PERFORMANCE OF EXISTING LOCK BASED PROTOCOLS FOR XML DOCUMENTS

V. GEETHA, ANN MARY JOSE, DHIVYA M. & FAMINA T.P.
Dept of Information Technology, Pondicherry Engineering College
Puducherry, India, (E-mail: {vgeetha@pec.edu;ann.22aug,dhivyavino02,femi614@gmail.com}

Abstract: Extensible Markup Language (XML) has a wide variety of application in the recent days. An effective concurrency control mechanism is needed when many users access the document simultaneously. Lock based protocols are found to provide a better concurrency among the existing methods. In this paper we are presenting a survey among all the existing lock based protocols to ensure which one will provide better concurrency and to find the drawbacks in the existing methods or works.

Keywords: XML Documents, Concurrency control techniques, Multigranularity lock model.

1. INTRODUCTION
Extensible Markup Language has become a standard format for data exchange on the internet. Many applications in science, biology and business require XML to represent data in their disciplines. The widespread use of XML in these areas prompted the development of a method for efficient synchronization of concurrent updates and queries for XML data. XML is extensible, because it only specifies the structural rules of tags. There is no specification on tag themselves. Users can create new tags with new names. Only the structure should be followed. That is why XML is called as Semi Structured. It also facilitates the sharing of structured data across different information systems through internet. It is used to both encode documents as well as serialize data. XML is simple text file, that can be managed by any text editor. Earlier XML documents were mapped onto relational database (RDBMS). But now XML documents can be stored in native XML databases (XDBMS).

Simultaneous execution of transactions on a shared database can create several data integrity and consistency problems like lost updates, uncommitted data and inconsistent retrievals. When several transactions have access to the same document at the same time, the consistency of the data gets effected. Concurrency control techniques allow transactions to be executed concurrently without effecting data consistency.

Three categories of concurrency control techniques namely Time stamp based protocols, optimistic concurrency control and lock based protocols are available. Among these, lock based protocols are found to be effective because they provide concurrency at multigranularity level and hence comparatively increases the number of transactions running per unit time.

A lock compatibility matrix is used to check the compatibility between the two transactions. If they can be executed at the same time, then both are allowed. Otherwise one transaction has to wait for the other to complete its execution. Multigranular lock models allow lock conversion. A transaction can change its lock mode without releasing the data item and requesting for it afresh. This creates locking overhead and leads to deadlock. Sometimes a transaction might want to change the lock mode without releasing the data item and requesting afresh. This is lock conversion. The rule for lock conversion is to be provided by a lock conversion matrix.

There are many Application Programming Interfaces (API) for XML documents. The APIs are DOM, SAX, XPath, XQuery.
DOM is an API for well-formed XML documents. It defines the logical structure of documents and the way a document is accessed and manipulated. The DOM is a platform-neutral and language-neutral interface that will allow programs and scripts to dynamically access and update the content, structure and style of documents. The document can be further processed and the results of that processing can be incorporated back into the presented page. DOM model defines a document model to represent XML documents together with DOM operations to manipulate them. The DOM document model abstracts the document as a tree, namely XML tree. The root of an XML document is named document element in the XML tree, while elements, attributes, and texts in an XML document are mapped to nodes in the XML tree [3]. Fig. 1.a shows an example of XML document and its corresponding XML tree in Fig.1.b. In Fig. 1.b, the element nodes, attribute nodes, and text nodes are denoted by circles, rhombuses, and rectangles, respectively. In the selected example name, lname, position, job, country are element nodes. fname is the only attribute node and the leaf nodes rahul, raj, PA, india are the text nodes. Text node gives a value to the attribute node and the element node.

DOM operations can be classified into observers and modifiers mainly for semantic analysis purpose [3]. An observer is a shared operation that reads nodes, values or navigates nodes in an XML tree, while a modifier is an exclusive operation that modifies nodes, values or document structures. Table 1 gives the list of observers and modifiers defined in DOM. There are six operations classified as observers and four operations as modifiers.

SAX is another API for XML documents. SAX stands for Simple API for XML documents. It is an event-based sequential access parser and it provides a mechanism for reading data from an XML document that is an alternative to that provided by the DOM. While the DOM operates on the document as a whole, SAX parsers operate on each piece of the XML document sequentially. The quantity of memory that a SAX parser must use in order to function is typically much smaller than that of a DOM parser. DOM parsers must have the entire tree in memory before any processing can begin. The memory of a SAX parser by contrast, is based only on the maximum depth of the XML file and the maximum data stored in XML attributes on a single XML element. So DOM can be used for large documents whereas SAX is used for small documents.

A parser that implements SAX (i.e., a SAX Parser) functions as a stream parser, with an event-driven API. The user defines a number
of callback methods that will be called when events occur during parsing. The SAX events include:

- XML Text nodes
- XML Element Starts and Ends
- XML Processing Instructions
- XML Comments.

XPath, the XML Path Language is a query language for selecting nodes from an XML document. It is based on a tree representation of the XML document, and provides the ability to navigate around the tree, selecting nodes by a variety of criteria. XPath may also be used to compute values (e.g., strings, numbers, or Boolean values) from the content of an XML document. XPath uses path expressions to select nodes or node-sets in an XML document. The node is selected by following a path or steps. Table 2 gives the list of most useful path expressions in XPath.

<table>
<thead>
<tr>
<th>Expression</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>nodename</td>
<td>Selects all child nodes of the named node</td>
</tr>
<tr>
<td>/</td>
<td>Selects from the root node</td>
</tr>
<tr>
<td>//</td>
<td>Selects nodes in the document from the current node that match the selection</td>
</tr>
<tr>
<td>.</td>
<td>Selects the current node</td>
</tr>
<tr>
<td>..</td>
<td>Selects the parent of the current node</td>
</tr>
<tr>
<td>@</td>
<td>Selects attributes</td>
</tr>
</tbody>
</table>

XQuery is a query and functional programming language that is designed to query collections of XML data. XQuery provides the means to extract and manipulate data from XML documents or any data source that can be viewed as XML, such as relational databases or office documents. XQuery uses XPath expression syntax to address specific parts of an XML document. It is based on a tree-structured model of the information content of an XML document, containing seven kinds of node: document nodes, elements, attributes, text nodes, comments, processing instructions, and namespaces.

This survey is mainly focusing on papers which provide concurrency in XML documents using lock protocol and optimistic concurrency control.

2. PERFORMANCE EXPECTATIONS OF CONCURRENCY CONTROL IN XML

2.1. Support of DOM, SAX, XPath and XQuery APIs

A good concurrency control technique is expected to provide fine granularity access for all the operations defined by XML APIs namely SAX, DOM, XPath and XQuery. Fine granularity access provides better concurrency and hence better performance.

2.2. Support of Different Types of Transactions

Generally two types of transactions access the XML documents. They are runtime transactions and designtime transactions. The runtime transactions access the XML documents for data. Earlier the XML data was mapped to relational databases. Now the data is mapped onto native XML databases (XDBMS). The runtime transactions do not modify the structure. They read or modify only the data. The designtime transactions access the XML schema. XML schema represents the structure of domain which is represented as XML document.

2.3. Support of XML Operations

The runtime and designtime operations have a set of allowed operations on the XML document. The runtime transactions can typically traverse the XML document, then read or update a data item. Designtime transactions can insert a new node to the XML schema and can move, delete or rename the existing nodes. It can also read and modify node definitions.

2.4. Multigranularity Support

Granularity may be supported at many levels such as node, tree, edge. Increased concurrency can be achieved if the granularity is supported at the finest level and it is node in the case of XML documents with reduced number of conflicts. Also there should not be any data or schema inconsistencies at any level.
3. LITERATURE SURVEY

A number of concurrency control techniques have been proposed so far. Among them most popular is locking-based protocols. In this method, a transaction must lock a data item before accessing that item and unlock it when its transaction is done. A lock is a variable associated with a data item in the database. Lock based protocols provide a better concurrency. A locking protocol describes rules, according to which a transaction should lock and unlock the data items. It comparatively increases the number of transactions running per unit time and it is possible to lock items in various granule levels depending on the operations. It can be either at fine granule levels (which include nodes) or can be at coarse granule levels (sub tree level). The concept of locking data items is one of the main techniques used for synchronizing the concurrent execution of transactions.

3.1. Concurrency Control Using Lock Protocols

Lock-based protocols use various types of locks to determine whether a transaction can proceed or it must have to wait for another transaction to complete. Many papers discuss on increasing concurrency of XML documents using locking protocols. Earlier concurrency control algorithms used a two phase locking (2PL) method where a lock can be obtained and released in two separate phases. A lock can be obtained for a data item only during growing phase and it can be released during shrinking phase. In short this protocol specifies that no data item can be unlocked until all data items to be accessed have been locked [9]. There is a variant of this 2PL namely strict two phase locking method where a lock is acquired and released in two separate phases. A lock can be obtained for a data item only during growing phase and it can be released during shrinking phase. In short this protocol specifies that no data item can be unlocked until all data items to be accessed have been locked [9]. There is a variant of this 2PL namely strict two phase locking method where a lock is acquired and released in two separate phases. A lock can be obtained for a data item only during growing phase and it can be released during shrinking phase. In short this protocol specifies that no data item can be unlocked until all data items to be accessed have been locked [9]. There is a variant of this 2PL namely strict two phase locking method where a lock is acquired and released in two separate phases. A lock can be obtained for a data item only during growing phase and it can be released during shrinking phase. In short this protocol specifies that no data item can be unlocked until all data items to be accessed have been locked [9]. A variant of this two phase locking method is called Multigranularity Locking. Multigranularity locking (MGL) also known as multigranularity locking (MGL), also denotes as hierarchical locking, allows for fine grained access by setting R (read) or X (exclusive) locks on objects at the lower levels in the hierarchy and coarse grained access by setting the locks at higher levels in the hierarchy. To avoid lock conflicts, intention locks with modes IR (intention read) or IX (Intention Exclusive) have to be acquired along the path from root to the object. This MGL protocol in most cases is too restricted because both R and X mode on a node would always lock the whole sub tree below. That is, it provides concurrency at coarse granule levels. In Ref [17], a Lightweight multigranularity protocol (LWMGL) is introduced which is a hybrid mechanism of both tree-based locking and multigranularity locking. It was mainly developed as a concurrency control scheme without the phantom phenomenon. The goal of this scheme is to realize locking at the level of precise elements in an XML database while preventing the phantom problems. For the purpose of locking granules they have introduced a new XML data management model called XML indexed element tables (XIET) [17]. This model handles a mapping methodology for transferring diverse XML document schema to relational database schema. This locking scheme enables execution of structural change operations as well as data access operations in parallel. But the paper lacks comparison with other existing protocols for increasing concurrency.

A locking protocol is defined in [8] for XPath language known as XLP. XLP has the features of rich lock modes, low lock conflict and lock conversion. There are 5 lock modes used by XLP protocol. They are P-lock mode, W-lock mode, I-lock mode, R-lock mode and D-lock mode [8]. P-lock mode is used for navigation through the tree where as W-lock mode and R-lock mode is used when a transaction requests for write and read operations. Similarly D-lock mode and I-lock mode is used for deletion and insertion of a node respectively. And XLP is a combination of six rules which includes 2PL, P-lock rule, granularity rule, upgrade rule, compatibility rule and release rule. Three major contributions of XLP supports most XPath axes and invention of a new lightweight lock mode called P-lock for better concurrency. The conflict conditions and rules in the XPath model are also derived in [8]. [8] has provided a comparison of this protocol with existing 2PL, tree based locking protocol...
and other XPath based protocols. But when there is an increased locking overhead, it leads to deadlock and detection of this deadlock needs a costly graph.

3.2. Dataguide Protocols

Another protocol used as an extension of MGL is Dataguide locking protocol. Dataguide is a data structure that summarizes an XML document. It is concise and accurate, because Dataguide describes every unique label path exactly once regardless of the number of times it appears in the document, and encodes no other label path that does not appear in that document. In [12][10], a XPath based DataGuide Locking (XDGL) protocol is developed which generalizes and extends on Multigranularity locking model. XDGL guarantees serializability and provide high degree of concurrency. It uses a subset of XPath language to access the document nodes and to modify the document. It also considers the semantics of operations. But this XDGL cannot provide support to full version of Xpath. In [7], Snapshot based Concurrency control protocol is proposed which is also a hybrid of MGL and dataguide locking protocols. A new protocol which is an extension of XDGL is used in [9]. Major advantage of this protocol is that it can be used not only for XML document structures but also for RDBMS. A major advantage of Dataguide locking protocol is reduced locking overhead. Because Dataguide is usually much smaller than the document itself. Hence it may be kept in main memory even for large XML documents.

3.3. taDOM \ Protocols

In order to increase the number of concurrent transactions on an XML document, a protocol called taDOM has been introduced [19][20]. This is mainly for DOM APIs. In this XML data is represented as a taDOM tree which is an extension of the XML tree representation of DOM. The lock modes are node locks, navigational or edge locks and logical locks. Node locks provide fine granularity to avoid phantom problems. It also provides lock escalation when there is a lock conflict. Four types of taDOM lock protocols have been introduced in paper [6]. These include taDOM2, taDOM2+, taDOM3, taDOM3+. These protocols provide maximum concurrency. The number of lock modes are increased as we move from taDOM2 to taDOM3+ to support higher versions of DOM API. A lock compatibility matrix is provided for each protocol depending on which the concurrent transactions are allowed. Lock conversion matrix is also provided which helps in lock conversion in case of lock conflicts. In [6], a bank document data is taken for their experimental data and a comparison among these four protocols is made in terms of transaction throughput and concurrently maintained locks. These protocols are compared with the already existing Node2PL, OO2PL lock protocols also. And taDOM protocols are found to provide better concurrency than other protocols. But this can be used only for DOM APIs. In [4], taDOM locking protocols are used to provide locks for queries in a functional update language XML-Ł. Here a transactional grammar is used to transfer the queries in XML-Ł to Document Object Model API calls and taDOM lock protocols are used in DOM.

3.4. Optimistic Concurrency Control

Optimistic concurrency control (OCC) is a concurrency control method that assumes that multiple transactions can complete without affecting each other and hence transactions can proceed without locking the data resources that they affect. Before committing, each transaction verifies that no other transaction has modified its data. If the check reveals conflicting modifications, the committing transaction rolls back.

In [2][6][11], optimistic concurrency control mechanism is used for increasing concurrency. But it is found that OCC is generally used in environments with less number of concurrent transactions and where there is less number of conflicts. OCC is effective only when conflicts are rare. Then the transactions can complete without the expense of managing locks and without having transactions to wait for other transactions locks to clear, leading to higher throughput. But if conflicts happen often, the cost of repeatedly restarting the transactions
affect the performance of system significantly, even leading to deadlock related problems.

3.5. Semantic Based Concurrency Control
In this method, semantics of operations are considered to check the conflicts and to provide a better concurrency. These protocols are found to provide a better concurrency by determining whether two operations commute at the semantic level. Two operations commute if their execution orders do not produce different results, such as their returned object structures or values. In [3], semantics of DOM operations are explored to increase concurrency by determining which pairs of operations commute. By using the commutativity concept from operations’ semantics, operations can run concurrently even though they are conflicting under those non-semantic-based protocols. Here DOM operations are analyzed in a semantic perspective and a new semantic based concurrency control protocol SCD is developed. In [3], each operation is classified into either a node operation or a subtree operation according to whether it operates on a node or a subtree. The only drawback in this paper is that it considers only semantics of DOM operations.

4. COMPARATIVE STUDY
A comparative study is done based on the operations used, granularity level considered and languages supported. Significant contributions in each category of concurrency control technique are considered for the comparative study. Operations identified are read, write, traversal/navigational, rename, insert, delete/remove, replace and update. And the granularity level considered includes dataguide, xpath, tree, node, graph and edge. All these papers are reviewed for the various APIs support for XML such as XPath, XQuery, SAX and DOM. A tick is marked against each operations, granularity level or query if it is supported in the corresponding paper.

Table 3 shows the comparative study prepared.

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Read</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Write</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Traverse</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Rename</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Insert</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Delete</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Update</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
</tbody>
</table>

Granularity Level

<table>
<thead>
<tr>
<th></th>
<th>Dataguide</th>
<th>✓</th>
<th>✓</th>
<th>✓</th>
<th>✓</th>
<th>✓</th>
<th>✓</th>
<th>✓</th>
<th>✓</th>
<th>✓</th>
<th>✓</th>
<th>✓</th>
</tr>
</thead>
<tbody>
<tr>
<td>Graph</td>
<td></td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>XPath</td>
<td></td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Tree</td>
<td></td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Node</td>
<td></td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Edge</td>
<td></td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
</tbody>
</table>

Query Support

<table>
<thead>
<tr>
<th></th>
<th>SAX</th>
<th>✓</th>
<th>✓</th>
<th>✓</th>
<th>✓</th>
<th>✓</th>
<th>✓</th>
<th>✓</th>
<th>✓</th>
<th>✓</th>
<th>✓</th>
<th>✓</th>
</tr>
</thead>
<tbody>
<tr>
<td>XPath</td>
<td></td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>XQuery</td>
<td></td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
</tbody>
</table>

Table 3
5. CONCLUSION

XML has widespread application in the field of science and web. Increased number of users accessing a single document simultaneously has led to the need for increasing the concurrency of XML documents. From the survey prepared it is clear that concurrency of XML documents can be increased using lock protocols than any other method. Granularity level also becomes an important issue with regard to concurrency in XML documents. It is able to provide an increased concurrent access to XML documents when the locks are considered or provided to finer granular level (to node level). In the existing works, there is a lack of concurrency control which provide multilingual support. A system which support all operations in a dynamic environment also need to be considered.

References


