeled and that is ordered among nodes and edges and that edges are directed on. When the XML document is represented as a graph $G=\{V,E,A\}$ where V is a vertex, i.e. set of nodes, E, a set of edges, and A, a set of attributes defined on start tag of element. An element in the XML document means all contents from the equivalent start tag to finish tag are related to the start tag, and each element of the XML document is represented as a sub-tree/graph that takes the tag of an equivalent element as a root node. The label of node V is a tag value in the case of a middle node and it corresponds with the value (contains NULL and EMPTY) in the case of a leaf node. The label of the edge Ei from node Vi corresponds with relation between node Vi and arriving child node Vj. If child node Vj is another element, it is labeled as

CHILD that indicates child element information of node *Vi* and if the child node *Vj* is a value of element, it is labeled as VALUE that manifests *Vi* value information. Thus, node set *V* and edge *E* on the graph can be represented as:

$$V = V_{tag} \cup V_{value}$$

$$E = E_{child} \cup E_{value}$$

3.2.1. Basic Attribute

(1) The basic attribute of node

V(G) is a node-set on the graph when *v* is satisfied with $v \in V(G)$.

- label(v) : A tag name is returned in the case of the root or middle node, and the value between tags is returned in the case of a leaf node. NULL is returned when there is no value between a start tag and a finish tag to distinguish two cases of element that have emptiness. EMPTY is returned when element is stated in a tag.
- level(v) : The root node of the document starts at 1 and has the hierarchy level of the equivalent node on the graph of all documents. The node that has different parents can have the same level.

(2) The basic attribute of attribute set

A(G) is all attribute sets in the document and value when attribute *a* satisfies $a \in A(G)$.

- name(a): returns the attribute name of attribute a.
- value(a): returns the value of attribute a.
- node(a): returns the node which attribute a is defined.
- type(a): returns the type of attribute a.

(3) The basic attribute of edge

E(G) is edge set on the graph and the basic attribute of edge *e* is below when satisfied $e \in E(G)$.

- *from(e)* : returns the node where edge *e* starts.
- *to(e)* : returns the node where edge *e* arrives.
- *relation(e)*: returns the relation (*CHILD* or *VALUE*) of the starting node and arriving node of edge *e*.
- *order(e)* : returns the order among edges that depart from the same parent node.

3.2.2. Additional Attribute

(1) The additional attribute of node

V(G) is a node-set on the graph and the additional attributes of node *v* that is derived from the basic attribute of node and edge *e* is below when $v \in V(G)$.

- *child(v)* : returns the child node of node *v*

  child(v) ≡ {to(e)|□ e□ E:from(e)=v}

- *parent(v)* : returns the parent node of node *v*

parent(v)≡ {from(e)|□ e□ E:to(e)=v}

- *ancestor(v); returns the set of ancestor node of node v*

  while ({e|□ e□ E(G):to(e)=v}≠ {}) {

  ancestorlist := ancestorlist□ from(e) ;

  v:=from(e);

  } end while;

  return ancestorlist;

- *descendent(v)* : returns the set of descendent node of node *v*

  void descendent(node v) {

  for (each {e|□ e□ E(G):from(e)=v})

  node u := to(e);

  descendentlist := descendentlist □ u;

  descendent(u) ; } }

- *root(v)* : returns the root node of node *v*

  while ({e|□e□ E(G):to(e)=v}≠ {}) {

  rootnode :=from(e);

  v:=from(e);

  } end while;

  return rootnode;

- *sibling(v)* : returns the sibling node set that has the same parent node as node *v*

  sibling(v)≡{to(e)|□e□ E(G): from(e)= {from(e)|□e□ E(G):to(e)=v} } - v

  ≡ {to(e)|□ e□ E(G): from(e)= parent(v) } - v

- *identity(v)* : return the sibling node that has the same label as node *v* among sibling nodes that have the same parent node

  identity(v)≡{to(e)|□ e□ E(G):

  from(e)={from(e)|□ e□ E(G):to(e)=v}□label(to(e))=label(v)}- v

The additional attribute of node v derived from basic attributes of attribute *a(a□ A(G)) is as follows:*

- *attributeS(v)* : returns the set of attribute defined at the start tag and the attribute value in the case of root or middle node of node *v*.

  attributeS(v)≡ {name(a), value(a) | □a□ A(G): node(a)=v}

- *@att_name(v)* : returns the attribute value that has a certain name defined at node *v*.

  @att_name(v)≡ {value(a) |□ a□ attributeS(v) : name(a)="att_name"}

(2) Additonal attribute of edge

The additional attribute of edge *e* derived from basic attribute of node *v* and edge *e* is as follows:

- *successor(e)/predecessor(e)* : returns the predecessor or successor of edge *e* among edges that have the same departure node as edge *e*.

  $successor(e) \equiv \{e_2 | \square\ e_2 \square\ E(G): from(e_2) = from(e) \square\ order(e_2) > order(e)\}$

  $predecessor(e) \equiv \{e_2 | \square\ e_2 \square\ E(G): from(e_2) = from(e) \square\ order(e_2) < order(e)\}$

- *after(e)/before(e)* : returns edges after or before edge *e* among edges that have the same departure node as edge *e*.

  $after(e) \equiv \{e_2 | \square\ e_2 \square\ E(G): from(e_2) = from(e) \square\ (order(e_2) = order(e)+1)\}$

  $before(e) \equiv \{e_2 | \square\ e_2 \square\ E(G): from(e_2) = from(e) \square\ (order(e_2) = order(e)-1)\}$

- *first(e)/last(e)* : returns the first or the last edge among edges that have the same departure node as edge *e*.

  $first(e) \equiv \{e_2 | \square\ e_2 \square\ E(G): from(e_2) = from(e) \square\ order(e_2) = 1\}$

  $last(e) \equiv \{e_2 | \square e_2 \square\ E(G): from(e_2) = from(e) \square\ (after\ e_2) = \{\})\}$

(3) Second additional attribute of node

The second additional attribute of node derived from additional attribute of edge $e(e \square\ E(G))$ is as follows:

- *next(v)/previous(v)*: returns next or previous nodes among sibling nodes of node *v*.

  $next(v) \equiv \{ to(after(e)) | \square e \square\ E(G) : to(e)=v\}$

  $previous(v) \equiv \{to(before(e)) | \square\ e \square\ E(G) : to(e)=v\}$

- *older(v)/younger(v)*: returns the node set of successor or predecessor among sibling nodes of node *v*.

  $older(v) \equiv \{to(predecessor(e)) | \square e \square E(G) : to(e)=v\}$

  $younger(v) \equiv \{to(successor(e)) | \square\ e \square\ E(G) : to(e)=v\}$

- *oldest(v)/youngest(v)* : returns the first or the last node among sibling nodes of node *v*.

  $oldest(v) \equiv \{to(first(e)) | \square e \square\ E(G) : to(e)=v\}$

  $youngest(v) \equiv \{to(last(e)) | \square\ e \square\ E(G) : to(e)=v\}$

3.3 Generalized Object Model Architecture

3.3.1. Extended Model considering DTD

DTD means a mutual agreement on the XML document transmitted when XML documents are exchanged. When accompanied by DTD, the XML document can get the reference information between elements from the data of aDTD document. The XML document is extended like below if all the reference edge sets that are added on the graph of the XML document are *R(G)*.

$$G=(V, E, R, A)$$

(1) Basic attribute of reference edge

R(G) is a reference edge set on the graph and when satisfied $r \square\ R(G)$, basic attributes of the reference edge *r* are as follows:

- *refTo(r)* : returns the node to which reference edge arrives.
- *refFrom(r)* : returns the node from which reference edge starts.
- *retAttr(r)*: returns the attribute name of reference element referring to another element.

(2) Additional attribute of node

Additional attributes of node *v(v□ V(G))* derived from the basic attribute of reference Edge *r(r□ R(G))* are as follows:

- *references(v)* : returns the node set which reference set started from node *v* arrives

  references(v) ≡ *{refTo(r)| □ r □ R(G) : refFrom(r)=v }*
- *referredBy(v)* : returns the node set which reference set arrived from node *v* starts

  referredBy(v) ≡ *{refFrom(r)| □ r□ R(G) : refTo(r)=v }*

3.3.2. Four Objects of Generalized Object Model Architecture

In the hybrid data model, XML documents are represented as a graph. The graph consists of nodes, edges, attributes, reference edges, etc. and each component has its own inherent attributes as an object. When tuples are represented as attributes that contain V, E, A, R, each component object of a graph G, they are:

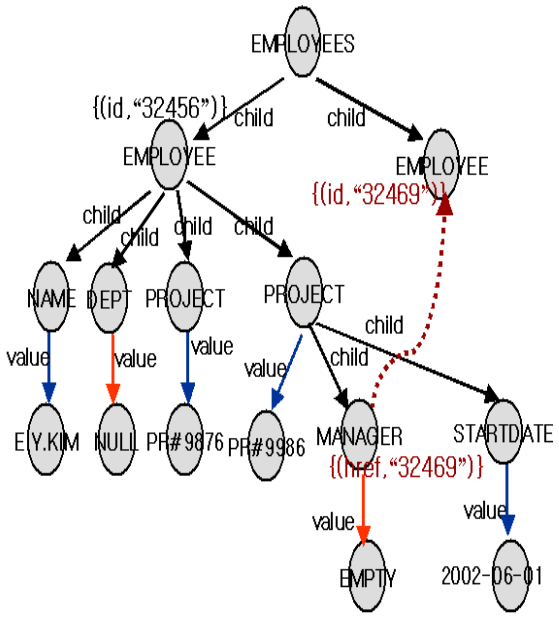
Vertex Object =(void, label, level)

Edge Object = (from, to, relation, order)

Attribute Object = (node, name, value, type)

Reference Edge Object =(refFrom, refTo, refAttr)

This tuple structure offers a unique structure regardless of the data and structure of the XML document. [Figure 1] is a representation of the data model of the XML document



(A) (B)

[Figure 1] (A) XML document, (B) XML document represented as a graph of the data model.

4. Characteristics of Generalized Object Model in Search Functions.

Search for the XML document in the hybrid data model of this paper is possible by the condition of structure information and data information, and it is possible without previous knowledge of the precise data structure.

4.1 Retrieval by Data Information

A search on the XML document containing certain data scans label of vertex object, and the search on the value of the definite attribute scans name and value of attribute object. A search on the XML document containing element that has a tag name *A* defined the value of *AT* attribute as VAL scans the area of the name and value of attribute object and seeks corresponding node information. Afterwards, by using this information, it searches matched vertex object and confirms whether the label is A or not. A search for the document where element value that has tag name *A* satisfies AVAL first scans the label of vertex object, finds Object *A* and maps this information with 'from' of edge object and scans 'to' area. With this information, it scans the label of vertex object again and makes sure whether it is AVAL or not.

4.2 Search by Structure Information

A search for element that has a definite hierarchy structure scans the label of the vertex object and 'from' and 'to' of edge object and search for the element that is in the definite position at the same level scans *from, to*, and the order of edge object. Also, a search for element that refers to a certain element scans *refFrom and refTo* of the Reference edge object.

```
<EMPLOYEES>
 <EMPLOYEE id="32456">
   <NAME> E.Y. KIM </NAME>
   <DEPT></DEPT>
  <PROJECT>PR#9876</PROJECT>
   <PROJECT>PR#9986
       <MANAGER href="32469"/>
       <STARTDATE>2002-06-01 </STARTDATE>
    </PROJECT>
  </EMPLOYEE>
  <EMPLOYEE id="32469">
   ...
  </EMPLOYEE>
```

When the element named *A* tag searches all elements named *B* tag, a k^{th} child bottom-up scan is effective. First, it searches Object *A* by scanning the label of vertex object and then it maps a found object with 'to' of edge object and finds k^{th} parents node by using 'from' data. Also, it matches vertex object by using the found 'from' data and confirms whether the label of found object is *B* or not. This kind of search method is applicable to the case where element named *B* tag searches all the element named *A* tag of k^{th} parent.

If element named *B* tag searches all the element named *A* tag as a descendent or if element named *B* tag searches all the element having *A* tag name as ancestor, it replaces the operation finding k^{th} parents node or k^{th} child node in the former process with recurrently summoning operation meeting a root node or finding a child node.

When element named *R* tag name searches all elements named *L* tag, k^{th} right sibling, it is accessed as left-right scan. First, it scans the label of vertex object and searches *L* object, maps the searched object with 'to' of edge object and stores order value, searches edge object by using *from* data and the condition of *saved order value* + *k*. Also, from the 'to' information of the found object, it checks whether the label of searched Object is *R* or not by matching vertex object.

5. Implications

The users see XML documents as a set of one or more objects. These objects can be vertex, edge, attribute, or reference edge objects. Each vertex object has a "label", which users can look at to perceive the hierarchy structure of XML documents. Each vertex can also contain links (called "Edge Object") to other attribute object in the system; the relationships between attribute objects can be displayed as connections (called “Reference Edge Object”). Under certain complex conditions, the user can expand and discern a data model for XML documents.

XML has been employed as a critical means for exchange enterprises’ information. The design objectives of XML documents must be clearly and easily understood at the outset and must be defined in terms of the business requirements. For those purposes, the techniques for modeling structures and data of XML documents have been pointed out (e.g., a graph of a data model). The most important reason to build a database of XML documents is to improve the usability of XML data from the user's perspective.

In spite of this usefulness, the hybrid XML data model has a weakness related to automatic problems. For higher efficiency of the data model, it is necessary to implement the automatic capability for obtaining XML documents in two ways: one is to make an automatic coordination possible with a CASE (Computer-Aided Software Engineering) tool which supports development of the graph of the hybrid XML data model and the other is to make possible automatic storages and reconstruction of XML documents.

6. Conclusion

This paper propose the new data model for the XML document that is suggested as a document standard to represent the data on the Internet and to exchange data mutually. The hybrid data model in this paper is an applicable model in the case of designing a new database system for the XML document and in the case of using an existing database system like a relational database system. The Limitations of this study are as follows: (1) it is not applied to a real-life case in order to illustrate its practical usefulness; (2) we suggest only a data model, not the phases of data modeling.

Because the data view and the structure view of the original XML document are lost in the data model for mapping to the existing relational database system or the object-oriented system, not only can't XML generate again from the stored data, but also XML sub-graph corresponding to element as search result can't be returned. However, in the proposed model, this problem is solved because the data view and structure view of the original XML document are stored. Therefore, since the proposed model can form a new XML document from the search result of the XML document, it supports reusability as another great feature.

The search about the XML document in the proposed model is possible on the condition of all the factors in the XML document, and not only top-down search but also bottom-up search is possible. Besides, as it has the order information between child elements that have same parents, left-right search is possible and the proposed model preserves order of same level as well as hierarchy structure between the elements of the XML document.

For a future study, we expect to do research on testing valuation by mapping the hybrid data model to a relational and an object-oriented system. We also expect to apply this modeling technique to XML based RFP(Request For Proposal) or RFQ(Request For Quotation) in business to business electronic commerce and other contents management system (CMS). Especially, the hybrid data model can be applicable to the ETL (Extracting, Transformation, and Loading) (King 2000) in a data warehouse. Data ETL is the process of changing the structure of cleaning, and reformatting the content of the operational database to load it into the data warehouse or a data mart. Some ETL product is using XML data model as the vehicle of an effective transformation process that verifies and improves the quality and value of the information and structures it for the data warehouse (e.g., Data Junction®). Yet, legacy products have limitations of past studies.

Also, the technique of the Hybrid XML data model can be applicable to HL 7 (Health Level 7) (van Bemmel et. al. 1997) message protoco which is similar to EDIFACT (Electronic Data Interchange for Administration, Commerce and Transport). The HL7 message can be used to exchange data with a laboratory. Segments of which a HL7 message is separated into a number contain a number of data fields. However, there is no consideration into the data management and the data retrieval.

7. References

A. Bonifati and S. Ceri (2000), "Comparative Analysis of Five XML Query Languages," ACM SIGMOD Record, 1(29).

Fernndez M., Simon J., Suciu C., and Wadler P (1999),□"A Data Model and Algebra for XML Query",□ Draft Manuscript Available at http://www.cs.bell-labs.com/~wadler/topics/xml.html#algebra

Gerti Kappel, Elisabeth Kapsammer, and Wemer Retschitzegger (2000), "X-Ray-Towards Intergrating XML and Relational Database Systems□, Technical Report," July, 2000.

Gyssens M., Paredaens J., Van der Bussche J., and Van Gucht D.(1994), "A Graph-oriented Object Database Model," IEEE Trans. on Knowledge and Data Eng., 6(4), pp. 572-586.

Jayavel Shanmugasundaram, H. Gang, Kristin Tufte, Chun Zhang, David J. DeWitt, and Jeffrey F. Naughton (1999), "Relational databases for querying XML documents: Limitations and opportunities," In VLDB'99, In Proc. of 25th International Conference on Very Large Data Bases, Edinburgh, Scotland, pp. 302-304.

King, E.(2000), Data Warehousing and Data Mining: Implementing Strategic Knowledge Management, Computer Technology Research Corp.

Peter Buneman, Susan Davidson, Gerd Hillebrand, and Dan Suciu (1996), □"A query language and optimization techniques for unstructured data," In Proceedings of ACMSIGMOD International Conference on Management of Data, pp. 505-516, Montreal, Canada, June 1996.

Peter Buneman, Susan Davidson, and Dan Suciu (1995), "Programming constructs for unstructured data," In Proceedings of 5th International Workshop on Database Programming Languages, Gubbio, Italy, September 1995.

P. Wadler (1999), "A formal semantics of patterns in XSLT," Markup Technologies, December 16, 1999.

R. Goldman, J. McHugh, and J. Widom (1999), □"From Semistructured Data to XML: Migrating the Lore Data Model and Query Language□," Proceedings of the 2nd International Workshop on the Web and Databases (WebDB '99), pp. 25-30, Philadelphia, Pennsylvania, June 1999.

S. Chawathe, S. Abiteboul, and J. Widom (1999), "Managing historical semistructured data," Theory and Practice of Object Systems, 5(3), pp. 143-162, August 1999.

S. Cluet (1997), "Modeling and querying semi-structured data," In International Summer School on Information Extraction (SCIE), volume 1299 of Lecture Notes in Computer Science, pp. 192-213, Springer-Verlag.

T. Shimura, M. Yoshikawa, and S. Uemura(1999). "Storage and Retrieval of XML Documents Using Object-Relational Databases," In Database and Expert Systems Applications, pp. 206-217, Springer.

Van Bemmel, Musen., M.A., Helder., J.C.(1997), Handbook of Medical Informatics., Springer.