

EIT - Escalonador Inteligente de Transações

Maristela Holanda^{1*}, Angelo Brayner², Sergio Fialho³

¹Departamento de Ciência da Computação
Universidade Católica de Brasília (UCB) – Brasília, DF – Brazil

²Mestrado em Informática Aplicada
Universidade de Fortaleza (UNIFOR) – Fortaleza, CE – Brazil

³Departamento de Engenharia Elétrica
Universidade Federal do Rio Grande do Norte (UFRN) – Natal, RN – Brazil
mholanda@ucb.br, brayner@unifor.br, fialho@pop-rn.rnp.br

Abstract. *Serializability has been used as a criterion to correctly synchronize the execution of concurrent operations belonging to several transactions. For that reason, concurrency control protocols should yield schedules using serializability. Concurrency control protocols may present an aggressive or conservative behavior. This paper presents the Transaction Intelligent Scheduler (EIT), which has the ability to synchronize database operations by using concurrency protocols with an aggressive or conservative behavior. EIT adapts its behavior dynamically according to characteristics of the computing environment where it is inserted. In order to make such changes, an expert system based on fuzzy logic is one of the components of this scheduler.*

Key words *transaction, concurrency control, expert system and fuzzy logic.*

Resumo. *Serializabilidade tem sido usada como um critério para sincronizar corretamente a execução das operações pertencentes às diversas transações. Por isso, os protocolos de controle de concorrência aplicam esse critério para gerar escalonamentos. Os protocolos de controle de concorrência podem ter comportamento agressivo ou conservador. Este artigo apresenta o Escalonador Inteligente de Transações (EIT), o qual tem a habilidade de sincronizar essas operações através de protocolos com comportamento agressivo ou conservador. O EIT adapta dinamicamente seu comportamento de acordo com as características do ambiente computacional onde está inserido, sendo composto por um sistema especialista baseado em lógica fuzzy.*

Palavras-chaves: *transação, controle de concorrência, sistema especialista e lógica fuzzy.*

* Autora financiada pela Fundação Nacional de Desenvolvimento do Ensino Superior Particular - Funadesp

1. Introdução

Os sistemas de banco de dados estão presentes em diversas aplicações que são utilizadas por diferentes usuários. Usuários interagem com os sistemas de banco de dados através de transações. Uma transação representa uma abstração de uma seqüência de operações no banco de dados resultante da execução de programas de aplicações [Brayner *et al* 1999].

Um sistema de banco de dados deve controlar a execução concorrente das transações para garantir que o estado do banco de dados permaneça consistente. Um banco de dados está em seu estado consistente, quando ele obedece todas as restrições de consistência (integridade) definidas sobre ele e se todos os itens de dados satisfazem às restrições de consistência específicas da aplicação [Bernstein *et al* 1987; Elmasri e Navathe 2000; Molina *et al* 2001; Silberschatz *et al* 1999].

O conceito de serializabilidade [Eswaran *et al* 1976] vem sendo utilizado a partir da década de 70 por vários protocolos de controle de concorrência, com o objetivo de garantir sempre a geração de estados consistentes para um determinado banco de dados. Este conceito determina que uma execução S de transações concorrentes para um conjunto T de transações é correta, se produzir um estado no banco de dados igual ao que seria produzido por alguma execução serial das transações pertencentes a T . O componente de um sistema de banco de dados responsável por executar protocolos de controle de concorrência é denominado de escalonador.

Os protocolos de controle de concorrência que implementam serializabilidade segundo a classificação apresentada em [Bernstein *et al* 1987] podem ter comportamento agressivo ou conservador. Os protocolos agressivos evitam atrasar o escalonamento (execução) de operações sobre objetos do banco de dados. Por outro lado, os protocolos conservadores podem atrasar as operações, para garantir um escalonamento de operações sobre o banco de dados que gere um estado consistente no banco de dados.

Os protocolos agressivos podem ser ainda classificados como otimistas ou pessimistas [Brayner *et al* 1999]. Um escalonador executando um protocolo pessimista deve decidir aceitar ou rejeitar uma operação de uma transação t assim que recebe a operação. Se decidir por rejeitar a operação, a transação t será abortada. Um escalonador com comportamento agressivo otimista imediatamente escalona a operação, e após um período de tempo, verifica se o escalonamento já processado está correto, decidindo então se continua sincronizando as operações ou se é necessário abortar uma ou mais transações.

Entre os protocolos com comportamento agressivo e que utilizam uma abordagem pessimista, destacam-se o de ordenação por marcas de tempo (TO) e o Teste de Grafo de Serialização (SGT). Dentre os protocolos agressivos e com abordagem otimista, destaca-se o protocolo de Validação (ou Certificador). Entre os protocolos conservadores, destaca-se o Protocolo de bloqueio em Duas fases (2PL).

Este artigo apresenta o Escalonador Inteligente de Transações (EIT), que se caracteriza por comportar-se de maneira agressiva ou conservadora. O comportamento do EIT, agressivo ou conservador, é alterado dinamicamente e sem interferência humana, de acordo com certas características do ambiente computacional onde está inserido. Essa característica de adaptabilidade do escalonador é importante em

ambientes computacionais onde o acesso aos dados através das transações é realizado de maneira não uniforme.

Este artigo está estruturado da seguinte maneira: na seção 2 é descrita a arquitetura do escalonador proposto. No decorrer da seção 3, uma análise dos resultados obtidos a partir dos testes realizados no EIT é apresentada. Na seção 4, algumas propostas encontradas na literatura são descritas e analisadas, e por fim as conclusões deste trabalho são apresentadas na seção 5.

2. EIT – Escalonador Inteligente de Transações

O EIT é um escalonador inteligente composto por um sistema especialista baseado em lógica *fuzzy* e pelos protocolos de controle de concorrência. A sua arquitetura é composta por dois módulos (figura 1): **Analizador**, com a função de tomar as decisões relacionadas à escolha do melhor comportamento do escalonador de acordo com as características do ambiente computacional; e o **Escalonador**, com a função de executar o protocolo de controle de concorrência definido (escolhido) pelo Analizador.

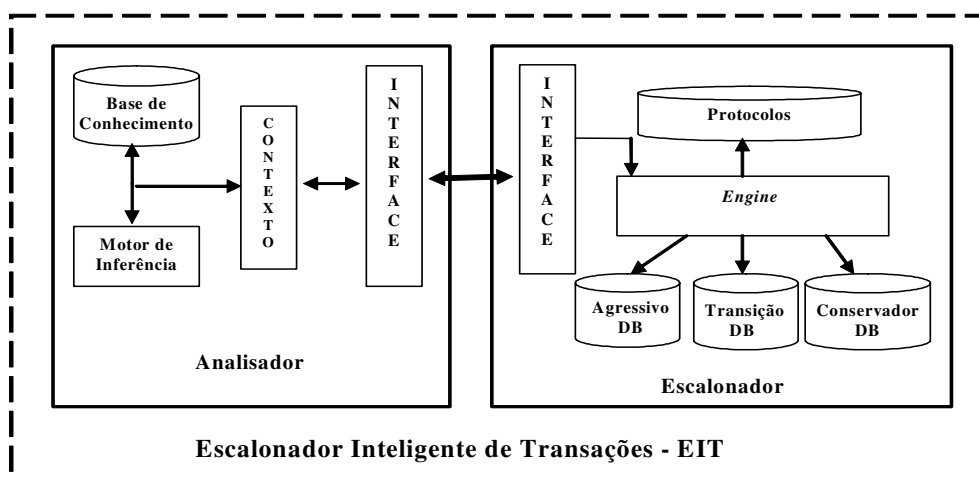


Figura 1: Modelo Abstrato do EIT

2.1. Analisador

O Analisador é o componente do EIT que tem como função decidir o comportamento para o Escalonador durante um determinado período de tempo, com o objetivo de aumentar o grau de concorrência entre as transações e diminuir a taxa de transações abortadas.

O Analisador é um Sistema Especialista (SE) baseado em lógica *fuzzy* [Zadeh 1965]. Um SE é um programa computacional que utiliza o conhecimento de um especialista para inferir conhecimento em um determinado domínio [Merrit 1989; Silver e Buckley 2005]. No módulo Analisador, apresentado na figura 1 são ilustrados os componentes básicos de um SE: Base de Conhecimento, que contém o conhecimento especializado a ser utilizado nas decisões; Mecanismo de Inferência, algoritmo capaz de elaborar as conclusões a partir dos dados fornecidos pelo usuário do sistema e do conhecimento armazenado em suas bases; Interface, realiza o diálogo entre o SE e o usuário (módulo Escalonador do EIT); Contexto, também chamado de memória de trabalho, é o componente que armazena os dados do problema que se deseja resolver.

2.1.1 Construindo o Analisador do EIT

Para o desenvolvimento e implementação do EIT foi utilizado o processo apresentado em [Canuto 2003], que especifica cinco etapas para construção de um sistema especialista baseado na lógica *fuzzy*: 1. Especificar o problema e definir as variáveis lingüísticas; 2. Definir os conjuntos *fuzzy*; 3. Construir as regras; 4. Construir o sistema especialista; e 5. Avaliar e melhorar o sistema.

1) Especificar o problema e Definir as variáveis lingüísticas

O problema que o SE deve resolver é: definir o comportamento mais apropriado para o escalonador (agressivo ou conservador) a partir das características do ambiente computacional onde o EIT está sendo utilizado. O comportamento mais apropriado é aquele que aumenta o grau de concorrência entre as transações e diminui o número de transações abortadas.

Durante a fase de aquisição do conhecimento foram definidas três variáveis lingüísticas de entrada e uma de saída. As variáveis lingüísticas de entrada são: taxa de transações abortadas (*tta*), a porcentagem do número de transações abortadas em relação ao número total de transações; taxa de *deadlocks* (*td*), a porcentagem do número de *deadlocks* em relação ao número total de transações; e taxa de operações de leitura (*tol*), a porcentagem do número de operações de leitura das transações em relação ao número total de todas as operações das transações. Para variável lingüística de saída, foi especificada a variável comportamento do escalonador (*ce*), que define o comportamento apropriado do escalonador.

2) Definir os conjuntos *fuzzy*

Essa etapa especifica os conjuntos *fuzzy* para cada variável lingüística. Os conjuntos *fuzzy* foram definidos usando a forma trapezoidal. Como pode ser observado na Figura 2, todas as variáveis lingüísticas de entrada têm os valores lingüísticos P (pequeno), M(médio) e G(grande), especificados no universo de discurso de 0 a 100. Para as variáveis lingüísticas *tta* e *td*, o valor P, significa que o escalonador está com seu comportamento apropriado, o M, um valor razoável, mas que seria interessante mudar para P, e o G, é inaceitável, ou seja, o escalonador deve mudar seu comportamento. Por sua vez, os valores da variável *tol*, são importantes para verificar a possibilidade de operações conflitantes, quando seu valor for P, significa uma probabilidade de muitas operações conflitantes, M, a probabilidade de operações conflitantes é razoável, e G, a probabilidade de operações conflitantes é pequena. Os pontos x_1 , x_2 , x_3 e x_4 (Figura 2) especificam os valores P, M e G, sendo fornecidos pelo administrador da aplicação.

A escolha da forma trapezoidal para esses conjuntos *fuzzy* de entrada levou em consideração que, essa forma representa bem os valores de entrada para a tomada de decisão do Analisador, tem um tempo de resposta pequeno (função linear), além de facilitar a configuração dos valores pelo administrador. Utilizando a forma trapezoidal, o administrador, deve definir o que realmente ele tem certeza, assim, do intervalo $[0, x_1]$ com certeza o valor é P, do intervalo de $[x_2, x_3]$ com certeza é M e do intervalo de $[x_4, 100]$ com certeza o valor é G. Os intervalos onde o especialista não tem certeza (conhecimento impreciso) são representados por $[x_1, x_2]$ e $[x_3, x_4]$.

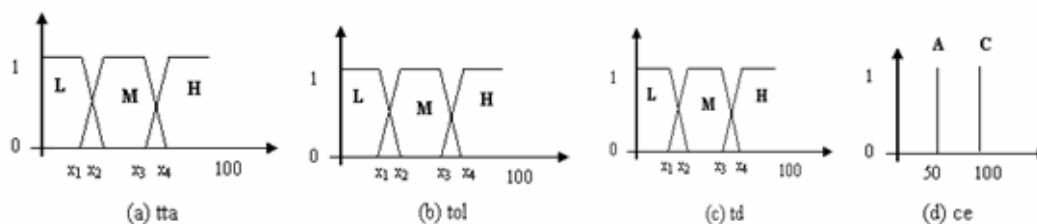


Figura 2: Conjuntos fuzzy do Analisador

A figura 2d mostra os conjuntos *fuzzy* da variável *ce*, a forma *singleton* foi escolhida uma vez que essa variável só pode ter os valores lingüísticos A (agressivo) ou C (conservador). O universo discurso dessa variável varia de 0 a 100.

3) Construindo as regras

O conhecimento adquirido foi formalizado através da representação por regras de produção, totalizando 12 regras, apresentadas a seguir:

Regra 1: IF *tta* = P AND *tol* = G THEN *c* = A
 Regra 2: IF *tta* = P AND *tol* = P THEN *c* = C
 Regra 3: IF *tta* = P AND *tol* = M THEN *c* = C
 Regra 4: IF *tta* = M AND *tol* = P THEN *c* = C
 Regra 5: IF *tta* = M AND *tol* = G AND *td* = M THEN *c* = A
 Regra 6: IF *tta* = M AND *tol* = M AND *td* = M THEN *c* = C
 Regra 7: IF *tta* = M AND *tol* = P THEN *c* = C

Regra 8: IF *tta* = G AND *tol* = P THEN *c* = C
 Regra 9: IF *tta* = G AND *tol* = G AND *td* = M THEN *c* = A
 Regra 10: IF *tta* = G AND *tol* = M AND *td* = M THEN *c* = C
 Regra 11: IF *tta* = G AND *tol* = P AND *td* = M THEN *c* = A
 Regra 12: IF *tta* = G AND *tol* = G THEN *c* = A

A regra 1, por exemplo, significa que, se a taxa de transações abortadas é P, taxa coerente com o comportamento apropriado do escalonador, e a taxa de transações de leitura é grande (G), então a possibilidade de conflitos é pequena e o escalonador pode usar o comportamento agressivo (A). As etapas 4 (Construir o sistema especialista) e 5 (Avaliar e melhorar o sistema) são descritas na seção 3.

2.2. Escalonador

O Escalonador do EIT é o módulo responsável pela execução de um protocolo de controle de concorrência de acordo com a decisão do Analisador. O Escalonador é composto pelos componentes apresentados a seguir: BD Protocolos, componente que armazena os algoritmos correspondentes aos protocolos de controle de concorrência; *Engine*, responsável pela execução dos protocolos de controle de concorrência; BD agressivo e BD conservador, armazenam os dados necessários para o escalonamento das transações quando o escalonador estiver com o comportamento agressivo e conservador, respectivamente; BD transição, armazena as informações necessárias para controlar as transações durante o período de transição; Interface, responsável pela comunicação do escalonador com outros componentes do EIT.

2.2.1 Comportamento do Escalonador

Nessa seção formaliza-se o comportamento do escalonador, após a apresentação inicial de definições básicas relacionadas ao controle de concorrência das transações.

Em relação ao controle de concorrência, uma transação (definição 1) é uma representação de seqüência de operações de leitura (*read*) e escrita (*write*) em objetos do banco de dados, sendo finalizada com uma operação de confirmação (*commit*) ou rejeição (*abort*), para indicar se a execução ocorreu com sucesso ou não, respectivamente [Bernstein *et al* 1987].

Definição 1 (Transação): uma transação T é uma seqüência de ações distintas a_1, a_2, \dots, a_n , onde cada ação a_i representa uma operação de leitura (*read* – r) ou escrita (*write* – w) desempenhada por T em objetos do banco de dados. Essa seqüência será finalizada com uma operação de confirmação (*commit* – c) ou rejeição (*abort* – a). ■

Escalonamento (definição 2) é uma seqüência ordenada no tempo de operações (de leitura ou escrita) pertencentes a uma ou mais transações. Outro conceito importante é o de projeção apresentado na definição 3.

Definição 2 (Escalonamento): Um escalonamento S sobre um conjunto de transações $T = \{T_1, T_2, \dots, T_n\}$ representa uma seqüência entrelaçada de operações das transações em T , onde S é um elemento do produto entrelaçado² $T_1 * T_2 * \dots * T_n$. A ordem indicada pelas transações em T deve ser preservada por qualquer escalonamento sobre T . O que significa, se uma operação p_i precede q_i em uma transação T_i (isto é, $p_i < q_i$), $T_i \in T$, então a execução de p_i deve acontecer antes de q_i , em qualquer escalonamento sobre T . O conjunto de todas as operações de S é representado por $OP(S)$. ■

Definição 3 (Projeção): Se S um escalonamento sobre o conjunto de transações T , e M um conjunto de transações, onde $T \supseteq M$. Uma projeção P de S sobre M é um escalonamento, no qual as seguintes condições devem ser obedecidas:

- (i) P apenas contém operações de transações pertencentes a M ;
- (ii) $\forall q \in OP(P)$, então $q \in OP(S)$, e;
- (iii) $\forall o, q \in OP(P)$, $o <_P q$ então $o <_S q$. ■

O conceito de serializabilidade, como dito anteriormente, vem sendo utilizado em sistemas de banco de dados tradicionais para identificar quais escalonamentos são corretos, isto é, escalonamentos que deixam o banco de dados em um estado consistente. Um escalonamento é serializável, se for equivalente a um escalonamento serial. Dois escalonamentos S e S' são ditos equivalentes, quando envolvem o mesmo conjunto de transações $T = \{T_1, T_2, \dots, T_n\}$, e produzem o mesmo estado final se executado no mesmo estado inicial do banco de dados. Há várias maneiras de se definir equivalência de escalonamentos, a equivalência por conflito é implementada na maioria dos SGBDs comerciais [Elmasri e Navathe 2000; Silberschatz, *et al* 1999].

Definição 4 (Operações conflitantes): Sejam $p_i \in OP(T_i)$ e $q_j \in OP(T_j)$ operações de T_i e T_j , com $i \neq j$, onde $OP(T_i)$ representa o conjunto de todas as operações T_i . Duas operações p_i e q_j conflitam (ou são conflitantes), se e apenas se elas são executadas sobre o mesmo objeto do banco de dados e ao menos uma delas é uma operação de escrita. ■

Dois escalonamentos S e S' definidos sobre o mesmo conjunto de transações são ditos equivalentes por conflito, se as operações conflitantes (Definição 4) nos dois

² Tradução livre para *shuffle product*.

escalonamentos aparecem na mesma ordem. Em outras palavras, S e S' são equivalentes por conflito se, para quaisquer operações conflitantes p_i e q_j pertencentes às transações T_i e T_j , respectivamente, se $p_i <_S q_j$, então $p_i <_{S'} q_j$, isto é, p_i e q_j apresentam a mesma ordem de serialização nos dois escalonamentos. Com base na noção de equivalência, pode-se definir um escalonamento S' , definido sobre um conjunto de transações T , como serializável por conflito, se este for equivalente por conflito a um escalonamento serial sobre o mesmo conjunto de transações T . Demonstra-se ainda que um escalonamento S é serializável por conflito se o seu grafo de serialização (SG – *Serialization Graph*) for acíclico [Bernstein *et al* 1987]. O grafo de serialização é um grafo direcionado, onde os nós são as transações e uma determinada aresta $T_i \rightarrow T_j$ existirá, se uma das operações de T_i precede e conflita com qualquer operação de T_j em S .

Após essas definições é apresentado então o comportamento do módulo Escalonador do EIT. O Escalonador, como já mencionado anteriormente, é capaz de sincronizar as operações das transações usando protocolos de controle de concorrência que se comportam de maneira agressiva ou conservadora. Dentro desse contexto, são definidos três períodos para o Escalonador: conservador, onde todas as transações são escalonadas utilizando um protocolo de controle de concorrência com comportamento conservador; transição, período de transição de comportamento do escalonador de conservador para agressivo ou de agressivo para conservador; e agressivo, quando todas as transações são escalonadas utilizando um protocolo de controle de concorrência com comportamento agressivo.

Para o EIT, o período de transição se inicia com a notificação da mudança de comportamento enviada pelo Analisador para o Escalonador e termina quando todas as transações ativas (Definição 5) no momento da notificação tenham se transformado em transações finalizadas (Definição 6).

Definição 5 (Transação Ativa): Uma transação T_i é ativa se:

$$\forall o \in OP(T_i), o \neq a \wedge o \neq c. \quad \blacksquare$$

Definição 6 (Transação finalizada): Uma transação T_i é definida como finalizada se:

$$\exists o \in OP(T_i), o = a \vee o = c. \quad \blacksquare$$

Antes da definição do escalonamento gerado por um escalonador EIT, faz-se necessário a definição de uma nova operação sobre escalonamentos, denominada junção ordenada de escalonamento e representada pelo símbolo \parallel . A junção ordenada de dois escalonamentos $S \parallel S'$ resultará em um escalonamento S^J , que contém uma seqüência de operações entrelaçadas pertencentes aos escalonamentos S e S' , de tal forma que, a ordem das operações, pertencentes a S e S' , é preservada em S^J . Se os escalonamentos S e S' possuírem transações iguais, só será possível aplicar a operação de junção ordenada $S \parallel S'$, se a ordem de serialização das operações conflitantes dessas transações for a mesma em S e S' .

Definição 7 (Junção Ordenada): Uma junção ordenada de dois escalonamentos S e S' , $S \parallel S'$, resultará em um escalonamento S^J , tal que:

(i) $OP(S^J) = OP(S) \cup OP(S')$;

(ii) $\forall p, q \in OP(S), p <_S q$ então $p <_{S^J} q$;

- (iii) $\forall p, q \in OP(S'), p <_{S'} q$ então $p <_{S^J} q$, e;
- (iv) $\forall p, q \in OP(S) \cap OP(S')$, se $p <_S q$, então $p <_{S'} q$, e $p <_{S^J} q$. ■

A figura 3 ilustra um exemplo de aplicação da operação de junção ordenada entre dois escalonamentos S e S' . O escalonamento S está definido sobre as transações T_1, T_2 e T_3 , enquanto S' está definido sobre as transações T_2, T_3 e T_4 . Observe que, em um determinado intervalo de tempo i , o escalonamento S^J apresenta o entrelaçamento de operações de S e S' , garantindo o item (iv) da Definição 7.

$$T_1 = r_1(x)w_1(y)r_1(v)w_1(v); T_2 = w_2(x)r_2(y); T_3 = r_3(x)w_3(z); T_4 = r_4(x)w_4(z)$$

$$S = r_1(x)w_2(x)r_3(x)w_1(y)r_1(v)w_3(z)r_2(y)w_1(v); S' = w_2(x)r_3(x)w_3(z)r_4(x)w_4(z)r_2(y)$$

$$S^J = r_1(x)w_2(x)r_3(x)w_1(y)r_1(v)w_3(z)r_4(x)w_4(z)r_2(y)w_1(v)$$

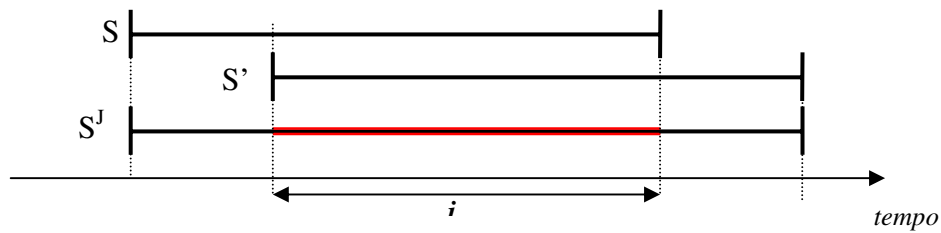


Figura 3: Exemplo de uma operação de junção ordenada

Agora, pode-se definir o conceito de escalonamento produzido pelo EIT, o qual é composto pelo escalonamento produzido por um protocolo conservador ou agressivo, acrescido do escalonamento no período de transição.

Definição 8 (EIT): Seja S_0 um escalonamento definido sobre um conjunto de transações T^0 e S_i um escalonamento definido sobre o conjunto de transações T^i , onde $T^0 \cap T^i = \emptyset$, $0 < i \leq n$. Um escalonamento S^{EIT} é definido como: $S^{EIT} = S_0 \parallel_{i=1}^n (S_{transição_i} \parallel S_i)$, onde S_0 é o escalonamento inicial, antes de qualquer mudança de comportamento do escalonador (agressivo ou conservador), S_i , $0 < i \leq n$, é o escalonamento gerado pelo escalonador com um outro protocolo em um novo tipo de comportamento. O escalonamento $S_{transição_i}$ é o escalonamento no período de transição e i é o número de mudanças de comportamento do escalonador. Se o escalonador não mudar o seu comportamento, $S^{EIT} = S_0$. ■

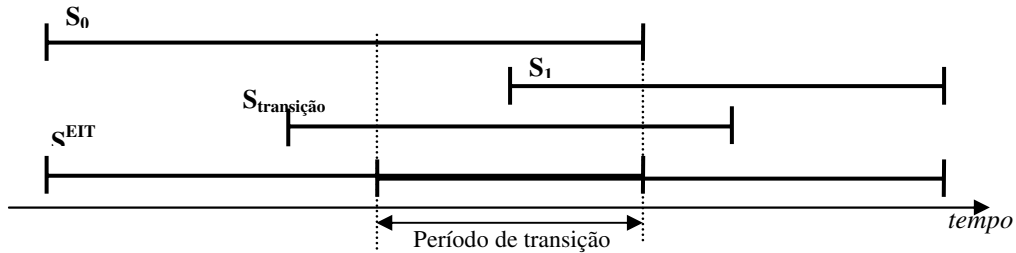


Figura 4: Escalonamento do EIT

A figura 4 ilustra a idéia de um escalonamento EIT. Observe que durante o período de transição de um protocolo para outro, o escalonamento deve ser executado obedecendo aos dois protocolos, com o objetivo de garantir a corretude do escalonamento S^{EIT} .

Para garantir escalonamentos corretos usando comportamento exclusivamente agressivo ou conservador, já existem propostas consolidadas na literatura, como é o caso do 2PL [Eswaran 1976], TO [Lamport 1978], SGT [Casanova 1981], dentre outros. Os códigos referentes às implementações destes protocolos estão armazenados no Banco de Protocolos (veja figura 1). Como o EIT opera com mudança dinâmica de protocolo, o problema chave é o período de transição, já que se deve garantir que o escalonamento gerado no período de transição, denominado de $S_{transição}$, gera estado consistente no banco dados sobre o qual está sendo executado. Em outras palavras, como sincronizar as transações no período de mudança de comportamento do EIT, de maneira que as transações concorrentes não produzam inconsistências no banco de dados.

O escalonamento $S_{transição}$ (gerado no período de transição) é definido sobre o conjunto de transações ativas no momento da notificação para mudança de protocolo, denominado de $T_{antigas}$, e sobre o conjunto das transações geradas durante o período de transição, com o novo comportamento, denominado de T_{novas} . Formalmente, $S_{transição}$ está definido sobre o conjunto $T_{transição} = T_{antigas} \cup T_{novas}$. O escalonamento no período de transição, $S_{transição}$, é a projeção de S^{EIT} (veja Definição 8) sobre o conjunto $T_{transição}$. Como as transações ativas estavam sendo escalonadas com um protocolo, e as transações novas devem ser escalonadas com um outro protocolo, $S_{transição}$ deve considerar os dois protocolos no período de transição, o protocolo antigo que já era usado nas $T_{antigas}$, e o protocolo novo que foi especificado para as T_{novas} .

Portanto, o escalonamento no período de transição deve levar em consideração tanto o escalonamento gerado a partir de um protocolo de controle de concorrência com comportamento agressivo, quanto o escalonamento gerado a partir do protocolo de concorrência com comportamento conservador. Para garantir que o escalonamento $S_{transição}$ seja correto, deve-se garantir que, a ordem de serializabilidade das operações conflitantes entre as transações do conjunto T_{novas} e $T_{antigas}$ seja a ordem de serializabilidade que obedeça tanto o protocolo de controle de concorrência agressivo quanto o conservador.

Lema 1: Seja $S_{transição}$ um escalonamento gerado pelo EIT no período de transição. $S_{transição}$ é um escalonamento correto.

Prova: Como $S_{transição}$ é um escalonamento gerado pelo EIT usando como critério de corretude a serializabilidade, então $SG(S_{transição})$ é acíclico. Pela teoria da serializabilidade, $S_{transição}$ é correto. ■

Teorema: Se S é um escalonamento representando uma execução produzida pelo EIT, então S é um escalonamento correto.

Prova: Considere S^{EIT} um escalonamento gerado pelo EIT, onde $S^{EIT} = S_0 \parallel_{i=1}^n (S_{transição_i} \parallel S_i)$. Os escalonamentos S_0 e S_i são gerados a partir de protocolos (com comportamento agressivo ou conservador) que utilizam serializabilidade como critério de corretude. O escalonamento $S_{transição}$ é correto conforme Lema 1. Portanto, os grafos de serialização para os escalonamentos S_0 , $S_{transição}$ e S_i não apresentam ciclos. As operações de junção $S_0 \parallel S_{transição}$ e $S_{transição} \parallel S_i$ são corretas, ou seja, não introduzem ciclos no grafo de serialização de S^{EIT} , pois (i) a ordem (de escalonamento) das operações comuns aos escalonamentos envolvidos em uma operação de junção ordenada é preservada (item (iv) da Definição 7), e (ii) S_0 e S_i

estão definidos sobre dois conjuntos de transações T^0 e T^i , onde $T^0 \cap T^i = \emptyset$, $0 < i \leq n$ (Definição 8). Pode-se, então, concluir que o grafo de serialização de S^{EIT} não contém ciclos. Portanto, S^{EIT} é correto. ■

A seguir, as regras necessárias para implementação do EIT são descritas. O escalonamento de operações de $\forall T_i, T_i \in \mathbf{T}_{antigas}$, e $\forall T_j, T_j \in \mathbf{T}_{novas}$, devem obedecer as seguintes regras:

- **R1:** as operações de T_i devem ser escalonadas com o protocolo anterior ao período de transição. As operações de T_j devem ser escalonadas com o novo protocolo.
- **R2:** as operações conflitantes entre T_j e T_i devem ser escalonadas no período de transição garantindo as regras dos protocolos antigo e novo;
- **R3:** as operações de T_i têm prioridade de execução em relação às operações de T_j . Desta forma, as operações de T_i que têm conflito com as operações de T_j induzem o cancelamento (*abort*) de T_j .

Observe que, para a definição das regras supracitadas, T_i estava ativa no início de um período de transição PT , pois $T_i \in \mathbf{T}_{antigas}$. Por outro lado, T_j é uma transação que iniciou sua execução durante o mesmo período de transição PT , já que $T_j \in \mathbf{T}_{novas}$. Portanto, a regra 1, garante que cada transação seja escalonada com o protocolo escolhido pelo Analisador. A regra 2 garante que durante o período de transição (nesse período o escalonador permanecerá com os dois comportamentos) o grafo de serialização (que contém as transações T_i e T_j) permaneça acíclico. Para tanto, as operações conflitantes devem apresentar a mesma ordem de serialização, independentemente do tipo de comportamento. A regra 3 refere-se à estratégia de implementação para garantir a regra 2, ou seja, pode ser necessário que o protocolo agressivo atrase as operações conflitantes de T_j ou o protocolo conservador rejeite as operações conflitantes de T_j , abortando a transação.

2.2.2 Exemplo de Funcionamento do EIT

Para ilustrar o funcionamento do escalonador EIT, considere o cenário descrito a seguir. Um escalonador EIT implementa os protocolos 2PL (conservador) e o TO (agressivo). O 2PL induz a serializabilidade através de bloqueios em duas fases. Por sua vez, o protocolo TO induz a serializabilidade através do ordenamento de operações com base em valores de marcas de tempo.

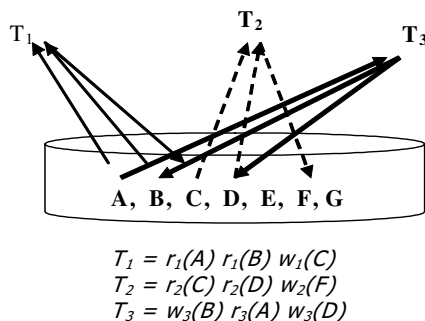


Figura 5: Exemplo de três transações concorrentes

A figura 5 apresenta uma ilustração de um sistema de um banco de dados $BD = \{A, B, C, D, E, F, G\}$, sobre o qual são executadas três transações T_1 , T_2 e T_3 concorrentemente.

Suponha que, inicialmente, o Escalonador executa o protocolo 2PL e que os eventos apresentados abaixo ocorreram na seguinte ordem (temporal):

1. O escalonador recebe a operação $r_1(A)$. Nesse caso, o Escalonador verifica se é possível conceder o bloqueio leitura para T_1 sobre o objeto A . Como essa é a primeira operação a ser escalonada, não existe bloqueio sobre A , o escalonador concede o bloqueio para a transação T_1 de acordo com as regras do protocolo 2PL;
2. O Escalonador recebe a notificação de mudança de comportamento, começando assim o período de transição do protocolo 2PL para o TO. Supondo que o *timestamp* da notificação ($ts_{transição}$) seja 1.5, o $ts(T_1)$ será 1.5;
3. O escalonador recebe $r_1(B)$. Como $r_1(B)$ é uma operação da transação T_1 , que começou a ser escalonada com o 2PL, esse protocolo deve ser utilizado para escalonar essa operação. Sendo assim, o escalonador verifica a possibilidade de conceder o bloqueio de acordo com o 2PL. Além disso, o escalonador verifica se existe conflito $r_1(B)$ com alguma operação de transações geradas após a notificação (transações pertencentes ao conjunto \mathbf{T}_{novas}). Como não existe o bloqueio do dado B e não existe nenhuma transação nova com operações conflitantes, o bloqueio de B é concedido a T_1 ;
4. Chega a operação $r_2(C)$. Nesse caso, $r_2(C)$ é uma operação de uma transação pertencente ao conjunto \mathbf{T}_{novas} (conjunto de transações iniciadas durante o período de transição). Portanto, o escalonador verifica se a operação $r_2(C)$ conflita com alguma operação de uma outra transação pertencente a \mathbf{T}_{novas} , ou seja, operações escalonadas com o TO. Além disso, o escalonador verifica se existe alguma operação de uma transação pertencente ao conjunto $\mathbf{T}_{antigas}$ (conjunto de transações iniciadas antes do período de transição), escalonada previamente com o 2PL que conflite com $r_2(C)$. Como não existe nenhum conflito para nenhum dos casos mencionados, $r_2(C)$ pode ser escalonada. Essa transação será escalonada com o protocolo TO. Suponha $ts(T_2)=2$;
5. Chega a operação $w_3(B)$. Pelo protocolo TO, esta operação poderia ser escalonada assim que ela chega. Contudo, o escalonador identifica um conflito com a operação $r_1(B)$, onde $T_1 \in \mathbf{T}_{antigas}$. Como $ts(T_1) < ts(T_3)$, essa operação será aceita, mas só após T_1 liberar o bloqueio sobre B ;
6. Chega a operação $w_1(C)$. Como $T_1 \in \mathbf{T}_{antigas}$, o escalonador verifica primeiramente se esta operação pode ser escalonada pelo protocolo 2PL. O escalonador identifica que $w_1(C)$ pode ser escalonada pelo 2PL, mas antes de escaloná-la, o escalonador deve verificar se esta operação conflita com alguma operação de uma transação pertencente ao conjunto \mathbf{T}_{novas} , que já foi escalonada. O escalonador identifica, então, um conflito com a operação $r_2(C)$. De acordo com a regra 3, como T_1 tem prioridade de execução, a transação T_2 deve ser abortada. A operação $w_1(C)$ é escalonada usando 2PL;
7. Chega o c_1 e a transação T_1 termina. O período de transição termina;
8. A operação $w_3(B)$ que estava na fila de espera é escalonada, a transação T_2 deve ser re-iniciada com um *timestamp* maior que o anterior. As operações seguintes são escalonadas com o protocolo TO.

O escalonamento S apresentado a seguir foi gerado a partir dos passos anteriormente referente a sincronização das operações das transações T_1 , T_2 e T_3 .

$$S = r_1(A) r_1(B) w_1(C) c_1 w_3(B) r_2(C) r_3(A) w_3(D) c_3 r_2(D) w_2(F) c_2$$

Como pode ser observado o grafo de serialização desse escalonamento é acíclico, portanto é um escalonamento correto.

3. Implementação do EIT e Análise dos Resultados

O EIT foi desenvolvido com a linguagem Java, sendo constituído de dois grandes módulos: o *analyzer* implementado com a API JFuzzy [JFuzzy 2005] e o *scheduler*. Os protocolos implementados no módulo *scheduler* foram o 2PL (conservador) e o TO (agressivo). Os testes que são apresentados a seguir baseiam-se nesses protocolos.

Para analisar o funcionamento do EIT foi implementado um simulador de transações, denominado TransactionSimulator. Com o simulador, é possível definir o número de transações, o número de operações de leitura, e a quantidade de dados que serão manipulados pelas transações. O simulador gera as transações randomicamente e encaminha as operações das transações para o EIT. O intervalo de envio das transações para o escalonador também pode ser configurado no simulador.

Para a utilização do EIT, o administrador da aplicação deve configurar os valores das variáveis *fuzzy* que são utilizados como referência na tomada de decisão em relação ao comportamento do escalonador. Para todos os testes apresentados nesse trabalho foram utilizados os seguintes valores: *tta* é considerada pequena se o valor for entre 0 e 30, considerada média quando varia de 40 a 70, e grande quando variar de 80 a 100; *tol* é considerada pequena quando seu valor estiver entre 0 e 30, considerada média quando varia de 40 a 70, e grande quando for de 80-100; e *td* é considerada pequena quando seu valor estiver entre 0 e 20, considerada média quando variar de 40 a 70, e grande quando for de 80-100. O administrador também deve especificar o comportamento inicial do EIT e o intervalo de tempo de análise. Nos testes realizados, o EIT iniciou com comportamento conservador (2PL) em todos os testes e o intervalo de análise foi de 10 segundos. Para simular as transações concorrentes foi considerado que a cada 1 milissegundo é encaminhada uma nova transação para o Escalonador. Cada transação é composta de 5 operações.

As figuras 6 e 7 apresentam dois gráficos de dois cenários de teste que são analisados a seguir. No primeiro cenário foi simulado um ambiente com transações compostas por 80% de operações de leitura (e 20% de operações de escrita), onde as operações das transações são executadas sobre banco de dados com apenas 5 objetos. Os testes foram realizados da seguinte forma. Primeiro, o EIT foi utilizado para sincronizar as operações das transações geradas pelo simulador de transações. Após isto, utilizou-se um escalonador que implementava apenas o protocolo 2PL preciso (*strict* 2PL) e finalmente foi utilizado um escalonador que implementava o protocolo TO para sincronizar as operações das transações geradas. Para o primeiro cenário, foram realizados três conjuntos de testes com cada um dos escalonadores, nos quais o simulador de transações gerava 100, 200 e 250 transações, respectivamente. Como pode ser observado na figura 6, até 100 transações, os escalonadores implementado as diferentes estratégias de escalonamento comportaram-se de maneira semelhante. Para escalonar operações de um número de transações maior que 200, o EIT apresentou um comportamento melhor do que o escalonador 2PL. A partir de 200 transações o EIT que

tinha como comportamento inicial conservador (2PL) mudou o comportamento para agressivo (TO), conforme ilustrado na figura 8. O protocolo que teve o pior desempenho nesse cenário foi o 2PL, isso ocorre pois, uma vez que o número de itens acessados por todas as transações é muito pequeno (um tipo de *hot spot*), há uma alta frequência na ocorrência de *deadlocks*. Na figura 7, é apresentado o gráfico do Cenário 2, onde foi utilizado uma taxa de leitura de 50% em um conjunto de dados de 20 elementos. Como pode ser observado o comportamento do EIT e do 2PL é semelhante, isso acontece porque o EIT inicia seu escalonamento com o comportamento 2PL, e continua com esse comportamento durante todo o período (Figura 9).

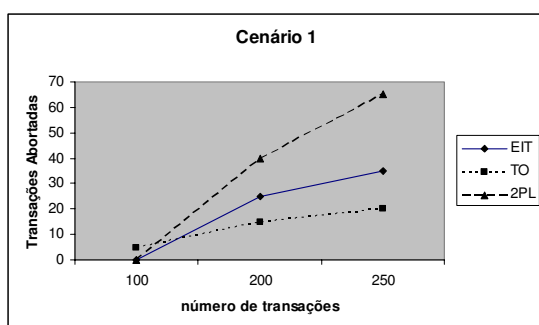


Figura 6: Gráfico Cenário 1

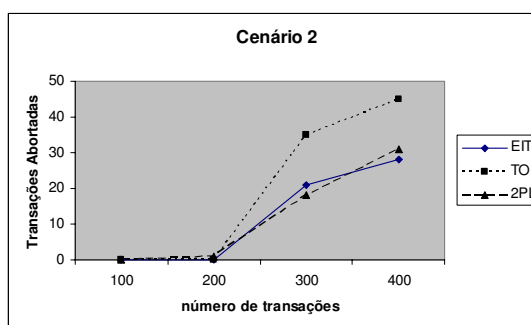


Figura 7: Gráfico Cenário 2

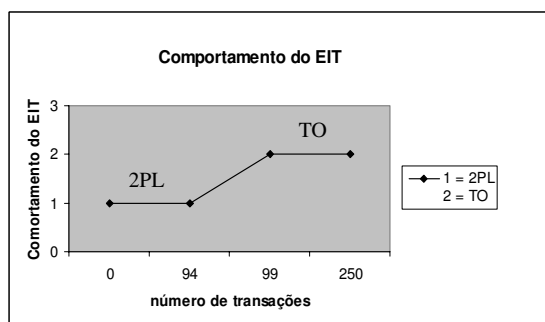


Figura 8: Comportamento do EIT Cenário 1

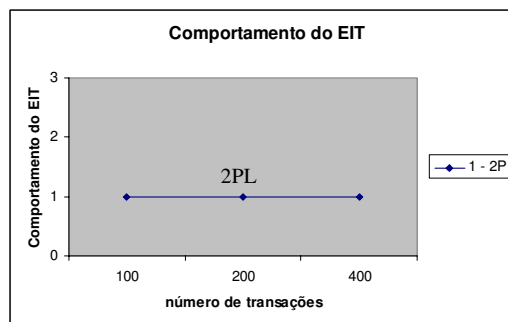


Figura 9: Comportamento do EIT Cenário 2

4. Trabalhos Relacionados

Na literatura existem algumas propostas de algoritmos de controle de concorrência com comportamento agressivo, conservador, e até mesmo algoritmos que integram características agressivas e conservadoras com o objetivo de obter um melhor desempenho do escalonador de transações. Porém, nenhuma dessas propostas utiliza um sistema especialista baseado em lógica *fuzzy* para auxiliar na tomada de decisão em relação a qual comportamento é mais apropriado a um escalonador em um determinado período de tempo. A maneira como o EIT garante escalonamentos corretos, onde no período de transição são considerados os escalonamentos gerados a partir de um protocolo de controle de concorrência com comportamento agressivo e também o escalonamento gerado a partir do protocolo de concorrência com comportamento conservador (veja seção 2.2), também é inovadora.

Bernstein e Goodman apresentaram em [Bernstein e Goodman 1981] algoritmos que integram características do 2PL e do TO. Nestes métodos as operações das

transações são divididas em operações *ww* (escrita-escrita) e *rwr* (leitura-escrita-leitura). Uma interface é criada para garantir escalonamentos acíclicos. Tal interface utiliza pontos de bloqueios para induzir *timestamps*. Nesse artigo são apresentados métodos que usam 2PL para sincronizar as operações *rw* e o TO para sincronização das operações *ww*, como também métodos onde o TO é utilizado para sincronizar as operações *rw* e o 2PL para as operações *ww*. Uma desvantagem dos métodos apresentados nessa proposta é que para o seu funcionamento, ou o conjunto de operações de escritas são pré-declaradas ou uma fila de espera de operação de escrita é necessária. Essa proposta diferentemente do EIT, não traz nenhum tipo de procedimento de mudança de comportamento do escalonador.

Bharat Bhagava em [Bhargava 1989] definiu adaptabilidade de algoritmo como o conjunto de técnica para mudança dinâmica de um algoritmo para outro algoritmo. Neste artigo o autor apresenta modelos de adaptabilidade que suportam a mudança de escalonadores em um sistema de transações distribuídas. Nessa proposta a adaptabilidade de algoritmo pode ser realizada de três maneiras: adaptabilidade temporal, onde um algoritmo muda ao longo do tempo para outro algoritmo, este modelo geralmente tem um período de transição; adaptabilidade por transação, permite que cada transação escolha o seu algoritmo, transações diferentes sendo executadas simultaneamente pode usar algoritmos diferentes; adaptabilidade espacial, uma variação da adaptabilidade por transação no qual transações escolhem seu algoritmo baseado nas propriedades dos itens de dados que ele acessa. O método apresentado por Bhagava que mais se aproxima ao apresentado pelo EIT é a adaptabilidade temporal sufixo-suficiente. A vantagem do método do EIT é que na proposta do Bhargava é necessário a definição de uma função de terminação para que o escalonador mude de algoritmo, enquanto que no EIT essa função não é necessária, uma vez que quando as transações antigas são terminadas, automaticamente o EIT passa a escalonar apenas com o novo protocolo.

Existem propostas na literatura que misturam técnicas de escalonamento baseado em protocolos otimistas e pessimistas, como apresentado em [Pavlova e Nekrestyanov 1997], onde um protocolo de controle de concorrência híbrido é apresentado para sincronizar as transações baseando-se em transações aninhadas para banco de dados de tempo-real. Nessa proposta, as transações são divididas em árvores, onde uma árvore de transações é serializada com uma outra transação usando a proposta otimista, mas transações dentro da mesma árvore são serializadas usando algoritmo pessimista. Como pode ser observado, a estratégia, utilizada na implementação do EIT, é mais flexível, já que dá suporte a execução de protocolos conservadores, como o 2PL.

5. Conclusão

Esse artigo apresentou o EIT, um escalonador de transações capaz de adaptar o seu comportamento dinamicamente e automaticamente (sem interferência humana) às características do ambiente computacional. Uma nova operação entre escalonamentos foi definida, chamada de junção ordenada, sendo essa utilizada para demonstrar a correteza dos escalonamentos gerados a partir do EIT. Para ilustrar o funcionamento do EIT um exemplo com os protocolos 2PL e TO foi apresentado, porém, o EIT pode ser utilizado com quaisquer protocolos baseado no critério de correteza de serializabilidade. Além disso, foram realizados teste de execução do EIT e os resultados destes testes foram apresentados e analisados.

O EIT mostrou-se uma implementação viável para o escalonamento de transações. Com os resultados obtidos, pode-se observar que o EIT atingiu seu objetivo, mesmo assim, outras variáveis para a escolha do comportamento do EIT pelo Analisador estão sendo investigadas, tais como, taxa de operações conflitantes, o tamanho do banco de dados, número de transações simultâneas, dentre outras. Outro ponto importante que está sendo verificado é o custo de processamento e o impacto em número de transações abortadas no período de transição entre a mudança de comportamento do escalonador.

Referências

- Bernstein, P. e Goodman, N. (1981) "Concurrency Control in Distributed Database Systems". *ACM Computing Surveys*, v. 13, n. 2, p.185-221.
- Bernstein, P., Hadzilacos, V. e Goodman, N. (1987) "Concurrency Control and Recovery in Database Systems". Massachusetts-USA: Addison-Wesley, 361 p.
- Bhargava, B. (1989) "A Model for Adaptable Systems for Transaction Processing", *IEEE Transactions on Knowledge and Data Engineering*, v. 1, n.4 p. 433-449.
- Brayner, A., Harder, T. e Ritter, N. (1999) "Semantic serializability: A correctness criterion for processing transaction in advanced database applications", *Data & knowledge Engineering*, v. 1, p. 1-24.
- Brayner, A. (1999) "Transaction Management in Multidatabase Systems". Shaker Verlag, 191 p.
- Canuto, A. (2003) "Inteligência Artificial: Uma Visão Introdutória.", Universidade Federal do Rio Grande do Norte.
- Casanova, M. (1981) "The Concurrency Control Problem of Database Systems", *Lecture Notes in Computer Science*, n. 116. Springer-Verlag.
- Elmasri, R., Navathe, S. (2000) "Fundamentals of Database Systems", Vancouver - Canada: Addison Wesley, 955 p.
- Eswaran, K., Gray, J., Lorie, R. e Traiger, I. (1976) "The Notions of Consistency and Predicate Locks in a Database System". *Commun. ACM*, v. 9, n. 11 p. 624-633.
- FuzzyJ Toolkit em http://www.iit.nrc.ca/IR_public/fuzzy/fuzzyJToolkit.html, acessado em agosto de 2005.
- Lamport, L. (1978) "Time, clocks, and ordering of events in a distributed systems". *Commun. ACM*, v. 21, p. 558-564.
- Merrit, D. (1989) "Building Expert Systems in Prolog", Lebanon, OH – USA: Springer-Verlag, 358 p.
- Molina, H., Ullman, J., Widom, J. (2001) "Implementação de Sistemas de Banco de Dados", Rio de Janeiro: Editora Campus, 685 p.
- Pavlova, E., Nekrestyanov, I. (1997) "Concurrency Control Protocol for Nested Transactions in Real-time Databases", In: *Advances in Databases and Information Systems*, p. 23-28.
- Silberschatz, A., Korth, H., Sudarshan, S. (1999) "Sistema de Banco de Dados", São Paulo: Editora Makron Books, 778 p..
- Silver, W., Buckley, J. (2005) "Fuzzy Expert Systems and Fuzzy Reasoning", New Jersey – USA: John Wiley & Sons. 405 p.
- Zadeh, L. (1965) "Fuzzy Set". *Information and Control*, v. 8, n. 1, p. 338-353.