J. Software Engineering & Applications, 2008, 1: 26-32
Published Online December 2008 in SciRes (www.SciRP.org/journal/jsea)

Scientific Research Publishing

Storing and Searching Metadata for Digital Broadcasting on Set-Top Box Environments

Jong-Hyun Park[1], Ji-Hoon Kang[2]

[1]Software Research Center, Chungnam National University, Gung-Dong, Yuseong-Gu, Daejeon, 305-764, South Korea, [2]Dept. of Computer Science and Engineering, Chungnam National University, Gung-Dong, Yuseong-Gu, Daejeon, 305-764, South Korea
Email: {[1]jonghyunpark, [2]jhkang}@cnu.ac.kr

Received October 30th, 2008; revised November 10th, 2008; accepted November 14th, 2008.

ABSTRACT

Digital broadcasting is a novel paradigm for the next generation broadcasting. Its goal is to provide not only better quality of pictures but also a variety of services that is impossible in traditional airwaves broadcasting. One of the important factors for this new broadcasting environment is the interoperability among broadcasting applications since the environment is distributed. Therefore the broadcasting metadata becomes increasingly important and one of the metadata standards for a digital broadcasting is TV-Anytime metadata. TV-Anytime metadata is defined using XML schema, so its instances are XML data. In order to fulfill interoperability, a standard query language is also required and XQuery is a natural choice. There are some researches for dealing with broadcasting metadata. In our previous study, we have proposed the method for efficiently managing the broadcasting metadata in a service provider. However, the environment of a Set-Top Box for digital broadcasting is limited such as low-cost and low-setting. Therefore there are some considerations to apply general approaches for managing the metadata into the Set-Top Box. This paper proposes a method for efficiently managing the broadcasting metadata based on the Set-Top Box and a prototype of metadata management system for evaluating our method. Our system consists of a storage engine to store the metadata and an XQuery engine to search the stored metadata and uses special index for storing and searching. Our two engines are designed independently with hardware platform therefore these engines can be used in any low-cost applications to manage broadcasting metadata.

Keywords: *Digital Broadcasting, Metadata Management, Storing and Searching XML Data, XQuery Processing, TV-Anytime metadata, Set-Top Box*

1. Introduction

Digital broadcasting is a novel paradigm for the next generation broadcasting. Its goal is to provide not only better quality of pictures but also a variety of services that is impossible in traditional airwaves broadcasting [1]. One of the important factors for this new broadcasting environment is the interoperability among applications since the environment is distributed. As the digital broadcasting is evolving to more complex and diverse environment due to rapid increase of channels and content, the broadcasting metadata becomes increasingly important. Therefore a standard metadata for digital broadcasting is required and TV-Anytime metadata [2] that is proposed by the TV-Anytime Forum is one of the metadata standards for digital broadcasting [3].

A Set-Top Box, which is called personal digital recorders (PDR), is responsible for receiving and managing the digital content and its metadata. Currently, a Set-Top Box is designed with limited hardware and relatively software. Therefore, it is necessary to develop technologies for efficiently storing of metadata and searching stored metadata based on The Set-Top Box with low-costing and low-setting. Of course, several researches have already proposed some methods for managing metadata on digital broadcasting environment for these necessaries [4]. However, we cannot confirm whether their methods run efficiently in a Set-Top box environment because they do not consider characteristics of a Set-Top Box. We have also proposed the method for efficiently managing the broadcasting metadata in a service provider before this study [4]. The result of our research was more effective than other methods. However, to apply our previous methods into Set-Top Box has several problems such as small storage, memory size, and limited software. Consequently, there are some issues to apply general approaches for managing the metadata into Set-Top Box and we have to consider these issues.

In this paper, we propose a method for storing and searching broadcasting metadata. Also we implement the prototype using the proposed method and evaluate our method on a Set-Top Box environment with low-cost and low-setting.

[2]He is a corresponding author

The remainder of this paper is organized as follows. Section 2 describes the related work. Section 3 and section 4 shows the index for Broadcasting metadata and a method for storing and searching metadata by our prototype system, respectively. In section 5, we describe the conformance evaluation and finally, section 6 provides concluding remarks.

2. Related Work

TV-Anytime forum is organized to develop specifications to enable services based on Local Storage and TV-Anytime Metadata is one of these specifications. TV-Anytime Metadata is used to describe various TV contents and is identified by CRID (Content Reference Identifier). The metadata allows consumer to find, navigate and manage content from a variety of sources, for example, broadcast, TV, internet. XML is the "representation format" used to define the schemas of the TV-Anytime Specification. Also, TV-Anytime metadata is technically defined using a single XML schema, so it is comprised of XML data. Figure 1 shows the structure of TV-Anytime metadata and Figure 2 is its sample instance.

TV-Anytime metadata is technically defined using single XML schema, and it's comprised of XML data. Therefore the method for storing and searching TV-Anytime metadata relates with the method for XML data. Many researchers have investigated different ways of storing XML data in relational databases [4,5,6,7], native XML databases [8,9], and file systems [10,11]. Some re searches including our previous research investigated methods for storing the broadcasting metadata into relational database and searching stored metadata [4,5]. [4,5] support both XPath and XQuery languages for searching. So, two systems

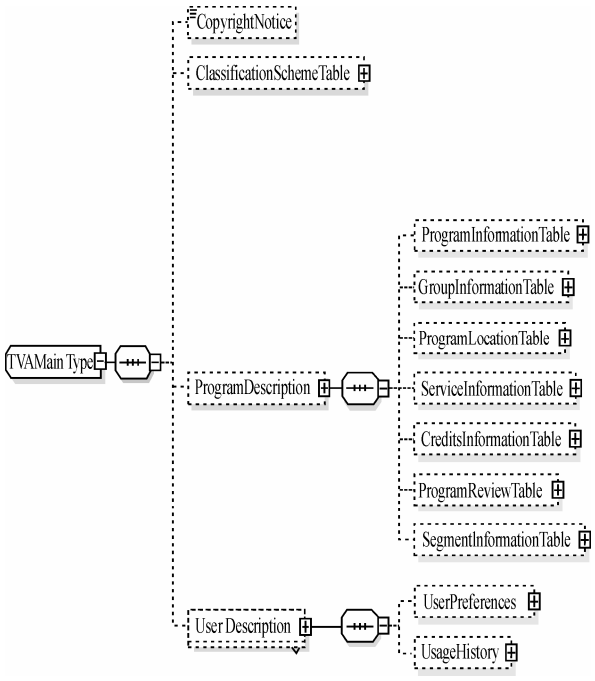


Figure 1. The structure of TV-Anytime Metadata

```
<TVAMain version="1">
<ProgramDescription>
<ProgramInformationTable>
<ProgramInformation
programId="crid://www.kbs.com/KBSNews9103052300001">
    <BasicDescription>
     <Title type="main">KBS News 9</Title>
     <Synopsis> Bank of Korea Cuts Key Rate, Kim Yu-na Captures
Skate America Title </Synopsis>
   <Keyword> Main News </Keyword>
   <Keyword> Night News </Keyword>
    </BasicDescription>
</ProgramInformation>
</ProgramInformationTable>
                        <ProgramLocationTable>
  <OnDemandProgram>
   <Program
crid="crid://www.kbs.com/KBSNews9103052300001"></Program>
   <ProgramURL>D:\Media\News\news_9.mpg</ProgramURL>
                         </OnDemandProgram>
</ProgramLocationTable>
<SegmentInformationTable timeUnit="PT1001N30000F">
<SegmentList>
<SegmentInformation segmentId="SID_0_0_148"> . . .
</SegmentInformation> . . .
</SegmentList>
</SegmentInformationTable>
</ProgramDescription>
</TVAMain>
```

Figure 2. A sample instance of TV-Anytime Metadata

have a module to convert from user query to SQL query and use a specialized indexing method for efficient searching (quick processing of selection, projection, and join). However these two systems use a commercial relational database management system to manage the large volume of metadata because they only focused on service provider systems. Of course, it seems that it is a natural choice to use the RDBMS or Native XML DB because the content service provider has to mange not only the large volume of broadcasting metadata but also a lot of multimedia contents. However, their cost is expensive for STB with low-cost and low-setting.

3. Index for Broadcasting Metadata

In order to store broadcasting metadata, we select the file system because of the cost and hardware power. Although we choose the file system, the basic idea for storing is similar to our previous approach for storing TV-anytime metadata into a relational database. In other words, the basic approach for storing is based on binary approach [12] and the approach for assigning an identifier into a node is the Dewey number labeling [13] to keep a parent-child relationship.

Also we use the path table concept [14] for direct accessing to every nodes and node position concept for obtaining partial document from the metadata instance. Every node which has same name is stored in a single file and information for searching is addressed by the index file. Figure 3 shows the structure for indexing a 'b' node.

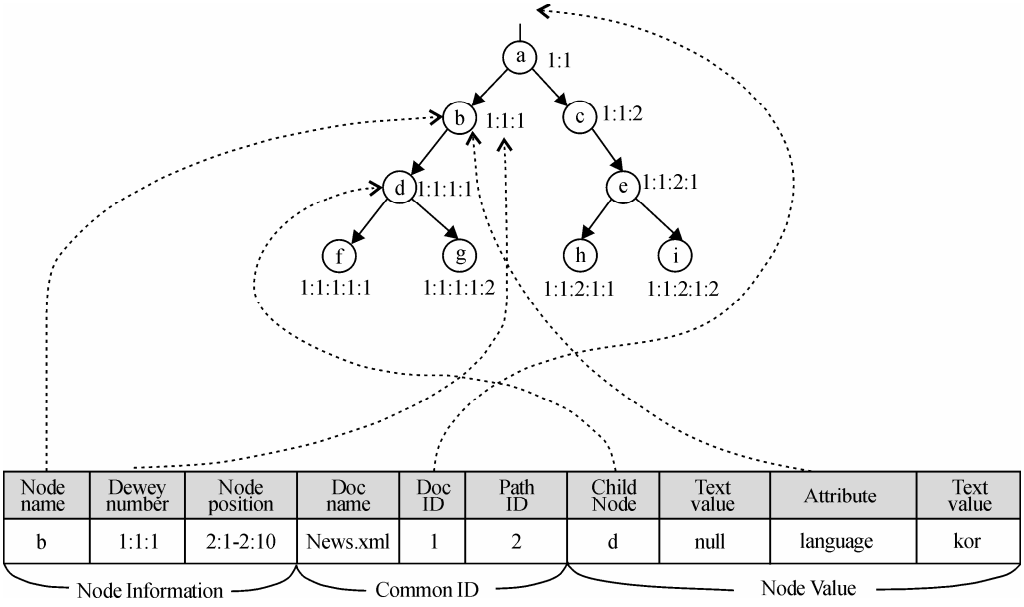


Figure 3. The structure of index

The structure for indexing a node consists of node information part, common ID part, and node values part. In the node information part, we store the name, ID, and position which address the position of current node in original TV-Anytime metadata instance. The common ID part includes the name and ID of TV-Anytime metadata instance and Path ID which links with the XPath expression from root node to current node. The node value parts stores the information of child nodes and attribute nodes.

Figure 4 shows an example XQuery query, Path Index, Node Index and document tree for obtaining result of the query briefly. In order to process the example XQuery query, a node has to satisfy following conditions. The full path expression to'd' node from root node is 'a/b/c/d', and its value have to contain "KBS News 9". Also the parent node 'c' of the 'd' node must have 'Month' attribute and its value have to equal to 'May'. If a node satisfies these conditions, we can obtain the partial documents of TV-Anytime metadata instance including the node by the Node_Position.

4. Metadata Management System for Storing and Searching

The goal of Metadata Management System is to store and search metadata efficiently in a Set-Top Box environment for digital broadcasting. Figure 5 shows the architecture and function of the metadata management system in the Set-Top Box. Our metadata management system consists of the Storage Engine and the XQuery Engine.

As shown in Figure 6, Storage Engine provides basically four interfaces: InsertDoc, DeleteDoc, UpdateDoc, and GetDoc for inserting, deleting, updating, and retrieving a metadata instance, respectively. In order to generate and store an index file including a metadata, InsertDoc parses the metadata received from Metadata Generator or Metadata Editor and then extracts and stores the information from the parsing Tree. DeleteDoc deletes the metadata matched with the user-inputted CRID. UpdateDoc deletes the old metadata that has the same CRID as the new metadata, and then inserts the new metadata. Since XQuery doesn't support update of XML data, we use the delete and insert instead of update command.

In this paper, we propose to use XQuery as query language for searching the broadcasting metadata. Since XQuery is standard query language proposed by W3C for querying XML data, it guarantees interoperability between digital broadcasting applications including a Set-Top Box. An XQuery Engine consists of an XQuery parser module for query validation and a SearchDoc module for query execution. The input of XQuery Engine is the XQuery query, and its output is either the whole document or one part of the document. Figure 7 shows the architecture of XQuery Engine for a search of stored metadata.

(1) Input XQuery query

```
For $d in input ("TVAnytime")
For $p1 in $d/a/b/c
For $p2 in $p1/d
Where contains (string($p2), "KBS News9") and $p1/ @ Month="May"
return <returns> {$p2} </returns>
```

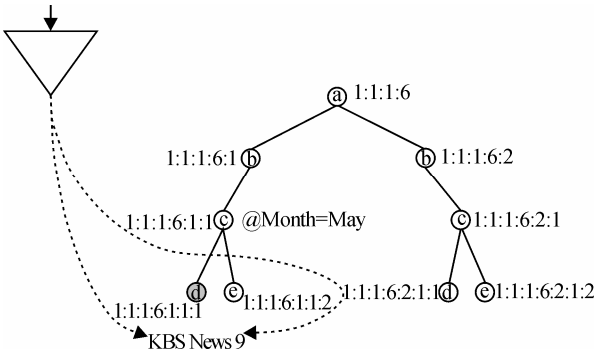
(2) Path Index

Path ID	Full Path expression
62	a/b/c
63	a/b/c/d
64	a/b/c/e

(3) Each Node Index

Node_Name	Dewey_Number	Node_Position	Doc_Name	Doc ID	Path ID	Child_Node	..	Attribute	..	Text_Value
d	1:1:1:6:1:1:1	9:13~9:45	News01.xml	1	63	null	..	null	..	KBS News 9
c	1:1:1:6:1:1	92:19~97:20	News01.xml	1	62	1:1:1:6:2:2:1	..	Month	..	null

(4) Document Tree



(5) Result Composer

<returns>KBS News 9</returns>

Figure 4. An example for processing a XQuery query

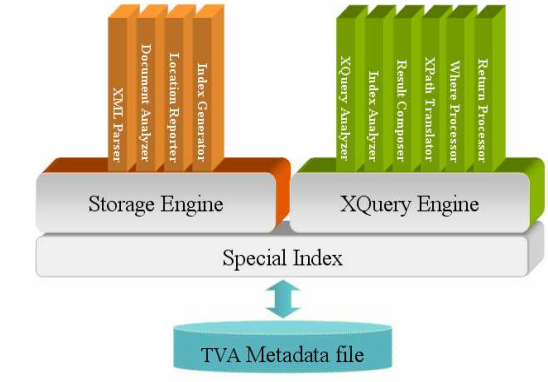


Figure 5. The architecture of TV-anytime metadata management system

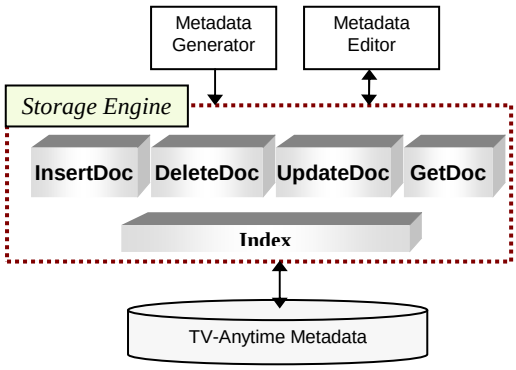


Figure 6. The architecture of storage engine

XQuery Analyzer gets a query in XQuery, parses the query using an XQuery parser and generates its syntax tree. XPath Translator module creates an XPath expression which consist of full path expression to current node from root node by merging XPath expressions defined in FOR and LET clauses in XQuery queries. WHERE Processor and RETURN Processor are used for processing conditions defined in a WHERE clause and for constructing the result structure defined in RETURN clause, respectively. Index Analyzer parses the index files and generates the information for obtaining result metadata fragments from the storage by using the selected index. Result Composer constructs the final result using the result structure and result metadata fragments.

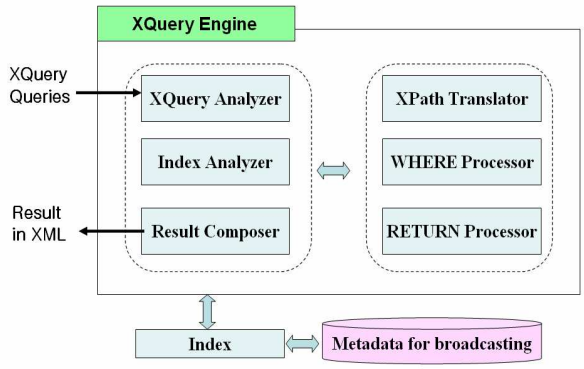


Figure 7. The architecture of XQuery engine

5. Performance Evaluation

In order to evaluate whether our choice of the strategies for the issues is relevant, we compare our prototype system with other general-purpose XQuery Engine and test their performance for various typical queries. We select two popular general XQuery Engines. One is the Oracle XQuery Engine [10]. The other is a Saxon-B XQuery Engine [11]. Two XQuery Engine it is all free source, a JAVA base, and a head of a family general XQuery Engine. The experimental setup is as follow: the CPU is Intel Pentium III Process 750 MHz, the memory size is 256 MB, the JDK version is 1.4 and the OS is LINUX 2.4.2.

Our system uses XQuery, which is a sub set of XQuery 1.0 (e.g. is not support 'OR' in WHERE clause and '//' in XPath path express). From the previous work [4, 15, 16, 17], we have found that the query processing performance depend on the XPath expression, number of predicates, and result size. By considering these factors, I use the XQuery in Table 1.

We omit some expressions in example queries except Q1. For example, the constructor '<Results>' is omitted because that is the same as in Q1. The queries Q1, Q2 and Q3 use single condition which is declared in the WHERE

Table 1. Example XQuery queries for experiment

	WHERE Conditions / RETURN Value
	XQuery query

Q1	*Single condition/ Single terminal node*
	<Results>{
	for $d in input("TVAnytime")
	return <Result>{
	for $p1 in
	$d/TVAMain/ProgramDescription/
	ProgramInformationTable/ProgramInformation
	for $p2 in $p1/@programId
	for $p3 in $p1/BasicDescription/Title
	where $p2="crid://www.kids17.net/amigonme
	103042200049"
	return <node>{ $p3 }</node>
	}</Result> }</Results>

Q2	*Single condition/ Single root node* for $p1 in $d/TVAMain for $p2 in $p1/ProgramDescription/ProgramInformation Table/ProgramInformation/@programId where $p2="crid://www.kids17.net/amigonme 103042200049" return <node>{ $p1 }</node>
Q3	*Single condition/ Multiple terminal nodes* for $p1 in $d/TVAMain for $p2 in $p1/ProgramDescription/ProgramInformationTable/ ProgramInformation/BasicDescription/Genre/Name where $p2="Education" return <node>{ $p1 }</node>
Q4	*Three conditions/ Single root node* for $p1 in $d/TVAMain for $p2 in $p1/ProgramDescription/ProgramInformation Table/ProgramInformation/BasicDescription for $p3 in $p2/Language for $p4 in $p2/ProductionDate/TimePoint for $p5 in $p2/ReleaseInformation/ReleaseDate/ DayAndYear where $p3="ko" and $p4>="2006" and $p5="2006-04-14" return <node>{ $p1 }</node>
Q5	*Five conditions/ Multiple root nodes* for $p1 in $d/TVAMain for $p2 in $p1/ProgramDescription/ProgramInformation Table/ProgramInformation/BasicDescription for $p3 in $p2/Language for $p4 in $p2/ProductionDate/TimePoint for $p5 in $p2/ReleaseInformation/ReleaseDate/ DayAndYear for $p6 in $p1/ProgramDescription/ProgramLocation Table/BroadcastEvent/Live/@value for $p7 in $p1/ProgramDescription/ServiceInformation Table/ServiceInformation/Name where $p3="ko" and $p4>="2006" and $p5="2006-04-14" and $p6="true" and $p7="KBS" return <node>{ $p1 }</node>
Q6	*Three conditions/ Multiple terminal & root nodes* for $p1 in $d/TVAMain for $p2 in $p1/ProgramDescription/ProgramInformation Table/ProgramInformation/BasicDescription for $p3 in $p2/Title for $p4 in $p2/Language for $p5 in $p2/ProductionDate/TimePoint for $p6 in $p2/ReleaseInformation/ReleaseDate/ DayAndYear for $p7 in $p1/ProgramDescription/ProgramLocation Table/BroadcastEvent/Live/@value for $p8 in $p1/ProgramDescription/ServiceInformation Table/ServiceInformation/Owner for $p9 in $p1/ProgramDescription/CreditsInformation Table/PersonName for $p10 in $p1/ProgramDescription/ServiceInformation Table/ServiceInformation/ParentService where $p3="KBS News 9" and $p4="ko" and $p5>="2006" and $p7="true" and $p8="KBS" return <node>{ $p3, $p4, $p5, $p6, $p9, $p10 }</node>

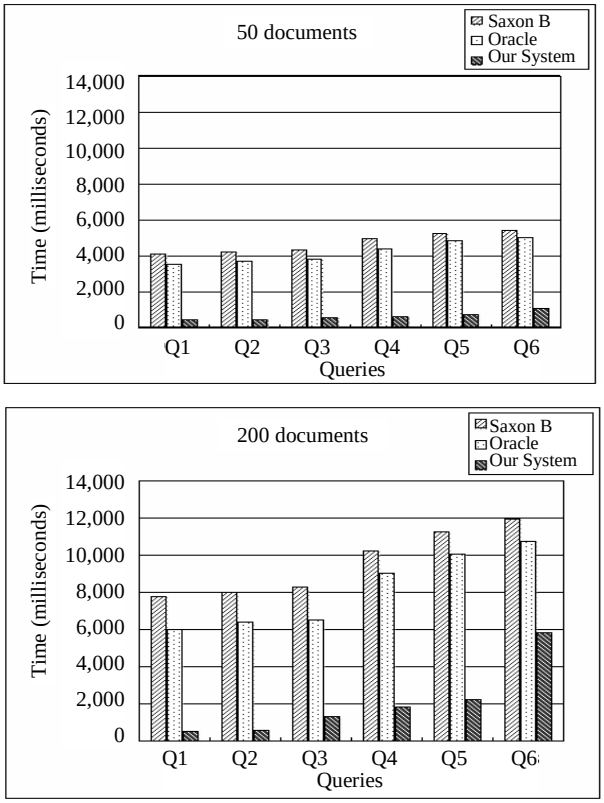


Figure 8. Comparison of query processing times

clause. However, the result data sizes are expected different because the result of each query is a leaf node, an root node, and multiple root nodes together with their descendent nodes, respectively. Q4, Q5 and Q6 use different number of conditions. The return value of each query is a single root node, multiple root nodes, and multiple terminal and root nodes, respectively.

Figure 8 summarizes the performance. The numbers of the test data are 50 and 200 TV-Anytime metadata instances respectively. The result shows that our system outperforms other methods for any queries except Q6. In case of Saxon B and Oracle, the complex queries Q4 and Q5, takes more execution time than simple query Q1, Q2, and Q3. However, our system does not so depend on the queries. In case of our system, Q6 takes more execution time than the other queries since we need time to compose result. However the case of Q6 is not general, because the result size of user queries is not large volume in a Set-Top Box, generally.

Figure 9 summarizes the scalability property of the systems. The size of the test data is 50 documents, 100 documents, 150 documents and 200 documents, respectively. In case of Saxon B and Oracle, the processing time increases linearly as the number of data increases. However, the processing time of our system is independent of the data size for searching. The result of the evaluation shows that our system outperforms so that our approach is believed to be on of the efficient approaches for managing metadata in the Set-Top Box.

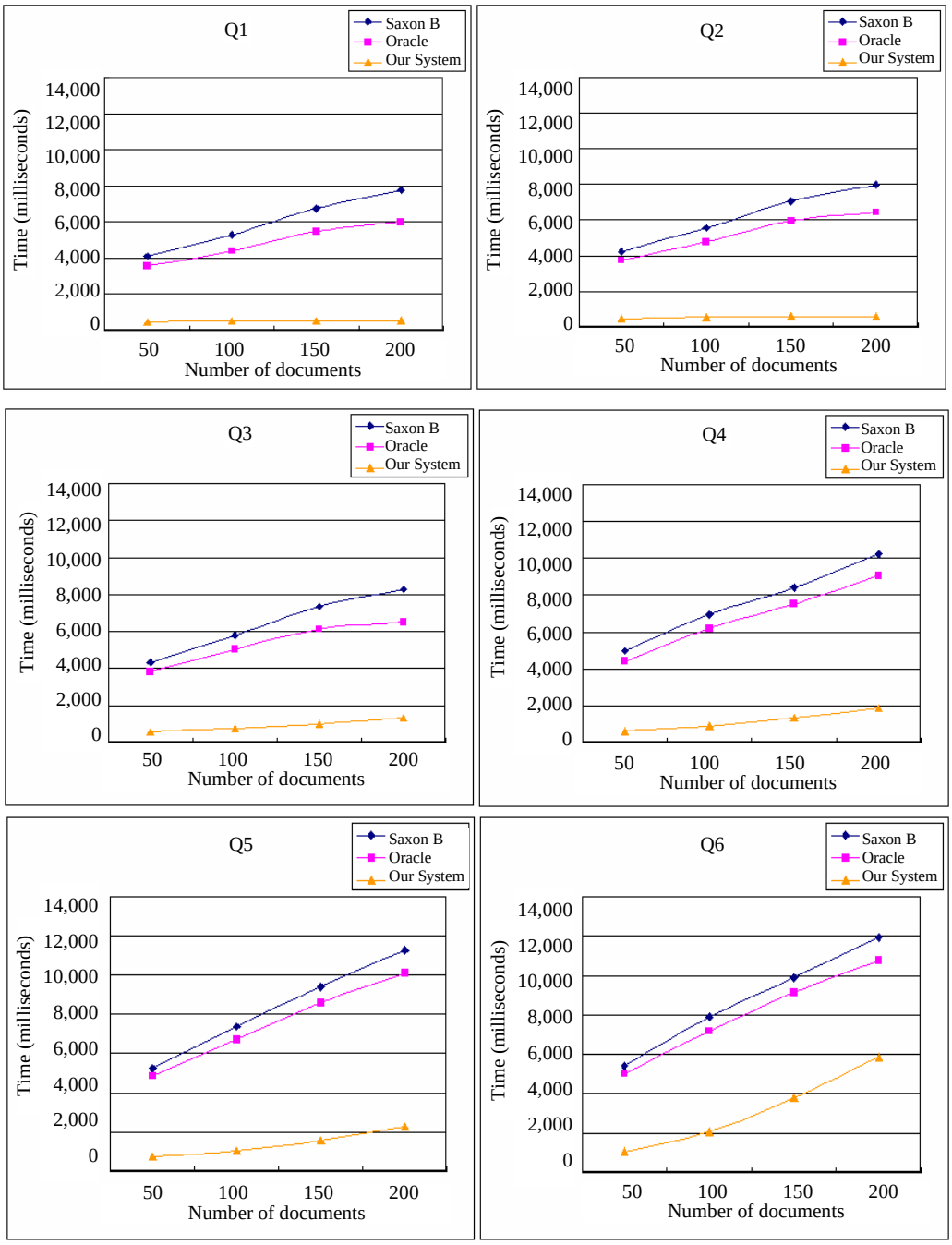


Figure 9. Performance evaluation for scalability property

6. Conclusions

In this paper, we have proposed a method for storing and searching TV-Anytime metadata for digital broadcasting based on a Set-Top Box which is low-cost and low-setting. Also we have implemented a prototype system for applying our method and evaluated our approach which seems important since our prototype system outperforms the other compared systems. Our system was developed on digital broadcast environments [18]. However our result can be applied to any XML management systems that focus on the performance of store and retrieval on low-cost environments.

7. Acknowledgement

This research is supported by MKE & IITA(08-Infrastructure-13, Ubiquitous Technology Research Center), and also by Foundation of ubiquitous computing and networking project (UCN) Project, the Ministry of Knowledge Economy (MKE) 21st Century Frontier R&D Program in Korea and a result of subproject UCN 08B3-O1-30S.

REFERENCES

[1] S. Pfeiffer and U. Srinivasan, "TV anytime as an application scenario for MPEG-7," In Proceedings ACM Multimedia 2000, Los Angeles, October 2000.

[2] "TV-anytime phase 1," Part 3 Metadata, ETSI TS 102 822−3−1, Vol. 1.1.1, October 2003.

[3] TV-Anytime Forum Website: http://www.tv-anytime.org.

[4] J. H. Park, J. H. Kang, B. K. Kim, Y. H. Lee, M. W. Lee and M. O. Jung, "An XQuery-based TV-anytime metadata management," Proceedings of DASFAA'05 Conference, April 2005.

[5] H. S. Shin, "A storage and retrieval method of XML-based metadata in PVR environment," IEEE Transactions on Consumer Electronics, Vol. 49, No. 4, pp. 1136−1140, November 2003.

[6] D. Florescu and D. Kossmann, "Storing and querying xml data using an RDBMS," IEEE Data Engineering Bulletin, Vol. 22, No. 3, 1999.

[7] I. Tatatinov, S. D. Viglas, K. Beyer, J. Shanmugasundaram, E. Shekita, and C. Zhang, "Storing and Querying ordered XML using a relational database system", Proceedings of ACM SIGMOD Conference, June 2002.

[8] ORACLE, "Berkeley DB introduction," http://www.oracle.com/database/berkeley-db/.

[9] T. Fiebig, S. Helmer, C. C. Kanne, J. Mildenberger, G. Moerkotte, R. Schiele, and T. Westmann, "Anatomy of a Native XML Base Management System," Technical Report 01, University of Mannheim, 2002.

[10] ORACLE, "Oracle XML data synthesis or XDS," http://www.oracle.com/technology/tech/xml/xds/.

[11] SAXONICA, "SAXON XQuery Engine," http://www. saxonica.com/.

[12] D. Florescu and D. Kossmann, "Storing and querying XML data using an RDBMS," IEEE Data Engineering Bulletin, Vol. 22, No. 3, pp. 27−34, September 1999.

[13] I. Tatarinov, S. D. Viglas, K. Beyer, J. Shanmugasundaram, E. Shekita, and C. Zhang, "Storing and querying ordered xml using a RDB system," Proceedings ACM SIGMOD Conference, June 2002.

[14] M. Yoshikawa, T. Amagasa, T. Shimura, and S. Uemura: "XRel: a path-based approach to storage and retrieval of XML documents using RDBs," Proceedings ACM Transactions on Internet Technology, Vol. 5, August 2001.

[15] T. Grust, "Accelerating XPath location steps," Proceedings of the ACM SIGMOD Confefence, pp.109−120, June 2006.

[16] M. Barg and R. K. Wong, "A fast and versatile path index for querying semi-structured data," Proceedings of the DASSFAA'03 Conference, pp. 249−256, March 2003.

[17] S. Hidaka, H. Kato and M. Yoshikawa, "A relative cost model for XQuery," Proceedings of the SAC'07 Conference, March 2007.

[18] K. Kang, J. G. Kim, H. K. Lee, H. S. Chang, S. J. Yang, Y. T. Kim, H. K. Lee, and J. W. Kim, "Metadata broadcasting for personalized service: A practical solution," ETRI Journal, Vol. 26, No. 5, pp. 452−466, October 2004.

