I.J.Modern Education and Computer Science, **2011, 2, 51-60**
Published Online April 2011 in MECS (http://www.mecs-press.org/)

A Scheme for Evaluating XML Engine on RDBMS

Guannan Si, Zhengji Zhou, Nan Li, Jing Xu, Jufeng Yang
College of Information Technical Science
Nankai University
Tianjin, China
sign@mail.nankai.edu.cn

Abstract— **There are an increasing number of DBMS vendors thinking of integrating XML data management into traditional relational database, with wider use of XML. In this case, a comprehensive evaluation methodology is needed to evaluate the XML engine in RDBMS correctly. In this paper, we analyze the characteristics of XML engine and propose an evaluation strategy of XML engine in a RDBMS. We believe that the evaluation should include functional evaluation and performance evaluation, and cover several major aspects of DB such as storage, query and update. Then we designed an evaluation scheme for the XML engine in RDBMS according the strategy. The scheme describes an evaluation scene and contains a data set, workload and index set. The data set reflects the characteristics of both data-centric and document-centric XML data. The workload covers all of the requirements of XQuery in W3C. The index set covers the aspects of storage, indexing, query and update. In the end, we complete an experiment to test an actual computer system using the proposal. The result shows that the proposal is proper.**

Index Terms—DBMS; XML; W3C; XQuery; Evaluation

I. INTRODUCTION

XML (Extensible Markup Language) is becoming a standard format for representing and exchanging data with the development of software communicating via the Internet. There are a great number of XML documents that are created by wide application of XML in more and more area. To manage large-scale XML documents effectively has been a research subject crying out for solutions in database research.

The best way to manage a large number of XML data is to store them into databases. Two main types of databases are promoted to manage XML data: Native XML databases and XML-Enabled databases. Native XML databases, like Software AG Tamino, Apache Xindice and Wolfgang Meier eXist, are tailored to XML requirements and thus promise performance benefits and improved support for specific XML requirements. They have a storage scheme and a query engine suited for XML, which can manage XML naturally. But the most important problem of Native XML databases is that they must reimplement many fully-fledged theories and technologies which have been researched and practiced for more than thirty years in field of database, such as storage management, transaction management, lock management, backup and recovery management, etc. On the other hand, XML-Enabled databases, typically relational databases, such as DB2, SQL Server, Oracle, provide extensions for transferring data between XML documents and themselves. Such databases are generally designed to store and retrieve data-centric XML documents. For these systems, XML management modules are integrated with relational databases in order that they can support existed relational data and additional XML data, which will make XML databases more useful applications.

For this reason, a number of database vendors, such as IBM, Oracle, Microsoft, are committing themselves to develop the technology of integrating XML engine with RDBMS, in order to describe XML data model and reuse existing system module. Nowadays, many new products have supported XML as well as XPath/XQuery language and SQL/XML: 2003 standard in various ways.

It is inevitable that special evaluation methods are used to assess the integrated XML module of RDBMS, with the development of XML engine in RDBMS and the requirement of XML data increasing. In this paper we propose an evaluation scheme according to characteristics of XML engine on RDBMS, and prove usability of the scheme.

The rest of the paper is structured as follows: in the next section we briefly discuss related benchmark work. We describe the design of the evaluation scheme in Section 3. In Section 4 we use the proposal to test a real system and analyze the result. Finally, we indicate the current directions that we are pursuing in Section 5.

II. RELATED WORK

Benchmark is an accepted method and standard of assessing performance characteristics of computer in IT industry. It is the act of running a computer program, a set of programs, or other operations, in order to assess the relative performance of an object. It is normally implemented by running a number of standard tests and trials against it.

There are many benchmarks for RDBMS, like the widely used family of TPC benchmarks. But all the elements of these benchmarks including data and workload are based on RDBMS, which is unavailable for XML databases. As XML data is widely used, many

benchmarks for XML databases have been proposed, such as: XMark[1], XOO7[2], XBench[3], XMach-1[4], TPoX[5], XPathMark[6], MBench[7], MemBeR[8]. They assess XML databases in various aspects.

XMarch-1 is a web-based multi-user benchmark for XML data management. Its database contains a directory structure and XML documents. Its query set is made up of a mix of XML queries and update operations for which system performance is determined. XMark is a single-user benchmark. Its database models an Internet auction site. It provides a more comprehensive set of queries to evaluate query performance of the system under test. XOO7 derives from OO7, a benchmark for object-oriented DBMS. Besides mapping the database from OO7, XOO7 also maps the original queries of OO7, and adds some specific queries for XML database. XBench categorizes XML databases into four database domains by their Application characteristics and Data characteristics, and uses XML Query Use Cases to design separate workloads for each of them. TPoX is developed by IBM's DB2 Performance and Development group mainly applied to their product, DB2. Its scenario is a simplification of a real-world brokerage application and the workload consists of a set of queries, inserts, updates, and deletes. In the benchmarks mentioned previously, XMark, XOO7, XBench, XPathMark, MBench and MemBeR place more weight on offering query challenges that are designed along the lines of XML query algebras, helping to analyze and improve the underlying query processor. The only performance metrics of them is response time. These benchmarks are not enough to evaluate the overall performance of a full-fledged XML database system. XMach-1, TPoX are application-oriented benchmarks which focus on evaluating a complete database system. More detailed analysis and comparison of the benchmarks can be found in [9], [10], [11] and hence is not repeated here.

We find that none of all these benchmarks is designed for the XML engine in RDBMS. We propose a list of requirements for a benchmark of XML engine in RDBMS drawing from our own experience of researching RDBMS, XML databases and XML engine. Subsequently we design and implement BenchXE as an attempt to meet these requirements.

III. DESIGN OF THE SCHEME

A. Evaluation Strategy of XML Engine in RDBMS

A XML engine in RDBMS is a module of RDBMS that can manage XML data. So the following issues should be considered, for evaluating the performance of XML engine in RDBMS correctly.

- Storage management. XML data is semi-structured. Storage concepts such as tables, rows and columns are structured. A XML engine should store both content information and structure information of XML data in relational tables nondestructively. When needed, the XML data, both content information and structure information should be retrieved from databases and be reconstructed accurately. So the evaluation scheme must contain operation of storing, retrieving and checking XML data.
- Indexing structure. A XML engine should create various structure indexes based on traditional indexing structure of RDBMS. These indexes can be used to locate the nodes which meet the structural relation indicated quickly. To assess indexing capability, the evaluation scheme should have a method to compare throughput of XML engine with and without indexing.
- Query optimization. A XML engine put XML data in DBMS environment, and integrates XML engine in RDBMS seamlessly. On this basis, Structural Summary Index can play a role in constructing optimized queries. The evaluation scheme should design proper queries to evaluate the query optimization.
- Extending basic SQL grammar, and supplying XML data management language. A XML engine should implement SQL/XML, an extension of SQL standard defined by ISO/IEC 9075-14:2003, in a relational query module. And it should also support the statements of embedding XPath/XQuery in SQL. Queries of the evaluation scheme should include not only SQL statements that operate relational data but also XPath/XQuery statements that operate XML data.
- Update operation and online transaction processing. For RDBMS, concurrency control module, being made up of transaction manager and lock manager, communicates with storage manager. It ensures the logical correctness when several transactions update data concurrently. A XML engine should integrate update operation of XML data with RDBMS transaction processing architecture which is fundamentally different in conceptual model. In this case, the evaluation scheme should also assess capability of update operation.

B. Evaluation Scene

The key criteria of a database benchmark are domain-specific, relevant, portable, scalable and simple. Meeting the key criteria, we design an evaluation scene that simulates a literature management system. The evaluation scene is shown in Fig. 1.

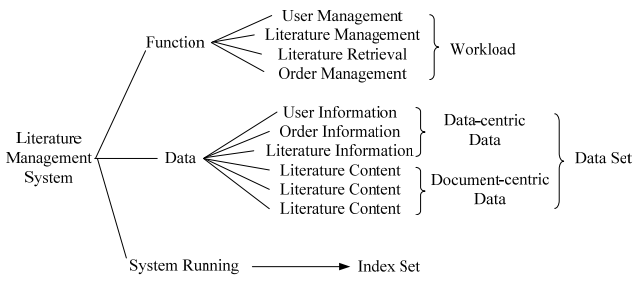

Figure 1. evaluation scene

We often search articles we need by various document retrieval systems, so we can comprehend their characteristics of data and process. For simplicity we decided not to simulate all real-world application logic. But the system is representative in all the aspects of data, transactions and XML schema. The system can be recognized from data set, workload and index set.

- Data set. Data set is made up of data information. Data information defined in the system includes user information, order information, literature information and literature content. User information, order information and literature information are data-centric XML data. Literature content contains a lot of XML documents which are document-centric XML data.
- Workload. Workload comprises four types of system function: user management, literature management, literature retrieval and order management. They simulate common functions of the system.
- Index set. When the literature management system is running on a SUT (system under test), a set of data can be record. These data are the index to assess performance of the SUT. They constitute the index set.

All these three parts of the evaluation scheme are presented in detail in next sections.

C. Design of Data Set

XML documents are characterized to two classes: data-centric documents and document-centric documents. Data-centric documents are documents that use XML as a data transport. They are characterized by fairly regular structure, fine-grained data, and little or no mixed content. They are designed for machine consumption. Examples of data-centric documents are sales orders, flight schedules, scientific data, and stock quotes. Document-centric documents are documents that are designed for human consumption. They are characterized by less regular or irregular structure, larger grained data, and lots of mixed content. They are usually written by hand in XML or some other format, such as RTF, PDF, or SGML, which is then converted to XML. Examples of document-centric documents are books, email, advertisements, and almost any hand-written XHTML document.

A XML engine can operate both data-centric documents and document-centric documents. So, data set of the evaluation scheme also includes these two classes of documents. The data set contains entities of User, Category, Order, Databases and Literature. There are references between entities, for example, Order has references of Database, Category and User. They are database--id, category—id and user—id elements. Fig. 2 shows the hierarchy of entities. The figure doesn't show attributes of entities due to limited space.

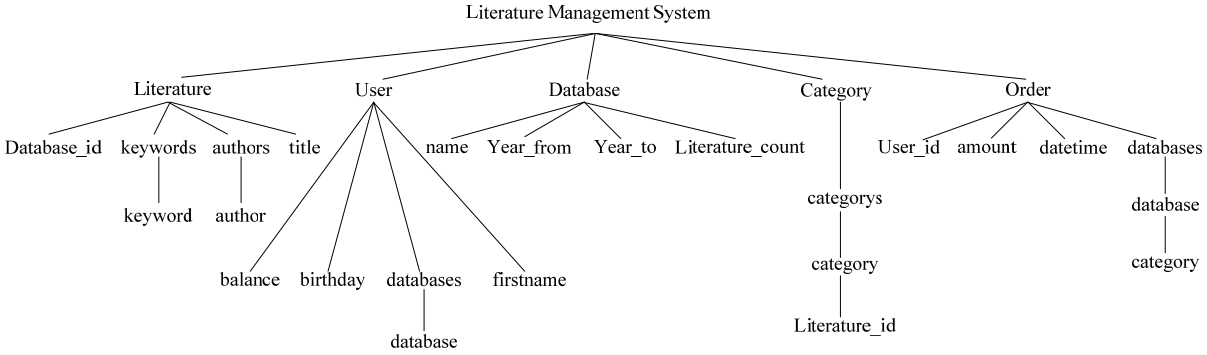

Figure 2. hierarchy of entities

User, Database, Category and Order are simulated data-centric documents that are generated by the schema, as Fig. 2 shows. For instance, Order shows the characteristic of data-centric XML data because of regular structure and the focus of data recording. The characteristic of document-centric XML data is mainly shown by Literature entities. Contents of Literature entities are real data which are obtained in various ways (e.g. real XML documents like websites content, books of digital library and DBLP documents from internet). All of the documents are organized to the format that we need. So the whole data set can have the characteristics of both data-centric XML data and document-centric XML data. We believe that the data set can reflect the characteristics of XML data properly.

D. Design of workload

The workload is defined according to the W3C standard. Each of the statements embodies some features of XML data processing, so that the statement set can cover all the functionalities that are necessary for XML engine.

1) Queries

We design fourteen queries. Each query performs certain functionality. Constants in statements are generated randomly, for example "32655062" in Q4. Queries are listed as follows:

Q1: Return user information by id, including databases and categories that he orders.

Retrieving detail information by id is to verify a XML engine is able to preserve the identity of items. Listing a list of databases and categories is a validation of collections operation. Querying several elements according to references among User, Order, Database and Category elements can test that a XML engine is able to combine related information from different parts of a given document or from multiple documents. All of the requirements are defined in XQuery standard.

Q2: Return all of users who order a category (e.g. agriculture).

This query uses references among Category, Order and User entities to verify that a XML engine is able to traverse intra- and inter-document references.

Q3: List databases that are order by a user, and the results should be separated by certain delimiters.

This query is to verify that a XML engine is able to express simple conditions on text, including conditions on text that spans element boundaries. In XQuery, this requirement is satisfied by the ability of the string () function to return the text content of an element, including text within sub-elements.

Q4: Judge the existence of databases that have more than 32655062 literatures.

This query is to verify that operations on collections of a XML engine include support for existential quantifiers.

Q5: Judge if all the databases have more than 32655062 literatures.

This query is to verify that operations on collections of a XML engine include support for universal quantifiers.

Q6: List all the categories of a level and all of their immediate children categories.

This query is to verify that a XML engine supports operations on hierarchy of document structures. The hierarchy of elements in input documents should be preserved in results. Additionally, the function of intra-documents is tested by this query.

Q7: List first author of every literatures.

This query is to verify that a XML engine supports operations on sequence of document structures. The XML engine need not only retrieve the certain item from a sequence of document structures, but also return a result with correct sequence.

Q8: Calculate quantity of orders, the total amount for the date, maximum amount, minimum amount and average amount of a date (e.g. 2005-7-22).

This query is to verify that a XML engine is able to compute summary information from a group of related document elements (This operation is sometimes called "aggregation"), and sort query results

Q9: Nested query for new orders that are subscribed by users who have the account balance of 2981.

A nested query is constructed. Its sub query is used as an operand to verify that a XML engine supports expressions in which operations can be composed, including the use of queries as operands.

Q10: List users who have no order.

If a user has no order, database and category elements of the User entity are both NULL. Therefore, all operators, including logical operators, should take NULL values into account. It is verified that a XML engine include support for NULL values.

Q11: Return information of a database by database name.

A XML engine should be able to operate on literal fragments of an XML document such as

<name><first>Joe</first><last>Doe</last></name>.

This query constructs a XML literal fragment that contains database name. The literal fragment is used as a condition to be compared with information in database. The query tests performance of a XML engine to operate literal fragment.

Q12: Create a function that receive user id and date and return information of an eligible order.

A XML engine should support the use of externally defined functions. The interface to such functions should be defined by the XML engine, and should distinguish these functions from functions defined in query language. It means that the implementation of externally defined functions is not part of the query language. This query defines a simple query, and put it in a user-defined function. It can test extensibility of the XML engine.

Q13: Calculate number of users for each age group.

A XML engine should provide access to information derived from the environment in which the query is executed, such as the current date, time, locale, time zone, or user. The function is tested by a query which contains a condition that system time minus birthday of a user.

Q14: Retrieve literatures that include a certain key word (e.g. customer).

This query is to verify that a XML engine supports full-text search.

2) Insert, Update, Delete

There are five update/delete/insert statements defined in this paper.

U1: Update a user's account balance by user id when he orders something.

This update is to verify that a XML engine is able to change the properties of existing nodes while preserving their identity. The XML engine should also be able to create a new copy of a node with a specific set of changes and change the value returned by the typed-value accessory for a node.

U2: Delete <new—order> tag after executing an order.

This update is to verify that a XML engine is able to delete nodes. The XML engine should also be able to modify some of the properties of a node such as the name, type, content, knelled, base-URI, etc.

U3: Insert child categories into a category.

This update is to verify that a XML engine is able to insert new nodes in specified positions.

U4: Update database and category of a user after his subscribing.

This update is to verify that a XML engine is able to replace a node according to certain condition. The XML engine should also be able to compose update operators with other update operators.

U5: Update the category that a user orders by a received parameter.

This update is to verify that a XML engine provides a means to parameterize update operations. Updating functions and external variables are both used to parameterize update operations.

TABLE I. MAPPING BETWEEN WORKLOAD STATEMENTS AND W3C REQUIREMENTS

Requirements	Statements
Supported Operations	Q1, Q2
Text and Element Boundaries	Q3
Universal and Existential Quantifier	Q4, Q5
Hierarchy and Sequence	Q6, Q7
Combination	Q1, Q2
Aggregation	Q8
Sorting	Q8
Composition of Operations	Q9
NULL Values	Q10
Structural Transformation	Q3, Q6, Q7
References	Q2, Q6
Identity Preservation	Q1
Operations on Literal Data	Q11
Operations on Names	Q3, Q6
Extensibility	Q12
Environment Information	Q13
Full-Text Search	Q14
Locus of Modifications	U1
Delete	U2
Insert	U3
Replace	U4
Changing Values	U1
Modifying Properties	U2
Conditional Updates	U4
Iterative Updates	Multi-Statements
Validation	Schema Validation
Compositionality	U4
Parameterization	U5

Table 1 lists the mapping between workload statements and W3C requirements. It shows that every requirements of W3C standard can be implemented by a statement or an operation like schema validation and concurrent load.

E. Design of the index set

The index set can be used for evaluating functionality and performance of a XML engine. It covers the aspects of storage, indexing, query and update, as Table 2 shows.

TABLE II. DESIGN OF INDEXES

Evaluation Interests	Evaluated Functions	Indexes
Storage ability	Storage of XML data in RDBMS	Time cost of loading data
		Space cost of loading data
	XML index structure	Time cost of indexing
		Space cost of indexing
		Time cost of index maintenance
Query ability	Coverage of query functionality	Coverage of all the requirement
	Ability of basic query: simple query to values, attributes, etc.	Response time with secondary index
		Response time without secondary index
		Response time with schema
		Response time without schema
	Ability of complex query: optimization of structural join algorithm in XML query	Response time with secondary index
		Response time without secondary index
		Response time with schema
		Response time without schema
	query optimization	Time cost and space cost of secondary index
Update ability	Coverage of update functionality	Coverage of all the requirement
	Basic update ability	Response time with schema
		Response time without schema
	Ability of concurrency control	Transaction rollback rate
		Transaction waiting time
		Transaction response time
		System throughput
		Occupancy rate of system resources
	Incremental validation of large scale XML documents	Response time with schema
		Response time without schema

The main interests of the index set are storage ability, query ability and update ability of a XML engine. Each interest evaluates some functionalities of the engine. The storage ability evaluates performance of storage and index functionalities. The query ability evaluates performance of simple query, complex query and query optimization. The update ability evaluates performance of common update, concurrency control and incremental validation of large scale XML documents. The performance of functionalities is shown by some indexes. It means that we use some indexes to evaluate performance of the XML engine in each aspect. For example, indexes of transaction rollback rate, transaction waiting time, transaction response time, system throughput and occupancy rate of system resources are used to evaluate ability of concurrency control. The function is an aspect of update ability of a XML engine.

F. Workload mix

We design three types of workload mix: update (write only), query (read only) and mixture (read-write). They are made up of selections from statements mentioned above. All the workloads assess performance of the system at different concurrency and amount of data. Testers can compare the performance of different XML databases by analyzing their results.

- Update: Large scale XML documents are inserted into Literature as initial data. The amount of data can be changed according to hardware

configuration. And then, five update/delete/insert statements are executed in different stages in order to simulate various concurrent operations. The update workload is mainly to assess storage ability and update ability of a XML engine.

- Query: It has fourteen query statements that are mentioned above. The statements have same weight, and can be executed in different concurrency. The purpose of this workload is to assess query ability of a XML engine by an incremental concurrency.
- Mixture: There are 70% query statements and 30% insert/update/delete statements in the workload mix. And 20% update, 40% delete and 40% insert statements constitute the insert/update/delete set. The mixture workload is to evaluate overall performance of a XML engine by an index of throughput.

G. Usage of the scheme

In this section, we present process of the scheme. The process is shown in Fig. 3.

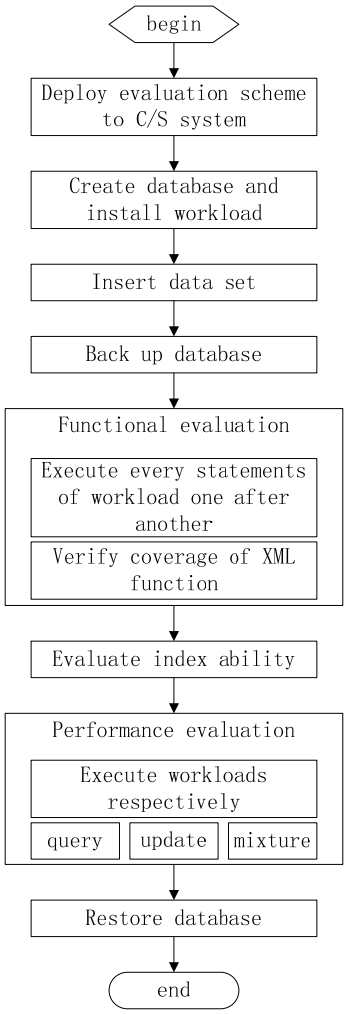

Figure 3. process of the scheme

The process is divided into two main stages: functional evaluation and performance evaluation. And there are several relational steps before or after each stage, as Fig. 3 shows.

The scheme should be deployed to a C/S system. The server is a RDBMS with the XML engine called SUT. The client simulates concurrent operation of many users. It may be multiple terminals or a terminal of multi-process. After deployment of the C/S system, the database is created in the server according to schema of the data set. And the workload is installed to the clients. It is implemented by SQL with XML extensions. Table 3 shows an example. The implement of other statements are shown in appendix.

TABLE III. IMPLEMENT U3 BY SQL WITH XML EXTENSIONS

```
Update c_category
set cateinfo = xmlquery('copy $newinfo := $CATEINFO
modify do insert <category><name>new
category</name><literatures></literatures></category>
into $newinfo/category/categorys
return  $newinfo' )
where xmlexists('$CATEINFO/category[@id=1]');
```

The data set is inserted into the database in the server for testing. Before evaluation, the database should be backed up for reuse. After all of the preparation above, every statements of the workload are executed one after another for functional evaluation. The evaluation can verify the XML function coverage of the XML engine. During the evaluation, the indexing ability can also be assessed by comparing response time of the XML engine with and without index. Performance evaluation is after functional evaluation. The query workload, update workload and mixture workload are executed respectively. During the evaluation, the indexes of the XML engine can be recorded for performance assessment. After the evaluation, the database should be restored for next round of evaluation.

IV. A CASE STUDY

We test an actual computer system using the proposal described before. The server is equipped with 1.6GHz Inter Pentium Dual processor and 1GB of main memory; operating system is Ubuntu9.10; DBMS is DB2 V9.7. We use Toxgene data generator to produce instance documents for User, Category, Order and Databases schemas. Every document is between 1KB and 10KB in size. And the documents of Literature are real XML documents from DBLP, UW XML Repository, Chinese Web Information Retrieval Forum and RSS of some Chinese websites. And the size is between 100KB and 10MB. Scale of initial data is shown in table 4.

TABLE IV. SCALE OF INITIAL DATA

Size	Documents				
	User	*Orders*	*Databases*	*Category*	*Literature*
10MB	600	3,000	5	500	6,000
100MB	6,000	30,000	10	1,000	60,000
1GB	60,000	300,000	20	2,000	600,000
10GB	600,000	3,000,000	40	4,000	6,000,000

Literature entities are document-centric data, so they take up the majority of data size. The Database and Category entities take up little size, and they are used to maintain data of Literature. So, when Literature multiplied by 10, Database and Category multiplied by 2.

Fig. 4 and Fig. 5 show a part of the test result. We select these figures intending to explain the correctness of this proposal.

Fig. 4 shows response time of update workload. We execute the statements of update workload in half an hour randomly, and record their average response time. We learn the characteristics of all statements from Fig. 4: U3 and U5 are simple statements, and they only operate one entity, so they have the shortest response time. U1 and U2 are also simple statements, but both of them involve two entities, therefore their response time is longer. Similarly, because that U4 deals with four entities, and it's the most complex, it needs the longest response time. Consequently, the proposal is correct from the point of statement.

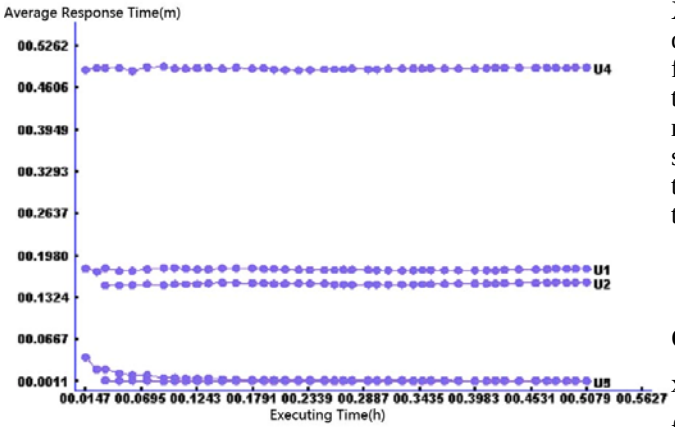

Figure 4. response time of update workload

We run query workload and mixture workload at concurrency of 20, 30, 50, 75 and 100 users respectively, and show throughput of system at various concurrencies in Fig. 5. We can see that the maximum throughput come out when the concurrency achieve 75. The throughput of query workload is 591 and that of mixture workload is 408, and they will never increase with the raise of concurrency, because that the system reach its utmost. Additionally, throughput of mixture workload is much lower than that of query workload. The reason is that executing write and read transaction may cause deadlock and rollback, this reduce the whole throughput of system. Accordingly, the proposal is proper from not only the point of whole workload but also the point of whole system.

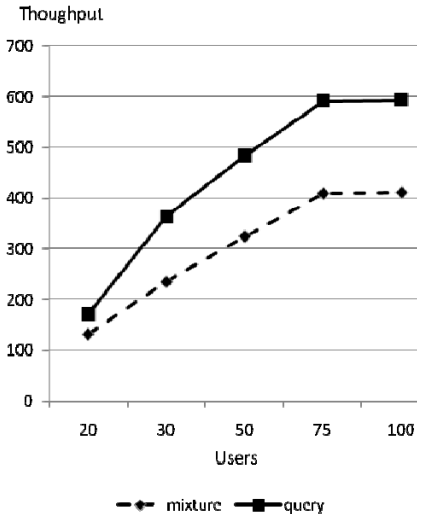

Figure 5. throughput of system at various concurrencies

V. CONCLUSION AND FUTURE WORKS

In this paper we propose an evaluation scheme of XML Engine in RDBMS. This evaluation scheme includes test scene, data set, workload and index set. Its statements have functions that cover all the criterion of XQuery of W3C. It focuses on the assessment of storage, query and update, and considers the aspect of both function and performance. Then we use the proposal to test an actual computer system. The testing result manifests that the proposal is correct. However there is still something that should be improved. How to optimize the data manipulating statements to reduce the impact of the application on testing result is our future work.

APPENDIX: IMPLEMENT OF STATEMENTS

Q1:

```
xquery
for $y in db2-fn:
xmlcolumn('C_USERS.USERINFO')/user
for $d in db2-fn:
xmlcolumn('C_DATABASES.DATABASEINFO')/datab
ase
where $y/@id =2
and $y/databases/database/id = $d/@id
return ($y ,$d/name)
```

Q2:

```
xquery
for $y in db2-
fn:xmlcolumn('C_USERS.USERINFO')/user
where $y//name = 'Agriculture'
order by $y/@id
return
```


<user>{$y/firstname/text()}</user>

Q3:

xquery let $u := db2-fn:xmlcolumn('C_USERS.USERINFO')/user

let $d := db2-fn:xmlcolumn('C_DATABASES.DATABASEINFO')/database

where $u/@id = 3 and $u/databases/database/id = $d/@id

return (fn:string-join(($d/name),'------'))

Q4:

xquery

some $x in db2-fn:xmlcolumn('C_DATABASES.DATABASEINFO')/

database/literature_count satisfies $x>3265506

Q5:

xquery

every $x in db2-fn:xmlcolumn('C_DATABASES.DATABASEINFO')/

database/literature_count satisfies $x>3265506

Q6:

xquery

for $s in db2-fn:xmlcolumn('C_CATEGORYS.CATEINFO')/category

return(($s/name),($s/categorys/category/*[1]))

Q7:

xquery

for $lit in db2-fn:xmlcolumn('C_LITERATURES.LITERATUREINFO')/literature

where $lit/@id = 61

return $lit/authors/author[1]

Q8:

xquery

let $a := for $t in db2-fn:xmlcolumn('C_ORDERS.ORDERINFO')/order

where $t/datetime = '2005-7-22'

return $t

return (

<count>{fn:count($a/amount)}</count>,

<sum>{fn:sum($a/amount)}</sum>,

<max>{fn:max($a/amount)}</max>,

<min>{fn:min($a/amount)}</min>,

<avg>{fn:avg($a/amount)}</avg>

)

Q9:

xquery

for $u in db2-fn:xmlcolumn('C_USERS.USERINFO')/user

for $o in db2-fn:xmlcolumn('C_ORDERS.ORDERINFO')/order

where $u/balance = 2981.82 and $o/user_id = $u/@id

return $o

Q10:

xquery

for $u in db2-fn:xmlcolumn('C_USERS.USERINFO')/user

where (fn:empty($u/databases))

return $u

Q11:

xquery

for $d in db2-fn:xmlcolumn('C_DATABASES.DATABASEINFO')/database

where $d/name = " The full text of Chinese doctoral thesis database "

return $d

Q12:

create function findOrder(id integer)

returns table(orderinfo xml)

specific findorder

begin atomic return select * from c_orders where xmlcast(xmlquery('$c/order/user_id' passing ORDERINFO as "c") as integer)=id; end

select * from table(findOrder(3245))

drop function findOrder

Q13:

xquery <age>

<kid>

{count(for $i in db2-fn:xmlcolumn("C_USERS.USERINFO")/user

where year-from-date(current-date())-year-from-date(xs:date($i/birthday/text()))<10

return $i)}

</kid>

<teenager>

{count(for $i in db2-fn:xmlcolumn("C_USERS.USERINFO")/user where year-from-date

(current-date())-year-from-date(xs:date($i/birthday/text()))>=10


```
and year-from-date(current-date())-year-from-
date(xs:date($i/birthday/text()))<20 return
$i)}
</teenager>
<middleage>
{count(for $i in db2-
fn:xmlcolumn("C_USERS.USERINFO")/user where
year-from-date
(current-date())-year-from-
date(xs:date($i/birthday/text()))>=20
and year-from-date(current-date())-year-from-
date(xs:date($i/birthday/text()))<50 return
$i)}
</middleage>
<elder>
{count(for $i in db2-
fn:xmlcolumn("C_USERS.USERINFO")/user where
year-from-date
(current-date())-year-from-
date(xs:date($i/birthday/text()))>=50 return $i)}
</elder>
</age>
```

Q14:

```
xquery for $i in db2-
fn:xmlcolumn('C_LITERATURES.LITERATUREINFO')
/literature
where contains($i/keywords,"customer")
return $i/title/text()
```

U1:

```
update c_users
set userinfo = xmlquery('
        transform
        copy $newinfo := $USERINFO
        modify do replace value of
$newinfo/user/balance
                with $newinfo/user/balance+10.00
        return $newinfo')
where xmlexists('$newinfo/user[@id=1]'passing
c_users.userinfo as "newinfo");
update c_users
set userinfo = xmlquery('
        transform
        copy $newinfo := $USERINFO
        modify do replace value of
$newinfo/user/balance
                with $newinfo/user/balance+10.00
        return $newinfo')
where 1=xmlcast(xmlquery('$newinfo/user/@id'passing
c_users.userinfo as "newinfo") as integer);
```

U2:

```
update c_orders
set orderinfo = xmlquery('copy $newinfo := $c
        modify do delete $newinfo/order/new_order
        return  $newinfo' passing c_orders.orderinfo as
"c")
where xmlexists(' $m/order[@id=1]' passing
c_orders.orderinfo as "m");
```

U3:

```
update c_categorys
set cateinfo = xmlquery('copy $newinfo := $CATEINFO
        modify do insert <category><name>new
category</name><literatures></literatures></category>
          into $newinfo/category/categorys
        return  $newinfo' )
where xmlexists('$CATEINFO/category[@id=1]');
```

U4:

```
update c_users
set userinfo = xmlquery('copy $newinfo := $USERINFO
        modify
      let $dbs:=db2-
fn:xmlcolumn("C_ORDERS.ORDERINFO")/order[@id=
1]/databases
        return
        (do replace $newinfo/user/databases with $dbs)
          return  $newinfo' )
where xmlcast(xmlquery('$u/user/@id' passing
c_users.userinfo as "u") as integer)=xmlcast(xmlquery('let
$id:=db2-
fn:xmlcolumn("C_ORDERS.ORDERINFO")/order[@id=
1]/user_id
          return  $id' ) as integer);
select * from c_users where
xmlexists('$USERINFO/user[@id=2429]')
```

U5:

```
create procedure putneworder(in  iuser_id bigint,
                in  idatabse_id bigint,
                    in  icate_id bigint,
                in  iamount DECIMAL (5, 2))
LANGUAGE SQL
BEGIN declare i_xml xml; declare itime
timestamp;declare ino bigint;set ino=1; set itime=current
timestamp; set i_xml
```



```
=XMLDOCUMENT(XMLELEMENT(name
"order",XMLCONCAT(XMLELEMENT(name
"user_id",iuser_id),XMLELEMENT(name
"database_id",idatabse_id),XMLELEMENT(name
"category_id",icate_id),XMLELEMENT(name
"amount",iamount),XMLELEMENT(name
"datetime",itime),XMLELEMENT(name
"new_order",ino)))); insert into c_orders(orderinfo)
values(i_xml);end

drop procedure putneworder

call putneworder(10002,2,3,2.00)

select * from c_orders where
xmlexists('$ORDERINFO/order[user_id=10002]')

values(current   date)

values(current   timestamp)
```

REFERENCES

[1] Schmidt, F. Waas, M. L. Kersten, M. J. Carey, I. Manolescu and R. Busse: “XMark: A Benchmark for XML Data Management”, Proceedings of the International Conference on Very Large Data Bases (VLDB), pp 974-985, August 2002.

[2] S. Bressan, G. Dobbie, Z. Lacroix, M. L. Lee, Y. G. Li, U. Nambiar: “XOO7: Applying OO7 Benchmark to XML Query Processing Tools”, Proceedings of the ACM International Conference on Information and Knowledge Management (CIKM), November 2001.

[3] B. Yao, M. T. Özsu, and J. Keenleyside: “XBench - A Family of Benchmarks for XML DBMSs”, Proceedings of EEXTT 2002 and DiWeb 2002, LNCS Vol. 2590, pages 162-164.

[4] T. Böhme, E. Rahm: XMach-1: “A Benchmark for XML Data Management”, Proceedings of German database conference BTW2001, pp 264-273, Springer, Berlin, March 2001.

[5] Matthias Nicola, Irina Kogan, Rekha Raghu, Agustin Gonzalez, Berni Schiefer, and Kevin Xie, “An XML database benchmark: transaction processing over XML (TPoX)”. IBM Corporation, June 2008.

[6] M. Franceschet, “XPathMark - an XPath benchmark for XMark generated data”, International XML Database Symposium (XSYM), Trondheim, Norway, pp 129-143, August 2005.

[7] K. Runapongsa, J. Patel, H. Jagadish, Y. Chen, and S. Al-Khalifa. The Michigan Benchmark: A Microbenchmark for XML Query Processing Systems. In Proceedings of EEXTT, pages 160–161, 2002.

[8] L. Afanasiev, I. Manolescu and P. Michiels: “MemBeR: A Micro-benchmark Repository for XQuery”, XML Symposium (XSym) 2005.

[9] J. Gray: The Benchmark Handbook. Morgan Kaufmann, San Mateo, CA, 1993.

[10] L. Afanasiev and M. Marx: “An analysis of the current XQuery benchmarks”, Experimental Evaluation of Data Management Systems (EXPDB), 2006.

[11] T. Böhme et al: “Multi-User Evaluation of XML Data Management Systems with XMach-1”, LNCS Vol. 2590, 2003.

Guannan Si was born in Shandong, China, in 1981. He received the M.S. degree in software engineering from Shandong University, Jinan, China, in 2008.

He is currently a PhD candidate at Nankai University, Tianjin, China. His research interests are software engineering and software evaluating technology.

Zhengji Zhou was born in Shandong, China, in 1985. He received the B.E. degree in Information Security from Central South University, Changsha, China, in 2009.He is currently working toward the M.S. degree in Computer Software and Theory at Nankai University, Tianjin, China.

His research interests include software engineering, software testing and information security.

Nan Li received the B.E. degree in Software Engineering from Tianjin University, Tianjin, China, in 2009. He is currently working toward the M.S. degree in Computer Application Technology at Nankai University, Tianjin, China.

His research interests are software engineering and software testing.

Jufeng Yang received the PhD degree in control theory and engineering from Nankai University in 2009.

Presently, he is an assistant professor of Nankai University in the institute of machine intelligence. His research fields include software engineering, pattern recognition and computer-human interaction.

Jing Xu received the PhD degree in control theory and engineering from Nankai University in 2003.

Presently, she is a professor of Nankai University in the institute of machine intelligence. Her research fields include software engineering, software testing and information technology security evaluation.

Prof. Xu is a member of China computer federation, software engineering technical committee.

