SXDGL: Snapshot Based Concurrency Control Protocol for XML Data*

Peter Pleshachkov[1] and Sergei Kuznetcov[2]

[1] Institute for System Programming RAS, Russia
peter@ispras.ru
[2] Institute for System Programming RAS, Russia
kuzloc@ispras.ru

Abstract. Nowadays, concurrency control for XML data is a big research problem. There are a number of researchers working on this problem, but most of the proposed methods are based on the two-phase locking protocol, which potentially leads to a high blocking rates in data-intensive XML-applications. In this paper we present and evaluate SXDGL, a new snapshot based concurrency control protocol for XML data. SXDGL completely eliminates data contention between read-only and update transactions. Moreover, SXDGL takes into account the hierarchical structure and semantics of XML data model determining conflicts between concurrent XML-operations of update transactions. The conducted evaluation shows significant benefits of SXDGL for processing concurrent transactions in data-intensive XML-applications.

1 Introduction

Over the last decade, native XML database management systems has received considerable attention from the research community and industrial software companies. As a result, there are a number of databases now on the market, which support XML data model. One of the key requirements for a native XML database is it's ability to provide data consistency while allowing multiple transactions to have concurrent read/write access to the XML documents. In order to meet this requirement database researchers adopted an existing multigranularity locking scheme based on 2PL protocol for XML data [10,6,17].

It is a well known fact [20] that 2PL does not efficiently support concurrent processing of complex read-only transactions (usually called *queries*) and update transactions (usually called *updaters*), causing update transactions to suffer from long delays due to data contention with read-only transactions. The problem seems to be increasingly important in the context of XML applications, because XQuery (with small extensions like XQueryP[2]) is used as a native language for development of XML-applications, and, as a result, transactions written in XQuery may be long-lived. Thus, delays for *short* updaters may be unacceptable.

This problem has been extensively studied by many researchers in the past years and various multiversion extensions to 2PL protocol (e.g. 2V2PL, MV2PL,

* This work was partially supported by the grant of RBRF N 05-07-90204.

D. Barbosa et al. (Eds.): XSym 2007, LNCS 4704, pp. 122–136, 2007.

ROMV) have been proposed in the literature [21]. Moreover, multiversion protocols have been widely accepted in the industry and implemented in many relational databases like Oracle, MS SQL Server 2005, PostgreSQL, etc. However, multiversion concurrency control for XML data has received little attention in the literature so far. The problem is challenging due to a number of reasons specific to XML data.

Firstly, it is not straightforward how to design a versioning scheme for XML database in a such way that the effectiveness of XML-document traversal operations would be preserved. It is a very important issue, because traversal operations are *intensively* performed during query/update execution, and therefore, of crucial importance to achieve high performance. Indeed, each time when we want to access a node by pointer we should perform pointer dereferencing and additionally follow the version chain in reverse chronological order to locate the appropriate version. Thus, the versioning scheme should be designed in a such way that additional overhead incurred by choosing an appropriate version would be minimized.

Secondly, to eliminate unnecessary conflicts between updaters we need to take into consideration hierarchical structure of XML data model and semantics of XQuery/XUpdate operations. Especially, we need carefully consider all different types of XML update operations and regular path expressions taking into account different conflict behavior.

Thirdly, we should reduce the locking overhead (in updaters), which can be tremendous for XML data as shown in [10].

As our main contribution we propose SXDGL, a new snapshot-based XML DataGuide locking protocol, which produces only serializable schedules and supports efficient processing of concurrent processing of transactions in XML database. Our protocol enables efficient processing by employing the following techniques. Firstly, SXDGL allows queries to execute without acquiring locks. It completely eliminates locking overhead for queries and interferences between queries and updaters. Secondly, multiversioning is implemented using adjusted memory-mapped architecture, which significantly reduces a total number of traverses of version chain to locate the appropriate version. Besides, the proposed versioning scheme restricts the maximum number of versions of data item to four. So, it significantly simplifies version management. Finally, we introduce a DataGuide-based locking scheme for isolating updaters. 11 types of new lock modes are introduced, which allow to capture the different conflict behavior of XQuery/XUpdate operations, and, as a consequence, avoid unnecessary delays. Besides, using a compact DataGuide structure for locking purposes guarantees that the locking overhead is low as opposed to approaches that set locks on the nodes of XML-document.

The SXDGL protocol have been prototyped in Sedna XML database system [13], which is successfully used in the content engineering projects, biological and Web-based applications, etc. In this paper, we present the results of some experiments which have been conducted with the aim of evaluating the performance