SISTEMAS DE BANCO DE DADOS 2

AULA 10

Transação e Concorrência

Vandor Roberto Vilardi Rissoli

APRESENTAÇÃO

- Transação em B. de Dados (revisão)
- Protocolos para Concorrência (alguns)
- Ordenações por Timestamp
- Técnicas de Validação e Esquemas de Multiversão
- Prevenção de *Deadlook*
- Referências

TRANSAÇÃO

As operações que formam uma única unidade lógica de trabalho são chamadas de **TRANSAÇÕES**.

É uma unidade de execução de programa que acessa e manipula dados no Banco de Dados, sendo executada geralmente por um programa (instrução contida em aplicativo) elaborado com:

- linguagem de manipulação de dados (alto nível);

- linguagem de programação;

→ A transação consiste em todas as operações a serem executadas a partir do começo até o fim da transação.

TRANSAÇÃO

Estas propriedades também são conhecidas pelo acrônimo **A C I D**

Atomicidade
Consistência
Isolamento
Durabilidade

Exemplo:

Suponha um sistema bancário simplificado com várias contas e diversas transações sobre estas contas. A transação (**T**) de transferência de cinquenta reais (R$ 50,00) de uma conta **A** para uma outra conta **B** seria representada pela escala ao lado:

T: leia(A);
A = A – 50;
escreva(A);
leia(B);
B = B + 50;
escreva(B);

TRANSAÇÃO

Para assegurar a integridade dos dados, um BD deve garantir sempre algumas propriedades das transações. Estas propriedades são:

Atomicidade: Todas as operações da transação são refletidas corretamente no BD ou nenhuma será

Consistência: A execução de uma transação isolada preserva a consistência do BD (situação inicial e final)

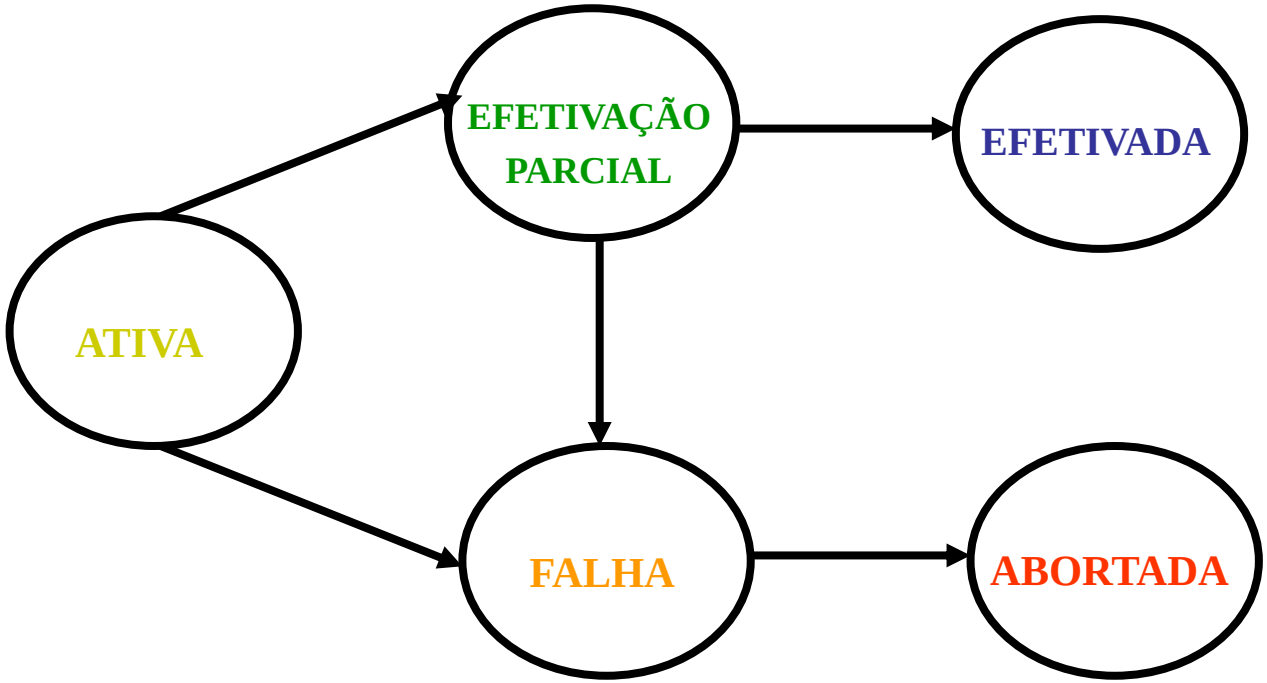
Isolamento: Cada transação não toma conhecimento das outras transações concorrentes

Durabilidade: Depois da transação completar-se com sucesso, as mudanças que ela faz no BD, persistem até mesmo se houver falhas no sistema

TRANSAÇÃO

ESTADO DA TRANSAÇÃO

Para melhor compreensão dos possíveis estados de uma transação será usado um modelo simples e abstrato destas situações:



TRANSAÇÃO

ATIVA: permanece neste estado enquanto está sendo executada a transação;

EFETIVAÇÃO PARCIAL: após execução da última declaração;

FALHA: descobre-se que a execução não poderá ser efetivada;

ABORTADA: transação desfeita, restabelecendo o BD ao início (para ser executada novamente deverá ser gerada uma nova transação);

EFETIVADA: após a conclusão com sucesso;

TRANSAÇÃO

- Diz-se que uma transação foi efetivada (*commited*) somente se ela entrar no estado de **EFETIVADA**;

- Diz-se que uma transação foi abortada (*rolled back*) somente se ela entrar no estado de **ABORTADA**;

- Uma transação é concluída se estiver no estado de **EFETIVADA** ou de **ABORTADA**.

TRANSAÇÃO

Uma transação no estado de **FALHA** (erro de hardware, ou lógico, ou de leitura, entre outros) não pode prosseguir com sua execução normal, devendo ser desfeita.

Assim, ela passa para o estado de **ABORTADA** e pode:

- Reiniciar a transação: possível somente para erros de hardware ou software e **não pela lógica da operação**;
- Encerrar a transação: **erro lógico** normalmente, pois a aplicação (ou programa) deverá ser refeito;

→ A operação que reinicia uma transação consiste na criação de uma **nova transação** para ser processada.

TRANSAÇÃO

TRANSAÇÕES CONCORRENTES

O processamento de transações concorrentes agiliza a realização da tarefa desejada, mas também traz diversas complicações em relação a consistência dos dados no BD.

Seria muito mais fácil manter as execuções das transações **sequenciais**, mas duas possibilidades básicas incentivam a concorrência e sua agilização:

- Operação da CPU e as E/S podem ser feitos em paralelo;
- Mistura de transações simultâneas no sistema
  - curtas;
  - longas;

TRANSAÇÃO

Ações executadas

→ Acessa diferentes partes do BD;

→ Reduz atrasos imprevisíveis;

→ Diminui o tempo médio de resposta;

→ Reduz ociosidade da CPU, discos e outros dispositivos.

O processamento concorrente compromete, principalmente, a propriedade de **CONSISTÊNCIA** do BD.

Para permitir a concorrência eficiente, sem comprometer a CONSISTÊNCIA, é analisada a escala de execução (*schedules*) das transações envolvidas.

TRANSAÇÃO

Exemplo:

No sistema bancário existem várias contas com vários lançamentos que acessam e atualizam estas contas. Supõem-se então duas transações T_1 e T_2 que transferem fundos de uma conta **A** para outra conta **B**.

T_1:	T_2:
leia(A);	leia(A);
A = A – 50;	aux = A * 0.10;
escreva(A);	A = A - aux;
leia(B);	escreva(A);
B = B + 50;	leia(B);
escreva(B);	B = B + aux;
	escreva(B);

Controle de Concorrência

A execução de diversas transações em modo **concorrente**, pode prejudicar a consistência do BD, sendo necessário ao sistema controlar a interação entre as transações que são executadas simultaneamente (ao mesmo tempo).

Para garantir a **serialização** são usados vários esquemas de controle sobre a concorrência, sendo os mais comuns sintetizados a seguir:

- Protocolos de Bloqueio;
- Ordenação por *Timestamp*;
- Técnicas de Validação;
- Esquemas de Multiversão.

Controle de Concorrência

PROTOCOLOS DE BLOQUEIO

Este protocolo é um conjunto de regras que estabelece quando uma transação pode bloquear e desbloquear um item de dados do SGBD.

Obriga que o acesso a um item de dados seja mutuamente exclusivo, ou seja:

→ enquanto uma transação acessa um item de dados, nenhuma outra transação pode modificá-lo.

Controle de Concorrência

Existem vários modos de bloqueio de dados, mas serão abordados somente os dois mais significativos.

COMPARTILHADO: (representado por **S**)

Se uma transação obteve um bloqueio compartilhado sobre um item de dados, ela só poderá ler o item, mas não escrevê-lo.

EXCLUSIVO: (representado por **X**)

Se uma transação obteve o bloqueio exclusivo sobre um item de dados, ela poderá ler e escrever neste item.

Controle de Concorrência

Exemplo:

Observe a transferência de 50 reais da conta **A** para conta **B,** que serão acessadas pelas respectivas transações (T_1 e T_2).

Supondo que **A** possui R$ 400,00 e **B** R$ 200,00 tem-se:

T_1: bloqueia-X(A);
leia(A);
A = A – 50;
escreva(A);
desbloqueia(A);
bloqueia-X(B);
leia(B);
B = B + 50;
escreva(B);
desbloqueia(B);

→ Esta transação apresenta como resultado a soma do saldo das contas:

T_2: bloqueia-S(A);
leia(A);
desbloqueia(A);
bloqueia-S(B);
leia(B);
desbloqueia(B);
apresente(A+B);

Controle de Concorrência

Os desbloqueios também podem ser solicitados ao final da transação, evitando o acesso e uso de informações momentaneamente inconsistentes.

T_3: bloqueia-**X**(A);
leia(A);
A = A – 50;
escreva(A);
bloqueia-**X**(B);
leia(B);
B = B + 50;
escreva(B);
desbloqueia(A);
desbloqueia(B);

T_4: bloqueia-**S**(A);
leia(A);
bloqueia-**S**(B);
leia(B);
apresente(A+B);
desbloqueia(A);
desbloqueia(B);

Controle de Concorrência

T_1	T_2	Gerenciador Concorrência
bloqueia-X(A);		concedido-X(A,T_1)

leia(A);
A = A – 50;
escreva(A);
desbloqueia(A);		

	bloqueia-S(B);	concedido-S(B,T_2)

	leia(B)
	desbloqueia(B);
	bloqueia-S(A);	

	leia(A);
	desbloqueia(A);
	apresente(A+B);
		concedido-S(A,T_2)

bloqueia-X(B);		concedido-X(B,T_2)

leia(B);
B = B + 50;
escreva(B);
desbloqueia(B);		

ESCALA 1

Controle de Concorrência

DEADLOCK

De forma geral, os *deadlocks* são problemas inerentes ao bloqueio, que garante a consistência do BD. (***deadlock*** pode ser traduzido como **impasse**)

Exemplo:

Observe a escala parcial de T_3 e T_4.

$\rightarrow$ T_4 espera que T_3 libere B

$\rightarrow$ T_3 também está esperando que T_4 libere A

Assim, chegasse a situação de que nenhuma dessas transações pode processar na sua forma normal, ocorrendo então um **impasse** (*deadlock*).

Controle de Concorrência

→ O uso de bloqueio pode causar situações indesejáveis;

→ Usar o bloqueio e o desbloqueio tão logo seja possível para evitar possíveis inconsistências;

T_3	T_4
bloqueia-S(B);	
leia(B);	
B = B – 50;	
escreva(B);	
	bloqueia-S(A);
	leia(A);
	bloqueia-S(B);
bloqueia-S(A);	

ESCALA 2-parcial

→ Desbloquear um item antes de solicitar o bloqueio de outro, reduzindo os possíveis *deadlocks*;

→ Na ocorrência de um *deadlock* o sistema deverá desfazer uma das transações, liberando um item de dado.

Controle de Concorrência

Exemplo:

Suponha que a transação T_2 tenha um bloqueio compartilhado (S) sobre um item de dado, e que T_1 solicite um bloqueio exclusivo sobre este mesmo item de dado.

→ T_1 esperará até que T_2 libere este item de dado;

→ porém T_3 solicita um bloqueio compartilhado sobre o mesmo item de dado, antes que T_2 libere-o;

→ o bloqueio de T_3 é compatível e será concedido, enquanto T_1 continua esperando a liberação;

→ novamente surge outra transação (T_4) solicitando um bloqueio compartilhado, e T_1 fica em espera;

Assim, T_1 poderá nunca ser processada, sendo necessário alguns cuidados para que isso não aconteça (inanição).

Controle de Concorrência

Concedendo bloqueio (mantendo cuidado com a **inanição**):

- Verificar se não existe nenhuma outra transação de bloqueio sobre um item de dado, cujo modo de bloqueio seja conflitante;
- Não existe nenhuma outra transação que esteja esperando um bloqueio sobre este item de dado e que tenha feita sua solicitação anteriormente;

Controle de Concorrência

PROTOCOLO DE BLOQUEIO EM DUAS FASES

Este protocolo permite que uma transação bloqueie um item de dado somente após desbloqueá-lo.

Fase de **Expansão**: transação pode obter bloqueios, mas NÃO pode desbloquear nenhum;

Fase de **Encolhimento**: transação pode liberar bloqueios, mas NÃO consegue obter nenhum outro bloqueio;

Esse protocolo garante a serialização, mas não esta livre de *deadlock*.

→ Com a falta de informações a respeito do acesso que é necessário sobre um item de dado, este protocolo será necessário e suficiente para garantir a serialização.

Controle de Concorrência

Conversão de Bloqueios

Consiste em um **refinamento** do protocolo básico de **bloqueio em duas fases** que pode:

- promover um bloqueio compartilhado para exclusivo (*upgrade*) na fase de **expansão**, ou

- rebaixar um bloqueio exclusivo para compartilhado (*downgrade*) no **encolhimento**.

→ Todos os bloqueios são desbloqueados após uma transação ser concluída (efetivada ou abortada).

Controle de Concorrência

Variações do bloqueio em duas fazes:

- **SEVERO**: em adição as características deste bloqueio, ele também exige que os bloqueios exclusivos sejam **mantidos** até a transação ser efetivada (encerrada);

- **RIGOROSO**: exige que todos os bloqueios sejam mantidos até que a transação seja encerrada (efetivada ou abortada).

→ Estas duas variações de bloqueios são usadas extensivamente em sistemas de BD comerciais.

Controle de Concorrência

Para obter escalas de **serialização de conflito**, sem usar o protocolo de bloqueio em duas fases, serão necessárias informações ADICIONAIS sobre a transação, ou a imposição de alguma estrutura ou ordenação sobre o conjunto de itens de dados do SGBD.

Existem diversos modelos, que precisam de quantidades diferentes de informações, de acordo com as características que este modelo irá proporcionar.

Controle de Concorrência

ORDENAÇÃO POR *TIMESTAMP*

Um outro método para determinação da ordem serializada é a seleção de uma ordenação entre transações em andamento.

Entre alguns métodos, o mais usado é o de ordenação por *timestamp*.

→ Cada transação recebe a associação de um único *timestamp* fixo, sendo ele criado pelo SGBD, antes que esta transação inicie sua execução.

Controle de Concorrência

Duas formas simples para esta implementação seriam, mas podem existir outras:

1- usar a hora do sistema (**relógio**);

2- usar um **contador** lógico automático;

O *timestamp* das transações determinam a ordem de serialização, sendo necessário garantir uma equivalência de escala serial na sua execução.

A ordenação por *timestamp* é realizada por meio da seleção na ordem de execução baseada no valor do *timestamp* entre pares de transação.

Controle de Concorrência

- Um único *timestamp* é associado a cada transação;
- Execução da transação com menor *timestamp*;
- Reversão da transação, sempre que a ordem for violada;
- Uma reversão, feita pelo controle de concorrência, recebe um novo *timestamp* e é re-iniciada;
- Este protocolo é resistente a *deadlock*, pois uma transação nunca espera.

Controle de Concorrência

Exemplo:

Observe a escala a seguir:

ESCALA 3

→ uma transação recebe um *timestamp*, antes da sua primeira instrução;

→ T_5 tem *timestamp* menor que T_6;

T_5	T_6
leia(A);	
	leia(A);
	A = A – 50;
	escreva(A);
leia(B);	
	leia(B);
apresente(A+B);	
	B = B + 50;
	escreva(B);
	apresente(A+B);

Controle de Concorrência

TÉCNICAS DE VALIDAÇÃO

Este método é mais adequado em situações que a maioria das transações sejam somente de leitura, com baixas taxas de conflito entre elas.

3 fases (**leitura,validação,escrita**) ⇒ 3 *timestamps* (1 transação)

- Associação de um único *timestamp* para cada transação;
- A ordem da serialização é determinada pelo *timestamp* associado;
- Necessidade de passar pelo teste de validação para completar-se ou será revertida até seu estado inicial;
- Uma transação nunca atrasa neste esquema.

Controle de Concorrência

AGREGAÇÃO DE ITENS DE DADOS

Em algumas situações pode ser vantajoso **agrupar alguns itens de dados** (bloqueio, ...).

Estes agrupamentos são tratados como itens de dados agregados que resultam em múltiplos níveis de granularidade.

Se estabelecem:

- tamanhos diferentes para cada item de dados;
- hierarquia entre eles;
- os menores itens são aninhados aos maiores;

→ Esta hierarquia pode ser representada graficamente como uma árvore.

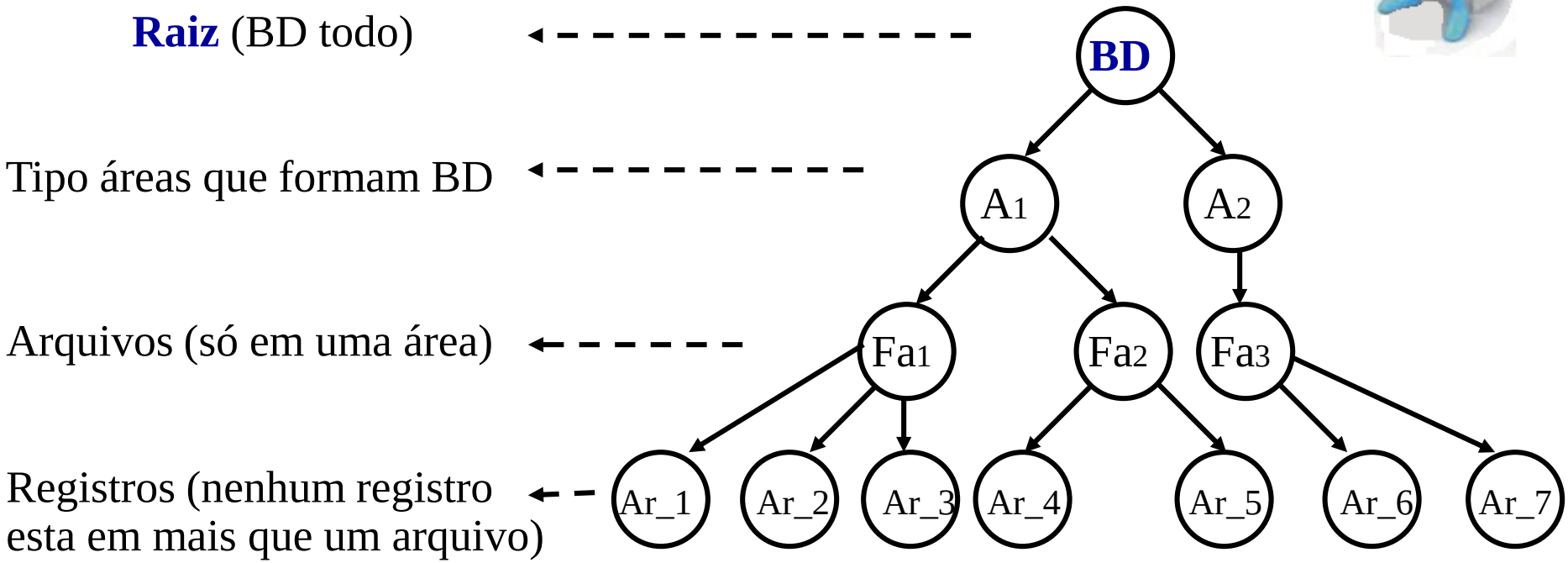
Controle de Concorrência

Agregação representada por meio de uma ÁRVORE:

- Bloqueios concedidos no sentido da raiz para as folhas;
- Liberação dos bloqueios no sentido contrário (das folhas para a raiz);
- O protocolo garante a serialização;
- Possibilidade da ocorrência de *deadlock*.

Controle de Concorrência

Exemplo:



→ Exemplo gráfico com nós consistindo em quatro níveis, sendo o nível mais alto a **Raiz**.

Controle de Concorrência

CONTROLE DE CONCORRÊNCIA MULTIVERSÃO

Os métodos estudados até aqui atrasam ou abortam uma transação, garantindo a serialização, e controlando a concorrência.

O **método de multiversão** tem por base a criação de uma **nova versão do item de dado** que será escrito.

→ Quando uma leitura é solicitada, o sistema seleciona uma das versões para realizar a leitura.

→ É crucial, para o desempenho, que uma transação possa determinar fácil e rapidamente qual a versão de item que será lida.

Controle de Concorrência

- O esquema de controle de concorrência garante que a versão a ser lida **será serializada** por meio do *timestamp*;
- Uma operação de **leitura sempre obtém sucesso**, porém na multiversão com ordenação por *timestamp* ela pode resultar em *rollback*;
- Na multiversão, com bloqueio em duas fases, uma operação de escrita pode ter que aguardar para efetivar um bloqueio, ou mesmo um *deadlock* pode ocorrer;
- Propriedades indesejáveis:
  - operação de leitura faz dois acesso a disco (própria leitura e a atualização do *R-timestamp*);
  - conflitos de transações são resolvidos por *rollback* (não usa tempo de espera).

Controle de Concorrência

MECANISMO DE MANIPULAÇÃO DE DEADLOCK

Vários protocolos de bloqueio ocasionam o *deadlock*, mas algumas abordagens podem preveni-los: - ciclo de espera poderá ocorrer ou

- todos os bloqueios serem solicitados juntos.

Um modo de evitar *deadlock* é usar a preempção e o *rollback* de transações.

Para controlar a preempção:

- marca-se um único ***timestamp*** para cada transação;
- *timestamp* auxilia na decisão da transação **esperar ou ser revertida** (desfeita);
- para transações revertidas o *timestamp* é mantido, quando ela for reiniciada;

Controle de Concorrência

PREVENÇÃO DE *DEADLOCK*

Dois esquemas de prevenção de *deadlock* baseados na **preempção** são:

ESPERAR-MORRER

- T_A solicita um item mantido por T_B;
- T_A pode esperar se tiver um *timestamp* **menor** que T_B;
- T_A é **mais antigo** que T_B;
- Caso contrário T_A será revertida (**morta**);
- tem por base uma técnica de não-preempção.

FERIR-ESPERAR

- T_A solicita um item mantido por T_B;
- T_A pode esperar se tiver um *timestamp* **maior** que T_B;
- T_A é **mais nova** que T_B;
- Caso contrário T_B será desfeita (**ferida**).
- tem por base uma técnica de preempção, sendo a contrapartida do Esperar-Morrer.

Controle de Concorrência

Exemplo:

Suponha as transações T_7, T_8 e T_9 com os respectivos *timestamps* 11, 15 e 17;

ESPERAR-MORRER

- Se T_9 solicitar um item de dado mantido por T_8, então T_9 será desfeita;

FERIR-ESPERAR

- Se T_7 solicitar um item de dado mantido por T_8, então o item de dado será liberado por T_8 , que será desfeita;
- Se T_9 solicitar um item de dado mantido por T_8, então T_9 esperará;

Controle de Concorrência

Uma outra ALTENATIVA seria o uso do método de detecção de *deadlock* e recuperação.

- Elabore um gráfico de espera;
  - A existência de um ciclo, geralmente lembrando um **losango**, neste gráfico indica um *deadlock*;
- Usar um algoritmo de detecção de *deadlock*;
- Na detecção de um *deadlock*, o sistema irá se recuperar, revertendo uma ou mais transações para "romper" (quebrar) o *deadlock*.

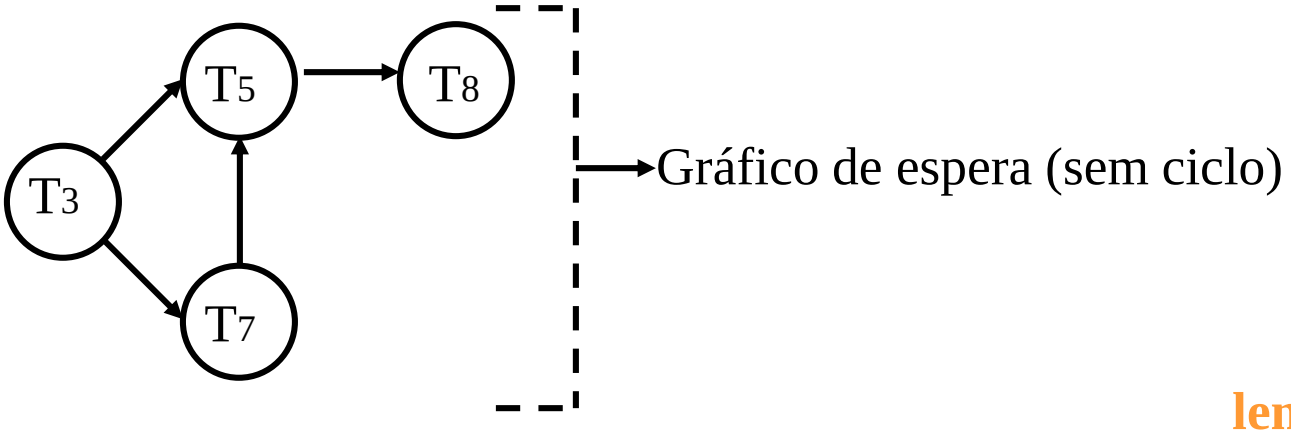
Controle de Concorrência

Exemplo - Suponha que:

→ A transação T_3 está esperando as transações T_5 eT_7;

→ A transação T_7 está esperando as transação T_5;

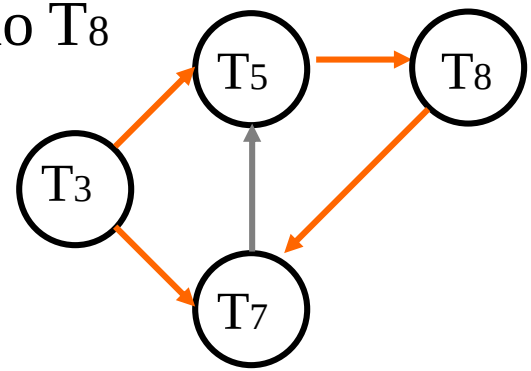
→ A transação T_5 está esperando as transações T_8.



Para existência de um ciclo imagine que a transação T_8 esteja solicitando um item preso por T_7 ($T_8 \rightarrow T_7$):

Este gráfico contém um ciclo, implicando em um *deadlock* entre T_5, T_7 e T_8

lembra losango



Controle de Concorrência

Após a detecção de um ***deadlock*** um sistema precisa recuperar-se (normalmente reverter uma ou mais transações).

- Selecionar a transação:
  - Tempo da transação (realizado e à processar);
  - Quantos itens de dados a transação já usou;
  - Quantos itens ela ainda usará para se completar;
  - Quantas transações serão envolvidas no *rollback*;
- Reversão ou *rollback*;
  - Reversão total ou somente até a quebra do *deadlock*;
- **Inanição** (escolha sempre da mesma "vítima" - custo).

Controle de Concorrência

OPERAÇÃO DE INSERÇÃO E REMOÇÃO

INSERÇÃO

- Necessita do bloqueio exclusivo sobre a NOVA tupla;
- Possível ocorrência do fenômeno do fantasma.

REMOÇÃO

- Necessita de bloqueio EXCLUSIVO sobre a tupla a ser excluída;

→ **FENÔMENO DO FANTASMA** – uma **inserção entra em conflito com uma consulta**, mesmo que cada uma delas não acessem uma tupla em comum.

Controle de Concorrência

Uma solução para este problema seria o bloqueio de índice (a ser estudado). Ele garante que todas as transações conflitantes estejam em conflito por itens de dados reais, e não por "**fantasmas**".

→ Para estruturas de dados especiais podem ser desenvolvidas técnicas especiais de controle de concorrência.

- Normalmente são aplicadas sobre árvores $\mathbf{B^+}$ visando o aumento da concorrência;
- Essas técnicas permitem acessos não-seriados sobre as árvores $\mathbf{B^+}$;
- Estrutura muito adequada, com acesso garantido ao SGBD, de forma seriada.

Referência de Criação e Apoio ao Estudo

Material para Consulta e Apoio ao Conteúdo

- ELMASRI, R. e NAVATHE, S. B., Fundamentals of Database Systems.
  - Capítulos 19 e 20
- SILBERSCHATZ, A., KORTH, H. F., Sistemas de Banco de Dados.
  - Capítulos 13 e 14
- DATE, C. J., Introdução a Sistemas de Banco de Dados, Editora Campus.
  - Páginas 411 - 436