Um Controle de Concorrência Distribuído para Dados XML

Leonardo O. Moreira[1], **Flávio R. C. Sousa** [1,2], **Javam C. Machado**[1]

[1] Grupo de Redes, Engenharia de Software e Sistemas (GREat)
Universidade Federal do Ceará (UFC) – Fortaleza, CE – Brasil

[2]Campus de Quixadá - Universidade Federal do Ceará (UFC)
Estrada do Cedro, Km 5 – Quixadá, CE – Brasil

{leoomoreira,sousa,javam}@ufc.br

Abstract. *XML has become a widely used standard for data exchange among applications. Consequently, a large amount of data is distributed on the Web and stored in different persistence models. DBMSs provide concurrency control techniques to manage such data. However, the structure of XML data makes the application of these techniques difficult. Regarding distributed environments, there are still few papers and they have limitations. This paper introduces DTX, a mechanism for distributed concurrency control for XML data. In order to evaluate DTX, experiments that measure its performance are presented.*

Resumo. *XML tem se tornado um padrão amplamente utilizado na troca de dados entre aplicações. Com isso, um grande volume desses dados está distribuído na Web e armazenado em diversos meios de persistência. SGBDs fornecem técnicas de controle de concorrência para gerenciar esses dados. Entretanto, a estrutura dos dados XML dificulta a aplicação dessas técnicas. Considerando o ambiente distribuído, ainda existem poucos trabalhos e estes apresentam algumas limitações. Este artigo apresenta o DTX, um mecanismo para o controle de concorrência distribuído para dados XML. Visando avaliar o DTX, são apresentados alguns experimentos que medem seu desempenho.*

1. Introdução

XML [W3C 2007] tem se tornado um padrão amplamente utilizado na representação e troca de dados em aplicações. Como conseqüência, um grande volume destes dados está espalhado ou distribuído na Web e armazenado em diversos meios de persistência. Com isso, aplicações necessitam ter acesso e manipular dados distribuídos. Portanto, torna-se necessária a existência de sistemas eficientes para o gerenciamento destes documentos XML em ambiente distribuído. Para gerenciar estes dados, os SGBDs tradicionais implementam protocolos de controle de concorrência distribuídos. Entretanto, a estrutura dos dados XML dificulta a aplicação destes protocolos, afetando o nível de concorrência, tanto em ambientes centralizados como distribuídos.

Em ambiente centralizado, existe uma grande quantidade de propostas de controle de concorrência para dados XML. Alguns protocolos são baseados em bloqueios hierárquicos em árvores, e estes ocorrem de forma *top-down*, ou seja, os nós desde o ponto inicial da consulta até o final do documento são bloqueados, dificultando a execução concorrente de consultas. Protocolos baseados no modelo DOM utilizam diferentes tipos de bloqueio para agrupar nós de níveis distintos, o que ocasiona um número elevado de

conflitos e, conseqüentemente, ocorre um número maior de *deadlocks*. Protocolos baseados em bloqueios de caminho aumentam a concorrência, fazendo com que utilizem um subconjunto muito limitado da linguagem XPath e métodos dispendiosos para determinar conflitos entre consultas complexas, que inviabilizam sua aplicação em sistemas práticos. Alguns protocolos se servem de estruturas como o *DataGuide* para gerenciar o acesso aos dados e apresentam melhores resultados [Grabs et al. 2002] [Pleshachkov et al. 2005].

Considerando o ambiente distribuído, há ainda poucos trabalhos como [Pagnamenta 2005] [Tamino 2007] [Seltzer 2007]. Estes apresentam um baixo nível de concorrência, afetando o tempo de resposta e não exibem estudos que avaliem explicitamente aspectos de desempenho. Visando a prover um gerenciamento eficaz em ambientes distribuídos, o artigo apresenta o DTX como mecanismo para o controle de concorrência distribuído para dados XML, que leva em consideração características estruturais destes dados e melhora a concorrência entre transações. O trabalho tem como foco seguir as propriedades transacionais de consistência e isolamento no âmbito distribuído, melhorando o desempenho através do acesso concorrente aos dados fisicamente distribuídos. O artigo está organizado da seguinte forma. A seção 2 descreve o DTX. Na seção 3, os algoritmos utilizados pelo DTX são discutidos. Vários testes de desempenho do DTX são apresentados na seção 4. A seção 5 comenta os trabalhos relacionados e, finalmente, a seção 6 apresenta as conclusões.

2. DTX

O DTX é um mecanismo de controle de concorrência distribuído para dados XML, que leva em consideração características estruturais destes dados. O DTX visa a melhorar o desempenho no acesso aos dados XML, utilizando um protocolo para controle de concorrência multi-granular que aumenta o paralelismo entre as transações e possui uma estrutura otimizada para representação dos dados. A manipulação dos dados XML é feita em memória principal. Para isso, o DTX recupera os documentos XML de uma estrutura de armazenamento, faz os devidos processamentos e, posteriormente, atualiza as alterações na estrutura de armazenamento. As estruturas de armazenamento destes documentos é independente, ou seja, o DTX oferece suporte de comunicação com qualquer método de armazenamento de documentos XML. O DTX opera sobre dados XML total ou parcialmente replicados.

Em seu controle de concorrência, o DTX utiliza o protocolo XDGL [Pleshachkov et al. 2005] adaptado com a finalidade de contemplar as propriedades de isolamento e consistência em âmbito distribuído. O XDGL é uma extensão do protocolo DGLOCK [Grabs et al. 2002], que utiliza uma estrutura *DataGuide* para representar os bloqueios. Esse protocolo pode ser implementado no topo de qualquer sistema que gerencia de dados XML, uma vez que possui uma estrutura própria de representação. Por utilizar uma estrutura otimizada para representar bloqueios, o XDGL possui uma maior eficiência na gerência destes bloqueios. Este protocolo requer que a transação esteja em acordo com o *Strict 2PL* [Özsu and Valduriez 1999]. Portanto, a transação adquire e mantém bloqueios até o seu término. São promovidos os princípios de bloqueios em granularidade múltipla a nível de elementos XML, o que torna esse protocolo mais concorrente. Assim, como no protocolo DGLOCK, são introduzidos os bloqueios de intenção. Ao contrário de seu protocolo base, o XDGL garante o critério de serializabilidade. O XDGL utiliza um subconjunto da linguagem XPath para recuperar informações

dos documentos XML. Para a atualização de dados nos documentos XML foi definida uma linguagem de atualização. Esta linguagem possui cinco tipos de operações para atualização: inserção, remoção, transposição, renomeação e troca.

Foram feitas três modificações no protocolo XDGL para seu funcionamento em ambiente distribuído: (i) inserção de uma infra-estrutura de comunicação entre os escalonadores, a fim de executar operações remotas, ao mesmo tempo em que adquire os bloqueios necessários e permite a consolidação e o cancelamento de uma transação distribuída; no cancelamento a transação desfaz todos os seus efeitos nos dados requeridos; (ii) distribuição do gerenciador de bloqueios em cada instância do DTX com o objetivo de descentralizar o módulo de aquisição e liberação de bloqueios, deixando a cargo de cada instância gerenciar os bloqueios sobre seus dados; (iii) alteração da forma pela qual o XDGL faz a detecção e o tratamento de *deadlocks*, bem como adição de um processo que periodicamente percorre todas as instâncias do DTX, e verifica se, na união dos grafos de espera, ocorre um ciclo, ou seja, um *deadlock*. É importante ressaltar que o DTX foi concebido de modo flexível, de forma que outros protocolos de controle de concorrência possam ser utilizados, tendo apenas que fazer as modificações descritas anteriormente.

Pelo fato de utilizar uma adaptação do protocolo para controle de concorrência, o DTX possui limitações quanto à linguagem de consulta e atualização. O subconjunto da linguagem XPath, empregado para recuperação de informações no protocolo XDGL, é comum à linguagem de consulta do DTX; quanto à linguagem de atualização, segue a mesma idéia. As regras de bloqueios para o processamento das operações do XDGL são comuns àquelas de bloqueios empregadas no processamento de operações do DTX [Pleshachkov et al. 2005].

No presente trabalho é usado o modelo de transações distribuídas baseado em *coordenadores* e *participantes* [Özsu and Valduriez 1999]. O DTX serve-se da abordagem síncrona para a execução das transações, com o emprego da técnica de bloqueios. Com esta abordagem é possível garantir o ordenamento das mensagens de comunicação entre sítios, e também permitir acesso exclusivo a recursos compartilhados. No DTX, cada instância faz a gerência de bloqueios em seus dados locais. Como o controle de concorrência utiliza técnicas de bloqueios, poderão eventualmente surgir *deadlocks*; então, o DTX implementa a política de detecção de *deadlocks*, servindo-se da técnica de união de grafos de espera das instâncias do DTX. Na ocorrência de *deadlock*, assim como no protocolo XDGL original, a transação mais recente envolvida no ciclo é cancelada. O DTX implementa a seqüência clássica de execução de transações e utiliza o nível de isolamento *read-committed*, no qual as transações concorrentes não vêem as alterações pendentes umas das outras.

2.1. Arquitetura

O DTX é dividido em alguns componentes de *software* que realizam comunicação entre si, implementando o controle de concorrência distribuído para dados XML. Uma visão geral da arquitetura do DTX pode ser observada na Figura 1.

O *Listener* é o componente responsável em receber as requisições dos clientes. Fornece uma interface simples para processar transações, recuperar seus resultados e encaminhá-los para o cliente. Uma outra função do *Listener* é receber, tratar e encaminhar requisições de outros escalonadores ao escalonador do DTX para realizar uma tarefa dis-

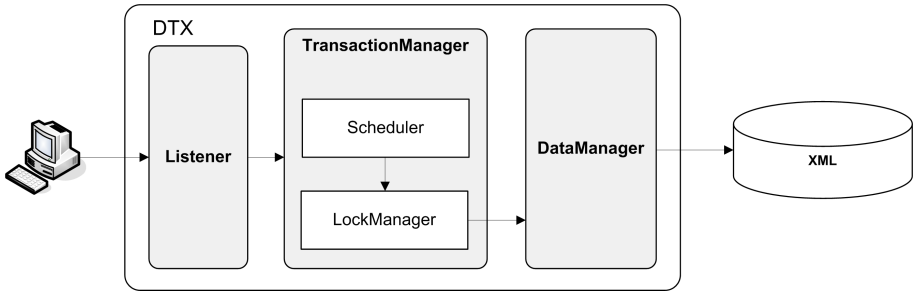


Figura 1. Arquitetura do DTX

tribuída ou de sincronização.

O *TransactionManager* é o componente responsável pela execução das transações e é composto de duas partes: o *Scheduler*, que tem por função escalonar a execução entre operações de transações, utilizando as regras do protocolo para controle de concorrência, detectar/tratar *deadlocks* e executar as operações através do *LockManager*; o *LockManager* que contém a estrutura de representação de dados e de bloqueios (i.e., *DataGuide*) utilizada para percorrer de forma otimizada os dados XML; esta segunda parte contém também as regras para concessão de bloqueios e as operações de manipulação dos dados XML. O *DataManager* é o componente utilizado pelo DTX para interagir com a estrutura de armazenamento dos dados XML. Ele é responsável em recuperar os dados XML da estrutura de armazenamento, convertê-los em uma estrutura própria de representação e fornecer meios de atualizar estes dados na estrutura de armazenamento.

2.2. Especificação

A cada sítio (i.e., nó do sistema) é acoplada uma instância do DTX entre os clientes e a estrutura de armazenamento dos dados XML; essas instâncias se comunicam entre si para executar uma transação distribuída. Para submeter uma transação ao DTX, o cliente realiza a conexão com uma instância do DTX e envia a transação. O componente *Listener* do DTX recebe as transações enviadas pelos clientes. Ao chegar uma transação, o *Listener* repassa-a ao gerenciador de transação para monitorar sua execução. O gerenciador de transação, por sua vez, envia essa transação ao escalonador para a execução em conjunto com outras concorrentes. O escalonador tem a função de decidir qual transação irá executar e obter bloqueios necessários à execução com o gerenciador de bloqueios do sítio corrente. O componente escalonador do sítio, em que a transação foi iniciada, é chamado de *coordenador*.

Se a transação contiver alguma operação a ser executada em outros sítios, o *coordenador* envia esta operação para os escalonadores dos sítios de destino, aguarda suas execuções e prossegue para a próxima operação. O sítio que recebe a operação enviada pelo *coordenador* é denominado *participante*. O *coordenador* também tem a responsabilidade de periodicamente verificar e tratar *deadlocks* distribuído bem como consolidar ou cancelar uma transação distribuída. O escalonador, ao conseguir os bloqueios necessários para uma operação, realiza-a interagindo com o gerenciador de bloqueios que, por sua vez, atualiza o gerenciador de dados.

O critério de *serializabilidade* global é obtido pela utilização de técnicas de bloqueios nos sítios envolvidos em transações distribuídas, uma vez que o protocolo para

controle de concorrência, utilizado no DTX, garante esse critério em um único sítio. Em [Türker et al. 2005] é ilustrado uma solução para o processamento de transações em grades computacionais. Os autores provam o critério de *serializabilidade* transacional global onde não há um coordenador central. O método de demonstração pode ser aplicado ao DTX, vez que tem essas características semelhantes, ou seja, não possui um escalonador central. Ainda segundo os autores, para garantir a *serializabilidade*, os escalonadores devem ter conhecimentos sobre o conflito entre transações. No DTX, isso é obtido na tentativa de aplicação de bloqueios em operações. Caso em algum sítio não seja possível obter o bloqueio, o sistema é informado de que houve um conflito de bloqueios entre transações.

Com relação à consolidação de uma transação, esta só poderá surgir se não depender de nenhuma outra transação ativa. Assim, apenas é permitida a consolidação de transações que executaram todas suas operações; para executar uma operação a transação deve obter os bloqueios necessários em todos os sítios alvos desta operação. Caso em algum sítio não se obtenham todos os bloqueios, a transação entra em modo de espera. Com isso, pode-se dizer que uma transação só consolida se não houver dependência de alguma outra ativa.

Para garantir o isolamento e a consistência, o *coordenador* deve obter os bloqueios necessários para cada operação e executar em todos os sítios *participantes*. Quando uma operação for executada em outros sítios e em algum destes não ocorrerem bloqueios necessários, a transação é posta em modo de espera e, nos sítios em que a operação for executada, suas ações serão desfeitas. Com isso, é garantido que uma operação só executa em sua totalidade, ou seja, ela tem que ocorrer em todos os sítios *participantes*. Ao término de uma transação, seja por sucesso ou falha, o *coordenador* deve consolidar ou cancelar as operações realizadas em todos os sítios envolvidos na transação distribuída. Caso em algum sítio não seja possível consolidar, a transação é cancelada. Se no cancelamento de uma transação não for possível realizar este procedimento em algum sítio, a transação falha. Em caso de falha, o DTX alerta a aplicação de que a transação falhou.

Com relação à garantia do término de uma transação, o DTX utiliza um processo que periodicamente percorre todos os escalonadores de todos os sítios com a finalidade de recuperar seus grafos de espera e verificar se na união ocorre um ciclo. Seguindo as regras do protocolo adotado, a transação mais recente envolvida no ciclo é cancelada. Quando uma transação consolida, aquelas que entram em modo de espera, aguardando por bloqueios da que consolidou, entram novamente em execução. Com isso, pode-se sempre dizer que uma transação consolida, cancela ou falha.

2.3. Cenário de Execução

Para melhor demonstrar o funcionamento do DTX, é ilustrado um cenário de execução. Neste exemplo, existem dois clientes: *c1* e *c2*. Eles estão alocados em diferentes sítios: *s1* e *s2* respectivamente. Para simplificar o entendimento, demonstra-se este exemplo com pequenos fragmentos de documentos XML. O primeiro documento *d1* contém informações a respeito de clientes de alguma instituição de vendas. O documento *d2* armazena informações sobre produtos que são vendidos nesta loja. Estruturalmente, o documento *d1* possui um elemento raiz chamado de `people`, que contém vários elementos `person`. Um elemento `person` possui dois sub-elementos, `id` e `name`, que representam um identificador único de `person` e seu nome respectivamente. Já o documento *d2*

possui um elemento raiz chamado `products`. O elemento `products` contém vários elementos `product` e este elemento possui os elementos `id`, `description` e `price`, que representa o identificador único do produto, sua descrição e preço.

Para a execução do cenário, definem-se três transações: *t1*, *t2* e *t3*. O cliente *c1* submete a transação *t1*, e o cliente *c2* envia as transações *t2* e *t3*. A transação *t1* contém duas operações: $t1_{op1}$ e $t1_{op2}$. A $t1_{op1}$ é uma consulta do cliente com identificador de número `4`. O identificador está relacionado aos atributos `id` contidos nas estruturas XML de `people` e `products`; é uma chave que identifica unicamente cada registro de clientes e produtos. Já a $t1_{op2}$ é uma inserção de um produto, denominado `Mouse`, de preço `10.30` e identificador de número `13`.

Transaction t1	Transaction t2	Transaction t3
t1op1	***t2op1***	***t3op1***
/people/person[@id=4]	/products	/products/product[@id=14]
t1op2	***t2op2***	***t3op2***
insert <product id="13">	**insert** <person id="22">	**insert** <product id="32">
<description>Mouse</description>	<name>Patrícia</name>	<description>Keyboard</description>
<price>10.30</price>	</person>	<price>9.90</price>
</product> **into** /products	**into** /people	</product> **into** /products

Figura 2. Descrição das transações

A transação *t2* contém duas operações: $t2_{op1}$ e $t2_{op2}$. A primeira é uma consulta que recupera todos os produtos da loja. A última operação de *t2* contém uma inserção de uma cliente, chamada `Patrícia`, com identificador `22`. Já a transação *t3* também possui duas operações: $t3_{op1}$ e $t3_{op2}$. A operação $t3_{op1}$ representa uma consulta do registro do produto que possui identificador `14`. A segunda operação de *t3*, a $t3_{op2}$ é uma inserção de um produto denominado `Keyboard`, de preço `9.90` e identificador `32`. Uma melhor visualização do conteúdo de todas essas transações pode ser obtida na Figura 2. O sítio *s1* gerencia uma cópia do documento *d1*; já o sítio *s2* contém uma cópia de todos os documentos: *d1* e *d2*. A Figura 3 ilustra a arquitetura, alocação dos clientes e documentos nos sítios que compõem este exemplo.

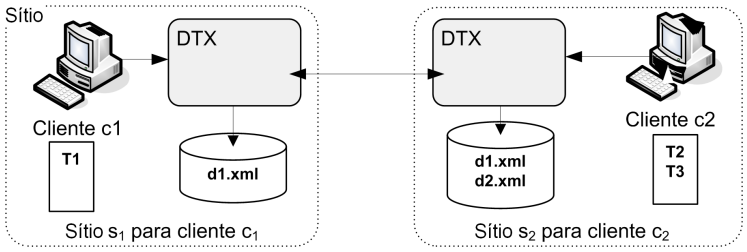


Figura 3. Exemplo do cenário

Ao iniciar este cenário, os clientes *c1* e *c2* submetem as transações *t1* e *t2* respectivamente. Considere-se que, para este caso, as submissões das transações ocorreram concomitantemente. Cada instância DTX dos sítios locais recebe as transações pelos seus componentes *Listener*, o qual, por sua vez, encaminha essas transações para seus respectivos *TransactionManager*. O *TransactionManager* envia as transações para o *Scheduler*, que armazena essas transações em uma fila, para que possam ser executadas concorrentemente com outras pendentes.

Suponha-se, para este cenário de execução, que o escalonador do sítio *s1* inicia a realização da $t1_{op1}$. Para executar esta operação, o escalonador deve obter os bloqueios necessários nos dois sítios, pois o documento *d1* está presente nestes. Supondo que não existam outras transações nas instâncias do DTX, os bloqueios são obtidos utilizando as regras do protocolo XDGL, e então a operação é processada. Neste instante, a primeira operação da transação *t2* inicia sua execução no sítio *s2*. O documento *d2* está presente apenas no sítio *s2*; como não existem transações executando sobre o documento *d2*, a operação $t2_{op1}$ adquire os bloqueios e é realizada. Os estados das estruturas de representação dos bloqueios e dados podem ser visualizados na Figura 4.

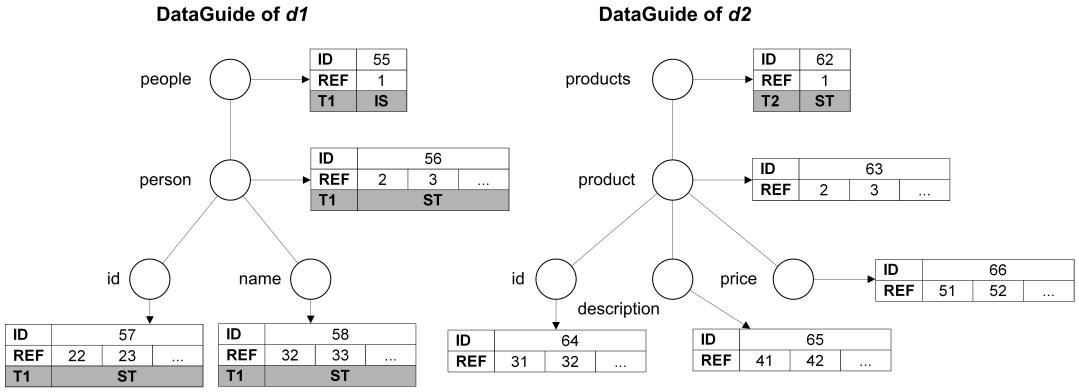


Figura 4. *DataGuides* dos documentos

O escalonador do sítio *s1* inicia a execução da operação $t1_{op2}$. Esta operação manipula o documento *d2*, então é enviada ao sítio *s2* para ser executada. Não é possível obter bloqueios de inserção para $t1_{op2}$, pois a transação *t2* mantém bloqueios de consulta em *d2*. A transação *t1* necessita realizar um bloqueio `IX` no nó identificado por `2` no *DataGuide*. Este nó possui um bloqueio `ST` que gera incompatibilidade entre bloqueios. Maiores detalhes sobre as regras de bloqueios podem ser obtidos em [Pleshachkov et al. 2005]. Neste momento, a transação *t1* é posta em estado de espera. O escalonador de *s2* decide então executar a operação $t2_{op2}$. O documento *d1*, alvo da próxima operação, está presente em todos os sítios. Portanto, é necessário enviar a operação e obter os bloqueios necessários nos dois sítios. Não é possível obter os bloqueios de inserção para $t2_{op2}$, pois a transação *t1* mantém bloqueios de consulta sobre o documento *d1*. A transação *t2* necessita realizar um bloqueio `IX` no nó identificado por `56` no *DataGuide*. Este nó possui um bloqueio `ST` que também ocasiona incompatibilidade entre bloqueios. Com isso, a transação *t2* também entra em modo de espera. A Figura 5 mostra a situação de incompatibilidade entre bloqueios de transações distintas nos *DataGuides* dos documentos.

Neste instante, visualiza-se uma situação de *deadlock* distribuído. O DTX possui um processo no escalonador que periodicamente recupera os grafos de espera de todos os sítios e verifica a ocorrência de *deadlocks*. Suponha-se que o escalonador do sítio *s1* inicie o processo de verificação de *deadlocks* e encontre as transações alvos do ciclo. Pelas regras do protocolo, a transação mais recente deve abortar; então a transação *t2* é cancelada, suas modificações desfeitas e seus bloqueios liberados.

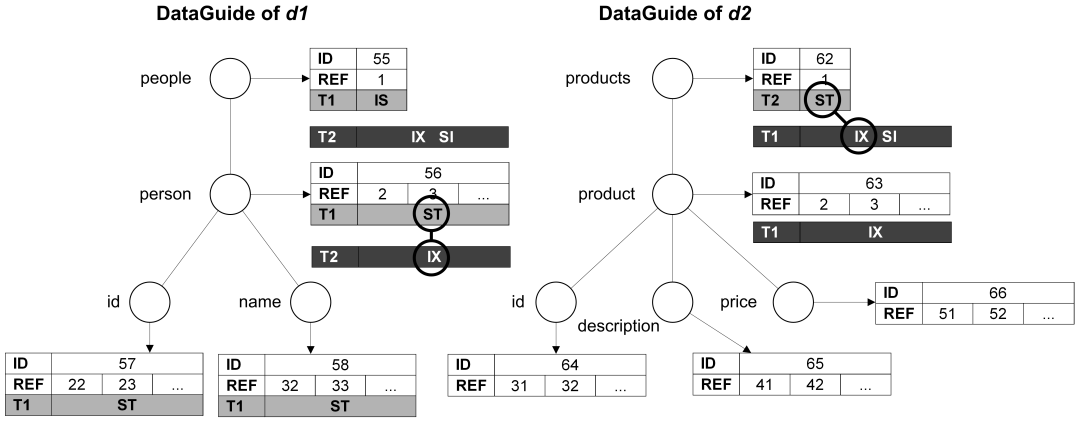


Figura 5. Incompatibilidade entre bloqueios

Considerando que *t1* pode entrar em execução, o escalonador de *s1* envia novamente a operação $t1_{op2}$ para execução no sítio *s2*. Neste caso, não existem bloqueios sobre o documento *d2*; esta operação adquire os bloqueios necessários para a inserção e processa. A transação *t1* não possui mais operações a serem executadas; então, esta inicia o processo de consolidação. Na consolidação, as modificações efetuadas pela transação são persistidas e seus bloqueios liberados. É de responsabilidade da aplicação cliente *c2* decidir se coloca novamente a transação *t2* para uma tentativa de execução. Considerando que o cliente descarta a transação *t2* e decide executar a transação *t3*, esta obtém os bloqueios necessários em todos os sítios e realiza seu processamento, visto que não há transações concorrentes.

3. Algoritmos

Esta seção descreve os principais algoritmos do DTX. O Algoritmo 1 é o procedimento executado pelo escalonador *coordenador* para o processamento das transações. O processo é repetitivo e circular, analisa e recupera a próxima transação disponível na lista de transações (l. 3). As transações disponíveis são aquelas que não estão no estado de espera. Após recuperar uma transação, é escolhida a primeira operação que ainda não foi executada (l. 4) para, em seguida, verificar onde esta operação será executada (l. 5). Se a operação for executada somente no sítio do *coordenador*, a operação será realizada no gerenciador de bloqueios local (l. 6 a 9); caso a operação seja executada em algum outro sítio, esta será enviada e executada em todos os *participantes* que contém o dado envolvido nesta operação (l. 12 a 22). Se a operação for executada somente no *coordenador* e não obtiver os bloqueios necessários, a transação entra em estado de espera (l. 9).

No caso de uma operação que será executada em algum outro sítio (l. 11), esta será enviada a todos os *participantes* (l. 11 e 12), inclusive para o *coordenador*, caso ele contenha o dado envolvido no processamento. A operação enviada para ser executada em outro sítio é chamada de operação remota. É válido ressaltar que o *coordenador* aguarda a execução de todos os sítios onde a operação foi enviada (l. 14). Caso a operação não adquira bloqueios em algum dos sítios *participantes* (l. 15), o procedimento da l. 16 desfaz as ações em todos os sítios em que a operação executou; após isto, a transação entra no estado de espera (l. 17). Se a operação falhar em algum dos sítios *participantes* ou gerar

Algoritmo 1 Procedimento executado pelo Coordenador

```
1:  procedure process_transactions
2:      loop
3:          transaction ← buffer.next_transaction_avaliable();
4:          operation ← transaction.next_operation();
5:          if operation contain only site of coordinator then
6:              if LockManager.process_operation(operation, wait_for_graph) then
7:                  operation.set_executed(true);
8:              else
9:                  transaction.wait();
10:             end if
11:         else
12:             participants ← sites.get_participants(operation.get_sites());
13:             participants.send_operation(operation);
14:             wait_for_responses();
15:             if operation.not_adquire_locking() then
16:                 undo_operation(operation);
17:                 transaction.wait();
18:             else
19:                 if operation.aborted() or operation.deadlock() then
20:                     abort_transaction(transaction);
21:                 end if
22:             end if
23:         end if
24:         if transaction does not have avaliable operation then
25:             commit_transaction(transaction);
26:         end if
27:     end loop
28: end procedure
```

(a) Procedimento executado pelo Coordenador

Algoritmo 2 Procedimento executado pelos Participantes

```
1:  procedure process_transactions
2:      loop
3:          remote_operation ← buffer.next_remote_operation_avaliable();
4:          if remote operation is valid then
5:              if LockManager.ps_operation(remote_operation, wait_for_graph) then
6:                  remote_operation.set_executed(true);
7:              else
8:                  remote_operation.set_adquire_locking(false);
9:              end if
10:             if remote_operation.failed() or remote_operation.deadlock() then
11:                 remote_operation.set_aborted(true);
12:             end if
13:             send_remote_operation_coordinator(remote_operation);
14:             process_commit_messages();
15:             process_abort_messages();
16:         end if
17:     end loop
18: end procedure
```

(b) Procedimento executado pelos Participantes

Algoritmo 3 Procedimento executado pelo Gerenciador de Bloqueios

```
1:  procedure process_operations
2:      result ← null;
3:      while not go through all the DataGuide elements of the operation do
4:          tr_conflited ← DataGuide.ps_locking(operation, locktable, currentnode);
5:          if tr_conflited is empty then
6:              result ← DataGuide.update(operation);
7:          else
8:              wait_for_graph.add_link(operation.get_transaction(), tr_conflited);
9:              if wait_for_graph.is_circle() then
10:                 operation.set_deadlock(true);
11:             end if
12:             DataGuide.undo_operation(operation);
13:             return null;
14:         end if
15:     end while
16:     return null;
17: end procedure
```

(c) Procedimento executado pelo Gerenciador de Bloqueios

Figura 6. Principais algoritmos do DTX

um *deadlock* (l. 19), a transação é cancelada (l. 20). Quando a transação analisada não contiver mais operações a serem executadas (l. 24), ela será consolidada (l. 25). Quando uma transação é cancelada, todas as operações são desfeitas em todos os sítios *participantes* e no *coordenador*, e todos os bloqueios sobre os dados envolvidos são liberados. Ao consolidar uma transação, as atualizações são efetivadas e todos os bloqueios sobre os dados envolvidos na transação são liberados.

O Algoritmo 2 descreve o comportamento dos escalonadores ao executar operações remotas nos sítios participantes. Ressalta-se que este procedimento também é comum ao coordenador. As operações remotas são aquelas que o coordenador envia para serem executadas em outros sítios. Todas essas operações são armazenadas em uma lista e recuperadas em modo seqüêncial (l. 3). Se houver alguma operação remota a ser executada (l. 4), os bloqueios necessários são adquiridos e a operação é processada no gerenciador de bloqueios do sítio participante (l. 5). Caso a operação não adquira bloqueios necessários, ela será marcada de modo a identificar esta ação pelo coordenador (l. 8). Se por algum motivo a operação falhar (l. 10), ela será marcada com o indicador de abort (l. 11). Ao término da execução da operação remota, seja por sucesso ou fracasso, o estado da operação é enviado ao coordenador (l. 13). Após processar as operações remotas, os participantes executam os procedimentos que tratam as mensagens remotas de consolidação e cancelamentos das transações distribuídas (l. 14 e 15).

O Algoritmo 3 descreve o procedimento de aquisição de bloqueios e realização das operações no gerenciador de bloqueios pelo escalonador. Este procedimento recebe como

parâmetro a operação a ser executada e o grafo de espera para o tratamento e detecção de *deadlocks*. O procedimento percorre todos os elementos do *DataGuide*, baseando-se nos nós utilizados na operação (l. 3). A cada elemento percorrido, é verificada a possibilidade de obtenção de bloqueio para o elemento em questão; caso não seja possível, é retornada a transação que mantém o bloqueio sobre o dado requerido (l. 4). Caso o bloqueio seja adquirido com sucesso (l. 5), o *DataGuide* é atualizado (l. 6). Se houver uma transação conflitante, será adicionada uma aresta que liga a transação conflitante com a transação da operação no grafo de espera (l. 8). Caso a adição de uma aresta no grafo de esperas gerar um ciclo (l. 9), isto é, ocorrer um *deadlock*, a operação será marcada com a ação de ocorrência de *deadlock* para ser tratada pelo escalonador (l. 10). Após a análise da ocorrência de *deadlock*, as modificações feitas pela operação no *DataGuide* e no gerenciador de bloqueios são desfeitas (l. 12). Este procedimento retorna o resultado da operação em caso de sucesso em sua execução (l. 17), e será considerado nulo, caso não adquira bloqueios e/ou ocorra *deadlocks* (l. 14).

O algoritmo de gerência de *deadlocks*, não presente neste artigo, é um processo que percorre periodicamente todos os sítios do sistema, captura o grafo de espera de cada sítio e faz as junções destes. Caso o grafo resultante contenha ciclo, a transação mais recente envolvida no ciclo é cancelada. Tanto na consolidação quanto no cancelamento distribuído de uma transação, os algoritmos percorrem todos os sítios alvos da transação, efetuam a consolidação ou cancelamento desta no sítio corrente e liberam os respectivos bloqueios mantidos. Maiores detalhes podem ser obtidos em [Moreira 2008].

4. Avaliação

A avaliação deste trabalho busca analisar o desempenho proporcionado pelo DTX. Para a avaliação do DTX, estende-se o *benchmark* XMark [Schmidt et al. 2002], adaptando suas consultas à linguagem XPath e adicionando operações de atualização, de forma a viabilizar a execução de experimentos. Desenvolve-se um simulador de clientes baseados em [Sousa et al. 2007] denominado DTXTester. Durante os experimentos com o DTX, utiliza-se o SGBD XML Nativo Sedna [Fomichev et al. 2006] por ser um sistema *open-source* e com as características de armazenamento e processamento de consultas, necessárias à execução dos experimentos.

4.1. Ambiente

É tomado um conjunto de sítios S=$\{S_1...S_N\}$. Cada sítio S_i possui um SGBD XML Nativo Sedna contendo os documentos XML adequados a cada experimento e uma instância do DTX acoplada ao Sedna. Considera-se um conjunto de clientes C=$\{C_1...C_M\}$, que contém os aplicativos que dão origem às transações. Para processar uma transação *t*, um cliente C conecta-se ao DTX e submete a transação *t*. Para cada transação *t*, somente um sítio S_i a inicia (*coordenador*) e, se *t* possuir operações a serem executadas em outros sítios, o DTX do sítio S_i envia estas para os demais sítios (*participantes*).

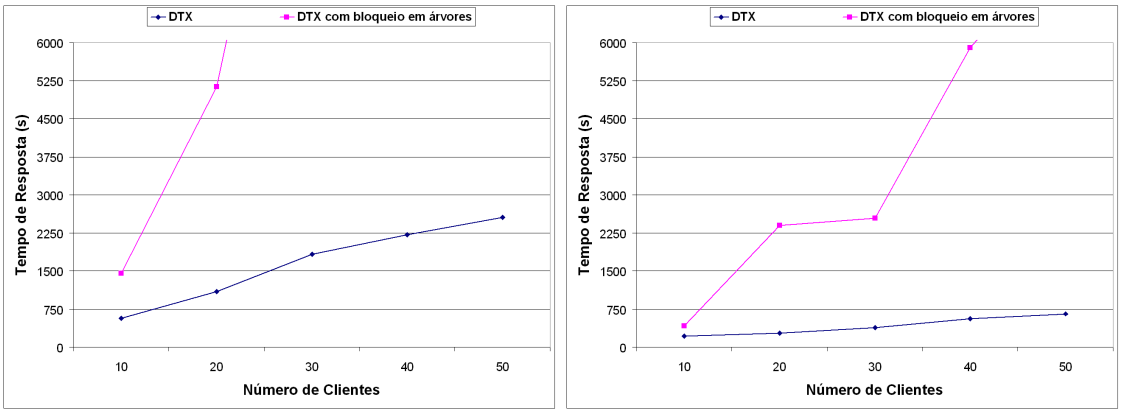
A concorrência de transações é simulada quando se usam múltiplos clientes. O simulador gera as transações de acordo com certos parâmetros, envia para o DTX e coleta os resultados ao final de cada execução. O ambiente utilizado para a avaliação foi um *cluster* de 8 PCs conectados através de um *Hub Ethernet*. Cada PC possui um processador de 3.0 GHz, 1 GB de RAM, sistema operacional Windows XP e interface de rede *full-*

duplex de 100 Mbit/s. A base de dados foi um documento de 40MB XML, gerado pelo XMark.

4.2. Experimentos

Cada experimento explora aspectos de desempenho em transações, tais como tempo de resposta e número de *deadlocks*. A comparação foi feita entre o DTX utilizando o protocolo Node2PL [Haustein et al. 2006] e o DTX, ambos acoplados ao SGBD XML Nativo Sedna. O objetivo dos experimentos é verificar o desempenho conseguido com o DTX adaptado ao protocolo XDGL. Pela dificuldade de ter acesso às implementações dos trabalhos relacionados, optou-se por adaptar o DTX e utilizar um protocolo de bloqueio em árvores (Node2PL), uma vez que a maioria dos trabalhos relacionados utiliza protocolos com essa característica, servindo-se da mesma lógica de sincronização entre sítios, persistência, recuperação e linguagem. Portanto, as únicas modificações feitas no DTX foram: a estrutura de representação de bloqueios/documento e as regras de aplicação/liberação de bloqueios por operação. Durante essas modificações, o DTX mostrou-se ser bastante flexível à mudança para novos protocolos.

Variação no número de clientes
Este experimento verifica o comportamento do DTX quanto à variação do número de clientes, com replicação total e parcial. Neste caso, varia-se o número de clientes de 10 a 50, cada cliente contendo 5 transações de leitura com 5 operações cada. Para a realização dos experimentos de replicação parcial, a base de dados foi fragmentada de acordo com a abordagem proposta por [Kurita et al. 2007]. Nesta abordagem, os dados são fragmentados considerando a estrutura e o tamanho do documento, de forma que cada fragmento gerado tenha tamanhos similares.



(a) Replicação total (b) Replicação parcial

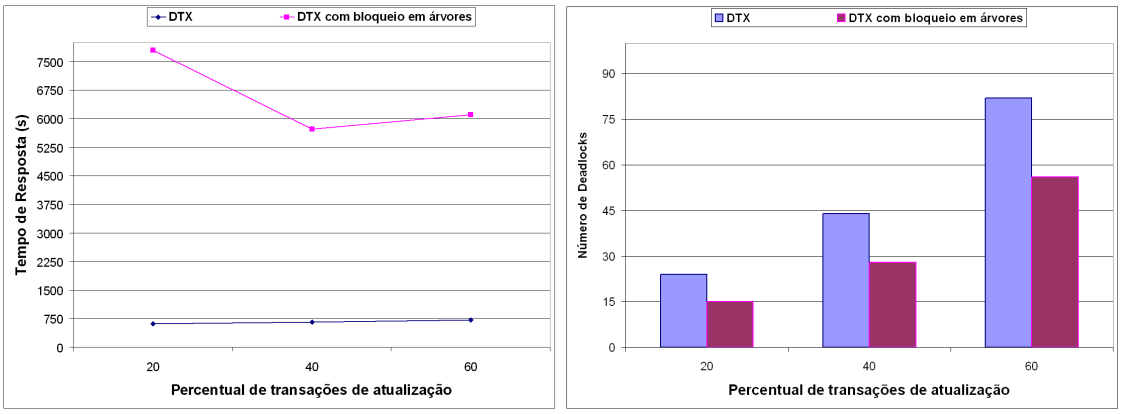
Figura 7. Variação no número de clientes

A Figura 7 mostra o tempo de resposta resultante deste experimento. Em ambas as abordagens de replicação, o tempo de resposta do DTX apresenta um melhor resultado do que o DTX com bloqueio em árvores. A razão disso é que a adaptação do protocolo XDGL, utilizado no DTX, possui uma granularidade bem menor ao do protocolo de bloqueio em árvores; neste caso, o *overhead* de gerência destes bloqueios é bem inferior. O tempo de resposta da replicação parcial é inferior ao da replicação total;

o motivo deve-se ao fato de que na replicação total há um *overhead* de comunicação e sincronização em todos os sítios do sistema, o que atrasa a execução das transações. Com base nestes resultados, optou-se por realizar os demais experimentos utilizando-se a abordagem de replicação parcial, já que em ambientes reais os dados XML geralmente estão distribuídos por vários sítios.

Variação no percentual de atualização

Este experimento demonstra o comportamento do DTX contendo uma variação no percentual de transações de atualização. Neste experimento, fixou-se o número de clientes em 50, onde cada cliente submete 5 transações com 5 operações. A variação do percentual de transações de atualização ocorre entre 20% a 60%. O percentual de operações de atualização por transação de atualização é de 20%.



(a) Tempo de resposta (b) Número de *deadlocks*

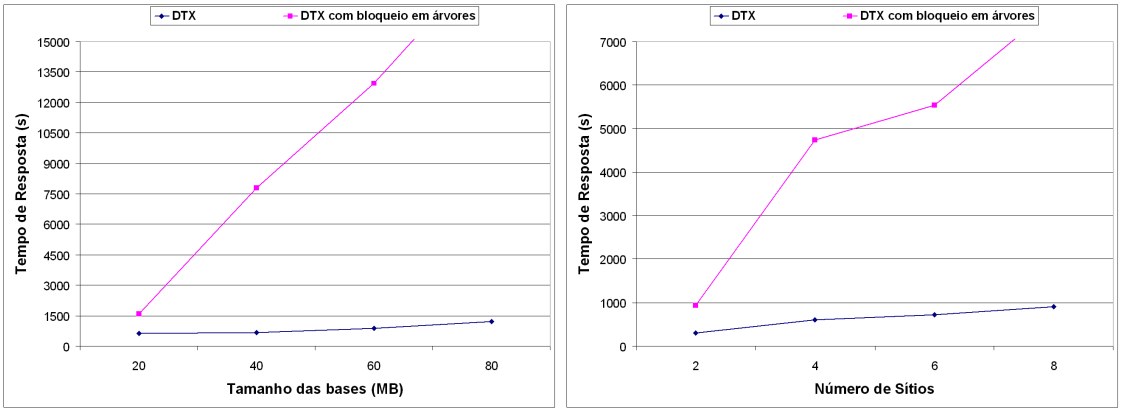
Figura 8. Variação no percentual de atualização

Na Figura 8 (a), o tempo de resposta do DTX se mantém bem inferior com o crescimento da proporção de atualização, enquanto o DTX com bloqueio em árvores possui um tempo de resposta superior. A justificativa, para que o DTX possua um tempo de resposta melhor, está relacionada ao menor *overhead* de gerência de bloqueios, e por usar uma estrutura sumarizada dos dados, o que agiliza a recuperação e modificação destes, além de manter uma estrutura de tamanho bem melhor do que o documento XML original. Analisando o gráfico (b) desta figura, o número de *deadlocks* obtido pelo DTX foi bem superior ao do DTX com bloqueio em árvores. Além disso, com o aumento do crescimento da proporção de atualização esses números tendem a crescer. O motivo para isso está ligado ao fato que o DTX consegue um maior paralelismo entre transações por utilizar uma granularidade de bloqueio muito menor do que a do outro protocolo. Isso faz com que um maior número de transações entre em execução concorrentemente no DTX e, eventualmente, por requerem dados em comum, entre em *deadlock* em maior quantidade do que o outro protocolo que é mais restrito e menos concorrente.

Variação no tamanho da base e número de sítios

Este último experimento verifica o comportamento do DTX quanto à variação do tamanho da base e ao número de sítios, servindo-se da replicação parcial. Para o experimento com variação no tamanho da base fixou-se o número de clientes por sítio em 50, cada cliente

submetendo 5 transações contendo 5 operações. O percentual de transação de atualização é 20% e o percentual de operações de atualização por transação de atualização é 20%. A variação do tamanho da base foi feita entre 20 a 80MB. Já para o experimento com variação no número de sítios, a base de 40MB foi fragmentada, alocada e carregada no Sedna dos sítios do experimento. Para o experimento de variação do número de sítios, foram utilizados os mesmos parâmetros para clientes, transações e seus percentuais de atualização. A variação entre o número de sítios foi feita entre 2 a 8 sítios.



(a) Variação no tamanho da base (b) Variação no número de sítios

Figura 9. Variação no tamanho da base e número de sítios

Na Figura 9 (a), o tempo de resposta do DTX mostra-se bem inferior diante do crescimento do tamanho das bases, enquanto o DTX com bloqueio em árvores ilustra um aumento do tempo de resposta. Uma razão para isso está relacionada ao fato de que o DTX utiliza uma estrutura compacta para representação dos documentos XML, o que leva a esses resultados. Uma outra razão está ligada ao fato de que, no DTX com bloqueio em árvores, a gerência de bloqueios é bem maior, uma vez que a aplicação destes bloqueios é em árvores e sub-árvores do documento em questão. Portanto, se o documento cresce o número de bloqueios também aumenta. A Figura 9 (b) ilustra que o tempo de resposta do DTX se mostra inferior com o aumento no número de sítios, e o DTX com bloqueio em árvores apresenta um resultado bem superior. Isso está relacionado ao fato de que com o crescimento do número de sítios, além do maior número de mensagens de sincronização, há um maior *overhead* de gerência de bloqueios nos sítios locais e remotos.

No crescimento do tamanho da base de dados, o DTX mostrou-se mais deficiente quanto ao número de *deadlocks*. O motivo deve-se ao fato de que, quando a base cresce, o DTX com bloqueio em árvores é mais lento por possuir um maior *overhead* de bloqueios. Este maior tempo de resposta diminui o grau de concorrência deste protocolo o que ocasiona um menor número de conflitos, ou seja, *deadlocks*. Com o aumento do número de sítios, é notado que o DTX apresenta resultados inferiores ao do DTX com bloqueios em árvores.

5. Trabalhos Relacionados

O Tamino é um SGBD XML Nativo que se serve do protocolo 2PL em seu controle de concorrência. Seu protocolo possui quatro níveis de granularidade: de banco de dados,

coleção de documentos XML, *doctype* e documento XML. O Tamino suporta transações distribuídas e utiliza o protocolo 2PC para garantir a consistência e a atomicidade das transações. O Berkeley DB XML [Oracle 2008] é um SGBD implementado como uma camada sobre Berkeley DB. O protocolo para controle de concorrência padrão utilizado é o 2PL. A granularidade do bloqueio pode ser a nível de documento ou de banco de dados. Este SGBD dá suporte à execução de transações distribuídas utilizando o protocolo 2PC para a garantia da atomicidade e da consistência.

Em [Pagnamenta 2005] é descrito um projeto e implementação inicial de SGBD XML distribuído. Este trabalho propõe uma camada arquitetural com uma infra-estrutura de acesso a dados que integram diferentes tipos de SGBDs que suportam XML. A atomicidade global é garantida pelo protocolo 2PC para coordenar as transações distribuídas. No trabalho, é enfatizado que a atomicidade é garantida quando os SGBD possuem um baixo grau de acesso concorrente. Também é tratada a questão de *deadlocks* e apresentada uma avaliação simples de desempenho.

Os trabalhos relacionados apresentam, em geral, um baixo nível de concorrência entre transações, por realizarem o bloqueio completo do documento, afetando o desempenho. Dentre eles, o Berkeley DB XML possui a menor granularidade, mas o usuário necessita decompor os documentos em fragmentos. Contudo, o Berkeley DB deixa a cargo da aplicação detectar e resolver *deadlock* distribuído. [Pagnamenta 2005] deixa o gerenciamento do controle de concorrência a cada SGBD que compõe o sistema e, neste caso, o trabalho garante o critério de *serializabilidade* apenas quando a granularidade do bloqueio é o documento completo. Além disso, as soluções apresentadas são implementadas utilizando informações específicas do SGBDs, dificultando a portabilidade destas soluções.

6. Conclusões

Com um volume muito grande de documentos XML espalhados pela Web, surge a necessidade de gerenciamento eficiente destes documentos. Neste artigo, foi apresentado o DTX como mecanismo de controle de concorrência distribuído para dados XML. O DTX apresenta uma melhora do tempo de resposta no processamento de transações distribuídas ao utilizar um protocolo que leva em consideração características de XML, e possui uma granularidade baixa. Avaliou-se o DTX considerando seu desempenho diante do tempo de resposta e número de *deadlocks*. Pela análise dos resultados obtidos, foi possível verificar que o DTX melhorou o tempo de resposta na execução de transações distribuídas em diversas situações. Foi possível verificar também que o DTX é flexível, permitindo a adaptação de outros protocolos de controle de concorrência, como o utilizado na avaliação.

Como trabalhos futuros, os autores pretendem desenvolver soluções para o DTX contemplar as propriedades de atomicidade e durabilidade, e assim oferecer um suporte transacional completo aos usuários do DTX. Com relação à avaliação do desempenho, também é proposto a avaliação do DTX em ambientes WAN. Durante a avaliação do DTX, foram observadas consideráveis quantidades de *deadlocks*. Portanto, é necessário um estudo aprofundado desses resultados, bem como da estrutura de funcionamento dos algoritmos visando a identificar os fatores que possam ter ocasionado esse problema. Para uma melhor robustez, confiabilidade e segurança ao DTX, pretende-se aplicar estratégias

de recuperação após falhas e também mecanismos para que o processamento no DTX não seja realizado todo em memória principal. Por fim, pretende-se adicionar outros protocolos de controle de concorrência e verificar o desempenho deles adaptados ao DTX.

Agradecimentos
Os autores agradecem ao ISPRAS/Rússia pela colaboração neste trabalho.

Referências

Fomichev, A., Grinev, M., and Kuznetsov, S. (2006). Sedna: A native xml dbms. In *SOFSEM 2006*, volume 3831 of *Lecture Notes in Computer Science*, pages 272–281.

Grabs, T., Böhm, K., and Schek, H.-J. (2002). Xmltm: efficient transaction management for xml documents. In *CIKM '02: Proceedings of the eleventh international conference on Information and knowledge management*, pages 142–152, McLean, Virginia, USA. ACM Press.

Haustein, M., Härder, T., and Luttenberger, K. (2006). Contest of xml lock protocols. In *VLDB '06*, pages 1069–1080. VLDB Endowment.

Kurita, H., Hatano, K., Miyazaki, J., and Uemura, S. (2007). Efficient query processing for large xml data in distributed environments. In *AINA '07: Proceedings of the 21st International Conference on Advanced Networking and Applications*, pages 317–322.

Moreira, L. O. (2008). Dtx: Um mecanismo de controle de concorrência distribuído para dados xml. Master's thesis, Universidade Federal do Ceará, Brasil.

Oracle (2008). *Berkeley Database XML*. `http://www.oracle.com/database/berkeley-db/xml/index.html`.

Özsu, M. T. and Valduriez, P. (1999). *Principles of distributed database systems*. Prentice-Hall, Inc.

Pagnamenta, F. (2005). Design and initial implementation of a distributed xml database. Master's thesis, University of Dublin, Irlanda.

Pleshachkov, P., Chardin, P., and Kuznetsov, S. (2005). A dataguide-based concurrency control protocol for cooperation on xml data. In *ADBIS*, volume 3631 of *Lecture Notes in Computer Science*, pages 268–282.

Schmidt, A., Waas, F., Kersten, M. L., Carey, M. J., Manolescu, I., and Busse, R. (2002). Xmark: A benchmark for xml data management. In *VLDB '02*, pages 974–985.

Seltzer, M. I. (2007). Berkeley db: A retrospective. *IEEE Data Eng. Bull.*, 30(3):21–28.

Sousa, F. R. C., Filho, H. J. A. C., Andrade, R. M. C., and Machado, J. C. (2007). Replix: Um mecanismo para a replicação de dados xml. In *SBBD*, pages 53–67.

Tamino (2007). *Tamino XML Server*. `http://www.softwareag.com/tamino`.

Türker, C., Haller, K., Schuler, C., and Schek, H.-J. (2005). How can we support grid transactions? towards peer-to-peer transaction processing. In *Conference on Innovative Data Systems Research*, pages 174–185.

W3C (2007). *Extensible Markup Language*. `http://www.w3.org/XML/`.