Transaction Validation for XML Documents based on XPath

Stefan Böttcher , Adelhard Türling
University of Paderborn
Fachbereich 17 (Mathematik-Informatik)
Fürstenallee 11 , D-33102 Paderborn , Germany
email : stb@uni-paderborn.de, Adelhard.Tuerling@uni-paderborn.de

Abstract: Whenever mobile devices modify copies of common XML data, then synchronization of multiple user transactions accessing this data is a key problem. However compared to standard XML database systems, transaction synchronization has to solve new problems, i.e. it has to manage lost connections and it furthermore should reduce data exchange over small bandwidth client connections for the purpose of synchronization. In order to avoid locks that are assigned to transactions of ”lost” clients, we suggest to use an optimistic approach to synchronization. In order to reduce the data exchange for synchronization, we furthermore suggest to exchange XPath expressions instead of larger XML fragments wherever possible. This allows us not only to treat lost connections more flexible but also to operate efficiently on mobile devices with small bandwidth connections to a server.

Keywords: mobile transactions, synchronization, small bandwidth connections, XML, XPath

1 Introduction

Problem origin: Whenever multiple client transactions access and modify server side XML documents concurrently, then synchronization of concurrent accesses to the XML documents is an important topic [3,12]. As soon as client transactions run on mobile devices, the transaction manager has to handle lost connections. Furthermore, when mobile devices use a small bandwidth connection to the server, the synchronization protocol should reduce data exchange between client and server.
Our work is motivated by an industrial CSCW application developed for web clients, that now shall be extended to mobile devices with small bandwidth connections. The clients use XPath expressions to access XML fragments that are stored on a central server. Our synchronization protocol is adapted to the needs of such mobile internet devices.

Related work and our focus: There are at least two major approaches to access persistent XML data: first, to map XML documents to relational or object oriented databases (e.g. [2,12]) and to use the given DBMS transaction scheduler for synchronization [3], and second, to use a specific XML database system (e.g. [11]) which allows to adapt the synchronization strategy to XML documents. We follow the second approach. Different from concurrency control in distributed database systems, we follow concurrency control protocols for client/server systems that interchange data to be modified by clients. Additionally, we consider the specific requirements of mobile transactions that are considered to be an additional challenge, because of limited bandwidth connections and frequent disconnection [9].

Within the area of mobile transactions, some contributions relax or redefine transaction properties (e.g. [6]), however, we follow the argumentation of e.g. [5] and support the classical ACID properties. Like other approaches in mobile transactions (e.g. [4]), we combine validation with client side data caching, however different from them we do not update the clients' caches by a server initiated broadcast, but leave the decision to refresh the read data up to the client. According to [1] and [8] this outperforms all other approaches to cache consistency in a client/server architecture like ours where the application processing is performed at the web client.
In contrast to all mentioned contributions, that validate mobile transactions on the basis of physically accessed data, we follow [10] and use a predicative approach to validation, i.e., in our case the validation uses XPath expressions. Different from all other approaches, our synchronization protocol is adapted to the specific needs of mobile web clients accessing XML-databases, i.e. it not only avoids problems with lost connections from web clients to a server-side XML database and is also well suited for small bandwidth connections. Furthermore, a key point of our contribution is to transfer (small) XPath expressions instead of transferring (larger) XML fragments wherever possible.

2. Problem description

2.1. XPath as access language to read fragments of the server's XML document

Several mobile clients share access to (fragments of) a server's XML document. XML fragments are transferred from server to client, where they are read or modified, and modifications of client transactions are transferred back to the server. The clients use XPath expressions to access XML fragments, i.e. all XML fragments read or written by clients can be described by XPath expressions. An XPath expression, e.g.

```
/doc/group[ @id='1' ]/item//*
```

describes a set of nodes of the underlying XML document. The path '/doc/group/item' starts at the root '/' of the XML document and uses the child axe to identify the selected elements, i.e. it selects 'item' elements under a 'group' element under a 'doc' element. The predicate filter [@id='1'] determines the 'group' elements to be considered, i.e. those with an attribute value of '1' for the attribute id. Finally, '//*' means that the nodes described by '/doc/group[@id='1']/item' are root nodes of the fragment that is accessed.

2.2. Requirements to our concurrency control protocol for mobile transactions

All client-access to XML fragments is done in the scope of transactions, however each transaction involves only a single client. A mobile client may loose its connection to the server during the execution of a transaction, but nevertheless the server should support completion of such a transaction after reestablishing the lost connection. Since for the server it is not acceptable to keep locks for a transaction of a mobile client that has lost the connection to the server, our approach uses a validation based protocol. Finally, most of the mobile client devices use a small bandwidth for the communication between client and server. This requires to avoid unnecessary exchange of large XML fragments for the purpose of validation.

3. Concurrency control for mobile clients

3.1 Predicative validation adapted to XPath

We present a validation protocol, that is adapted to the specific needs of mobile clients that use XPath expressions to access a server-side XML document. As in standard validation [7], first, clients work on local copies of XML fragments during a read phase, second, the server validates the clients' operations and eventually during a write phase modifies the XML document. Our protocol collects XML fragments instead of database tuples in the *write sets* of the transactions (i.e. in the sets containing old and new values of inserted, updated or deleted nodes of the XML document). Due to space limitations on the clients, write sets (and read sets) of transactions are only collected on the server.

The read phase: In contrast to standard validation [7], our server does *not* use sets of nodes (i.e. XML fragments) as read sets. Instead our server collects those XPath expressions in the read set that the client submits during the read phase. Since these XPath expressions will be used during the validation phase, no additional data transfer of read XML fragments or XPath expressions from client to server is needed, what we consider to be a competitive advantage compared to other validation based protocols.

We apply the same optimization that transfers XPath expressions instead of XML fragments from client to server wherever possible to delete and update operations. By this we avoid, that the client sends a deleted XML fragment (or the old values of an updated XML fragment) back to the server. This is possible because these XML fragments are already known to the server. Instead, clients send XPath expressions to the server that identify the deleted or updated XML fragment, such that the server can copy the old values for that fragment to its write set for that transaction. The client transfers only new values of inserted or updated fragments to the server, together with an XPath expression pointing to that fragment. For the purpose of synchronization, the server separates the XPath expressions used for insert, update or delete operations (the *write expression set*) from XPath expressions used for read operations (the *read set*).

The validation phase: Our validation protocol applies XPath expressions of the validating transaction to XML fragments collected in the write sets of older concurrent transactions. Similar to predicative validation [10], we use this as follows not only for read-write conflicts, but also for write-write conflicts.

Read-write conflicts of a validating transaction Tv with an older concurrent transaction To (i.e. a transaction that entered its validation phase before Tv, but was not completed before Tv started) are treated as follows:

```
For each XPath expression XPE in the read set of Tv:
    For each modified XML fragment MXF in the write set of To,
        If XPE applied to MXF is not empty, then return validation of Tv fails .
```

Write-write conflicts have to be checked only between Tv and an older transaction To2 that is validating concurrently to Tv (i.e. To2 enters its validation phase before Tv, but To2 is not completed at the time when Tv enters its validation phase). They can be checked as follows, because the write expression set contains all XPath expressions which are used to write a fragment of the XML document:

For each XPath expression XPE in the write expression set of Tv:
For each modified XML fragment MXF in the write set of To2,
If XPE applied to MXF is not empty, then return validation of Tv fails

If none of the validation checks fails, then return: Validation of Tv succeeds.

Note that the validation can be performed by a usual XPath query evaluator. Furthermore, our protocol has the following advantages. Our validation applies (read or written) XPath expressions only to small modified XML fragments, i.e. *not* to the whole XML document. We avoid to send XML fragments back from the client to the server wherever possible in order to adapt our protocol to a small communication bandwidth. We use this optimization, i.e. to exchange XPath expressions instead of XML fragments, not only for read operations but also for delete operations and old values of update operations. Note further that, since XPath expressions are considerably smaller than read XML fragments, main-memory storage of XPath expressions may be still possible, where storage of large read XML fragments would require to swap them to disk. Additionally, this avoids the phantom problem, because conflicting insert and read operations are found by querying the inserted fragments.
The write phase: Similar to standard validation [7], after successful validation, a transaction is committed and starts its write phase, during which it transfers the changes collected in its write sets to the original XML document.

3.2. How to treat lost connections

When a client looses its connection during a transaction before end of validation, this can never interfere with work of any other transaction, because changes on the XML fragment are made on local copies and are not yet transferred to the server's XML document.
As soon as the connection is reestablished, the client has the following alternatives. First, it can ignore its work, i.e. abort and restart the transaction, which includes to reload the XML fragments. Second, client can ask the server for an additional intermediate validation step that validates the current read set of the client (i.e. checks whether or not the current read set is up to date or has conflicts to modified XML fragments written in the meantime). Note that this can be done without transferring XPath expressions or even XML fragments from the client to the server, because the server keeps a copy of the XPath expressions in the transaction's read set. Third, the client can continue the transaction as if nothing happened, i.e. check everything in the usual way in the validation phase. Which of the three decisions after reestablishing a connection is appropriate, may depend on the work the client has done (i.e. if the client has not done much work, it may decide to restart the transaction) and on the duration the connection was lost (i.e. if the time was short, the client may decide to continue as if nothing happened). Note that whatever a client decides to do after reestablishing a connection, no other client has to take care of or can be damaged or delayed by that lost connection, which we consider to be an advantage of the optimistic approach that we use.
Lost connections during the validation phase or later are no problem, because at that time the data is already completely on the server. As soon as the connection is reestablished (and the transaction is completed), the server can inform the client about the commit status of its transaction.

4. Summary and Conclusions

We have presented a transaction synchronization protocol for client/server systems that exchange and modify XML fragments. Our protocol can handle lost client connections during transaction execution in multiple ways without disturbing the work of other clients, which makes it suitable for mobile clients. Our protocol is based on validation, but uses transferred XPath expression for validation purposes instead of transferring read or written XML fragments wherever possible, which we consider to be a competitive advantage for devices with small bandwidth connections. This includes not only the validation of read operations, but also the validation of delete operations, for which no XML fragments have to be transferred from client to server. Furthermore, our protocol avoids the phantom problem and can be implemented with standard XPath queries.
Although, we presented and developed our protocol specifically for the needs of mobile XML database clients with small bandwidth connections to the server, the protocol seems to be well suited for other XML databases using optimistic transactions as well.

References:

[1] Adya, A., Gruber, R., Liskov, B., Maheshwari, U.: Efficient Optimistic Concurrency Control Using Loosely Synchronized Clocks, ACM SIGMOD Int. Conf. on Management of Data, 1995.
[2] Bourret, R., Bornhövd, C., Buchmann, A.P.: A Generic Load/Extract Utility for Data Transfer Between XML Documents and Relational Databases. 2nd Int. Workshop on Advanced Issues of EC and Web-based Information Systems (WECWIS), San Jose, California, June, 2000.
[3] Böttcher, S., Türling, A.: Transaction Synchronization for XML Data in Client-Server Web Applications. Informatik 2001, Jahrestagung der GI, Wien, 2001.
[4] Chung, I.-Y., Hwang, C.-S.: Transactional Cache Management with Aperiodic Invalidation Scheme in Mobile Environments. ASIAN 1999: 50-61.
[5] Gore, M.M., Ghosh, R.K.: Recovery of Mobile Transactions. DEXA Workshop 2000: 23-27.
[6] Ku, K.I., Yoo-Sung, K.: Moflex Transaction Model for Mobile Heterogeneous Multidatabase Systems. RIDE 2000: 39-46.
[7] Kung, H.T., Robinson, J.T.: On Optimistic Methods for Concurrency Control. ACM TODS, 6, 2, 1981.
[8] Öszu, M.T., Valduriez, P.: Distributed Database Systems, 2nd Ed., Prentice Hall, 1999.
[9] Rasheed, A., Zaslavsky, A.B.: A Transaction Model to Support Disconnected Operations in a Mobile Computing Environment. OOIS 1997: 120-130
[10] Reimer, M.: Solving the Phantom Problem by Predicative Optimistic Concurrency Control, 9th VLDB, Florenz, 1983.
[11] Schöning, H., Wäsch, J.: Tamino -- An Internet Database System. Proc. of the 7 th Int. Conf. on Extending Database Technology (EDBT), Springer, LNCS 1777, Konstanz, March, 2000.
[12] Tatarinov, I., Ives, Z.G., Halevy, A.Y., Weld, D.S.: Updating XML, ACM SIGMOD Int. Conf. on Management of Data, 2001.