TÉCNICA DE CONTROLE DE CONCORRÊNCIA SEMÂNTICO PARA BANCO DE DADOS EM TEMPO-REAL

Yáskara Ygara Menescal Pinto FERNANDES (1); Talles da Silva LOPES (2)
(1) Centro Federal de Educação Tecnológica do Rio Grande do Norte
Unidade de Ensino Descentralizada de Mossoró, Rua: Raimundo Firmino de Oliveira, nº 400 - Costa e Silva Mossoró-RN - CEP: 59.628-330, fone: (0xx84) 3315.2752/2760, Fax: (0xx84) 3315.2761 ,email:yaskaramenescal@gmail.com
(2) UERN, e-mail: tallessl@gmail.com

RESUMO

O grande avanço tecnológico na área de redes de sensores tem motivado o uso de tais redes em uma variedade de aplicações, entre elas podemos destacar: sistemas de monitoramento, detecção de enchentes e rastreamento de produtos estocados em um armazém. A principal característica dessas aplicações é o grande volume de dados com restrições temporais que precisa ser processado. Conseqüentemente, as transações que os utilizam também devem ter restrições temporais de forma que consigam usar tais dados enquanto os mesmos ainda sejam válidos. Tais características impõem a necessidade de tais aplicações utilizarem um Sistema de Gerenciamento de Banco de Dados em Tempo-Real, que além de serem projetados para tratar com grandes volumes de dados, também garantem a validade lógica e temporal dos dados e transações. Apresenta-se uma técnica de controle de concorrência para Sistemas de Gerenciamento de Banco de Dados em Tempo-Real, denominado Técnica de Controle de Concorrência Semântico. A mesma permite a negociação entre a consistência lógica e temporal quando houver conflito entre ambas, é baseada em uma Função de Compatibilidade, definida pelo usuário, para os métodos de leitura e escrita de um objeto da aplicação.

Palavras-chave: Banco de dados em tempo-real, controle de concorrência, função de compatibilidade, redes de sensores.

1. INTRODUÇÃO

O avanço tecnológico das últimas décadas colocou de forma definitiva os computadores no cotidiano da vida moderna, seja num sistema bancário, de transporte, de saúde, de comunicação, de manufatura, entre tantos outros. Se de um lado tal inserção traz inúmeros benefícios às nossas vidas, de outro coloca novos desafios tecnológicos para a construção de sistemas computacionais cada vez mais confiáveis e seguros, sem os quais prejuízos e desastres tornar-se-ão inevitáveis.

Os Sistemas em Tempo-Real (STR) é um caso especial de sistemas computacionais em que o computador interage diretamente com o ambiente, não somente por meio de uma interface homem-máquina (vídeo e teclado), mas, sobretudo através de dispositivos que captam informações e que interferem no ambiente, denominados de sensores. Exemplos desses sistemas vão desde simples fornos de microondas até sistemas de controle automático de vôo. Esses sistemas executam as ações desejadas dentro de intervalos de tempos determinados pelo ambiente. Por exemplo, o controle de pouso de um avião tem que acionar o trem de pouso tão logo seja detectado a proximidade do solo através de um sensor de altitude, caso contrário, pode resultar em conseqüências catastróficas.

Os Sistemas de Gerenciamento de Bancos de Dados Convencionais (SGBD) são projetados para gerenciar grandes volumes de dados e controlar a execução concorrente das transações, no entanto não suportam o tratamento das restrições de tempo inerentes a diversas aplicações tais como: redes de sensores, sistemas de monitoramento de pacientes, sistemas de controle de trafego aéreo e ferroviário, bolsa de valores *on-line*, dentre outros. Desta forma, pode ser inviável utilizar a tecnologia de um SGBD convencional em tais aplicações.

Por outro lado, um Sistemas de Gerenciamento de Banco de Dados em Tempo-Real (SGBD-TR) possui como principal característica o tratamento de dados e transações com restrições explícitas de tempo. Tal característica impõe aos protocolos de controle de concorrência desses sistemas à necessidade de considerar tanto a manutenção da consistência lógica quanto a consistência temporal dos dados e transações. Entretanto, em algumas situações a manutenção simultânea de tais restrições é impraticável.

Por exemplo, para manter a consistência temporal dos dados, uma transação que atualiza um item de dados pode interromper outra transação que esta lendo o mesmo item de dados. Se o item de dados em questão perdeu sua validade temporal, é interessante permitir que a sua consistência temporal seja restabelecida através da execução da transação de atualização. No entanto, esta execução poderia violar a consistência lógica da transação de leitura.

Então, existe a necessidade de se negociar entre a manutenção da consistência temporal ou consistência lógica. Se a consistência lógica for escolhida, existe a possibilidade que o item de dado torne-se temporalmente inválido, ou que uma transação viole sua restrição temporal. Se por outro lado, consistência temporal for escolhida, a consistência lógica do dado ou da transação pode ser comprometida.

Portanto, é fundamental disponibilizar uma técnica de controle de concorrência para SGBD-TR que permita a negociação entre a consistência lógica e consistência temporal dos dados e transações.

Em Dipippo (1995), foi definida uma técnica baseada em informação semântica para SGBD-TR, denominada Técnica de Controle de Concorrência Semântico que permite a negociação entre a consistência lógica e temporal dos dados e transações, e limita a imprecisão resultante dessa negociação através de uma função denominada Função de Compatibilidade (FC).

A técnica de controle de concorrência semântico foi estendida visando implementá-la. Após a implementação da técnica uma aplicação focada em redes de sensores também foi implementada, e as restrições lógicas e temporais dos dados e das transações resultantes da execução concorrente das transações da aplicação foram analisadas.

2. FUNDAMENTAÇÃO TEÓRICA

Nos últimos anos, várias pesquisas têm sido voltadas para técnicas ou protocolos de controle de concorrência para SGBD-TR (LAM et al., 2002; LINDSTRÖM, 2003; LINDSTRÖM, 2006). Muitos destas técnicas são baseadas em técnicas de controle de concorrência tradicionais. A maioria destas técnicas baseia-se no mecanismo de *serialização* como critério de corretude para determinar quais transações pode executar concorrentemente.

Dos protocolos de concorrência bastante utilizados, temos:

- Protocolo de Bloqueio de Duas Fases Pessimista (2TPL) (NYSTRÖM et al., 2004): As transações são bloqueadas antes de executarem suas operações ou esperam pelo bloqueio se este não puder ser adquirido;
- Protocolo de Controle de Concorrência Otimista (CCO) (LINDSTROM, 2002, 2003; RAATIKAINEN; LINDSTROM, 2002): A resolução de conflitos é atrasada até a transação está próximo de comprometer;

No Protocolo de Bloqueio de Duas Fases Pessimista as transações adquirem os bloqueios antes de executarem suas operações no banco de dados ou esperam pelo bloqueio se este não puder ser adquirido. No entanto, esse protocolo pode gerar tempos indeterminados de espera pelo bloqueio. Dessa forma, a utilização desse mecanismo pode ser vantajosa por manter a consistência lógica do banco de dados sem impasses, em contrapartida pode comprometer as restrições de tempo impostas às transações.

Outro protocolo utilizado é o Protocolo de Controle de Concorrência Otimista, onde a resolução de conflito é atrasada até a transação está próxima de comprometer. Nesse protocolo não existe a indeterminação do tempo de espera, porém a quantidade de transações que podem ser reiniciadas é grande, o que pode ser fatal em sistemas com restrições temporais.

Em um SGBD-TR, os aspectos temporais dos dados e as restrições de tempo das transações também devem ser considerados. Portanto, um protocolo de controle de concorrência para esses sistemas deve buscar manter a *consistência temporal dos dados e transações* simultaneamente, ou seja, precisa dar suporte à consistência lógica e temporal dos dados e transações, e quando a manutenção simultânea de ambas não for possível, precisa permitir a negociação entre elas, além de controlar a imprecisão resultante dessa negociação.

2.1. Sistemas de Gerenciamento de Banco de Dados em Tempo-Real (SGBD-TR)

Nos últimos anos, várias pesquisas têm sido realizadas na área de SGBD-TR , uma vez que esses sistemas tratam com aplicações que envolvem o gerenciamento de grandes quantidades de dados, além de permitir tratar com dados e transações com restrições em tempo-real. Os SGBD-TR surgiram com uma publicação especial, registrado em *ACM SIGMOD* em março 1988. Atualmente, muitas aplicações requerem o uso de tais sistemas, dentre elas: os sistemas de recuperação da informação, os sistemas do reserva de passagem aérea, sistemas bancários, sistema de controle de avião e de nave espacial, robótica, a automação de fábrica, dentre outros.

A importância funcional dos SGBD-TR se dá pelo fato destes suportarem as características de SGBD, de processar transações e garantir a consistência lógica dos dados e transações, como também as características de um STR, de satisfazer às restrições de tempo impostas as tarefas. Assim, como os SGBD tradicionais, os SGBD-TR servem como repositórios para dados e permitem o processamento eficiente dos mesmos.

2.2. Técnicas de Controle de Concorrência para SGBD-TR

Como já mencionado anteriormente, a técnicas de controle de concorrência que utilizam serialização, são ineficientes para SGBD-TR. Transações em tempo-real não necessitam ser serializáveis, entretanto, seu relaxamento poderia gerar uma inconsistência no banco de dados.

A técnica de controle de concorrência semântica, orientada a objetos, apresentada em Dipippo (1995), suporta a consistência lógica e temporal dos dados e transações, por permitir a especificação de técnicas de escalonamentos mais maleáveis que aquelas permitidas pela serialização. É definida uma função de compatibilidade (FC) para cada par de métodos de um objeto com o objetivo de controlar o acesso concorrente ao banco de dados.

Alem da FC, a técnica utilizada, baseia-se também no critério de corretude (*Serialização Epsilon - ES*), ele generaliza a serialização permitindo imprecisão limitada no processo de transação, assumindo usar apenas dados mensuráveis.

O Algoritmo da técnica de controle de concorrência semântica é bastante genérico, e o nosso trabalho propõe a definição de um algoritmo mais específico, para ser utilizados em redes de sensores. O algoritmo definido, consiste em que:

Para cada par de métodos é considerado as transações T1 e T2 com as operações: (rT1(x), wT2(x)), (wT1(x), rT2(x)) e (wT1(x), wT2(x)) onde r representa leitura e w representa escrita. A FC será avaliada de acordo com as situações descritas abaixo:

- Quando as operações pertencem a transações diferentes;
- Quando as operações das transações requisitam a mesma tabela, a mesma tupla e acessam o mesmo atributo.

Os três casos que requerem a avaliação da FC são mostrados a seguir:

Quando ocorre (rT1(x), wT2(x)), a FC será avaliada como apresentado anteriormente, e ilustrada pelo algoritmo abaixo:

1: FC verifica se as operações são da mesma transação
2: Se forem da mesma transação as operações são executadas normalmente
3: Se forem de diferentes transações
4: Verifica se as operações acessam a mesma tabela, tupla e atributo
5: se verdadeiro então
6: Execute FC (leitura, escrita)
7: Avalia parâmetros temporais e lógicos
8: Concedido em verdadeiro
9: Incrementa Imprecisão
10: Concedido em falso
11: Aborta wT2(X)
12: senão
13: A FC não é invocada
14: fim se

O segundo caso possível é se a operação (wT1(x), rT2(x)) for invocada, sua execução concorrente só é possível se elas forem compatíveis. O algoritmo para este caso e mostrado a seguir.

1:FC verifica se as operações são da mesma transação
2:Se forem da mesma transação as operações são executadas normalmente
3: Se forem de diferentes transações
4:Verifica se as operações acessam a mesma tabela, tupla e atributo
5:se verdadeiro então
6: Execute FC (escrita, leitura)
7:Avalia parâmetros temporais e lógicos
8:Concedido em verdadeiro
9: Incrementa Imprecisão
10:Concedido em falso
11:Aborta rT1(X)
12: senão
12:A FC não é invocada
13:fim se

O último caso é (wT1(x), wT2(x)), e definido no algoritmo abaixo:

1:FC verifica se as operações são da mesma transação
2:Se forem da mesma transação as operações são executadas normalmente
3: S forem de diferentes transações
4:Verifica se as operações acessam a mesma tabela, tupla e atributo
5:se verdadeiro então
6: Execute FC (escrita, escrita)
7:Avalia os parâmetros temporais e lógicos
8:Concedido em verdadeiro
9: Incrementa Imprecisão

10:Concedido em falso
11:Aborta wT1(X)
12: senão
13:A FC não é invocada
14:fim se

3. IMPLEMENTAÇÃO DA TÉCNICA DE CONTROLE DE CONCORRÊNCIA

A técnica de controle de concorrência implementada, utiliza uma FC que permite a negociação entre a consistência lógica e temporal de acordo com as necessidades da aplicação, e quando do sacrifício de uma das consistências, esta FC controla a imprecisão gerada no Banco de Dados.

Inicialmente é considerado um par de operações de transações, sendo possíveis quatro tipos: leitura-leitura; leitura-escrita; escrita-leitura; escrita-escrita.

É necessária a observação da existência de conflito entre as operações, obedecendo aos critérios da FC. Quando o sistema recebe uma única transação, obviamente, a análise é desnecessária, sendo a mesma executada sem restrições, como também, sendo mais de uma transação, mas não atendendo pelo menos um dos requisitos apresentados na FC, as transações serão executadas normalmente.

Quando todos os requisitos são atendidos, a FC faz a análise semântica para identificar se poder ocorrer a execução concorrente ou não (tal análise é baseada no par de operações leitura e escrita). Sendo determinado o tipo do par, é invocado o método apropriado.

O método leitura-leitura executa as operações na ordem em que estas chegam ao escalonador, havendo o acesso concorrente ao Banco de dados.

No método leitura-escrita é aplicada a expressão booleana *((TempoCorrente - TimestampReal) <= absoluteValidationInterval)* para identificar se os parâmetros temporais dos dados da operação de escrita são temporalmente válidos. Caso esta expressão seja avaliada como FALSO, a escrita é abortada e a leitura continua sua execução, porem, caso a avaliação seja VERDADEIRO, os parâmetro lógicos dos dados são verificados para identificar se estes são logicamente válidos. Se forem FALSO, uma imprecisão poderá ser introduzida no Banco de dados sendo calculada pela expressão *Imprecisão = ValorOperaçãoEscrita – ValorBancodeDados*. Ainda deve ser verificado se esta imprecisão é menos ou igual a imprecisão permitida, caso esta verificação seja verdadeira, é calculado a imprecisão acumulada usando a expressão (ImprecisãoAcumulada + Imprecisão) <= ImprecisãoPermitida. Caso seja falso, a leitura é executada e a escrita é abortada, e se for verdadeiro, a execução continua e a Imprecisão é somada a ImprecisãoAcumulada. Finalmente, após todas as verificações, as operações podem ser executadas concorrentemente.

O terceiro método possível é o de escrita-leitura, onde inicialmente são avaliados os parâmetros temporais da operação de leitura, buscando identificar a validade temporal dos dados, usando a seguinte operação: *((TempoCorrente - TimestampReal) <= AbsoluteValidationInterval)*. Caso esta seja FALSA, a escrita continua sua execução com a operação de leitura abortada. Por outro lado, a expressão sendo avaliada em VERDADEIRO, é observada a consistência lógica dos dados, caso sejam avaliados em FALSO, a Imprecisão gerada no sistema é calculada subtraindo o valor da operação de leitura pelo valor do bando de dados, verifica-se também se o resultado desta diferença é menor ou igual a imprecisão permitida.

Se esta verificação for VERDADEIRA, calcula-se a imprecisão acumulada visando determinar se a imprecisão permitida não é ultrapassada, somando-se a Imprecisão acumulada com a Imprecisão. Sendo esta verificação avaliada como FALSO, a escrita continua e a leitura é abortada, porem se esta for avaliada em VERDADEIRO, a operação continua e a Imprecisão é somada a Imprecisão acumulada.

Finalmente, concluindo todas as verificações, e estas sendo avaliadas em verdadeiro, as duas operações podem ser executadas concorrentemente.

O ultimo método possível é o de escrita1-escrita2. Neste caso, avaliam-se, os parâmetros temporais de escrita2, visando identificar se estes são validos temporalmente. É utilizada a seguinte expressão booleana: *((TempoCorrente - TimestampReal) <= AbsoluteValidationInterval)*. Essa sendo avaliada em FALSO, a operação escrita1 é executada e escrita2 é abortada, porem, se esta é VERDADEIRA, é feita uma verificação nos parâmetros lógicos, para identificar se estes são logicamente validos, e definir sua Imprecisão através da diferença entre o valor das duas operações de escrita, e verificando se a mesma é

menor ou igual a Imprecisão permitida. Se este resultado for avaliado em VERDADEIRO, é calculada a Imprecisão acumulada para averiguar se esta não ultrapassa a Imprecisão permitida, através da expressão: *ImprecisãoAcumulada + Imprecisão <= ImprecisãoPermitida*. Se esta for FALSA, continua a operação de escrita1 e escrita2 é abortada. Caso seja VERDADEIRO, a Imprecisão é somada a ImprecisãoAcumulada e a execução continua. Finalmente, após todas as verificações terem sido avaliadas como VERDADEIRAS, as duas operações podem ser executadas concorrentemente.

A técnica implementada foi divida em três módulos: A Interface, que é um interface gráfica que permite ao projetista parametrizar os dados que serão utilizados na aplicação; A Simulação, sendo este modulo responsável pela simulação do funcionamento dos sensores; E o ultimo módulos é o Banco de Dados Servidor, sendo o responsável pelo acesso ao banco de dados.

3.1. Estudo de caso

Nos últimos anos, muitas pesquisas têm sido realizadas na área de redes de sensores. Dentre as diversas aplicações para tais redes destacam-se: rastreamento dos produtos estocados em um armazém, monitoramento residencial, detecção de enchente, e monitoramanto de tráfego de veículos \cite{yong1,yong2}.

Para validar o método implementado considera-se uma rede de sensores que é utilizada para monitorar a distância percorrida por um determinado veículo em um sistema de controle de transportes urbanos. Portanto a aplicação precisa ler dos sensores os atributos de tempo e velocidade do veículo para calcular a distância percorrida pelo mesmo. Observe que os atributos de tempo e velocidade devem ser relativamente consistentes de forma que o espaço percorrido seja calculado corretamente. Além disso, também é importante considerar uma imprecisão em relação à consistência absoluta entre o valor calculado e armazenado no banco de dados e a nova posição do veículo. No entanto esta imprecisão deve refletir aproximadamente a posição correta do mesmo.

Os dados em uma rede de sensores possuem um tempo de vida útil limitado. No caso da aplicação em questão, tal restrição temporal é considerada, tanto no caso da recuperação quanto no processamento dos mesmos.

No entanto é importante ressaltar que algumas aplicações precisam garantir tanto a consistência lógica quanto a consistência temporal do banco de dados. Observe que o método implementado, através da FC permite negociar entre a consistência lógica e temporal a fim de atender ao propósito de diversos tipos de aplicações, além de considerar e controlar a imprecisão gerada no banco.

Dentre as possíveis transações que devem ser executadas concorrentemente pela aplicação, temos:

- Transações de Atualização (Sensor)- que adquire dados (tempo e velocidade) dos sensores e escreve no banco de dados;
- Transações de Consulta (Aplicação)- que lêem dados (tempo e velocidade) do banco de dados.

Considerando uma situação em que o veículo ultrapassou a velocidade permitida, então o sensor é disparado e fotografa a placa do veículo que ultrapassou a velocidade permitida. O banco de dados servidor é atualizado com o dado adquirido do sensor através da transação de atualização. Nesse instante de tempo a aplicação requisita uma consulta ao banco de dados servidor para verificar as placas dos veículos que foram multados. Esta é uma situação clássica de conflito de transações, onde duas operações, uma de leitura e uma de escrita, pertencentes a transações diferentes tentam acessar o mesmo item de dado, concorrentemente no banco de dados. Essas duas operações só podem executar concorrentemente se a FC for avaliada em VERDADEIRO.

4. CONCLUSÃO

O objetivo principal deste trabalho foi implementar uma técnica de controle de concorrência para SGBD-TR, denominada Técnica de Controle de Concorrência Semântico. A técnica é baseada em uma extensão da serialização clássica, denominada *Serialização Epsilon* , e em uma função de compatibilidade. Ela permite o gerenciamento da execução concorrente de transações com restrições temporais, pois tais transações geralmente manipulam dados com tempo de vida útil limitado, portanto precisam ser utilizados enquanto são válidos. Como, freqüentemente, não é possível garantir a consistência lógica e temporal simultaneamente no banco de dados, ela permite a negociação entre ambas quando necessário através da introdução de uma imprecisão no banco de dados. No entanto, a imprecisão resultante desta negociação é controlada.

A técnica foi implementada através da linguagem de programação Java e o gerenciador utilizado foi o Microsoft SQL Server. É importante destacar que apesar do SGBD utilizado ter sido o Microsoft SQL Server a técnica pode ser utilizada com qualquer SGBD, desde que, a técnica de controle de concorrência do mesmo seja desativada, assim como foi feito neste trabalho.

Visando validar a técnica implementada foi desenvolvida uma aplicação para redes de sensores com seu respectivo banco de dados. Através dela verificou-se que a técnica implementada foi executada com sucesso. Algumas das situações conflitantes foram consideradas e os resultados produzidos atenderam os objetivos.

REFERÊNCIAS

DIPIPPO, L. **Semantic Real-Time Object-based Concurrency Control**. Tese (Doutorado) - Department of Computer Science and Statistics, University of Rhode Island, 1995.

LINDSTRÖM, J. **Relaxed Correctness for Firm Real-Time Databases**. IEEE International Conference on Embedded and Real-Time Computing Systems and Applications, 2006.

NYSTRöM, D. et al**. Pessimistic concurrency control and versioning to support database pointers in real-time databases**. In: Proceedings of Euromicro conference on Real-Time Systems (ECRTS). [S.l.: s.n.], 2004.

LINDSTROM, J. **Integrated and adaptive optimistic concurrency control method for real-time databases**. VIII International Conference on Real-Time Computing Systems and Application, University of Helsinki Finland, 2002. VIII International Conference on Real-Time Computing Systems and Application.

LINDSTROM, J. **Optimistic Concurrency Control Methods for Real-Time Database.** Tese (Doutorado) — Department of Computer Science, University of Helsinki Finland, January 2003.

RAATIKAINEN, K.; LINDSTROM, J. **Using real-time serializability and optimistic concurrency control in firm real-time databases**. University of Helsinki Finland, 2002.

LAM, K.-Y. et al. **Evaluation of concurrency control strategies for mixed soft real-time database systems**. *Inf. Syst.*, Elsevier Science Ltd., v. 27, n. 2, p. 123–149, 2002. ISSN 0306-4379.