Available online at www.sciencedirect.com
SCIENCE DIRECT®

Information and Software Technology 48 (2006) 708–716

INFORMATION AND SOFTWARE TECHNOLOGY
www.elsevier.com/locate/infsof

A high concurrency XPath-based locking protocol for XML databases☆

Kuen-Fang Jea*, Shih-Ying Chen

Department of Computer Science, National Chung-Hsing University, 250, Kuo-Kuang Road, Taichung 40227, Taiwan, ROC

Received 27 January 2005; revised 24 July 2005; accepted 2 August 2005
Available online 16 September 2005

Abstract

Providing efficient access to XML documents becomes crucial in XML database systems. More and more concurrency control protocols for XML database systems were proposed in the past few years. Being an important language for addressing data in XML documents, XPath expressions are the basis of several query languages, such as XQurey and XSLT. In this paper, we propose a lock-based concurrency control protocol, called XLP, for transactions accessing XML data by the XPath model. XLP is based on the XPath model and has the features of rich lock modes, low lock conflict and lock conversion. XLP is also proved to ensure conflict serializability. In sum, there are three major contributions in this paper. The proposed XLP supports most XPath axes, rather than simple path expressions only. Conflict conditions and rules in the XPath model are analyzed and derived. Moreover, a lightweighted lock mode, P-lock, is invented and integrated into XLP for better concurrency.

Keywords: XML document databases; Concurrency control; Locking protocol; XPath

1. Introduction

XML has become a standard format for data exchange on the Internet. Many applications such as Science, Biology and Business require XML to represent data in their disciplines. Since data in these areas are usually very large, it is common to store them in databases for efficient retrieval and storage.

Concurrency control is one of the most important techniques to achieve efficient access (including read and write operations) in database systems. This is especially true of current web-based XML applications, as it can provide higher transaction throughput and better scalability for XML database servers. By scheduling incoming transactions, concurrency control techniques allow transactions to be executed concurrently and thus diminish the waiting time. The correct execution of concurrent transactions scheduled by a concurrency control protocol can be ensured by serializability [8,12,20,22].

Many concurrency control protocols were proposed for traditional database systems. Among them, the lock-based protocols [1,12,17–19,22] are widely used. In these protocols, write locks and read locks are two fundamental types of locks. A transaction can proceed if the requested lock on the desired object is compatible with locks held by other transactions on the same object; otherwise, the transaction must wait until other transactions release the incompatible locks. The most famous lock-based protocol is the two-phase locking protocol (2PL) [7,12,18,22]. In 2PL, a transaction acquires locks only in the growing phase and releases locks in the shrinking phase to ensure serializability.

Graph-based locking protocols [17] treat data items in the database as a partial ordering set $D=\{d_1, d_2,\dots, d_n\}$. The set D, called database graph, forms a directed acyclic graph (DAG) where an edge $d_i \rightarrow d_j$ indicates item d_i must be accessed before item d_j. The tree locking protocol [21], whose database graph displays a tree structure, is a special case of graph-based protocols. A transaction can release a lock and subsequently obtain another lock. However, if a transaction previously released a lock on a data item, it can no longer relock that data item. Multi-granularity locking protocols [11,19] consider data items as an ordering abstraction with different granularity. A data item with

☆ This research was supported in part by NSC in Taiwan, R.O.C. under Grant No. NSC-90-2213-E005-014.

* Corresponding author.

E-mail addresses: kfjea@cs.nchu.edu.tw (K.-F. Jea), sychen@cs.nchu.edu.tw (S.-Y. Chen).

coarse granularity includes many smaller ones with finer granularity. Transactions only need to acquire the lock on data items of coarse granularity, and then they can access all of the descendent data items.

Unfortunately traditional concurrency control protocols are not tailored to XML database systems and therefore not optimized for the execution of XML transactions. The research in this area has rapidly gained importance in recent years, and several techniques [2–6,10,13–16] have been proposed. In general, these concurrency control techniques deal with three different types of access methods, including DOM [23], DataGuides [9] and XPath [3].

DOM [23] supports a standard set of application program interfaces (APIs) to manipulate XML documents. It allows programs to dynamically access and update the content and structures of XML documents. By exploiting the DOM access methods, [13,14,16] proposed protocols with a rich set of locking modes to achieve high concurrency. In contrast, DataGuides [9] is the data abstraction of an XML document. By exploiting the structural relationships among nodes in DataGuides, a 2PL-based DGLOCK protocol [10] was proposed. In addition to shared locks and exclusive locks, the intension locks are included in the protocol.

Besides DOM and DataGuides, XPath [3] is a popular language for addressing data in XML documents. Several query languages for XML databases are XPath-like query languages, such as XQuery [25] and XSLT [24]. By storing and comparing the database states for each concurrent transaction, [2] proposed a validation-based concurrency control protocol [12,22] for transactions including XPath expressions. The result of each write access in a transaction is reflected in its local database state. A conflict check proceeds to see if there exists any conflict with local database states of other transactions. If a transaction commits, its local database state becomes the new database state. The protocol may be expensive for storing database states and checking conflicts.

On the other hand, [4–6] proposed two locking protocols: path locks propagation (PL-PROP) and path locks satisfiability (PL-SAT). Both are similar except for the read locks being set. PL-SAT requires fewer read locks on the nodes in an XPath expression and leads to more complex checks of conflicts. By contrast, PL-PROP propagates read locks from the root of an XPath expression to its descendents and results in more read locks but simpler checks of conflicts. However, both schedulers can guarantee serializability [4,5]. Based on PL-PROP, a commit scheduler [4,6] and a conflict scheduler [5,6] were proposed. A commit scheduler lets a transaction wait if its request cannot proceed, while a conflict scheduler keeps a transaction proceeding unless it fails.

The work in [13], although based on the DOM model, also proposes a locking procedure to support part of axis operations (including the parent, child and sibling axes) similar to the XPath's access. Ref. [13] provides ten lock modes comprising shared, exclusive and intension locks: seven lock modes are applied on nodes, while three are on edges between nodes. To access a node, nodes in its path to the root are locked in the up-to-root direction. Edges are locked in both directions. Lock conflicts are checked against the compatibility matrices for both node and edge locks. Its lock granularity includes locks on the context node, the context node plus its direct-child nodes and the context node plus its edges. A lock conversion to a more restrictive one is also allowed.

In this paper, a new lock-based concurrency control protocol, namely *XLP* (XPath Locking Protocol), is proposed for XML database systems. The access behavior and operation conflicts in XPath expressions are analyzed to increase the concurrency of XPath-like queries in XML transactions. The protocol is intended not only to minimize the number of locks held by releasing them as soon as they are no longer needed, but also to support various concurrency enhancement features including rich lock modes, low lock conflict, and lock conversion.

The rest of this paper is organized as follows. Section 2 reviews the XPath model and describes the terms and notions used in XLP. Section 3 analyzes the conflicts in operations in the XPath model. Section 4 presents the new XLP protocol, while Section 5 analyzes XLP and makes a comparison with other protocols. Finally, Section 6 concludes this study and discusses the future work.

2. Preliminary

2.1. XPath model and XPath expression

XPath [3] models an XML document as a tree of nodes. There are seven types of nodes: the root node, element nodes, text nodes, attribute nodes, namespace nodes, processing instruction nodes, and comment nodes. On the other hand, being a query language, XPath expressions are used to indicate the requested nodes in the XML tree. Basically, an XPath expression includes *structural constraints* and *predicates*. The XPath expression consists of a *location path* [3], which in turn consists of a sequence of one or more *location steps* separated by the symbol '/'. Each location step starts from a set of nodes, called *context nodes*. A location step is represented by *Axis*::*Node-Test*[*Predicate*], where *Axis* specifies the node relationships (e.g. parent, child or sibling, etc.) between the context nodes and the nodes whose type is identified by *Node-Test*. The location path and location steps are considered as structural constraints. In contrast, *Predicate* is used to sieve out the results from nodes satisfying the structural constraint *Axis*::*Node-Test* of a location step. As a result, the set of nodes satisfying both the structural constraint and the predicate of a location step becomes the result of that location step, which in turn becomes the context nodes of the next location step. The result of an XPath expression is