Retrieval Performance of XML Documents Using Object-Relational Databases

객체-관계형 데이터베이스에 의한 XML문헌의 검색성능 평가

김 희 섭(Hee-Sop Kim)*

ABSTRACT

The purpose of this study is to evaluate the performance of XML retrieval based on ORDBMSs(Object-Relational Database Management Systems) approach. This paper describes indexing and retrieval methods for XML documents and the methodologies of experiments at INEX(Initiative for the Evaluation of XML retrieval). Like any other traditional information retrieval experiment, the test collection was consists of documents, topics/queries, task, relevance assessments and evaluation. EXIMA ™ Supply, a kind of native XML DB based on ORDBMS technologies, is used for this experiment. Although this approach has many benefits, for example, no delay in storing and searching XML documents, but it showed relatively disappointed retrieval performance at INEX 2002. This result may caused since the given topics had to be decomposed and modified to be processed by the XPath processor, and during this modification the original meaning of topics can be changed inevitably and some important information may pass over.

초 록

본 연구의 목적은 객체-관계형 데이터베이스 접근에 의한 XML 문헌의 검색 성능을 평가하는 것이다. 본 논문에서는 INEX(Initiative for the Evaluation of XML retrieval) 에서의 XML 문헌의 색인 및 검색 방법에 대하여, 그리고 실험 방법론들에 대하여 기술하고 있다. 대부분의 전통적인 정보검색 성능평가 실험에서와 같이 본 연구에서 사용된 테스트 콜렉션(test collection)은 문헌(즉, XML 문헌), 토픽, ad hoc 검색, 적합성 판단, 평가로 이루어졌다. 그리고 ORDBMS 기술들을 기반으로 개발된 전용XML 데이터베이스의 일종인 EXIMA™ Supply을 사용하여 INEX에서 제공한 대규모 XML 문헌들을 저장하고 검색하였다. 본 논문에서는 실험에서 사용한 시스템에 대한 개략적인 기능들과 색인 및 검색 과정 그리고 INEX 2002에서의 성능평가 결과에 대하여, 앞으로 개선되어야 할 기능에 대하여 논하고 있다.

키워드: XML documents, EXIMA supply, object-relational DBMS, IR performance evaluation, INEX, XML 문헌, 객체 - 관계형 DBMSs, 정보 검색, 성능 평가

---

* 경북대학교 문헌정보학과 전임강사(heesop@knu.ac.kr)

- 논문접수일자 : 2004년 5월 21일
- 게재확정일자 : 2004년 6월 16일

1. Introduction

XML(eXtensible Markup Language) is an emerging standard for the representation and exchange of Internet data. Most research on storing and indexing XML documents has been based on the work on semi-structured data. Models for representing XML data have been proposed from a database perspective and they have been tailored to facilitate querying processing on semi-structured data.

The nature of this semi-structured data is that it is self-descriptive, and that it incorporates an operational DTD(Document Type Definition). DTDs define the structure of an XML document, for example, what elements and attributes are permitted in the document. XML documents contain data, and must contain exactly one root element which will contain all the other elements. It might happen that an element contains a set of sub-elements in addition to the data. For query purposes XPath can be used. XPath is a language for finding information in an XML document. Using XPath, we can specify the locations of document structures or data in an XML document, and then process this information using XSLT (the Extensible Stylesheet Language Family Transformations).

Generally, three data models can be deployed for the persistent storage of XML documents. First, the development of specialized data management systems which is tailored to store and retrieve XML documents using special purpose indices and techniques of query optimization(e.g., McHugh, et al. 1997; Fernandez et al. 1998). Second, for an object-oriented DBMS(Database Management System), an O_2(Bancihon et al. 1988), or Object store, can be used to store XML document because of the rich capability of this database system(e.g., Bancihon, et al. 1988). Third, when a relational DBMS is employed XML data is mapped into relations and queries posed in a semi-structured query language which is then translated into SQL(Structured Query Language) queries(e.g., Khan and Rao 2001).

However, it is impossible to predict which of these three approaches will be widely accepted. The first, the use of a specialized or special purpose database system, may work best, once needs are met concerning scalability and the level of maturity required for the handling of huge amounts of data. The second, an object-oriented database system, seems well-suited to complex data like XML, but vulnerable in the area of evaluating

queries addressed to a very large database. The third approach, RDBMSs (Relational DBMSs), provides maturity, stability, portability, and scalability. Furthermore, since a majority of the data on the web currently resides in and will continue to be stored in RDBMSs, the opportunity arises for constructing a system using a RDBMS to store XML documents, making it possible to seamlessly query of data with one system and one query language.

It has become crucial to address the question of how large collections of XML documents can be stored and retrieved efficiently and effectively. To date, most work on storing, indexing, querying, and searching documents in XML has stemmed from the database community's work on semi-structured data. However, among the early studies in the community, ORDBMS has received less attention to date. This was the motivation for the present study which sought how performs the ORDBMSs approach in its retrieval under large test collections of XML docu ments.

2. Related Work

Many previous studies address storage strategies for XML documents in order to improve query processing. Most of these studies suggest storing XML documents in traditional DBMS due to the advantages of storing both legacy structured data and XML data in a single system, as well as using typical functionalities of a DBMS. However, there are still many open issues when storing native XML documents in a traditional DBMS. First of all, most studies are based on the relational data model, which offers limited structures to represent the features of XML data, such as nested relationships and ordering of XML elements. Also, their DBMS schema representations are proprietary, and querying these structures is usually complex since the final users are not familiar with them. Finally, few studies support the typical features of XQuery, i.e., the standard XML query language, such as the FLWR(FOR-LET-WHERE-RETURN) expressions, XPath expressions, and wildcard query operators(e.g., the '//' operator).

Figure 1 shows a brief comparison between RDBMS approaches and XML DB approaches.

Regardless of the DBMS data model, Vieria, Ruberg and Mattoso(2003) identified four basic representation aproaces to store XML data in a traditional data

RDBMS	XML DB
- 2-dimensional table - XML to DB conversion - multiple JOINs for retrieval - Find books with text "XML native database model"	- multi-dimensional - structural searching - native structure - Find abstracts of articles that have "XML native database model" in their reference

〈Figure 1〉 Comparison between RDBMS approaches and XML DB approaches

tbase system as follows:

- Black-box - uses special data type such as CLOBs(Character Binary Large Objects) to store an XML document with its original format;
- Universal representation for the elements - uses a single data structure for all types of an XML document;
- Specific representation for the elements - maps each named XML element to a specific data structure(e.g., tables in RDBMSs or object collections in ODBMSs); and
- Generic representation for the elements - adopts a generic schema to represent the XML document structure, such as the DOM(Document Object Model) tree.

Figure 2 shows a category of above four approaches according to their data model.

The black-box approach is very simple and closer in many aspects to the file system approach, thus not offering much flexibility for the manipulation of XML data. The use of special data types can be found in the Oracle database system and in the DB2(Cheng and Xu 2000). In the universal representation for the elements, the XML representation schema is very simple. However, there is no flexibility to efficiently manipulate the XML schema. The specific representation for the ele

	Black-box	Universal representation	Specific representation	Generic representation
Object data model	Special types (e.g., Cheng and XU 2000)	Object based approach (e.g., Tian et al. 2002)	Schema-based (e.g., Florescu and Lossman 1999)	DOM (e.g., W3C Document Object Model)
Relational data model	Special types (e.g., Oracle)	Universal Inlining/ (e.g., Florescu and Elmasri 2001)	Binary DTD (e.g., The Apache XML project)	Relational approach (e.g., Manolescu et al. 2000)

〈Figure 2〉 Category of XML data storage in traditional DBMSs

ments is the most flexible representation approach, where each named XML element has its own structure on the DMBS, so that attributes of a given XML element are directly mapped to the attributes of its equivalent database structure. This approach optimizes the access to XML elements of a given type, but it may generate a high number of type structures on the DBMS. Additionally, XML elements with distinct types but with identical names may not be properly stored in the database due to name conflicts. Finally, adopting a generic schema to represent XML documents corresponds to the DOM approach presented in W3C Document Object Model(see, http://www.w3.org/DOM). This approach does not generate a large number of data structure, hence it prevents possible name conflicts among XML documents. However, users must know the specific structures of the generic representation schema in order to query the XML database.

An example of ORDBMS approach Shimura, Yoshikawa and Uemura(1999) decompose an XML document into a set of nodes and store these nodes in several tables along with encoded path information from the root to each node. However, this path does not serve as a primary key. One of the shortcomings of their path encoding is that it does not facilitate the construction of XML document on the fly from the result set of database SQL query.

The transparent execution of XQuery queries in a traditional DBMS requires a process of automatic translation from XQuery to the DBMS query language. This task is not trivial because the translator must know the schema structure representation used to store the XML documents in the DBMS. Therefore, the choice of the representation schema in a DBMS determines the basis to the success of efficient XQuery execution. The translation process in the relational database model generates inefficient results for recursive queries because the mapping results in long SQL statements containing a great number of joins. On the other hand, the object-oriented database model offers more semantic to the representation of relationships, and it provides specific algorithms to the execution of queries containing path expressions. Fegaras and Elmasri(2001) have presented an XML query language, called XML-OQL, and an algebra that allows its translation to the OQL(Object Query Language) language. Although their mapping approach is more semantic-conservative

than those in the relational model, the standard language XQuery is not adopted, and a specific schema is used to store the XML data.

Vieira, Ruberg and Mattoso(2003) present a model of storage architecture of XML documents in ORDBMS while providing transparent execution of XQuery queries. Their proposal is based on adopting the DOM schema to map XML elements to object classes, and to store them in an ORDBMS. The main module of their proposal is the XVerter translator, which represents a set of transformation rules to covert XQuery queries on XML documents into SQL3 statements on the corresponding DOM classes. The DOM format was chosen because it has already been widely used in applications that manipulate XML documents in main memory, supported by programming languages. Beside that, the DOM format facilitates the total or partial recovery of XML documents in such a way that the original features of the XML data are preserved, such as the order of the XML elements. The DOM format allows better performance for typical tree-based queries. However, for traditional queries over element types, such approach does not fit well. The DOM format is very close to the object-based representation schema analyzed in Tian et al(2002) which presents a good performance for query processing.

Many approaches, which have been mainly proposed by the database community, are close to the research done on semi-structured databases(McHugh et al. 1997; Ceri, Fraternali and Paraboschi 2000). In this case, the main goal is to build semi-structured management systems that facilitate query processing. Other approaches are stemmed by the work done on structured documents from the perspective of information retrieval(Jang, Kim and Shin 1999; Moffat and Zobel 1996; Navarro and Baeza-Yates 1997). Among the database approaches is the Lore system. Lore accomplishes the uploading of new documents by adding the elements of the documents in a tree like structure and updating several indexes(value index, text index, link index and path index). From that point on, any data access is performed by considering the whole database as a huge tree that contains XML elements. Lore approach seems to view an XML document as a database and a set of documents as a single large database where all documents are mixed together into a tree like structure.

In Kotsakis(2002) approach, a set of XML documents is handled as is(set of

documents) and the index facilitates the retrieval of XML documents that match better the query terms. The output of the query evaluation is a set of XML documents. The proposed approach is closer to the information retrieval end and the objective is to retrieve the XML documents that match the user preferences expressed by way of Boolean query terms. Indexing structures for structured documents are discussed in Lee et al(1996).

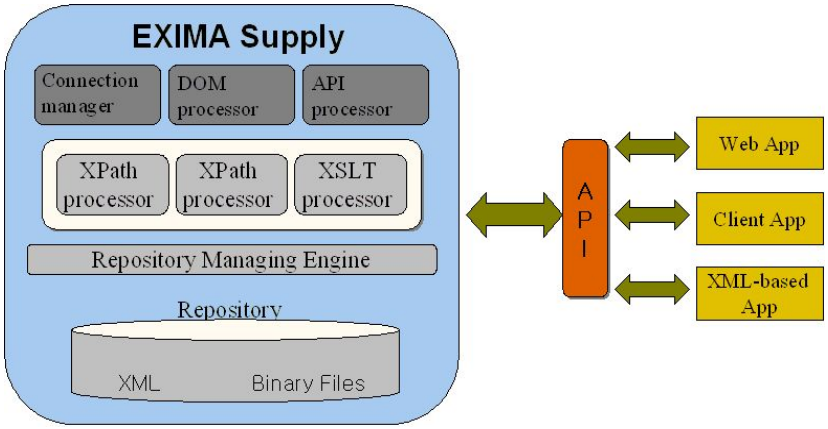
3. Experiment

3. 1 INEX

The aim of INEX(Initiative for the Evaluation of XML retrieval) is to provide means, in the form of a large test collection and appropriate scoring methods, for the evaluation of retrieval of XML document. It is organized by the DELOS Network of Excellence for Digital Libraries, initiates an international, coordinated effort to promote evaluation procedures for XML retrieval and supports a forum to compare results of each participant. 36 teams took part in INEX 2002 from all around world.

3. 2 An XML DB - EXIMA™ Supply

EXIMA™ Supply is a kind of native XML DB which is based on ORDBMS technologies to store and manage XML documents developed by Incom I&C Co. Ltd. It can store and retrieve XML and its related documents(e.g., DTD, XSL) fast enough to process XML information. EXIMA™ Supply is supporting



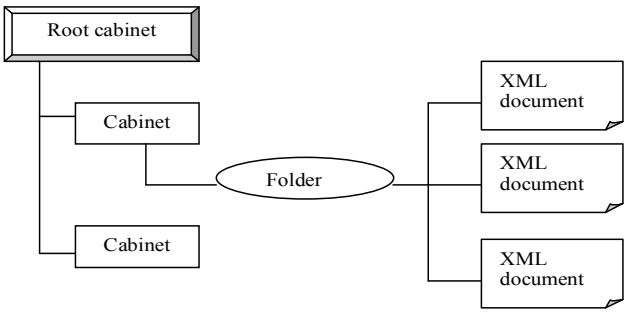
〈Figure 3〉 Architecture of EXIMA™ Supply

XPath Standard to search elements in XML documents. However, itis not provide any functionality of a searching engine. This means that it cannot search information as intelligently as most searching engines do.

Therefore, it can preserve the native features of XML documents by representing and storing them in object-oriented structures. This is one of the important features of EXIMA™ Supply. By means of this feature, the data and hierarchical information of XML documents can be stored without modification or distortion. Besides, EXIMA™ Supply helps manage and utilize XML documents with ease by providing the standard XPath query language. With EXIMA™ Supply, there is no need to transform XML documents into other formats such as relational tables of commercial DBMS while many XML servers are using relation DBMS and therefore XML documents must be transformed into relational tables, because it can treat the hierarchical structures of XML documents as it is. As a result, there is no delay in storing and searching XML documents and it is possible to process XML data on the fly.

EXIMA™ Supply provides a logically hierarchical structure to manage the storage of XML documents. The logically hierarchical structure is the storage structure that is transparently accessible by users regardless of the internal physical storage structure. EXIMA™ Supply has two kinds of storage types, “Cabinet” and “Folder.” Cabinet is a logical storage that can contain cabinets and folders. Cabinet can be used to manage storage hierar- chically.

Folder is the storage where XML and related documents are actually stored. A folder can contain one DTD and corresponding XML and XSL documents. On the other hand, XML documents correspond to a DTD can be stored in multi-



〈Figure 4〉 Storage Types of EXIMA™ Supply

ple folders, if necessary.

The system was tested under the following software environment.

- OS: Windows 2000 Professional
- XML Server: EXIMA™ Supply 1.0
- DBMS: UniSQL 5.1
- Web Server: Tomcat
- Searching client: Web application developed with JSP
- Client/Server: Pentium III PC(256 MB memory)

3. 3 Test Collection

The test collection consists of three parts:(1) a set of XML documents which were donated by the IEEE Computer Society,(2) a set of topics which consists of 30 CAS(content- and-structured) queries, and of 30 CO(content only) queries, and(3) relevance assessments which were provided by the INEX participants.

3. 3. 1 Documents

IEEE provided the XML documents which consist of the full texts of 12,107 articles from 12 magazines and 6 transactions of the IEEE Computer Society's publications, covering the period of 1995-2002, and totaling 494 megabytes in size. Although the collection is relatively small compared with TREC, it has a suitably complex XML structure(e.g., 192 different content models in DTD). On average, an article contains 1,532 XML nodes, where the average depth of a node is 6.9.

The overall structure of a typical article consists of a frontmatter, a body, and a backmatter. The frontmatter contains the article's metadata, such as title, author, publication information, and abstract. The body is structured into sections, subsections, and sub-subsections. These logical units start with a title, and contain a number of paragraphs, tables, figures, item lists, references, etc. the backmatter includes a bibliography and further information about the article's authors.

3. 3. 2 Topics

The topics are created by the participating groups. Topics are two types:(1) CAS(content-and-structure) and(2) CO(content-only). Each group created a set of candidate topics that were representative of what real users might ask and the type of the service that operational systems may provide.

The topic format and the topic development procedure were based on TREC guidelines, which were modified to allow for the definition of containment conditions and target elements in CAS

```
<INEX-Topic topic-id="05" query-type="CAS">
  <Title>
    <te>article//tig</te>
<cw>QBIC</cw>  <ce>bibl</ce>
<cw>image retrieval</cw>
</Title>
<Description>
Retrieve the title from all articles which deal with image retrieval
and cite the image retrieval system QBIC.
</Description>
<Narrative>
To be relevant a document should deal with image retrieval and also should contain(at least)
one bibliographic reference to the retrieval system QBIC.
</Narrative>
<Keywords>
QBIC, IBM, image, video, content query, retrieval system
</Keywords>
</INEX-Topic>
```

〈Figure 5〉 Example of CAS topic from the INEX test collection

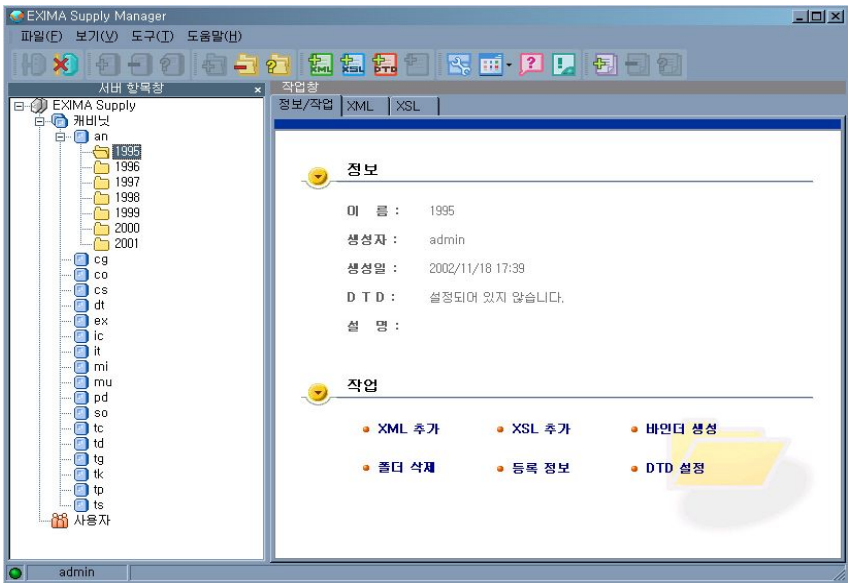
queries. The overall structure of an INEX topic consists of the standard title, description, and narrative fields and a new keywords field. Figure 5 shows an example of a CAS topic. 60(30 for CAS and 30 for CO) topics were selected into the final set.

3. 3. 3 Tasks

The task is the ad-hoc retrieval of XML documents. The answer to a query is a ranked list of XML elements, the top 100 elements of which is submitted as the retrieval result. The given topics had to be decomposed and modified to be processed by the Xpath processor in EXIMA™ Supply. The modified topics were expressed in one or several Xpath queries. Some complicated topics had to be decomposed into several Xpath queries. During this process of modification, the original meanings of topics were changed inevitably and some information was lost.

In set up the XML documents provided by INEX into EXIMA™ Supply, the directory structure of XML documents was mapped into the logical structure of EXIMA™ Supply. For example, XML documents in "E:\an\1995" directory are stored in the folder "1995" in the cabinet "an."

Figure 6 shows the example storage structure of EXIMA™ Supply shown in EXIMA™ Manager.



〈Figure 6〉 Example of the Storage Structure of EXIMA™ Supply

3. 4 Indexing

Two index schemes are well known for structured documents: position based indexing and path-based indexing to access document collections by content, structure, or attributes. In position-based indexing, queries are processed by manipulating ranges of offsets of words, elements or attributes. In path-based indexing, the paths in tree structures are used. Our storage method of XML documents adopts both of two indexing schemes and enjoys the advantages of them. XML processors guarantee that XML documents stored in databases follow tagging rules prescribed in XML or conform to a DTD. Hence, XML documents stored in databases are valid or well-formed.

An XML document can be represented as a tree, and node types in the tree are of the following three kinds: Element, Attribute and Text. Nodes of type Element have an element type name as a label. Element nodes have zero or more children. The type of each child node is of one of the tree, that is, Element, Attribute and Text. Nodes of type Attribute have an attribute name and an attribute value as a label. Attribute nodes have no child node. If there are plural attributes, the order of the attributes is not distinguished. This is because there is no order in XML attributes. Nodes of type Text have

character data specified in the XML Recommendation as a label. Text nodes have no child node.

We have the following policies for the storage of XML documents:

- Database schemas should not depend on DTDs or element types, and databases shall store any XML documents
- Index structures which are provided in DBMSs shall be used
- Storage method shall be realized by doing minimal extension to object-relational databases
- Functionalities of XML query languages shall be supported

EXIMA™ Supply has the functionality of indexing of elements of XML documents. Especially, EXIMA™ Supply makes indexes of elements when an XML document is stored. Therefore, it doesn't need any extra indexing process. Elements in one folder are indexed together and the searching speed is almost same among elements in one folder. However, the indexing is done in each folders, the searching speed may be different from each folder.

3. 5 Retrieval process

XML documents are decomposed into paths of their tree representation, and stored in the four relations, that is, Element, Attribute, Text and Path. Their relational tables, in which XML documents are stored, are hidden from users or applications. Users or application consider XML documents as trees, and they specify queries in XML query languages. XPath(XML Path Language) was adopted in this experiment.

3. 5. 1 XPath query generation

XPath is a language for addressing parts of an XML document, designed to be used by both XSLT(Extensible Stylesheet Language Transformation) and XPointer(XML Pointer Language). XPath is the result of an effort to provide a common syntax and semantics for functionality shared between XSLT and XPointer. The primary purpose of XPath is to address parts of an XML document. In support of this primary purpose, it also provides basic facilities for manipulation of strings, numbers and booleans. XPath uses a compact, non-XML syntax to facilitate use of XPath within URIs(Uniform Resource Identifiers) and XML attribute values. XPath operates on the abstract, logical structure of an XML document, rather than its surface syntax. XPath gets its name from its use of a path notation as in URLs

for navigating through the hierarchical structure of an XML document.

In addition to its use for addressing, XPath is also designed so that it has a natural subset that can be used for matching(testing whether or not a node matches a pattern).

XPath models an XML document as a tree of nodes. There are different types of nodes, including element nodes, attribute nodes and text nodes. The primary syntactic construct in XPath is the expression. An expression matches the production Expr. An expression is evaluated to yield an object, which has one of the following four basic types: (1) Node-set(an unordered collection of nodes without duplicates), (2) Boolean(true or false), (3) Number(a floating-point number), and (4) String(a sequence of Universal Character Set characters).

Expression evaluation occurs with respect to a context. XSLT and XPointer specify how the context is determined for XPath expressions used in XSLT and XPointer respectively. The context consist of: (1) a node(the context node), (2) a pair of non-zero positive integers(the context position and the context size), (3) a set of variable bindings, (4) a function library, (5) the set of namespace declarations in scope for the expression.

EXIMA™ Supply supports XPath searching functionality. Therefore, searching topics from INEX has to be converted to XPath queries for searching information. '/' is the child operator which selects from immediate child nodes. '//' is the descendant operator which selects from arbitrary descendant nodes. The '//' can be thought of as a substitute for one or more levels of hierarchy. Also, in the query, filer clause '[ ]' which is analogous to the SQL WHERE clause, indexing which is easy to find a specific node within a set of nodes can be specified. All the INEX topics were expressed in XPath in this stage. For instance, topic 01 of INEX 2002 expressed in XPath queries as follows:

Topic 01:

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE INEX-Topic SYSTEM "inex-topics.dtd">
<INEX-Topic topic-id="01" query-type="CAS" ct-no="010">
  <Title>
   <te>article/fm/au</te>
<cw>description logics</cw><ce>abs, kwd</ce>
</Title>
```

```
<Description>
   Retrieve the names of authors of articles on description logic, in particular articles in
which the abstract or the list of keywords contains a reference to description logic.
</Description>
<Narrative>
   The rating should reflect the likeliness that a person is an expert on description logic.
</Narrative>
<Keywords>
description logic DL ABox TBox reasoning
</Keywords>
</INEX-Topic>
```

Xpath query:

```
"article/fm[abs//*/text('*')[contains('description logic')]]/au"
```

Complicated topics that can not be expressed in one XPath query can be divided into several XPath queries. For instance, topic 06 of INEX 2002 expressed in XPath queries as follows:

Topic 06:

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE INEX-Topic SYSTEM "inex-topics.dtd">
<INEX-Topic topic-id="06" query-type="CAS" ct-no="034">
  <Title>
   <te>tig</te>
   <cw>Survey on Software Engineering</cw>
   <cw>
    software engineering survey, programming survey, programming tutorial,
software engineering tutorial
   </cw>
   <ce>tig</ce>
   <cw>programming languages</cw><ce>sec</ce>
  </Title>
  <Description>
   Retrieve the article title from all articles which are a tutorial or survey on software engineering or
programming dealing with programming languages.
</Description>
 <Narrative>
   To be relevant an article should offer a tutorial or survey on software engineering
   or programming containing sections dealing with programming languages.
 </Narrative>
 <Keywords>
survey, tutorial software engineering, programming language
</Keywords>
</INEX-Topic>
```

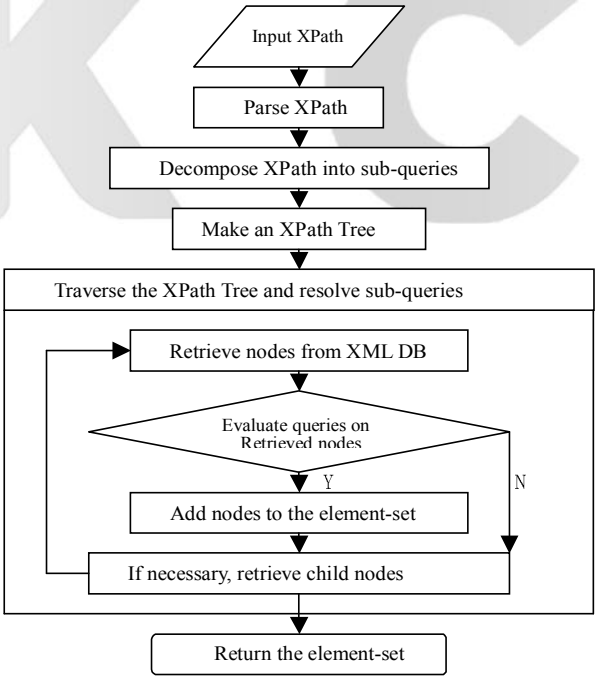
Xpath queries:

```
"article[//tig//*/text('*')[contains('Survey on Software Engineering')]]//tig"
"article[//tig//*/text('*')[contains('software')][contains('engineering')][contains('survey')]
[contains('tutorial')]]//tig"
"article[//sec//*/text('*')[contains('programming')][contains('languages')]]//tig"
```

If a topic can not be expressed in XPath queries, just keywords can use for searching as an alternative approach.

3. 5. 2 Searching process of XPath queries

The XPath queries processed as shown in the following Figure 7. As the following diagram illustrate, the given XPath query is first parsed and then decomposed into several sub-queries. And based on these sub-queries, a query tree that represents the hierarchical relation of sub-queries is constructed. Once the query tree is constructed, the tree is traversed and evaluated to get the corresponding nodes. The traversing of query tree starts from the current context element. The system first retrieves the child elements of the current element as candidate elements from storage. And then the candidate elements are evaluated and elements that satisfy conditions are added to the element-set. The



〈Figure 7〉 Flow of query processing

traversing is done recursively along to the child nodes of the query tree. If all nodes of the query tree are traversed and evaluated, the element-set is returned as the result of the search.

3. 6 Relevance assessments

For an XML test collection it is necessary to obtain assessments for the following two dimensions: (1) topical relevance, which describes the extent to which the in formation contained in a document component is relevant to the topic of the request, and (2) document coverage, which describes how much of the document component is relevant to the topic of request.

To assess the topical relevance dimension, the following 4-point relevant degree scale was adopted in INEX.

To assess the document coverage, the following four dimensions has defined which shown in Table 2.

The two dimensions are orthogonal to each other, that is, relevance measures the exhaustiveness aspect of a topic, whereas coverage measures the specificity of a document component with regards to the topic. This means that a document component can be assessed as having exact coverage even if it only mentions the topic of the request(marginally relevant) or discusses only some of the topic's sub-themes(fairly relevant) as long as the relevant information is the main or only theme of the component. According to the above definitions, however, an irrelevant document component should have no coverage and vice versa.

〈Table 1〉 Assessment of the topical relevance dimension

0	Irrelevant, the document component does not contain any information about the topic of the request
1	Marginally relevant, the document component mentions the topic of the request, but only in passing
2	Fairly relevant, the document component contains more information than the topic description, but this information is not exhaustive. In the case of multi-faceted topics, only some of the sub-themes or viewpoints are discussed
3	Highly relevant, the document component discusses the topic of the request exhaustively. In the case of multi-faceted topics, all or most sub-themes or viewpoints are discussed

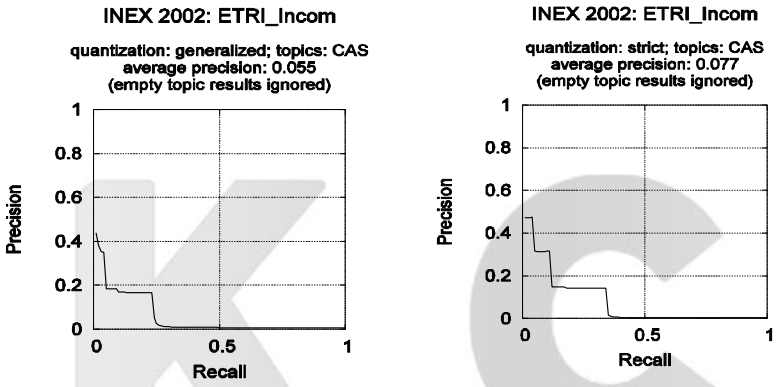
〈Table 2〉 Assessment of the document coverage dimension

N	No coverage, the topic or an aspect of the topic is not a theme of the document component
L	Too large, the topic or an aspect of the topic is only a minor theme of the document component
S	Too small, the topic or an aspect of the topic is the main or only theme of the document component, but the component is too small to act as a meaningful unit of information when retrieved by itself
E	Exact coverage, the topic or an aspect of the topic is the main or only theme of the document component, and the component acts as a meaningful unit of information when retrieved by itself

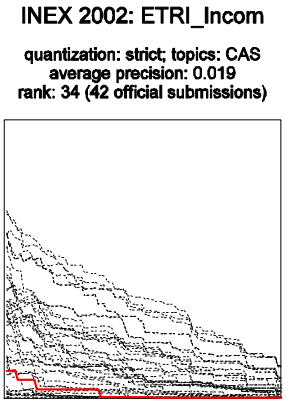
4. Results

Evaluation of the retrieval perforance of the proposed retrieval enginesfol-lowed the way of the INEX method which is based on the constructed test collection and uniform scoring techni-ques, including Recall-Precision mea-sures. The author only submitted the results of CAS queries in INEX 2002. Figure 8 presents Recall-Precision gra-phs for the evaluations results of the subsets of CAS topics, i.e., #01, #04, #05, #06, #11, #21. Applying the strict evalua-tion gave slightly higher score(average precision: 0.077) than the generalized evaluation result(average precision: 0.055).

Our overall result, rather than empty topic results ignored, showed relatively poor(average precision: 0.019) as shown in Figure 9.



〈Figure 8〉 Recall-Precision Graph for(a) Generalized and(b) Strict CAS topic ignored empty results



〈Figure 9〉 Recall-Precision Graph for Overall Results and Rank

5. Conclusion

This paper described an object-relational DBMS approach for XML documents to measure the retrieval performance as a result of INEX participant. Although the proposed approach has many benefits, for example, no delay in storing and searching XML documents, it showed relatively poor performance in overall evaluation among the participants at INEX 2002.

This result may be caused since the given topics had to be decomposed and modified to be processed by the XPath processor in EXIMA^{TM} Supply, and during this modification the original meaning of topics can be changed inevitably and some important information may missed. Some other possibilities are that because the system targets only for Korean and no support for aid tools of indices construction for INEX collection which will be investigated in the future study.

On the other hand, the proposed system lacks some of important information retrieval features such as weighting of search results, vague predicates to measure similarity, relevance feedback oriented search, semantic relativism among different XML tags.

Information retrieval(IR) techniques have traditionally been applied to search large sets of textual data and should thus be extended to encode the structure and semantics inherent in XML documents. Integrating IR and XML search techniques will enable more sophisticated search on the structure as well as the content of these documents, while leveraging the success of IR techniques in document similarity ranking and keyword search. Therefore approach to IR model, rather than DB model or maybe need a hybrid model, will be a good challenge. For example, adopting ORDBMS model for storage of XML documents and developing more sophisticated retrieval engine for XML documents. As Fuhr and GroBjohann (2001) pointed out a query language for the data-centric view should be very much in the line of database query languages, whereas the document-centric view should be supported by a language that builds on concepts developed in the area of information retrieval.

Acknowledgements

The author is grateful to Incom I&C Co. Ltd. who kindly permitted to use EXIMA^{TM} Supply. Any opinion, finding, or conclusion expressed in this paper are those of the author, and do not nec-

essarily reflect those of the sponsor.

References

Al-Khalifa, S., C. Yu and H.V. Jagadish (2003). "Querying Structured Text in an XML Database." *SIGMOD* 2003, pp. 4-15. June 9-12, 2003, San Diego, CA, USA.

Baeza-Yates, R., N. Fuhr and Y.S. Maarek(2002). "Second Edition of the XML and Information Retrieval Workshop." *SIGIR Forum,* 36(2): 53-57.

Bancihon, F., G. Barbedette, V. Benzaken, C. Delobel, S. Gamerman, C. Lecluse, P. Pfeffer, P. Richard and F. Velez(1988). "The Design and Implementation of O2, an Object-oriented Database System." *Proc. of the Second International Workshop on Object-oriented Database.*

Carmel, D., Y.S. Maarek, M. Mandelbrod, Y. Mass and A. Soffer(2003). "Searching XML Documents via XML Fragments." *SIGIR 2003,* pp. 151-158, July 28-August 1, 2003, Toronto, Canada.

Carmel, D., Y. S. Maarek and A. Soffer(2001). "XML and Information Retrieval: a SIGIR 2000 Workshop." *SIGMOD Record,* 30(1): 62-65.

Ceri, S., P. Fraternali and S. Paraboschi(2000). "XML: Current Developments and Future Challenges for the Database Community." *Proceedings of the 7th International Conference on Extending Database Technology (EDBT 2000),* pp 3-17, Konstanz, Germany.

Cheng, J., J. XU(2000). "XML and DB2." *ICDE 2000,* pp. 569-573.

Chinenyanga, T.T. and N. Kushmerick(2001). "Expressive Retrieval from XML documents." *SIGIR 2001,* pp.163-171, New Orleans, Louisiana, USA.

Despotopoulos, Y., G. Patikis, J. Soldatos, L. Polymenakos, J. Kleindienst and J. Geric(2001). "Accessing and Transforming Dynamic Content based on XML: Alternative Techniques and a Practical Implementation." In W. Winiwarter, S. Bressan, and I.K. Ibrahim, editor, *Third International Conference on Information*

Integration and Web-based Applications and Services(IIWAS 2001). Oster- reichische Comput. Gesellschaft. 2001, pp. 95-105. Wien, Austria.

Deutsch, A., M. Fernandez, D. Florescu, A. Levy and D. Suciu(1999). "XML-QL: A Query Language for XML." *Proceedings of the International WWW Conference.*

Fegaras, L. and R. Elmasri(2001). "Query Engines for Web-accessible XML Data." *VLDB 2001,* pp. 251-260.

Fernandez, M., D.F. Florescu, J. King, A. Levy and D. Suciu(1998). "Catching the Boat with Strudel: Experiences with a Web-Site Management System." *SIGMOD Record,* 27(2): 414-425.

Florescu, D. and D. Kossman(1999). "Storing and Querying XML Data using an RDBMS." *IEEE Data Engineering Bulletin,* 22(3): 27-34.

Fuhr, N., N. Goevert, G. Kazai, and M. Lalmas(2002). "INEX: Initiative for the Evaluation of XML Retrieval." *ACM SIGIR Workshop on XML and Information Retrieval,* August 2002, Tampere, Finland.

Fuhr, N. and K. GroBjohann(2001). "XIRQL: A Query Language for Information Retrieval in XML Documents." *SIGIR 2001,* pp. 172-180, September 9-12, 2001, New Orleans, Louisiana, USA.

Ha, S. and K. Kim(2001). "Mapping XML Documents to the Object-Relational Form." *ISIE 2001. 2001 IEEE International Symposium on Industrial Electronics Proceedings. Part Vol. 3,* 2001, pp. 1757-61. Piscataway, NJ, USA.

INEX homepage [Online] at: http://qmir.dcs.qmul.ac.uk/INEX/

INEX {down, up} load area [Online] at: http://ls6-www.cs.uni-dortmund.de/ir/projects/inex/download/

Jang, H., Y. Kim and D. Shin(1999). "An Effective Mechanism for Index Update in Structured Documents." *Proceedings of the 8th International Conference on Information Knowledge Management(CIKM '99),* pp. 383-390.

Kazai, G., M. Lalmas, N. Fuhr and N. Govert(2004). "A Report on the First Year of the Initiative for the Evaluation of XML Retrieval: INEX'02." *Journal of the American Society for Information Science and Technology,* 55(6): 551-556.

Khan, L. and Y. Rao(2001). "A Performance Evaluation of Storing XML Data in Relational Database Management Systems." *WIDM 2001,* pp. 31-38, Atlanta, GA, USA.

Kotsakis, E(2002). "Structured Information Retrieval in XML documents." *SAC 2002,* pp. 663-667, Madrid, Spain.

Lee, Y.K., S.J. Yoo, K. Yoon and P.B. Berra(1996). "Index Structures for Structured Documents." *Proceedings of the 1st ACM International Conference on Digital Libraries(DL '96),* pp.91-99.

Manolescu, I., D. Florescu, and D. Kossmann(2001). "Answering XML Queries Over Heterogeneous Data Sources." *VLDB 2001,* pp. 241-250.

McHugh, J., S. Abiteboul, R. Goldman, D. Quass, J. Widom(1997). "Lore: A Database Management System for Semi-structured Data." *SIGMOD Record,* 26(3): 54-66.

Miller, J. A. and S. Sheth(2000). "Querying XML documents." *IEEE Potentials,* 19(1): 24-26.

Moffat, A. and J. Zobel(1996). "Self-Indexing Inverted Files for Fast Text Retrieval." *ACM Trans. Inf. Sys,* 14(4): 349-379.

Navarro, G. and R. Baeza-Yates(1997). "Proximal Nodes: A Model to Query Document Databases by Content and Structure." *ACM Trans. Inf. Sys,* 15(4): 400-435.

Oracle. "The New XML Type Datatype." [Online] at: http://otn.oracle.com/products/oracle9i/daily/nov08.html

Schmidt, A., M. Kersten, M. Windhouwer, F. Wass(2001). "Efficient Relational Storage and Retrieval of XML Documents." *Lecture notes in Computer Science,* no. 1997: 137-150.

Shanmugasundaram, J., K. Tufte, C. Zhang, G. He, D.J. DeWitt and J.F. Naughton(1999). "Relational Databases for Querying XML Documents: Limitations and Opportunities." *VLDB,* pp. 302-314.

Shin, D(2001). "XML Indexing and Retrieval with a Hybrid Storage Model." *Knowledge & Information Systems,* 3(2): 252-261.

Tian, F., D. DeWitt, J. Chen and C. Zhang(2002). "The Design and Performance Evaluation of Alternative XML Storage Strategies," SIGMOD Record, 31(1): 5-10.

Varlamis, I. and M. Vazirgiannis(2001). "Bridging XML-Schema and

Relational Databases: A System for Generating and Manipulating Relational Databases using valid XML Documents." *DocEng'OI,* pp. 105-114, November 9-10, 2001, Atlanta, Georgia, USA.

Vieira, H., G. Ruberg and M. Mattoso (2003). "XVerter: Querying XML Data with OR-DBMS." *WIDM 2003,* pp. 37-44, November 7-8, 2003, New Orleans, Louisiana, USA.

W3C Document Object Model. [Online] at: http://www.w3.org/DOM

W3C XPath. [Online] at: http://www.w3.org/TR/xpath

Zhang, C., J. Naughton, D. DeWitt, O. Luo and G. Lohman(2001). "On Supporting Containment Queries in Relational Database Management Systems." *ACM. SIGMOD 2001,* pp. 425-436, May 21-24, 2001, Santa Barbara, CA, USA.