XML Repository System Using DBMS and IRS

Hyung-Il Kang
Dept. of Information and Communication Eng.,
Juseong College, Cheongwon, Korea

Jae Soo Yoo
Dept. of Computer and Communication Eng.,
Chungbuk National University, Cheongju, Korea

Byoung Yup Lee*
Dept. of Electronic Commerce,
PaiChai University, Daejeon, Korea

ABSTRACT

In this paper, we design and implement a XML Repository System(XRS) that exploits the advantages of DBMSs and IRSs. Our scheme uses BRS to support full text indexing and content-based queries efficiently, and ORACLE to store XML documents, multimedia data, DTD and structure information. We design databases to manage XML documents including audio, video, images as well as text. We employ the non-composition model when storing XML documents into ORACLE. We represent structured information as ETID(Element Type Id), SORD(Sibling ORDer) and SSORD(Same Sibling ORDer). ETID is a unique value assigned to each element of DTD. SORD and SSORD represent an order information between sibling nodes and an order information among the sibling nodes with the same element respectively. In order to show superiority of our XRS, we perform various experiments in terms of the document loading time, document extracting time and contents retrieval time. It is shown through experiments that our XRS outperforms the existing XML document management systems. We also show that it supports various types of queries through performance experiments.

Keywords: *XML Repository System, Database Management System, Electronic Commerce, IRS.*

1. INTRODUCTION

Recently, internet has significantly changed the concept of document preparation and distribution. Many documents tend to be produced as structured ones using markup languages like SGML or XML so that they are readable without any formatting information and easily interchangeable between platforms. They are called structured documents. XML that is a subset of SGML has been proposed by W3C as a new markup language that supports user-defined tags, and encourages the separation of document content from presentation[6]. XML is a meta language that allows the user to define a language for composing structured documents. It can automate web information processing and in particular is suitable for data exchange and interoperability. Currently, the research on XML proceeds in various domain such as digital library, EDI(Electronic Data Interchange), EC(Electronic Commerce), CALS(Commerce At Light Speed), MathML(Mathematics Markup Language), and so on. The main emphasis of this paper is on the XML. Because of the proliferation of XML documents, administering very large XML documents with logical structured information is becoming more and more important[8-9].

The XML Repository system(XRS)s should be designed with considering the following requirements[3][13][17]. First, since XML documents contain contents as well as structured information, XRS needs to represent the structured information efficiently. Second, XML documents may contain various types of multimedia data, so it must be capable of storing the multimedia data and managing them. Finally, queries in XRS are much more complex and diverse than those of other document management systems. That is, XRS must support a wide range of queries such as content-based queries and structure-attribute queries at all levels of the document hierarchy. Most of the existing XRSs are implemented based on DBMS(Database Management System). Even though they achieve good performance in storing and managing data, they show poor performance in content based search and full text

* *Corresponding author. E-mail : bylee@pcu.ac.kr*
Manuscript received Sep. 11, 2007 ; accepted Sep. 27, 2007

search compared to IRS(Information Retrieval System)[4]. In that reason, existing XRSs based on DBMS have limitation to satisfy the requirements mentioned above.

In this paper, we design and implement a XRS that exploits the advantages of DBMSs and IRSs. It uses BRS to support full text indexing and content-based queries efficiently and ORACLE to store XML documents, structured information, DTD(Document Type Definition), and multimedia. We employ the non-composition model when storing XML documents into ORACLE to solve the above problems. Well-organized logical structured information of XML documents enables XRSs to process the various queries efficiently. We define ETID(Element ID), SORD(Sibling ORDer) and SSORD(Same Sibling ORDer) for XRSs to represent the logical structured information of XML documents. The implemented XRSs efficiently process the structure and attribute queries as well as content-based queries.

The paper is organized as follows. In the section 2, we describe the related works. Section 3 explains designed structured information of XML documents. In section 4, we describe the XML data modeling. In section 5, we show the overall architecture of proposed XRS and explain the components in detail. In section 6 we do evaluate the performance of and finally in section 7, we make a conclusion.

2. RELATED WORKS

There are three possible approaches to construct XRS systems. The first approach is to build a special-purpose database system that only manages XML documents such as Lore and DocBase[2][10]. This kind of system is particularly tailored to store and retrieve structured documents, but it will take long time that such systems become mature and scale well for large amounts of data. The second approach is to build XRS based on object-oriented database system(OODBMS)[13, 15, 16]. This approach may be the best since the object-oriented data modeling technique can accommodate the hierarchical characteristics of structured information. However, OODBMSs are not yet mature enough to handle very large amount of database. The last approach is to build XRS based on RDBMS(relational database system)[1][5][11]. Many researches on building XRSs based on RDBMSs are being worked on because RDBMSs are mature and scalable very well and the most prevalent DBMSs over the world. In the non-composition based XRSs, however, XML documents are split into more than two tables so join operation that is one of the most expensive operations in RDBMS are needed to present whole XML documents.

Additionally, the second and third approaches have the following common properties. XRSs based on DBMS cannot support full text indexing unless additional modules that support full text indexing are added on. Consequently they cannot process content-based queries more efficiently compared to IRSs. In that reason, recently coupling based XRSs that exploit DBMS and IRS have been major research issues in XML communities. Yan [20] have integrated structured-text retrieval system(TextMachine) into a object-oriented system(OpenODB). Documents are stored in the TextMachine. It also manages index and retrieves documents. Structured information is stored in the database system. In this system, the user can retrieve components of documents and browse documents based on their structure using four functions : up, down, prev, next. But, a basic assumption is that text objects may not be modified. Also, Volz[14] have coupled the IRS INQUERY with the OODBMS VODAC. The coupling consists of OODBMS classes encapsulating the IRS functionality. They proposed an indexing to avoid the duplication of index information by modeling the elements of SGML documents using object-oriented techniques. But, it has a burden to calculate similarity between a user query and an element and it should record the location of keywords appeared in an element.

3. STRUCTURED INFORMATION OF XML DOCUMENTS

Existing methods to represent structured information of XML documents cannot access directly the specific element in documents. They need complex operations to access the sibling, parent or child elements. In this paper, we propose a method to represent the structured information that solves the above problems. We define the ETID(Element Type ID), SORD(Sibling ORDer) and SSORD(Same Sibling ORDer) to represent the structured information efficiently. ETID is a unique value assigned to each element of DTD. In order to assign ETID to the elements, we construct a tree structure with the relationship of elements in DTD. ETIDs have the UNIX directory-like appearance, for example '/' means root element, '/1'and '/2' mean the children of the root. Figure 1 shows an example of construction of a tree structure with a sample DTD and assignment of ETID to each element in the tree.

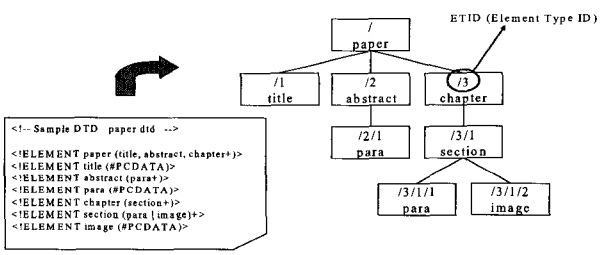


Figure 1. Assignment of ETID

SORD is order information between sibling nodes on the same level while SSORD is order information between the same sibling nodes on the same level. A SORD is a unique value in a XML document and it is used to search a specific element. SORDs are represented by the same way of ETID and it also has order information between parent elements. The order information is very helpful to deal with the following example queries : "Find documents where the third child element of chapter element is section" or "Find documents where the third child element of chapter element includes a word 'XML'." Therefore we can search a specific element

with ETID, SORD, and SSORD easily. Figure 2 shows an example that presents the structured information of a XML document as a tree with ETID, SORD and SSORD. For example, the second child element of a chapter whose SORD is '/3' has two child elements whose name is para. The ETIDs of the two 'para' are same but their SORD are '/3/2/1' and '/3/2/3' and their SSORD are '/1/2/1' and '/1/2/3' respectively.

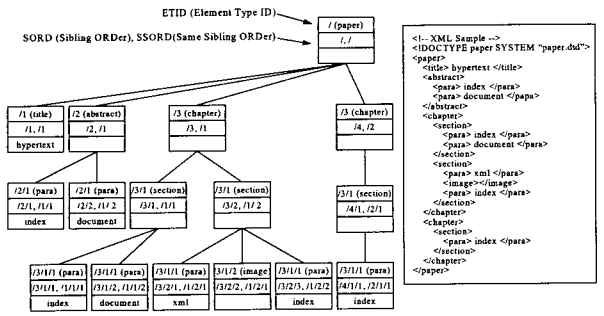


Figure 2. Structured information of XML document

The structured information of XML documents are extracted by a structured information extractor that consists of ETID extractor and structured information generator. Figure 3 shows the architecture of the structured information extractor. The ETID extractor extracts element type from DTD and constructs a mapping table that maps the element name to ETID. The XML structured information extractor analyzes XML documents and extracts the structured information by referencing the mapping table.

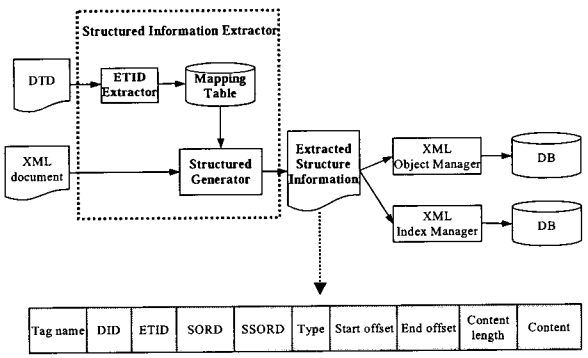


Figure 3. Structured Information Extractor Architecture

The information is extracted by structured information extractor such as DID, ETID, SORD, SSORD, Tag Name, Start_offset, End_offset, Content, and Content Length. Table 1 explains the meaning of each extracted structured information.

Table 1. Extracted Structured Information

Symbols	Descriptions
Tag Name	element name or attribute name
DID	unique document identification
ETID	element type identification
SORD	order information between sibling nodes
SSORD	order information between same type of sibling nodes
TYPE	element or attribute
START_OFFSET	start offset of element or attribute in the document
END_OFFSET	end offset of element or attribute in the document
CONTENT	value of PCDATA or attribute contained in the element

Figure 4 exemplifies that the structured information extractor extracts structured information from a XML document including images. The start offset and end offset of <figGrp> element in a document is 21666 and 21958 respectively and the attribute 'name' of <figure> element represents image file 'fig1.gif'.

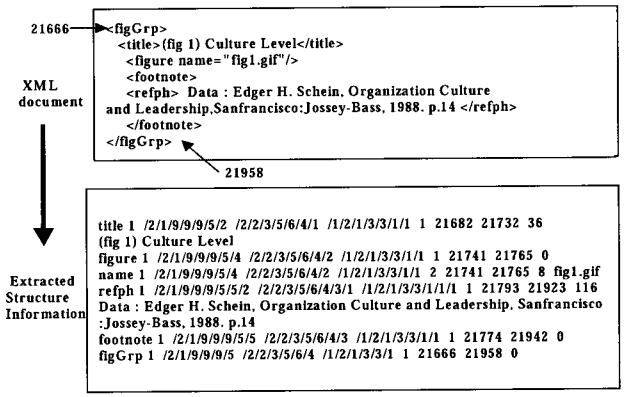


Figure 4. Structured Information of XML documents including image data

4. XML DATA MODELING

XRSs must be able to store and manage XML documents, DTD, structured information, various types of media in XML documents and so on. In order to do that we need data modeling of XML documents. In this paper, we use relational data modeling based on ORACLE used as storage system and employ composition storage model to guarantee fast retrieval of whole documents. The proposed XML data modeling structure in this paper is shown in the figure 5.

<DTD table>

DTDID	DTDNAME	FULLTEXT
number	char	long char

<DOCUMENT table>

DID	DTDID	NAME	FULLTEXT
number	number	char	long char

<ELEMENT table>

DID	SORD	DTDID	STARTOFFSET	LENGTH
number	char	number	number	number

<MULTIMEDIA table>

DID	NAME	STARTOFFSET	LENGTH	FULLTEXT
number	char	number	number	long raw

Figure 5. XML Data Modeling

Our XRS typically consists of four tables such as DTD table, Document table, Element table and Multimedia table. DTDs of XML documents are stored and managed in the DTD table. DTDID is a unique value of a DTD instance and DTDNAME is the file name of a DTD instance. DTDFULLTEXT is contents of a DTD instance and its data type is long type. When XML documents are requested, their DTDs are provided by DTDIDs. The Document table stores a whole document with document id(DID) and document name. DID is a unique id of a XML instance and NAME is the file name of a XML instance. FULLTEXT is the contents of a XML instance and it also has long type. In element table, structured information of each element of XML documents is stored and managed. To indicate the position of an element contained in XML documents, we store DID, DTDID, SORD, the start offset and length of an element into ELEMENT table. We need to store and manage media since XML documents may contain multimedia data. Multimedia data are managed in the Multimedia Table. The DID, start offsets, length and name that means the multimedia file name are included in the table. FULLTEXT is contents of multimedia and it has long type.

5. DESIGN AND IMPLEMENTATION OF XRS

The proposed XRS in this paper use ORACLE 7.3. RDBMS to manage XML documents and BRS search engine to provide efficient content-based queries and full text indexing. Figure 6 shows the overall architecture of the proposed XRS. It consists of a query analyzer, a result generator, a XML object manager, a XML index manager, a BRS search engine, and a structured query processor. In the followings we describe the each component in detail.

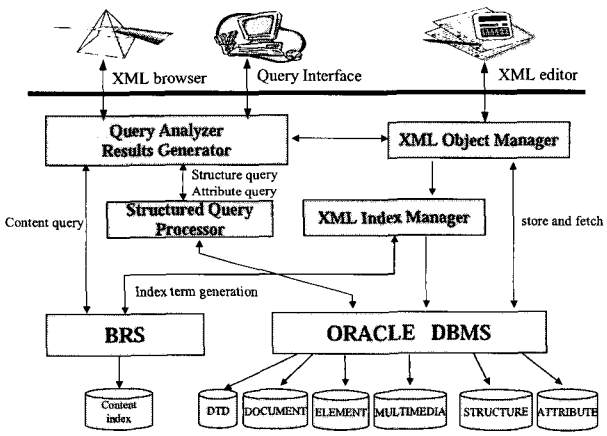


Figure 6. XML Repository System Architecture

The XML object manager that is in charge of the most important part of the system creates schemas for XML documents. It also extracts the structured information from XML documents and stores the XML documents into ORACLE DBMS. The sub-components of XML object manager are a object storage manager, a structured information extractor, a XML instance manager, a XML instance storage manager and a XML schema generator. The object storage manager provides other modules with unified interfaces of all of the sub-components of the XML object manager. The XML index manager creates and manages index structures to support various types of queries. It consists of three sub-components such as content index manager, structure index manager and attribute index manager. The BRS search engine creates and manages full text index that process content-based queries. However, it cannot process the structured query, the attribute query, and the hybrid query. Therefore we implement the structured query processor that can provide all of them. The query analyzer analyzes queries and distributes the query to BRS search engine or structured query processor according to the type of the query. Also, result generator provides the user's view of the search results from BRS search engine and structured search engine. In this section, we describe process flow between each module of our proposed XRS.

5.1 Schema Generation

The XML schema generator of the XML object manager creates schemas for XML documents as designed in the section 4. The XML schema generator creates DTD tables when the system initialized. After this, it creates document tables, element tables and multimedia tables dynamically whenever new DTD is required to be stored. Each table is named DOCUMENT_DTDname, ELEMENT_DTDname, and MULTIMEDIA_DTDname respectively. The process of schema creation is shown in figure 7.

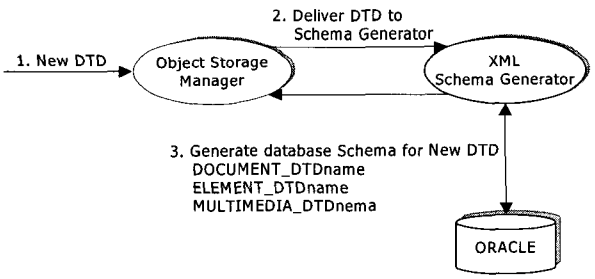


Figure 7. Process of Schema Generation

5.2 Store and Fetch of XML Documents

The XML instance storage manager that is one of the subcomponents of XML Object Manager stores DTD, XML documents, structured information, and multimedia in the database. We assign DTDID and DID to DTD and a XML document to be stored respectively and store structured information of the documents i.e., DID, SORD, Start_offset, length extracted by structured information extractor. A multimedia contained in XML documents as an attribute is stored with structured information of the attribute. Figure 8 shows the process for storing XML documents.

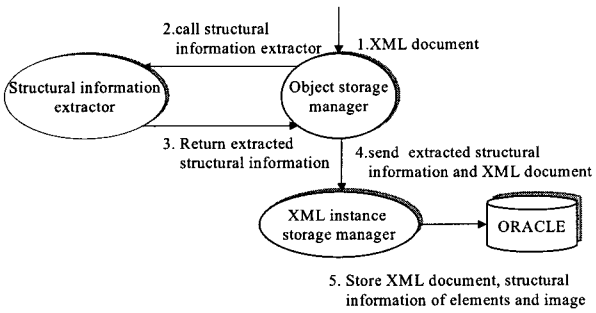


Figure 8. Process of storing XML documents

The XML instance manager fetches full documents or specific elements from the database. Figure 9 shows the process of fetching XML documents. If the documents or elements contain multimedia, it fetches them together. When fetching full documents from the database, we use DIDs or file names. On the other hand, when fetching an element, we first get position information (Start_offset, length) with DID and SORD of the element, and extract the element by using the position information. Multimedia is extracted like the followings. In the case of fetching a full document, the XML instance manager fetches all of the medias contained in the required documents. In the case of fetching an element, we fetch all medias that their start_offset is greater than that of the element and their end_offset is less than that of the element. In the figure 4, when figGrp is fetched, the start offset and end offset of figGrp is 2166 and 2195 respectively. Consequently, just fig1.gif between 2166 and 2195 is fetched.

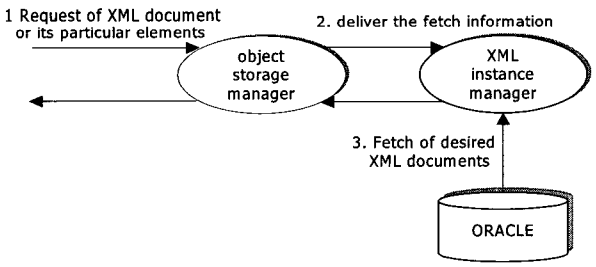


Figure 9. Process of fetching XML documents

5.3 Index for efficient Search

The XML index manager creates and manages index structures to support various type of queries. It consists of three sub-components such as a content index manager for content-based queries, a structure index manager for supporting structure queries and an attribute index manager for attribute queries. When documents are reached to the content index manager, it extracts DIDs of the documents, removes tags in XML documents and delivers them to BRS index manager. The BRS index manager creates index for content queries. We exploit the search engine of BRS to support fast full text search. To provide XML documents in ORACLE as search results to users, it creates a DOCN_MAP table that converts DOCN(document number) of BRS to the DID(Document ID) of a XML object manager and stores the table in ORACLE since only BRS perform content queries and XML documents are stored and managed in ORACLE. Figure 10 shows the process of creation of the content index.

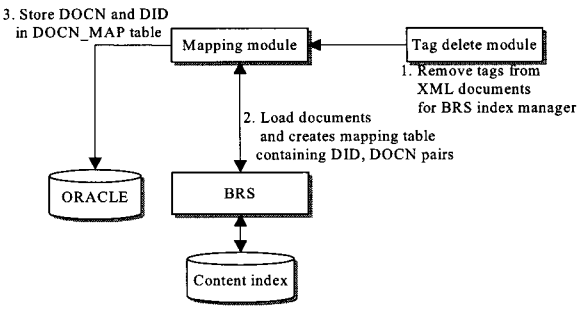


Figure 10. Process of creating content index

To process structure queries and structure-content queries efficiently, we index ETID and keywords together. It makes hybrid query processing more efficient. The structure index table in the figure 11 contains (ETID, term) pair as an index key. The SORD and SSORD make possible to access the unique element in large amount of XML documents. This index structures even can support queries like "Find documents which include keyword 'mobile' in <title> element" efficiently.

Figure 11 shows the structure and attribute index table and DOCN_MAP table. The key of the structure index table consists of ETID, TERM and DID and the key of the ATTRIBUTE index table is composed of ATTR_NAME and VALUE.

DOCN_MAP table

DOCN	DID
char	char

STRUCTURE index table

ETID	TERM	DID	SORD	SSORD
char	char	char	char	char

ATTRIBUTE index table

ATTR_NAME	VALUE	DID	SORD	ETID	SSORD
char	char	char	char	char	char

Figure 11. Structure, Attribute index table and DOCN_MAP table

5.4 Search

Queries are processed by BRS when they are content-based type or by structured query processor when they are structure, attribute, or hybrid type. The result generator handles the search results and provides the whole or the part of documents with users. It obtains the documents stored in ORACLE through a XML instance manager.

The content search module consists of two sub-modules such as BRS DOCN search module and ORACLE DID search module. The DOCN search module of BRS performs content search and

consists of BRS APIs. When handling content queries like "Find documents that include keyword 'parallel' ", the queries are delivered to the BRS search engine and receives DOCNs as search results. The ORACLE DID search module searches DIDs that correspond to the DOCNs from DOCN search module. The result generator uses the DIDs to provide the documents to users.

Structure queries are based on logical structures of documents. They should be processed with consideration of relationship between elements on hierarchy (parents, children, ancestors and descendants), relationship between elements on the same hierarchy(siblings) and the ordered relationship between the same elements on the same hierarchy. The proposed XRS can deal with all kind of structure queries with our proposed structured information(ETID, SORD, SSORD). Structure queries are classified into five types such as parent, child, ancestor, descendant and sibling. There are five query processing API sets corresponding to each query type. Each API translates corresponding structure queries to SQL statements.

Figure 12 shows the steps of processing the hybrid query, for example "Find the parent element of the first <ThesisTitle> including a keyword 'mobile'". In the first step, we get the ETID of start element by referencing ETID mapping table. The ETID becomes a key together with keyword 'mobile'. With this key we can get DID, SORD, SSORD of a document element whose type is ThesisTile and which contains a keyword 'mobile'. In the second step in order to find the first ThesisTitle element from the results of the first step, we select SSORDs that satisfy "/*/*/1". In third step, to find parent element of the first ThesisTitle, we use SORD. For example, if SORD is "/3/2/1", the SORD of its parent element is "/3/1". In the last step, we finally obtain document elements by using the SORD and DID of the parent.

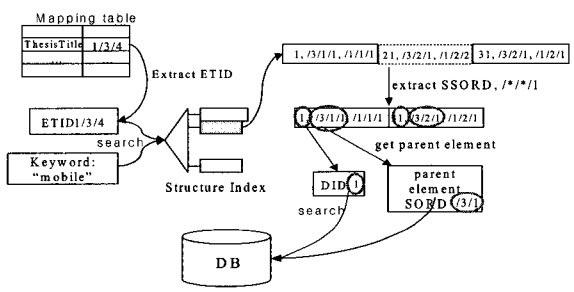


Figure 12. Structure Search Processing

Attribute queries specify values for attributes associated with elements. For example "Find documents such that <SEX> attribute has 'female'". Attribute queries are classified into four types. There are four query processing API sets corresponding to each query type. Each API translates corresponding attribute queries to SQL statements.

5.5 Query Interface

In this section, we design a web based query interface for users to search XML documents easily on the web. Our user interface supports the content-based queries, structure queries, attribute queries and mixed queries. In the case of content-based queries, users just type a keyword on textfields, "Keyword". For structure queries, there are several textfields such as start element, relationship, order and target element. The target element has logical relationship with start element. For a query like "Find the <section> element which is the first child of <chapter> element", chapter is the start element, section is the target element and the relationship between two elements is the child relationship. We use relationships between elements such as ancestor, descendant and sibling. Also we represent the order of elements through +, -, numeric. To process the attribute search, various relational operations are provided. Next, we get the results in a document or an element unit according to the value of Output Scope field.

Figure 13 shows the designed query interface. It exemplifies "Find the parent element of the first <ThesisTitle> that contains 'mobile'". Since the keyword is contained in the first ThesisTitle, we specify the start element's ranking value as 1, element as ThesisTitle and keyword as 'mobile'. Because the relationship between the start element and the target element is parent, we specify the relation ranking value as 1 and the relationship value as ancestor. Figures 14 shows the search results on the query.

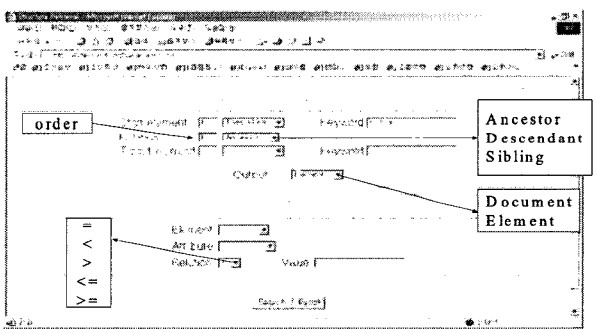


Figure 13. Query interface

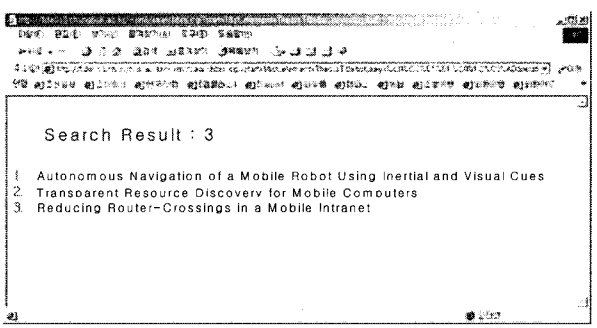


Figure 14. Query Results

6. PERFORMANCE EVALUATION

In this section, we show through experiments that our proposed XRS outperforms existing XRSs in terms of loading time, fetching time and search time. The type of XML documents used in this experiment is thesis. The number of thesis documents is 300 and the total size of them except images is about 70MBytes. The average number of images in a document is 13 and their average size is 37.5KBytes. We compare our proposed XRS with non-composition based XRS. The platform used in this experiments is SUN Enterprise 3000 with solaris 2.7.

Figure 15 shows the documents loading time. The loading time of our proposed XRS is about 28.71 seconds while that of non-composition based XRS is 59.42 seconds. The proposed XRS loads documents about 200% faster than composition-based XRS.

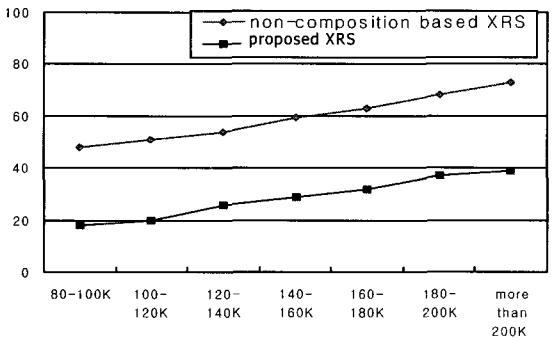


Figure 15. Loading Time

Figure 16 shows the documents fetching time. Since a XML document is split into elements and the elements are stored in non-composition based XRSs, the XRSs must assemble elements before presenting the whole document. The assembling process takes rather long time.

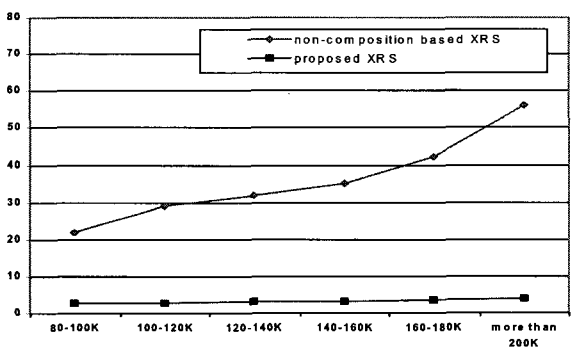


Figure 16. Fetching Time

In order to measure search time we use seven keywords such as 'mobile', 'communication', 'development', 'agriculture', 'contemporary', 'society' and 'research'. We measure time of processing each query. Figure 17 shows the measured search time. The search time of non-composition based XRS increases almost linearly as the size of documents increases. Generally DBMS cannot create index for variable size of texts. However, our proposed XRS use BRS that create inverse file so it can deal with content queries efficiently.

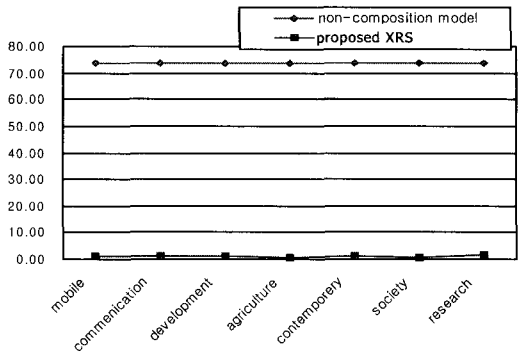


Figure 17. Search time of Content based Query

We cannot compare the search time of structure query with other XRSs, because none of existing XRSs can support structured queries fully. Therefore we only measure recall and precision to show retrieval effectiveness of our XRS. Recall is the ratio of relevant documents retrieved for a given query over the number of relevant documents for that query in the database. Precision is the ratio of the number of relevant documents retrieved over the total number of documents retrieved. The table 2 shows the recall and precision of processing the various structure queries. We can easily know from the table the proposed XRS support the various type of queries such as content queries, structure queries, attribute queries and hybrid queries correctly.

Table 2. Recall and Precision of Structure based Query

Queries	Recall	Precision
Structure queries (hierarchy relationship)	100%	100%
Structure queries (relationship on sibling)	100%	100%
Structure queries (order relationship)	100%	100%
Attributed queries	100%	100%
Hybrid queries	100%	100%

7. CONCLUSION

In this paper, we have designed and implemented a XRS that exploits the advantages of DBMSs and IRSs. Our scheme uses BRS to support full text indexing and content-based queries efficiently, and ORACLE to store XML documents, multimedia data, DTD and structured information. In order to process structure queries, our XRS represents structured information of a XML document as ETID(Element Type Id), SORD(Sibling ORDer) and SSORD(Same Sibling ORDer).

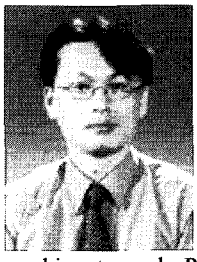
We have also designed databases to manage XML documents including audio, video, images as well as text. We employ the non-composition method when storing XML documents into ORACLE. To show superiority of our XRS over the existing methods, we have performed various experiments in terms of the document loading time, the document fetching time, search time of content-based queries, precision and recall. It has been shown through experiments that our XRS outperforms the existing XML document management systems. In the further work, we will extend XRSs by adding some functions such as versioning and collaborations.

ACKNOWLEDGEMENT

This work was supported by the Korea Science and Engineering Foundation(KOSEF) grant funded by the Korea government(MOST) (No. R01-2006-000-1080900)
And the Korea Research Foundation Grant funded by the Korean Government(MOEHRD)" (The Regional Research Universities Program/Chungbuk BIT Research-Oriented University Consortium)

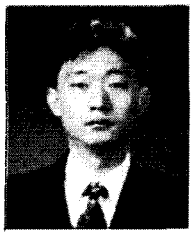
REFERENCES

[1] Albrecht Schmidt, Martin L. Kersten, Menzo Windhouwer, Florian Waas, "Efficient Relational Storage and Retrieval of XML Documents," WebDB (Informal Proceedings), 2000.
[2] Arijit Sengupta. "Toward the union of databases and document management: The design of DocBase," Conference on Management of Data (COMAD'98), Hyderabad, India, 1998.
[3] Brian Lowe, Justin Zobel and Ron Sacks-Davis, "A Formal Model for Databases of Structured Text," DASFAA 1995.
[4] C.H. Park, J.H. Cheong, D.I.Shim, S.G. Lee, "Performance Comparsion between DBMS and IRS for Structured Documents", Proceedings of the Korean Database Conference. 1999.
[5] D. Florescu, D.Kossman, "Storing and Querying XML Data using a RDBMS," IEEE Data Engineering Bulletin, Vol. 22, No. 3, 1999.
[6] Extensible Markup Language(XML) 1.0, "http://www.w3.org/TR/1998/REC-xml-19980210"
[7] Francois, "Generalized SGML repositories: Requirements and Modeling," Computer Standards & Interfaces, 1996.
[8] Ian A. Macleod, "Storage and Retrieval of Structured Documents, Information Processing and Management," Information Processing and Management, Vol. 26, 1990.
[9] J. McHugh, S. Abiteboul, R. Goldman, D. Quass and J. Widom., "Lore: A database management system for semi-structured data," SIGMOD Record, 26(3), September 1997.
[10] J. Shanmugasundaram, K. Tufte, G. He, C. Zhang, D. DeWitt, J. Naughton, " Relational Databases for Querying XML Documents : Limitations and Opportunities," VLDB Conference, Edinburgh, Scotland, September 1999.
[11] Klemens Bohm, Adrian Muller, Erich Neuhold, "Structured Document Handling- a Case for Integrating Database and Information Retrieval," CIKM. 1994.
[12] Kyuchul Lee, Yongkyu Lee and Bruce Berra, "Management of Multi-structured Hypermedia Documents: A Data Model, Query Language, and Indexing Scheme, Multimedia Tools and Applications," Vol. 4, 1997.
[13] Marc Volz, Karl Aberer, Klemens Bohm, "Applying a Flexible OODBMS-IRS-Coupling to Structured Document Handling," ICDE. 1996.
[14] Patricia Francois, Pierre Bazex, "SGML/HyTime Repositories: Requirements and Data Modeling Using Object-Oriented Database Concepts," DEXA, 1995.
[15] Philippe Futtersack, Didier Bolf "The Electronic Library Project: SGML Document Management System Based on ODBMS," TAPOS 5(4), 1999.
[16] Ron Sacks-Davis, Tuong Dao, James A. Thorn, and Justin Zobel, "Indexing documents for queries on structure, content and attributes," In International Symposium on Digital Media Information Base(DMIB'97), Nov. 1997.
[17] Tuong Dao, "An Indexing Model for Structured Document to Support Queries on Content, Structure and Attributes," In Proc. of IEEE ADL '98, 1998
[18] T.W.Yan, J.Annevelink, "Integrating a Structured-Text Retrieval System with an Object-Oriented Database System," VLDB. 1994.



yung-Il Kang
He received the B.S. degree in the department of Computer Communication Engineering from Howon University in 1999. And he received the M.S. degree in department of Computer and Industrial Engineering, Chungbuk National University in 2002. He is currently working towards PhD. degree on Bioinformatic system. His research interests also include Database System, Multimedia Database and Storage management system.

Jae Soo Yoo
He received the B.S. degree in Computer Engineering in 1989 from Chunbuk National University, Chunju, South Korea. And he received the M.S. and Ph.D. degrees in Computer Science in 1991 and 1995 from Korea Advanced Institute of Science and Technology, Taejeon, South Korea. He is now a professor in the department of Computer and Communication Engineering, Chungbuk National University, Cheongju, South Korea, where his research interests are the database system, multimedia database, distributed computing and storage management system.



young Yup Lee

He received the B.S. and M.S. degree in Computer Science in 1991 and 1993 from Korea Advanced Institute of Science and Technology, Daejeon, South Korea. And he received the Ph.D. degree in the Management Information Systems from Korea Advanced Institute of Science and Technology, Daejeon, South Korea in 1997. He is now a professor in the department of Electronic Commerce, Paichai University, Taejeon, South Korea, where his research interests are the datamining, XML, artificial intelligence, multimedia database, distributed computing.