International Journal of Database Theory and Application
Vol.7, No.6 (2014), pp.95-104
http://dx.doi.org/10.14257/ijdta.2014.7.6.09

Increasing Concurrency in XML Documents Using Semantic Locks

V. Geetha

Department of Information Technology, Pondicherry Engineering College, Puducherry, 605014, INDIA

vgeetha@pec.edu

Abstract

Several lock based protocols were introduced to increase the concurrency of a tree based XML structure. Concurrency can be increased when locking of resources is done at finer granular level. ***.*** *When the locking is done at the finest granular level like at node, more number of users can access the same document simultaneously. In the existing systems, all operations are done using only one of the API's like SAX, DOM or XPath. For this a new compatibility matrix is proposed which includes all the operations possible to an XML document in order to increase the concurrent access to the same XML document and also to lock the resources to the finest granular level.*

.

Keywords*: XML Documents, semantic locks, Multi-granularity 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).

XML document can be viewed as a containment hierarchy. It is organized as a tree. The main components of the tree are *nodes* and *axes*. The nodes can be *element nodes*, *text nodes* or *attribute nodes*. *Text nodes* can have text *string* as their content. *Attribute nodes* represent properties of the elements. They can be of any data type element nodes represent entities and they can contain one or more attribute nodes and text nodes. The text and element nodes are ordered. The relationships among entities (represented by nodes) are expressed using *links* or *axes*. The possible relationships among nodes are *parent*, *child* and *sibling*. There exists explicit links between Parent nodes and children nodes. As XML is tree structured, any child node can have only one parent. There is a root node which is the parent of all other nodes. The tree spreads as each parent spawns with many children. Figure 1.a shows a sample XML document and figure 1.b shows its tree structure.

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 multi-granularity 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. Multi-granular 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.

Clients can access XDBMS for data and schema. There are several standard language models like SAX, DOM, XPath, XQuery. These languages provide standard constructs for reading, navigating and modifying the schema and data of XML databases. These constructs use the relationships among the nodes for retrieval.

```
<root>

        <person>

                <name age= “16”>

                <fname>rahul<fname>

                <lname>raj</lname>

        </name>

        <job>

                <position>pa</name>

                <country>india</country>

        </job> </ person >

</root>
```

Figure 1.a- A Sample XML Document

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.

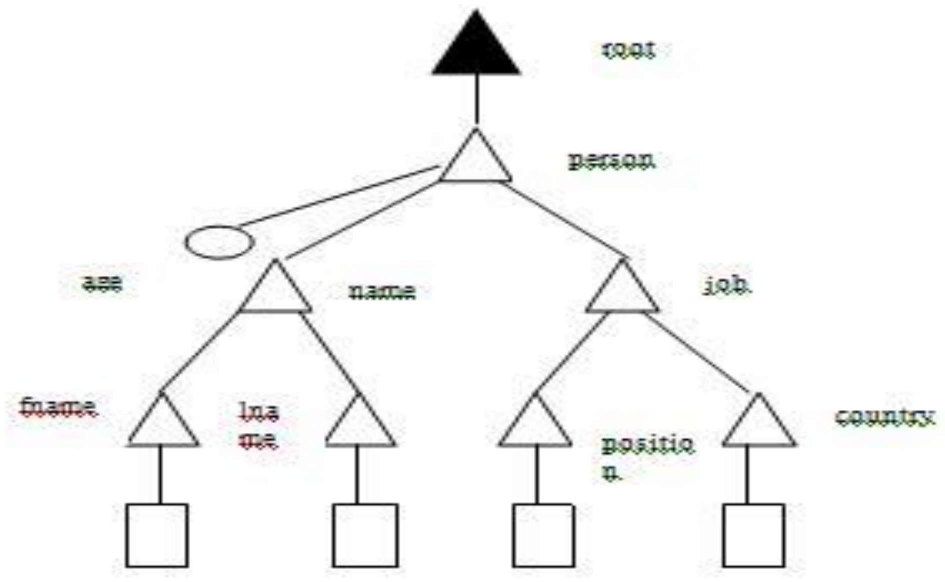


Figure 1.b. Sample XML Tree

DOM operations can be classified into observers and modifiers mainly for semantic analysis purpose [12]. 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.

Table 1. Classification of DOM Operations

Observers	Modifiers
FirstChild	InsertBefore
NextSibling	RemoveChild
LastChild	AppendChild
PreviousSibling	ReplaceChild
NodeName	
NodeValue	

2. Related Works

2.1 Semantics of XML

There are several data representation models available in the literature to speed up the execution of queries supported by the fore said language APIs. In taDOM tree model [5, 7, 8, 9, 20], three new node types are introduced: - *Document root, attribute root and string.*

The top of the taDOM tree consists of *document root* and topmost root element of the document. Attribute nodes are not directly connected to the element nodes. Instead an attribute root is connected to each element and organizes the attributes of the corresponding element as descendant nodes. In order to build a *NamedNodeMap.* Figure 2 shows the corresponding document tree for xml document in Figure 1.a.

In Dewey ID model [3,6] prefix based labeling is done. In this each label represents the path from the document root to the related node. It preserves local order with respect to parent node. It provides sparse numbering facilities node insertions and deletions. Figure 3 shows the Dewey ID model for the sample XML document in Figure 1.a.

Dataguide [15, 16, 17, 19] is a compact structural summary of XML tree. In Dataguide model, it gives unique label path of the document exactly once. Each Dataguide node has a label (like element, attribute, text, *etc.*) and pointer to data pages where nodes corresponding to Dataguide node are stored. Datapages of one Dataguide node are linked via pointers into a bidirectional list. Here structural part and test parts of a node are separated. Figure 4 shows the Dataguide.

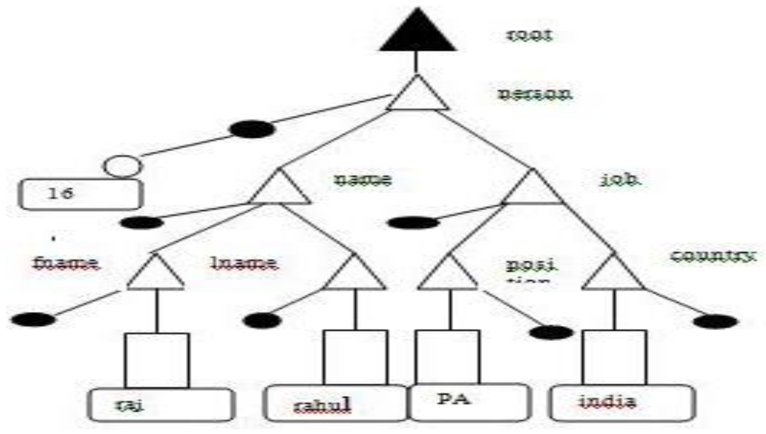


Figure 2. taDOM Tree

There are many operations which a user can request for an XML document. It include insert, delete, update, read, move, *etc.*, These operations be classified as Run time requests (RTR), and Design time requests(DTR). RTR consists of those operations which changes the values on an XML document. It does not change the schema or structure of the XML document. For example, an operation of insert in RTR means inserting a value or data to the existing nodes of the XML tree as well as in the database. No change is made in the XML tree structure. On the other hand, an operation of insert in DTR means inserting a new node, edge or subtree within the XML document which changes the structure of the XML schema. In other words we can say that RTR deals with data operations and DTR deals with schema operations. In this project we are including run time request operations like insert, delete, update and read and design time request operations which include insert, delete, modify and move. Our work considers all possible operations on an XML document and a new compatibility matrix is proposed to check the compatibility between the various operations requested by concurrent users at the same time.

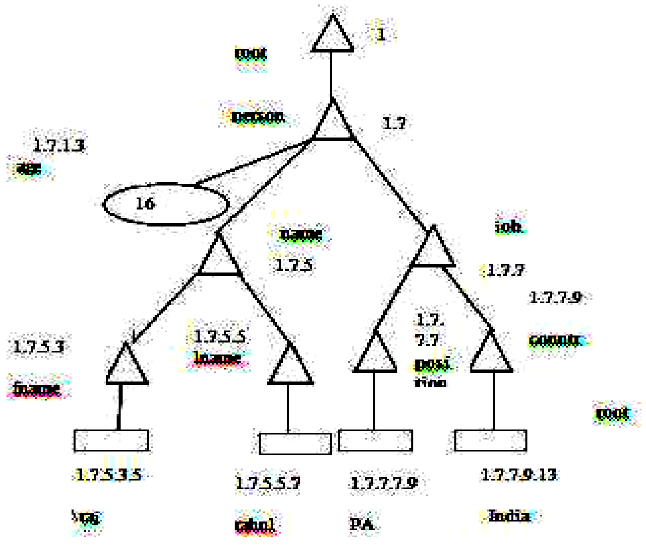


Figure 3. Dewey ID Model

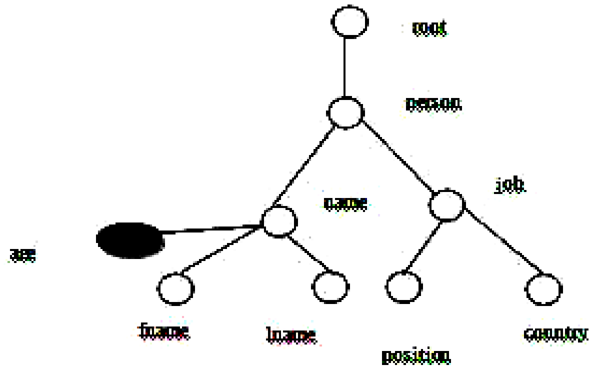


Figure 4. Dataguide Model

2.2 Literature Survey

Performance of concurrency control mechanisms can be assessed based on its support to parallel execution of RTR and DTR. The conflicts arising from RTR and DTR can be categorized into

1. Conflicts among RTR
2. Conflicts among DTR

3. Conflicts among DTR and RTR

Conflicts among RTR

In [17], P lock mode is used for navigation. R and W locks are used for reading and modifying data items. In [19], two protocols namely Optix and SnaX are defined. OptiX is based on optimistic concurrency control. In [17], the general problems of locks with annotated predicates remain unsolved. In [11], there exist conflicts between read-rename operations and update operations.

The locks of Dataguide protocol are set on the node of Dataguide, not on the document tree. But the node in the Dataguide usually corresponds to two or more nodes in the document tree. If we want to access a node or a sub tree, we must use predicate to differentiate the nodes which have same names which results in expensive computing [15, 16, 17, 19].

In taDOM [5, 7, 8, 9, 20], the same lock mode is shared for both text as well as element node. So, parallel execution of data modification and reading of node definition is not possible.

Conflicts among DTR

In [18] for all operations, conflict arise between intention Exclusive, sub tree operation, rename, insert and delete operations. While performing sub tree operation, no other operation can be done. Only read operation can be done by performing rename operation. Also during the insert/delete operation, conflicts arise with all other operations. In [13] while performing delete operation no other operation is allowed other then navigation.

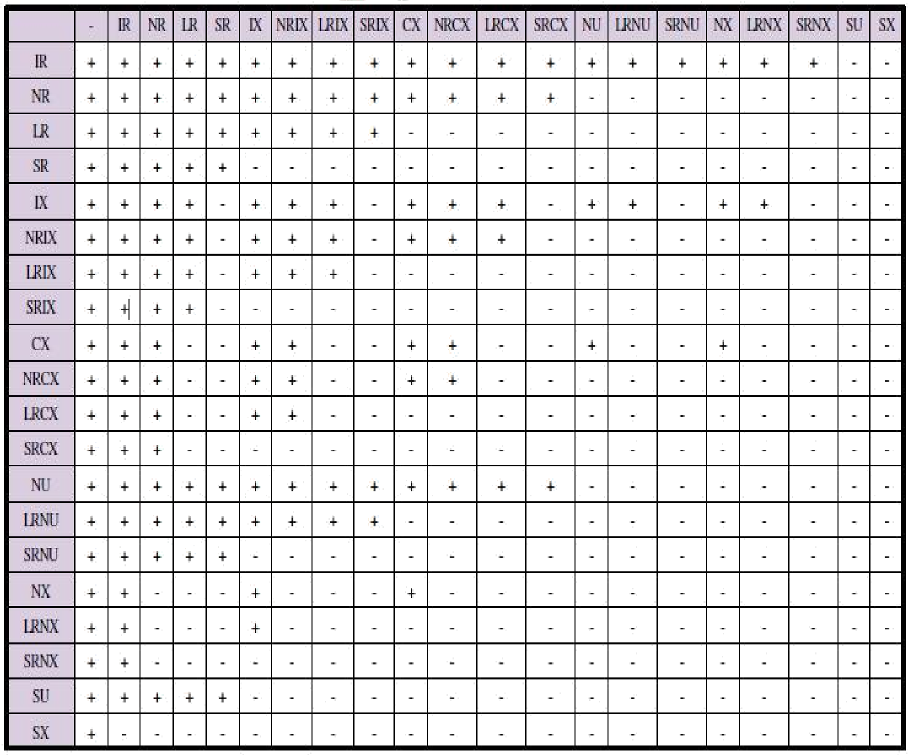
In [10] Exclusive lock on left sibling conflicts with shared lock on left sibling and Exclusive lock on left sibling .When there is an Exclusive lock on right sibling ,conflicts arise with shared lock on right sibling and Exclusive lock on right sibling. If there is an Exclusive lock on first child, conflict arises with shared lock on first child and Exclusive lock on first child. Similarly when there is an exclusive lock on last child, conflicts arise with shared lock on first child and also with Exclusive lock on first child. Also there is a lack of support for direct jumps to inner document nodes.

In [11] no other operations can be done while performing the operations like replace and remove. In [17] while inserting a child at a node, conflicts arise while inserting a child at another node and also no other operations on Dataguide and Dataguide's sub tree can be done. Similarly while inserting a sibling is performed, inserting a sibling at another node cannot be done. Also no operations can be done if sub tree is locked in exclusive mode.

Conflicts between RTR and DTR

In [18] read operation conflicts with insert/ delete operation on sub tree and deletion on parent. Also while performing intention read, operations like rename on the node and deletion on the parent cannot be done.

Table 2. Compatibility Matrix for the Existing Work [5]



	-	IR	NR	LR	SR	IX	NRIX	LRIX	SRIX	CX	NRCX	LRCX	SRCX	NU	LRNU	SRNU	NX	LRNX	SRNX	SU	SX
IR	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	-	-
NR	+	+	+	+	+	+	+	+	+	+	+	+	+	-	-	-	-	-	-	-	-
LR	+	+	+	+	+	+	+	+	+	-	-	-	-	-	-	-	-	-	-	-	-
SR	+	+	+	+	+	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
IX	+	+	+	+	-	+	+	+	-	+	+	+	-	+	+	-	+	+	-	-	-
NRIX	+	+	+	+	-	+	+	+	-	+	+	+	-	-	-	-	-	-	-	-	-
LRIX	+	+	+	+	-	+	+	+	-	-	-	-	-	-	-	-	-	-	-	-	-
SRIX	+	+	+	+	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
CX	+	+	+	-	-	+	+	-	-	+	+	-	-	+	-	-	+	-	-	-	-
NRCX	+	+	+	-	-	+	+	-	-	+	+	-	-	-	-	-	-	-	-	-	-
LRCX	+	+	+	-	-	+	+	-	-	-	-	-	-	-	-	-	-	-	-	-	-
SRCX	+	+	+	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
NU	+	+	+	+	+	+	+	+	+	+	+	+	+	-	-	-	-	-	-	-	-
LRNU	+	+	+	+	+	+	+	+	+	-	-	-	-	-	-	-	-	-	-	-	-
SRNU	+	+	+	+	+	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
NX	+	+	-	-	-	+	-	-	-	+	-	-	-	-	-	-	-	-	-	-	-
LRNX	+	+	-	-	-	+	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
SRNX	+	+	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
SU	+	+	+	+	+	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
SX	+	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-

In [13] if traversal is done on same granularity sub tree, operations like write, insert and delete cannot be done. Similarly while performing read and write operations, delete operation cannot be performed.

In [10] when read operation is performed on left sibling, exclusive lock on left sibling is not allowed. Similarly while read operation on right sibling is done, Exclusive lock on right sibling is not allowed. Also shared lock on first child, conflicts with Exclusive lock first child. Similarly shared lock on last child conflicts with Exclusive lock on last child.

In [11] operations like read, rename and update operations conflict with all the other operations. Survey is done on existing works on the topic increasing concurrency in XML documents. Many works regarding concurrency is done using optimistic concurrency control, semantic based concurrency control and lock protocols out of which use of lock protocols have been found effective in increasing the concurrency. Most of the works were carried only using DOM API. In the existing systems, there is a lack for a system which provides support for many API's (say XPath, DOM, SAX, XQuery etc...). Granularity level also becomes a important issue with regard to concurrency in XML documents. It is possible to provide an increased concurrent access to XML documents when the locks are considered or provided to finer granular level (to node level). Systems which support many numbers of operations in a dynamic environment also need to be considered.

Table 3. Compatibility Matrix of the Proposed Work

	RN	RL	RS	IPN	ICN	ISN	IL	IN	ISC	ISS	DPN	DCN	DSN	DL	DN	DSC	DSS	UPN	UCN	USN	UL	MCN	MSN	MN
RN	+	+	+	+	+	+	+	+	+	+	+	+	+	+	-	+	+	+	+	+	+	+	+	+
RL	+	+	+	+	+	+	+	+	+	+	+	+	+	-	+	+	+	+	+	+	+	+	+	+
RS	+	+	+	+	+	+	+	+	+	+	-	-	-	-	-	-	-	+	+	+	+	+	+	+
IPN	+	+	+	+	+	+	+	+	+	+	+	+	+	-	-	+	+	-	+	+	-	+	+	-
ICN	+	+	+	+	+	+	+	+	-	+	-	-	+	-	-	-	+	+	-	+	+	-	+	-
ISN	+	+	+	+	+	+	+	+	+	+	-	+	-	-	-	+	+	-	-	+	-	-	-	+
IL	+	+	+	+	+	+	+	+	+	+	-	-	+	-	-	-	+	+	+	+	+	+	+	-
IN	+	+	+	+	+	+	+	+	+	+	-	-	+	-	-	-	+	+	+	+	+	+	+	-
ISC	+	+	+	+	+	+	+	+	-	+	-	-	+	-	-	-	+	-	-	+	-	-	+	+
ISS	+	+	+	+	+	+	+	+	+	-	-	+	-	-	-	+	-	-	+	-	-	+	-	-
DPN	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
DCN	-	-	-	+	-	+	+	+	-	+	-	-	+	-	-	-	+	+	-	+	-	-	+	-
DSN	+	-	-	-	+	-	-	+	+	-	-	+	-	-	+	+	-	-	+	-	-	+	-	+
DL	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
DN	-	-	-	+	-	+	-	-	-	+	-	-	+	-	-	-	+	+	-	+	-	-	+	-
DSC	-	-	-	+	-	+	-	-	+	+	-	-	+	-	-	-	-	+	-	-	-	-	+	-
DSS	+	-	-	+	+	-	-	-	+	-	-	+	-	-	-	-	+	-	+	-	-	+	-	+
UPN	+	+	+	-	+	+	-	-	+	+	-	+	+	-	+	+	+	-	+	+	-	-	-	-
UCN	-	-	-	+	-	+	-	+	-	+	+	-	+	-	-	-	+	+	-	+	+	-	+	-
USN	+	-	-	+	+	-	-	+	+	-	+	+	-	-	+	+	-	+	+	-	+	+	-	+
UL	+	-	+	+	+	+	-	+	+	+	-	-	-	-	-	-	-	-	-	-	-	-	-	-
MCN	+	-	-	+	-	+	-	-	-	+	+	-	+	-	-	-	+	+	-	+	+	-	+	-
MSN	+	-	-	+	+	-	-	+	+	-	+	+	-	-	+	+	-	+	+	-	-	+	-	-
MN	+	+	+	+	-	+	-	-	-	+	+	-	+	-	-	-	+	+	-	+	-	-	+	-

[5] has provided a comparison between four important lock modes. They have made comparison by means of throughputs and response time. In that paper [5], taDOM3+ protocol is found to be the best when compared to remaining three protocols.

There are many shortcomings in the existing system which can be improved further. The proposed scheme aims to provide access of fine granularity, while maintaining the consistency of database. DOM is a widely used API for XML. DOM API also defines a set of operations which are classified as Observers and Moderators. Observers are set of operations to read from the database. Moderators's set of operations is used to update the database. The authors of existing works have proposed lock modes based on the API used. Due to this, there is a possibility of omitting overloaded. When the compatibility matrix is proposed based on operations, this can be avoided. Further in all the existing works, the same exclusive lock mode is shared for all types of operations like insert, delete, modify and move. In this paper, separate lock modes are provided for all operations.

Similarly, the possible relationships between two nodes could be parent, child and sibling. The semantics of these relationships is also exploited in proposed scheme.

When a node is specified its relation with its parent sibling and child is considered. The expansion of the lock modes is as follows.

- RN-Read Node- Reading the value of the specified node. This requires the node to be locked in shared mode.
- RL-Read Level- It denotes the navigation or traversal through the XML tree to reach a specific node.

- RS-Read Sub tree- Reading the contents of sub tree.
- IN-Insert Node-Lock mode is used when we need to insert a new node at the specified node. This requires node to be locked in the exclusive mode.
- DN-Delete Node-Deleting the specified node.
- MN-Move Node-This lock mode is used to move the node from one position to another.
- IPN-Insert at Parent Node-Insert a node at the parent of the specified node. Similarly a sibling node can be inserted anodes can be inserted at child also.
- UPN-Update Parents Node – Update or modify the content or structure of the parent node. Similarly separate lock modes are provided for updating the child node and the sibling node.
- MCN-Move Child Node- This lock is used when we need to move the child of the specified node from one position to another which may entirely change the structure of the tree.
- ICN-Insert at Child Node
- ISN-Insert Sibling Node
- IL-Insert Level/Edge
- ISC-Insert Sub tree at Child
- ISS-Insert Sub tree at Sibling
- DPN-Delete Parent Node
- DCN-Delete at Child Node
- DSN-Delete at Sibling Node
- DL-Delete Level/Edge
- DSC-Delete Sub tree at Child
- DSS-Delete Sub tree at Sibling
- UCN-Update Child Node
- USN-Update Sibling Node
- UL-Update Level/Edge
- MSN-Move Sibling Node

4. Conclusion

There are many shortcomings in the existing works. The main drawback in the existing work is that there are many inconsistencies between the operations and granularity is not considered to the finest level. These are rectified in our proposed work. This is achieved by proposing a compatibility matrix in which all operations are taken into account and separate lock modes are provided for all operations. The relationship of a node with its parent, child and siblings are considered and the semantics of these operations are exploited in the proposed scheme. The proposed scheme is expected to provide a better concurrency to the finest granular level without any inconsistencies.

References

[1] C. Byun, "A New Optimistic Concurrency Control in Valid XML", Journal of Information Science and Engineering , vol. 25, **(2009)**, pp. 11-31.
[2] D.Berrabah, "Optimistic path-based concurrency control over XML Documents", CSTST '08 Proceedings of the 5th international conference on Soft computing as transdisciplinary science and technology.
[3] Sebastian B¨achle, "Implementing and Optimizing Fine-Granular Lock Management for XML Document Trees", DASFAA, **(2009)**, pp. 631-645.
[4] Y. Choi and S. Moon, "Lightweight multigranularity locking for transaction management in XML database systems", The Journal of Systems and Software, vol. 78, **(2005)**, pp. 37–46.
[5] M. Haustein and T. Ha¨rder, "Optimizing lock protocols for native XML processing", Data & Knowledge Engineering, **(2008)**, pp. 147-173.
[6] M. Haustein, T. H¨arder, "An Efficient Infrastructure for Native Transactional XML Processing", **(2006)**.
[7] Michael P. Haustein, Theo Härder, "A Lock Manager for Collaborative Processing of Natively Stored XML Documents", SBBD, **(2004)**, pp. 230-244.
[8] M. P. Haustein, Theo H¨arder, "taDOM:A Synchronization Concept for the DOM API", Grundlagen von Datenbanken, **(2003)**, pp. 80-84.
[9] M. P. Haustein, T. H¨arder, "taDOM:A Tailored Synchronization Concept with Tunable Lock Granularity for the DOM API", ADBIS, **(2003)**, pp. 88-102 .
[10] S. Helmer, "Evaluating Lock based Protocols for Cooperation on XML Documents", ACM SIGMOD Record, v.33 n.
[11] Y.-F. Huang, M.-L. Guo, "A Locking Protocol for DOM API on XML Documents", Proceedings of the 10th International Conference on Enterprise Information Systems, vol. 1, **(2008)**, pp.105-110.
[12] K.Jea, "A Semantic-Based Protocol for Concurrency Control in DOM Database Systems", Journal of Information Science AND Engineering, vol. 25, **(2009)**, pp. 1617-1639.
[13] K.-F. Jea and S.-Y. Chen, "A high concurrency XPath-based locking protocol for XML databases", Information and Software Technology xx, **(2005)**, pp. 1–9.
[14] P. Pleshachkov and S. Kuznetcov, "SXDGL: Snapshot based Concurrency Control Protocol for XML Data", **(2007)**.
[15] P. Pleshachkov and P. Chardin, "A Locking Protocol for Scheduling Transactions on XML Data", Proceedings of the Spring Young Researcher's Colloquium on Database and Information Systems SYRCoDIS, St.-Petersburg, Russia, **(2005)**.
[16] P.Pleshachkov, "A DataGuide-Based Concurrency Control Protocol for Cooperation on XML Data*", ADBIS 2005, LNCS 3631, **(2005)**, pp. 268– 282.
[17] P. Pleshachkov, "XDGL: XPath-Based Concurrency Control Protocol for XML Data*", BNCOD 2005, LNCS 3567, **(2005)**, pp. 145–154.
[18] C. Rong, "SeCCX: Semantics-Based Fine Granular Concurrency Control for XML Data", WAIM 2010 Workshops, LNCS 6185, **(2010)**, pp. 146– 155.
[19] Z. Sardar, B. Kemme, "Don't be a Pessimist: Use Snapshot based Concurrency Control for XML", Technical Report SOCS-TR, **(2005)**.
[20] P. Strnad and P. Loupal, "Using taDOM Locking Protocol in a Functional XML Update Language", **(2008)**, pp.25–36.

Author

V. Geetha is working as Assistant Professor in the Department of Information Technology in Pondicherry Engineering College, Puducherry, INDIA. She has done her B.Tech, M.Tech and Ph.D in Pondicherry University. Her research areas of interest includes client/ server architecture, distributed systems, object oriented system design and middleware technologies.