Available online at www.sciencedirect.com

SCIENCE DIRECT®

The Journal of Systems and Software 78 (2005) 37–46

The Journal of Systems and Software

www.elsevier.com/locate/jss

Lightweight multigranularity locking for transaction management in XML database systems

Yonggoo Choi *, Songchun Moon

Database Laboratory, Department of Management Engineering, Korea Advanced Institute of Science and Technology, Graduate School of Management, 207-43 Cheongryangri-dong, Dongdaemungu, Seoul 130-012, Republic of Korea

Received 26 December 2003; received in revised form 29 November 2004; accepted 13 December 2004
Available online 4 January 2005

Abstract

As *eXtensible Markup Language* (XML) provides a capability for describing data structures, and can accommodate many kinds of *semistructured data*. The semistructured data format is flexible for changing data structures through insertion and deletion of data elements in mission-critical applications. In the case of concurrently changing such a data format, this flexibility could be endangered by a phantom problem which might lead to inconsistent information flow. For the purpose of developing a concurrency control scheme without the phantom phenomenon, we propose a *lightweight multigranularity locking* (LWMGL) scheme that is a hybrid mechanism of *Tree-based Locking* and *Multigranularity Locking*. The goal of this scheme is to realize locking at the level of precise elements in an XML database while preventing the phantom problems. Since these precise locks could considerably reduce the number of pseudo-conflicts that are regarded as unnecessary locks, they provide high concurrency compared with other concurrency control schemes. In order to realize the LWMGL scheme we also devised a new data model of *XML indexed element tables* (XIETs) for transferring diverse XML documents. This data model does not only can preserve the XML tree structure in application levels, but also enables execution of the structural change operations as well as the data access operations in parallel.

Keywords: Concurrency control; Semistructured data; XML database; XML

1. Introduction

As *eXtensible Markup Language* (XML) provides a capability for describing data structures, and can accommodate many kinds of *semistructured data* (Bourret, 2004; McHugh et al., 1997; Suciu, 2002). The semistructured data format is characteristic of providing flexibility for its structural change through insertion and deletion of data elements in mission-critical applications such as electronic commerce and bioinformatics (Achard et al., 2001). In the case of concurrently changing such a data format, this flexibility could be endangered by a phantom problem which might lead to inconsistent information flow. For example, assume that a customer C_1 searches books in an XML book catalog document such as illustrated in Fig. 1(a). At the moment, assume that another customer C_2 is going to insert a new element, *Discount*, into the location given by an XML path expression, */Books/Book*, on a *Book* XML tree such as in Fig. 1(b) while C_1 can only see the book title element, *Title*, and the book price element, *Price*.

If C_1 tries to search collections of elements contained in */Books/Book* once more, it would find that the results that are different from those before searching for the phantom element, *Discount*. The phantom phenomenon has its origin in inserting *Discount* before C_1 finishes its book search. If a transaction scheduler could control the

* Corresponding author. Tel.: +88 2 958 3333; fax: +88 2 958 3604.
E-mail addresses: ygoochoi@kaist.ac.kr (Y. Choi), scmoon@kgsm.kaist.ac.kr (S. Moon).

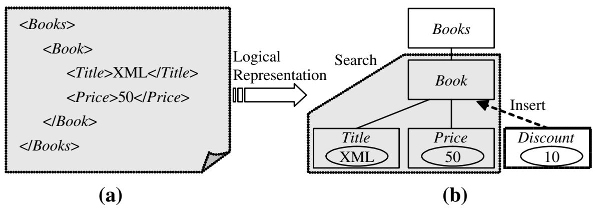

Fig. 1. An instance of a *Books* XML: (a) a *Books* XML document and (b) insertion of a new element into a *Books* XML tree.

schedules in order to insert the element after finishing the search, and vice versa, this phantom problem would not arise. There have been various attempts to resolve phantom problems in *dynamic databases* (Chaudhri and Hadzilocos, 1995) that are databases that contain a dynamic collection of dependent data elements, which can insert and delete at an application level. Most of these attempts have concentrated on hierarchical locking schemes i.e., *Tree-based Locking* (TL) (Bernstein et al., 1987) and *Multigranularity Locking* (MGL) (Bernstein et al., 1987; Helmer et al., 2004).

In order to prevent phantom problems, the TL scheme first holds a lock on the parent node before acquiring locks on the child node. A distinctive feature of TL is that a transaction releases a lock on a parent node immediately upon obtaining locks on all the children nodes. This handshake between locking children and unlocking their parent is known as *lock coupling* (Bernstein et al., 1987). Concurrency control schemes using such a lock coupling can achieve high concurrency because unnecessary locks are released earlier. Due to low locking cost and a useful index structure such as B-Tree (Gray and Reuter, 1993), TL has been universally applicable in traditional relational database management systems. However, TL does not be applicable for XML data management due to the absence of an efficient index structure for an XML document model and the phantom problems.

The MGL scheme has been the subject of considerable attention with regard to transaction schedules for data structures with containment relationships among granules. In order to manage subtree-level locks on a hierarchical data structure, this scheme involves a shared lock, an exclusive lock, and intention locks which are composed of intention shared locks and intention exclusive locks. Since these intention locks can give a previous notice to the other transactions, they can reduce the cost of searching entire elements on specified subtree for locking. Even though the MGL effectively preserves serializability from phantom problems, it may fail to achieve a high concurrency due to *pseudo-conflicts* (Grabs et al., 2002) in the case of XML databases. Such conflicts arise from inserting new elements into, or deleting child elements from, intention lock elements while other transactions perform on an XML database.

With a view to achieving an appropriate concurrency control scheme for typical XML applications, there have been several attempts to merge the hierarchical locking schemes with *Predicated Locking* (PL) (Choi and Kanai, 2003; Dekeyser and Hidders, 2004) and to combine these schemes with some properties such as multiple versions and operation semantics. Most of these attempts have concentrated on providing high concurrency as well as guaranteeing serializability, which could be endangered by phantoms. Unfortunately, the schemes do not focus sufficient attention on concurrency of transactions containing multiple operations which are composed of structural change operations as well as data access operations.

For the purpose of developing a concurrency control scheme without phantom phenomenon, we propose a *lightweight multigranularity locking* (LWMGL) scheme that is a hybrid mechanism of TL and MGL. The basic goal of this scheme is to realize locking at the unit of precise granule in the XML database while resolving the phantom problem. Since these precise locks could considerably reduce the number of pseudo-conflicts that are regarded as unnecessary locks, this approach provides high concurrency compared with other concurrency control schemes. In order to realize the LWMGL scheme we also devise a new data model of *XML indexed element tables* (XIETs) for transferring diverse XML document schemata to a relational database schema. This data model does not only preserve the XML tree structure at application levels, but enables execution of structural change operations as well as the data access operations in parallel.

2. Related works

Due to the nested hierarchical nature of the XML data model, the concurrency schemes are based on hierarchical locking protocols: *Tree-based Locking* (TL) (Bernstein et al., 1987; Fiebig et al., 2002) and *Multigranularity Locking* (MGL) (Bernstein et al., 1987; Helmer et al., 2004) for concurrency of transactions. Since these schemes have been developed appropriately in structured relational databases, they are inappropriate in semistructured XML databases. In this section, we describe the side effects and the system overheads for concurrency in the case of applying such schemes to transaction management in XML database systems.

2.1. Multigranularity locking

In the data structures nested as hierarchical granules, it is important that we can place locks on granules at different levels in the hierarchy for lock management overheads. Among the hierarchical locking schemes, MGL has been developed by organizing the database in a format of a lockable granule hierarchy represented as a