Transaction Isolation In the Sedna Native XML DBMS

© Peter Pleshachkov Leonid Novak

Institute for System Programming of the Russian Academy of Sciences,
B. Kommunisticheskaya, 25, Moscow 109004, Russia
peter@ispras.ru, novak@ispras.ru
Ph.D. Advisor S. D. Kuznetsov

Abstract

XML has become the most important technique to exchange data in World Wide Web. As consequence, an interest to native XML databases has surfaced. Concurrency control methods for traditional databases are not adequate for XML databases, because they do not capture the specific of XML data model. In this paper we propose a locking mechanism, developed in Sedna, which allows to achive a high degree of concurrency and takes into account the properties of structural and manipulation parts of XML model.

1 Introduction

The use of the extensible markup language XML [13] for electronic data interchange leads to an enormous growth of the number of XML documents. The number of different applications supporting XML grows rapidly too. So the challenge of isolating different applications in XML environment from each other is very actual. There are essentially three possibilities of storing XML documents: to use file system, RDBMS [4] or a native XML database management system (XDBMS for short). The first and second approaches poss an obvious imperfections from an isolation point of view [10, 2], and this is the reason why synchronizing protocols for XDBMSes are extensively being studied and developed now. This paper introduces synchronizing method of XML-data, based on locking techniques, which was developed in Sedna [8]. Our synchronizing method takes into account the properties of structural and manipulation parts of XML model.

Sedna is a full-featured native XML database management system, which is being developed by the MODIS R&D group of ISP RAS. It supports XQuery [14] language for query facilities. In order to support update facilities XQuery language was extended. Sedna update language is very close to [11].

There are several protocols [1, 7], which allow to synchronize hierarchical data via data locking, but most of them exploit the concept of intention locks. It means that for locking a subtree all ancestors of the root of this subtree must be locked in the intention mode. Thus, the operation of locking an entire subtree of XML document is needed information about nodes, which are distributed over XML document, i.e. the locking operation does not poss the locality property.

In Sedna physical representation of XML documents [5], using of these protocols leads to a very expensive locking operation, because all ancestors of subtree root are stored in different blocks and all these blocks must be read to the main memory.

This is the main reason why we do not incorporate these protocols in Sedna. Our locking method allows to lock subtree without locking ancestors of the root in intention mode. To lock the whole subtree only descriptor of the root node is needed. We have achieved this result by using numbering scheme for locking nodes and subtrees of XML document.

The rest of the paper is organized as follows. In Section 2 we introduce the storage schema of XML documents, which we use in Sedna. In Section 3 we introduce some of the basics relating to the synchronization methods. In Section 4 we present our locking method, which allows to acquire the arbitrary entire subtrees of XML documents in efficient way. In section 5 we present logical locks, which are useful for prevention conflicts at the logical level such as lost updates, dirty reads and unrepeatable reads. In Section 7 we discuss the locks, which are useful for preventing conflicts at the physical level, and we name them as physical locks. Finally, in Section 8 and 9, we give a brief overview of related work about XML synchronization and conclusion of the paper.

2 XML storage model

In this section we will describe the main concepts of Sedna XML storage model, over which the locking techniques have been developed.

The choice of our physical representation of XML documents is based on two fundamental principles of XML-data management:

- The efficiency of XPath evaluation.
- The efficiency of propagating updates.

Proceedings of the Spring Young Researcher's Colloquium On Database and Information Systems SYRCoDIS, St.-Petersburg, Russia, 2004

Consider the main characteristics of physical representation of XML documents in Sedna.

1. A descriptive schema driven storage stategy which consists of clustering nodes of an XML document according to their positions in the descriptive schema of the XML document. Descriptive schema presents a concise and accurate structure summary of the XML document. Formally speaking, descriptive schema is a tree. Every path of the document has exactly one path in the descriptive schema, and, vice versa, every path of the descriptive schema is a path of the document.

2. Each descriptive schema node is labeled with an XML node kind name (e.g. element, attribute, text, etc.) and has a pointer to data blocks where nodes corresponding to the descriptive schema node are stored. Some schema nodes depending on their node kinds are also labeled with names. Data blocks belonging to one schema node are linked via pointers into a bidirectional list. Nodes are ordered between blocks in document order.

3. The structural part and text values of nodes are separated. Text values are stored in blocks according to the well-known slotted-page structure method [12] developed specifically for data of variable length. The structural part of a node reflects its relationship to other nodes (i. e. parent, children, sibling nodes) and is presented in the form of node descriptor.

4. Each node descriptor contains a pointer to a numbering scheme. A numbering scheme assigns a unique numeric number to each node of an XML document according to some scheme-specific rules. The numeric numbers encode information about the relative position of the node in the document. Thus, the main purpose of a numbering scheme is to provide mechanisms to quickly determine the structural relationship between a pair of nodes. It provides the mechanism of quickly determining the following structural relationships between a pair of nodes: (1) ancestor-descendant relationship, (2) the document order relationship. Numbering scheme allows to support many XQuery-specific operations.

3 Synchronization Preliminaries

In this section we give an overview of concurrency control method that are of interest here.

To ensure serializability of transactions we use a well-known strict two phase locking protocol. We use logical locks to prevent conflicts at the logical level such as lost updates, dirty reads and unrepeatable reads. For prevention physical conflicts we use physical locks. A problem at the physical level can occur if one transaction follows a pointer to a record on some page, while the other transaction updates a second record on the same page and causes a data compaction routine to reassign record locations. Physical locking is handled by setting and holding locks on one or more pages during the execution of a single physical operation. Logical locking is handled by setting locks on such objects as nodes and subtrees and holding them either until they are explicitly released or to the end of the transaction. For synchronizing read and write operations, we introduce three kinds of locks: shared locks (S), exclusive locks (X) and update locks (U). Read access requires a shared lock while write access requires an exclusive lock. An update lock supports read with (potential) write access and prevents further shared locks for the same object. An update lock can be converted to an exclusive lock after the release of the existing read locks or back to an shared lock if no update action is needed. The compatibility matrix for lock modes is depicted in Figure 1.

	S	X	U
S	+	—	—
X	—	—	—
U	+	—	—

Figure 1: Compatibility Matrix for lock Modes

For two phase locking protocol the standard rules have to be obeyed. Before performing an operation, the corresponding lock has to be acquired. During lock acquisition a check for conflicting locks is performed, if a conflict exists the lock requiring transaction is blocked, and locks are held till the end of transaction. If a transaction is blocked, the wait graph is updated and if it contains a cycle, the transaction that completes the cycle is aborted. The selection of a victim is based on the relative ages of transactions in deadlock cycle. In general, the the youngest transaction is selected as the victim.

Maintaining locks, granting and declining lock requests are managed by a lock manager component.

In Sedna logical locks are implemented by means of numbering scheme, which was introduced in Section 2. Numbering scheme also can be used for preventing phantoms, because locking of the interval of numbering numbers, allows to lock all nodes (including nodes which are not presented in database), which labeled with numeric numbers included in this interval. For example, the interval of numeric numbers, which includes all nodes of the certain subtree (including phantoms) can be calculated by the root of this subtree.

Our main contribution in this paper is using and adopting numbering scheme for logical locking mechanisms (see Section 5).

4 Numeric Schema as Basics for Locking

In this section we introduce the details of Sedna numbering schema, and the main idea of our locking methos based on the certain properties of numbering scheme.

A numbering schema of XML document provides a one-to-one mapping of nodes of XML document onto numeric numbers. In other words, each node of XML document has a unique numeric number.

The numeric number of the node consists of two components. The first one is a prefix and the second one is a delimiter. Prefix is a string of characters, while delimiter is a character. We will refer to the numeric number of node n as (p_n, d_n), where p_n is the prefix of node n and d_n is its delimeter, or simply as N if prefix and delimeter of numeric number are not significant.

Below we give a set of definitions concerning numeric numbers.

Definition 1 *Let (p_{n_1}, d_{n_1}) and (p_{n_2}, d_{n_2}) are the numeric numbers of nodes n_1 and n_2 of some XML document correspondingly. We regard that the numeric number of n_1 is less ($<$) than numeric number of n_2 if and only if $p_{n_1} \prec p_{n_2}$, where $\prec$ is the lexlexicographical comparison of strings.*

Definition 2 *Let (p_{n_1}, d_{n_1}) and (p_{n_2}, d_{n_2}) are the numeric numbers of nodes n_1 and n_2 of some XML document correspondingly. We regard that interval of numeric numbers $[(p_{n_1}, d_{n_1}), (p_{n_2}, d_{n_2})]$ (or simply $[p_{n_1}, p_{n_2}]$) consists of all numeric numbers (p_{n_i}, d_{n_i}), which satisfy the condition $p_{n_1} \preceq p_{n_i} \preceq p_{n_2}$.*

Definition 3 *Let p_1 and p_2 are the strings of characters. Then the string $p_1 \cdot p_2$ is the concatenation of strings p_1 and p_2.*

The idea of numbering scheme, which provides (1) and (2) mechanisms (these mechanisms were introduced in Section 2) is based on the following facts:

- For two given strings p_1 and p_2 such that $p_1 \preceq p_2$ there exists string p such that $p_1 \preceq p \preceq p_2$. For example, if p_1 = "*abc*" and p_2 = "*abcd*", then p = "*abca*" (the number of such p is infinite).

- Assume, that node n is the root of the entire subtree in the document. Then the string interval $(p_n, p_n \cdot d_n)$, sets the range of numeric numbers for all descendants of node n.

Numbering numbers are asigned to the nodes of a document in the following way:

- For two given nodes n_1 and n_2, n_1 preceeds n_2 in the document order if and only if $p_{n_1} \prec p_{n_2}$.

- For two given nodes n_1 and n_2, n_1 is an ancestor of n_2 if and only if $p_{n_1} \prec p_{n_2} \prec p_{n_1} \cdot d_{n_1}$

```
<lib>
    <book>
      <title>Transaction Procesing:
                Concepts and Techniques
      </title>
      <price>120</price>
      <author>J. Gray</author>
      <author>A. Reuter</author>
    </book>
    <book>
      <title>Concurrency Control and
              Recovery in Database Systems
      </title>
      <author>P. A. Bernstein</author>
      <author>V. Hadzilakos</author>
    </book>
    <book>
      <title>Data on the Web</title>
      <price>50</price>
      <author>Abiteboul</author>
      <author>Buneman</author>
      <author>Suciu</author>
    </book>
</lib>
```

Figure 2: Example of an XML document

The exact algorithm of assigning numeric numbers to the nodes of XML document is not very significant here and we don't present it here.

The idea of locking entire subtree of XML document using numeric numbers is based on the fundamental property of numeric numbers, which is stated in the following proposition

Proposition 1 *Let (p_n, d_n) is the numeric number of node* n. *Then the interval $[p_n, p_n \cdot d_n]$ includes the numeric numbers of all descendants of node* n.

Thus, the numeric numbers can be used for locking nodes and subtrees of XML document. The locking of the node is implemented of locking its numeric number. The locking of the subtree (assume node n is the root of this subtree) is corresponds to the locking of interval $[p_n, p_n \cdot d_n]$. The correctness of this idea is guaranteed by proposition 1 .

Below we consider the example of using numeric schema for locking purposes.

Figure 3 depicts a small XML document *lib.xml*. The document contains the list of three books where each book is described by *title*, *price* (optional) and *author*, respectively.

Figure 3 shows the tree representation of the structure of the XML document defined in Figure 2. The outer element *lib* is the document node. Nested elements are connected with edges in tree. To each node of the tree (except text nodes) the numeric number is assigned.

To lock the first book element transaction is required to lock interval $["ab", "abm"]$, while to lock the last book elemnt transaction is needed to lock interval $["ap", "apm"]$. To lock the whole XML document interval $["a", "az"]$ must be locked.

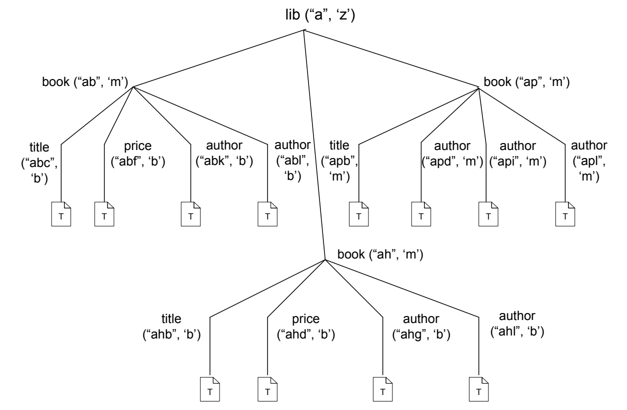

Figure 3: Example of assigning numbering numbers to the XML document

The lock all book elements of the document the following three intervals must be locked: $[''ab'', ''abm'']$, $[''ah'', ''ahm'']$ and $[''ap'', ''apm'']$ or simply one interval $[''ab'', ''apm'']$ should be locked.

5 Logical Locks

Logical locks serve for preventing conflicts at the logical level. As mentioned in Section 4 the set of nodes is locked by means of intervals of numeric numbers. The locking of one node with numeric number (p, d) corresponds to the locking of interval $[p, p]$, while the locking of the whole subtree corresponds to the locking of interval $[p, p \cdot d]$, where (p, d) is the numeric number of the root of the subtree to be locked.

Assume that there are m active transactions, and the intervals I_{i_j} $(j = 1..k_i)$ are locked by transaction T_i with modes M_{i_j} correspondingly. The request for interval $I_{i_{k_i+1}}$ with mode $M_{i_{k_i+1}}$ by transaction T_i will be granted by lock manager if and only if one of the following statements is obeyed:

- $I_{i_{k_i+1}}$ does not intersect with all intervals acquired by transactions T_j, $j = 1..m, j \neq i$.
- if $I_{i_{k_i+1}}$ intersects with several intervals acquired by transactions T_j, $j = 1..m, j \neq i$, then $M_{i_{k_i+1}}$ is compatible (according to Figure 1) with each of these intervals.

Actually, intervals can be useful for preventing phantoms [3]. We demonstrate this idea on example. Assume that transaction T_1 retrieves all books (see lib.xml) by the following path expression: */lib/book*, while the transaction T_2 inserts the *price* element as the first child of the first book element. It is obvious that the *price* element is the phantom for transaction T_1. But the transactions T_1 and T_2 will not run concurrently because interval $[''ab'', ''apm'']$ includes the numeric numbers for *price* element to be inserted. The algorithm of assigning numeric numbers to the new elements ensures it.

So far, we only considered the locking method for one node or the whole subtree of XML document. Below we present some extensions of our locking method which allow to increasing the degree of concurrency.

Sometimes, transaction is needed to lock the part of the subtree, for example all nodes of the subtree with depth less than or equal to n, while the depth of the whole subtree is greater than n. In this case the lock request will consist of two parts. The first one is the interval of numeric numbers, which covers the whole subtree, and the second one is the part of descriptive schema to which the nodes to be locked belong. Actually, the part of descriptive scheme consists of the set of scheme nodes. For example, the part of descriptive scheme, which covers *book* element in *lib* document is the set of nodes: *(book, title, price, author)*. We will refer to the part of descriptive scheme using S letter.

Thus, two locks (I_1, S_1) and (I_2, S_2) are compatible if one of the following statements is obeyed:

- I_1 does not intersect with I_2
- I_1 intersects with I_2, but their modes M_1 and M_2 are compatible.
- The first and second statements are not obeyed, but the descriptive scheme parts S_1 and S_2 does not intersect.

It is obvious, that the using of descriptive scheme allows to achieve higher degree of concurrency, but the logic of lock manager becomes more complex.

6 Logical locks escalation

In Sedna, the lock manager component maintains a count of the locks held by the transaction. If the number of locks held by one transaction becomes too large then the lock manager runs the escalation procedure: the conversion of many fine-granularity locks into a single coarse-granularity lock.

To tune the lock escalation, we introduce one parameter: the escalation thresold. If lock manager detects that the number of locks acquired by one transaction exceeds the percentage thresold value defined by the escalation threshold then the locks held by this transaction are replaced with a suitable single lock, which covers all these locks. In our locking method the escalation can be implemented by means of replacing a set of intervals I_i with one interval I, which covers all I_i.

It is obvious, there is a trade-off to be observed for lock escalation. On the one hand lock escalation leads to the decreasing of concurrency, but on the other hand a reduction of the number of held locks and the number of calls to the lock manager leads to saving lock manager overhead.

7 Physiacal Locks

Physical locks serve for preventing conflicts at the physical level. Physical locks are held during the evaluation of one physical operation. In most relational databases the granularity of physical locks is block.

In Sedna we also use blocks as granule for physical locking. And the simplest way to ensure physical consistency is to lock the block, where the needed nodes are stored.

To understand the interrelations between logical and physical locks consider the evaluation of the XPath expression */lib/book[title=”Data On the Web”]*.

We start evaluation of the query with traversing the descriptive schema (note: in this paper we do not consider the synchronization of descriptive schema). The result of traverse is one schema node that contain pointer to the list of blocks where the descriptors of *book* nodes are stored. Then transaction will lock the first block from the list. Traversing this block transaction will lock the blockes, where the *title* node descriptors of the *book* nodes are stored. If transaction find a book, which *title* is equal to *”Data on the Web”* then the one will lock this *book* node at the logical level. When the list of *book* blocks will be passed the physical locks can be released, while the logical locks on the *book* nodes, which satisfy to the predicate will be released only at the end of the transaction.

8 Related Work

Processing of concurrent querying and update of XML-data has received only little attention so far.

In [9], the synchronization of concurrent transactions is considered in the context of DOM API. The authors present three types of locks. Node locks are acquired for the actual nodes, navigational locks are acquired on virtual navigation edges to synchronize operations on the navigation paths, while logical locks are introduced to prevent phantom problem. Authors offer rich options to enhance transaction concurrency. But synchronization of non-navigational APIs (like XQuery) is part of future work.

In [10] the discussion is also based on the DOM API, several isolation protocols are proposed. But node locks are not acquired in a hierarchical context, and lock granularity is fixed for each protocol.

Grabs et. al [6] proposed a combination of well-known granularity locking and predicate locking which provides high concurrency, but their locking is applicable to only restricted XML documents with simple XPath query for transaction access.

9 Conclusion

Efficient concurrent processing of updates and queries of XML-data in a consistent and reliable way is an important practical problem. There are only few extensions of commercial database systems, which poorly support XML document processing.

In our paper we presented the physical representation of XML documents, which we use in Sedna. Based on this representation, we have introduced the locking mechanism, which allows acquiring the nodes and subtrees of XML document in efficient way. The logical and physical locks have been discussed. The idea how phantoms problem can be solved by means of intervals of numeric numbers is also presented. We pointed out that descriptive scheme knowledge can improve the degree of concurrency.

Future work includes efficient deadlock detection mechanism for proposed locking method. A recovery mechanism is also one of the next steps in our plan.

References

[1] P. A. Bernstein, V. Hadzilacos, and N. Goodman, ”Concurrency Control and Recovery in Database Systems” Addison-Wesley, 1987.

[2] S. Dekeyser, J. Hidders, J. Paredaens, ”A transaction model for XML databases”, World Wide Web Journal, Kluwer, 2003.

[3] K. P. Eswaran, J. Gray, R. Lorie, and I. Traiger, ”The notions of consistency and predicate locks in a database systems”, Comm. of ACM, Vol. 19, No. 11, pp. 624-633, November 1976.

[4] D. Florescu and D. Kossman, ”Storing and Querying XML Data using an RDBMS”, IEEE Data Engineering Bulletin, 1999.

[5] A. Fomichev, M. Grinev, S. Kuznetsov, ”Descriptive Schema Driven XML Storage”, Submitted to ADBIS 2004.

[6] T. Grabs, K. Bohmd, and H. Schek, ”XMLTM: efficient transaction management for XML documents”, Proc. of the 19th CIKM Conference, pp 142-152, 2002.

[7] J. Gray, and A. Reuter, ”Transaction Processing: Concepts and Techniques”, Morgan Kaufmann, 1993.

[8] M. Grinev, A. Fomichev, S. Kuznetsov, K. Antipin, A. Boldakov, D. Lizorkin, L. Novak, M. Rekouts, P. Pleshachkov, ”Sedna: A Native XML DBMS”, Submitted to International Workshop on XQuery Implementation, Experience and Perspectives (XIME-P), 2004.

[9] M. P. Haustein, and Theo Harder, ”taDOM: a Tailored Synchronization Concept with Tunable Lock Granularity for the DOM API”, In Proc. ADBIS Conference, LNCS 2798, Springer, 2003.

[10] S. Helmer, C.-C Kanne, and G. Moerkotte, ”Isolation in XML Bases”, Technical report, University of Mannheim, Germany, 2001.

[11] P. Lehti, ”Design and Implementation of a Data Manipulation Processor for an XML Query Language”, Technische Universitt Darmstadt Technical Report No. KOM-D-149, http://www.ipsi.fhg.de/ lehti/diplomarbeit.pdf, August, 2001.

[12] A. Silberschatz, H. Korth, S. Sudarshan, ”Database System Concepts”, Third Edition, McGraw-Hill, 1997.

[13] World Wide Web Consortium, "Extensible Markup Language (XML) 1.0 (2nd edition)", W3C Recommendation.

[14] World Wide Web Consortium, "XQuery 1.0: An XML Query Language", W3C Working Draft, 13 November 2003.