Ciência da Computação

GBC043 Sistemas de Banco de Dados

Processamento de Transações
Controle de Concorrência

Profa. Maria Camila Nardini Barioni

camila.barioni@ufu.br

Bloco B - sala 1B137

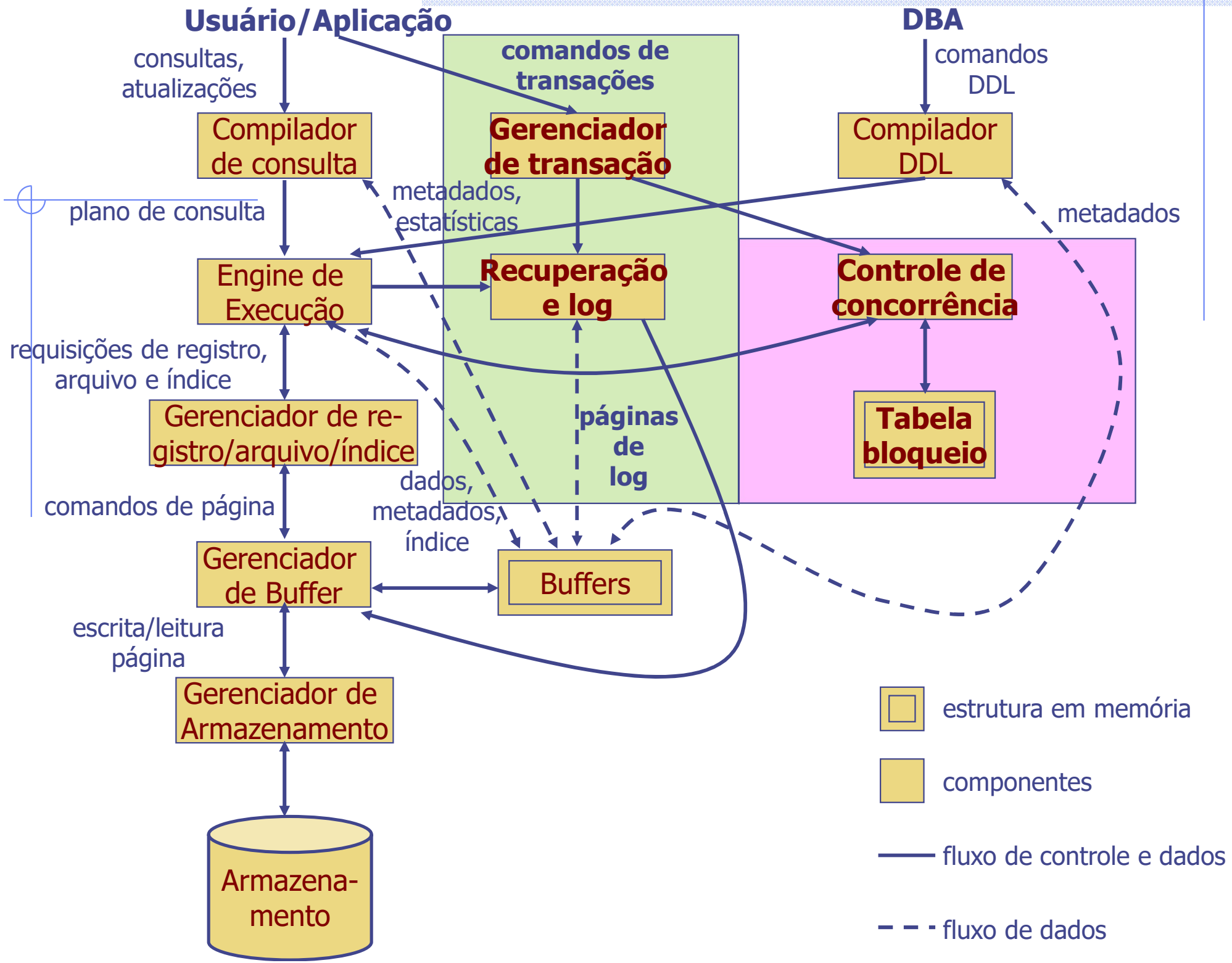
1° semestre de 2024

Roteiro Próximas aulas

- 24/10: vista da segunda prova
- Hoje e 31/10: Continuação da apresentação dos conceitos sobre transações
- 25/10 Aula reservada para finalização do projeto. Grupos irão se reunir para finalizar e entregar o projeto
- 01/11, 07/11, 08/11 e 14/11: Apresentações dos projetos
- 21/11: Prova Substitutiva (Toda a matéria!!!)
- 22/11: Vista final

Roteiro da aula

- Protocolos baseados em bloqueio
- Lidando com *Deadlock* (impasse) e *Starvation* (inanição)


Usuário/Aplicação
DBA
consultas, atualizações
comandos de transações
comandos DDL
Compilador de consulta
Gerenciador de transação
Compilador DDL
plano de consulta
metadados, estatísticas
metadados
Engine de Execução
Recuperação e log
Controle de concorrência
requisições de registro, arquivo e índice
Gerenciador de re-gistro/arquivo/índice
páginas de log
Tabela bloqueio
comandos de página
dados, metadados, índice
Gerenciador de Buffer
Buffers
escrita/leitura página
Gerenciador de Armazenamento
Armazena-mento
estrutura em memória
componentes
fluxo de controle e dados
fluxo de dados

Protocolos baseados em bloqueio

- Um **bloqueio** é um mecanismo para controlar o acesso simultâneo a um item de dados
- Um **bloqueio (lock)** é uma variável associada a um item de dados que descreve a condição do item em relação às possíveis operações que podem ser aplicadas a ele
- Vários tipos de bloqueios
  - Binários
  - Compartilhados/Exclusivos (ou Leitura/Escrita)
  - Conversão de Bloqueios

Tipos de bloqueio: Binários

- Podem ter dois **estados ou valores**
  - bloqueado (ou 1): Nesse caso, o item X que possuir esse valor de bloqueio não poderá ser acessado por uma operação do BD que solicite o item
  - desbloqueado (ou 0): Nesse caso, o item X que possuir esse valor de bloqueio poderá ser acessado quando solicitado
- Valor corrente do bloqueio associado a um item X → **lock(X)**
- Duas operações para o bloqueio binário: **lock_item(X)** e **unlock_item(X)**

Tipos de bloqueio: Binários

Figura 18.1 Operações de bloqueio e desbloqueio para bloqueios binários.

```
lock_item(X)
B: if LOCK(X)=0 (*item está desbloqueado*)
   then LOCK (X)←1 (*bloquear o item*)
   else begin
     wait (até que lock(X) = 0 e
     o gerenciador de bloqueio reinicia a transação);
     go to B
     end;

unlock_item (X):
LOCK (X)←0; (*desbloquear o item*)
 se  alguma transação estiver esperando
 então reiniciar uma das transações em espera;
```

Tipos de bloqueio: Binários

- Nesse esquema toda transação deve obedecer as seguintes regras

1. Uma transação T deve garantir a operação lock_item(X) antes que qualquer operação ler_item(X) ou escrever_item(X) seja executada em T
2. Uma transação T deve garantir a operação unlock_item(X) depois que todas operações ler_item(X) e escrever_item(X) sejam completadas em T
3. Uma transação T não resultará em uma operação lock_item(X) se ela já tiver o bloqueio no item X
4. Uma transação T não resultará em uma operação unlock_item(X), a menos que ela já tenha o bloqueio no item X

Tipos de bloqueio: Compartilhados/Exclusivos

- O esquema anterior é muito restritivo pois, no máximo, uma transação pode assumir o bloqueio em um dado item
- É interessante permitir que várias transações acessem um mesmo item, se todas elas tiverem o propósito de leitura
- Os itens de dados podem ser bloqueados em dois modos:

  1. Modo *exclusivo (X) / escrita*. O item de dados pode ser lido e também escrito. O bloqueio X é solicitado pela instrução write_lock(X) (ou lock-X).

  2. Modo *compartilhado (S) / leitura*. O item de dados só pode ser lido. O bloqueio S é solicitado pela instrução read_lock(X) (ou lock-S).

- As solicitações de bloqueio são feitas ao gerenciador de controle de concorrência. A transação só pode prosseguir após a concessão da solicitação.

Tipos de bloqueio: Compartilhados/Exclusivos

- Matriz de compatibilidade de bloqueio

	S	X
S	true	false
X	false	false

- Uma transação pode receber um bloqueio sobre um item se o bloqueio solicitado for compatível com os bloqueios já mantidos sobre o item por outras transações
- Qualquer quantidade de transações pode manter bloqueios compartilhados sobre um item, mas se qualquer transação mantiver um bloqueio exclusivo sobre um item, nenhuma outra pode manter qualquer bloqueio sobre o item
- Se um bloqueio não puder ser concedido, a transação solicitante deve esperar até que todos os bloqueios incompatíveis mantidos por outras transações tenham sido liberados. O bloqueio é então concedido.

Tipos de bloqueio: Compartilhados/Exclusivos

- Nesse esquema toda transação deve obedecer as seguintes regras

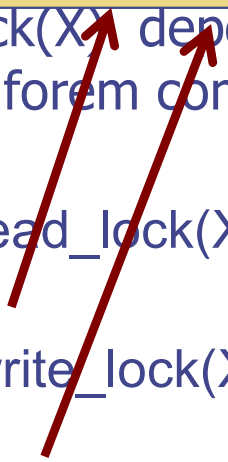
1. Uma transação T deve garantir a operação read_lock(X) ou write_lock(X) antes de qualquer operação ler_item(X) ser executada em T
2. Uma transação T deve garantir a operação write_lock(X) antes de qualquer operação escrever_item(X) ser executada em T
3. Uma transação T deve garantir a operação unlock(X) depois que todas operações ler_item(X) e escrever_item(X) forem completadas em T
4. Uma transação T não vai gerar uma operação read_lock(X) se ela já tiver um bloqueio de leitura ou escrita no item X
5. Uma transação T não vai gerar uma operação write_lock(X) se ela já tiver um bloqueio de escrita no item X
6. Uma transação T não resultará em uma operação unlock_item(X), a menos que ela já controle um bloqueio de leitura ou escrita no item X

Tipos de bloqueio: Compartilhados/Exclusivos

- Nesse esquema toda transação deve obedecer as seguintes regras

1. Uma transação T deve garantir a operação read_lock(X) ou write_lock(X) antes de qualquer operação ler_item(X) ser executada em T
2. Uma transação T deve garantir a operaç
   qualquer operação escrever_item(X) ser
3. Uma transação T deve garantir a operação unlock(X) depois que todas operações ler_item(X) e escrever_item(X) forem completadas em T
4. Uma transação T não vai gerar uma operação read_lock(X) se ela já tiver um bloqueio de leitura ou escrita no item X
5. Uma transação T não vai gerar uma operação write_lock(X) se ela já tiver um bloqueio de escrita no item X
6. Uma transação T não resultará em uma operação unlock_item(X), a menos que ela já controle um bloqueio de leitura ou escrita no item X

podem ser relaxadas



Tipos de bloqueio: Conversão de Bloqueios

- Nesse esquema é permitido, sob certas condições, que uma transação que já controla um bloqueio no item X converta o bloqueio de um estado para outro
  - promoção de read_lock(X) para write_lock(X) → Se T é a única transação que controla um bloqueio de leitura em X
  - rebaixamento de write_lock(X) para read_lock(X)

Tipos de bloqueio: Compartilhados/Exclusivos

- **Exemplo** de uma transação realizando bloqueio:

  T_2: lock-S*(A)*;

  read *(A)*;

  unlock*(A)*;

  lock-S*(B)*;

  read *(B)*;

  unlock*(B)*;

  display*(A+B)*

- O bloqueio acima não é suficiente para garantir a seriação - se *A* e *B* fossem atualizados entre a leitura de *A* e *B*, a soma exibida estaria errada.

- Um protocolo de bloqueio é um conjunto de regras seguidas por todas as transações enquanto solicita e libera bloqueios. Os protocolos de bloqueio restringem o conjunto de planos de execução possíveis.

Tipos de bloqueio: Compartilhados/Exclusivos

(a)

T_1	T_2
read_lock(*Y*);	read_lock(*X*);
read_item(*Y*);	read_item(*X*);
unlock(*Y*);	unlock(*X*);
write_lock(*X*);	write_lock(*Y*);
read_item(*X*);	read_item(*Y*);
X:=*X*+*Y*;	*Y*:=*X*+*Y*;
write_item(*X*);	write_item(*Y*);
unlock(*X*);	unlock(*Y*);

(b) Valores iniciais: *X*=20, *Y*=30
Resultado do plano de execução serial T_1 seguido por T_2:
X=50, *Y*=80
Resultado do plano serial T_2 seguido por T_1:
X=70, *Y*=50

(c)

T_1	T_2
read_lock(*Y*);	
read_item(*Y*);	
unlock(*Y*);	
	read_lock(*X*);
	read_item(*X*);
	unlock(*X*);
	write_lock(*Y*);
	read_item(*Y*);
	Y:=*X*+*Y*;
	write_item(*Y*);
	unlock(*Y*);
write_lock(*X*);	
read_item(*X*);	
X:=*X*+*Y*;	
write_item(*X*);	
unlock(*X*);	

Tempo

Resultado do plano S:
X=50, *Y*=50
(não serializavel)

Figura 18.3 Exemplo que pode produzir um plano de execução não serializável.

O protocolo de bloqueio em duas fases

- Esse é um protocolo que garante planos de execução seriáveis por conflito.
- Fase 1: **Fase de crescimento ou expansão**
  - transação pode obter bloqueios
  - transação não pode liberar bloqueios
- Fase 2: **Fase de encurtamento ou encolhimento**
  - transação pode liberar bloqueios
  - transação não pode obter bloqueios
- O protocolo garante a seriação. Pode ser provado que as transações podem ser seriadas na ordem de seus pontos de bloqueio (ou seja, o ponto onde uma transação adquiriu seu bloqueio final)

Tipos de bloqueio: Compartilhados/Exclusivos

(a)

T_1	T_2
read_lock(*Y*);	read_lock(*X*);
read_item(*Y*);	read_item(*X*);
unlock(*Y*);	unlock(*X*);
write_lock(*X*);	write_lock(*Y*);
read_item(*X*);	read_item(*Y*);
X:=*X*+*Y*;	*Y*:=*X*+*Y*;
write_item(*X*);	write_item(*Y*);
unlock(*X*);	unlock(*Y*);

não seguem o protocolo de bloqueio em duas fases

seguem o protocolo de bloqueio em duas fases

T_1'	T_2'
read_lock (*Y*);	read_lock (*X*);
read_item (*Y*);	read_item (*X*);
write_lock (*X*);	write_lock (*Y*);
unlock (*Y*);	unlock (*X*);
read_item (*X*);	read_item (*Y*);
X:=*X*+*Y*;	*Y*:=*X*+*Y*;
write_item (*X*);	write_item (*Y*);
unlock (*X*);	unlock (*Y*);

O protocolo de bloqueio em duas fases

- Variações:
  - **2PL conservador (ou estático)** → requer uma transação para bloquear todos os itens que ela acessa antes da transação iniciar pela pré-declaração de seus conjuntos de leitura (*read-set*) e escrita (*write-set)*. Se algum dos itens não puder ser bloqueado, a transação espera até que todos estejam disponíveis para bloqueio
  - **2PL estrito** → a transação não libera nenhum de seus bloqueios exclusivos (escrita) até que ela efetive ou aborte
  - **2PL rigoroso** → a transação não libera nenhum de seus bloqueios (exclusivos ou compartilhados) até que ela efetive ou aborte

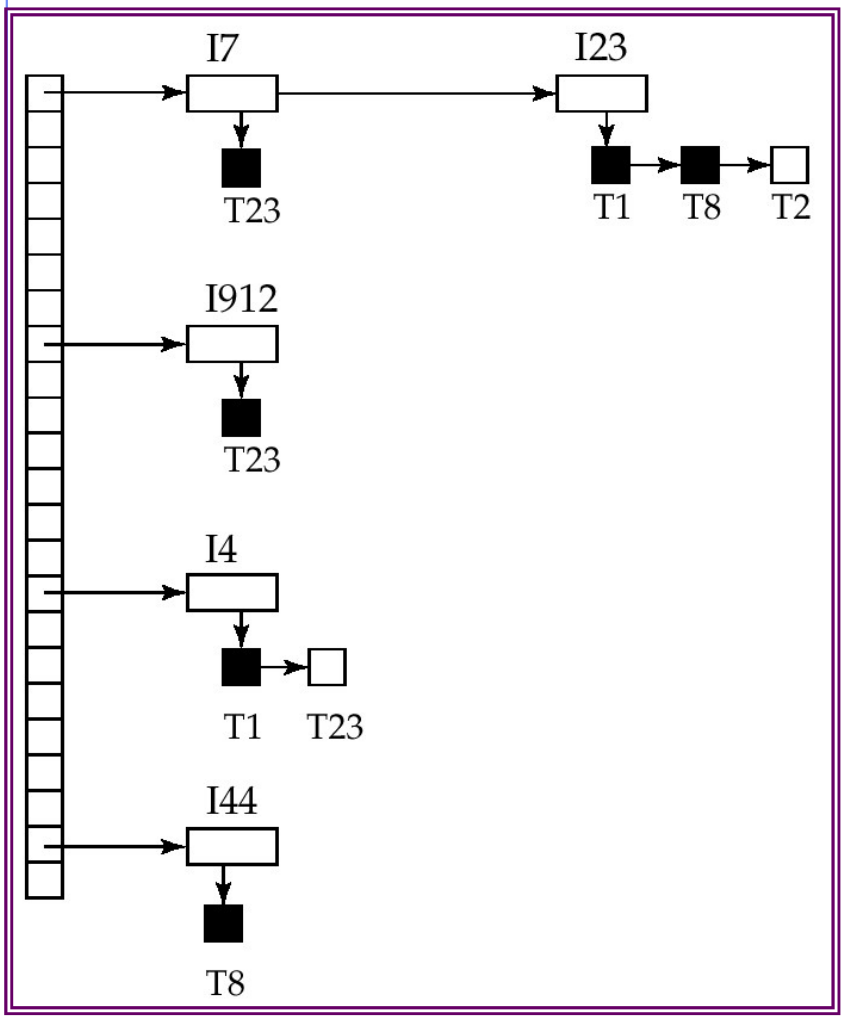
Conversões de bloqueio

- Bloqueio em duas fases com conversões de bloqueio:
  - Primeira fase:
    - pode adquirir um bloqueio-S sobre o item
    - pode adquirir um bloqueio-X sobre o item
    - pode converter um bloqueio-S para um bloqueio-X (upgrade)
  - Segunda fase:
    - pode liberar um bloqueio-S
    - pode liberar um bloqueio-X
    - pode converter um bloqueio-X para um bloqueio-S (downgrade)
- Esse protocolo garante a seriação. Mas ainda conta com o programador para inserir as diversas instruções de bloqueio.

Implementação do bloqueio

- Um gerenciador de bloqueio pode ser implementado como um processo separado para o qual as transações enviam solicitações de bloqueio e desbloqueio
- O gerenciador de bloqueio responde a uma solicitação de bloqueio enviando uma mensagem de concessão de bloqueio (ou uma mensagem pedindo à transação para reverter, no caso de um impasse)
- A transação solicitante espera até que sua solicitação seja respondida
- O gerenciador de bloqueio mantém uma estrutura de dados chamada **tabela de bloqueio** para registrar bloqueios concedidos e solicitações pendentes
- A tabela de bloqueio normalmente é implementada como uma tabela de hash na memória indexada sobre o nome do item de dados sendo bloqueado

Tabela de bloqueio



- Retângulos pretos indicam bloqueios concedidos, brancos indicam solicitações aguardando
- A tabela de bloqueio também registra o tipo de bloqueio concedido ou solicitado
- A nova solicitação é acrescentada ao final da fila de solicitações para o item de dados, e concedida se for compatível com todos os bloqueios anteriores
- As solicitações de desbloqueio resultam na solicitação sendo excluída e solicitações posteriores são verificadas para saber se agora podem ser concedidas
- Se a transação abortar, todas as solicitações aguardando ou concedidas da transação são excluídas
  - o gerenciador de bloqueio pode manter uma lista de bloqueios mantidos por cada transação, para implementar isso de forma eficiente

Armadilhas dos protocolos baseados em bloqueio

- Considere o plano de execução parcial

T_3	T_4
lock-X(B)	
read(B)	
$B := B - 50$	
write(B)	
	lock-S(A)
	read(A)
	lock-S(B)
lock-X(A)	

  - Nem T_3 nem T_4 podem ter progresso - a execução de lock-S*(B)* faz com que T_4 espere que T_3 libere seu bloqueio sobre *B*, enquanto a execução de lock-X*(A)* faz com que T_3 espere que T_4 libere seu bloqueio sobre *A*.
  - Essa situação é chamada de **impasse (*deadlock*)**
    - Para lidar com um impasse, um dentre T_3 ou T_4 precisa ser revertido e seus bloqueios liberados.

Armadilhas dos protocolos baseados em bloqueio

- O potencial para impasse existe na maioria dos protocolos de bloqueio. Os impasses são um mal necessário.
- **Inanição** (***Starvation***) também é possível se o gerenciador de controle de concorrência for mal projetado. Por exemplo:
  - Uma transação pode estar esperando por um bloqueio exclusivo sobre um item, enquanto uma seqüência de outras transações solicita e recebe um bloqueio compartilhado sobre o mesmo item.
  - A mesma transação é repetidamente revertida, devido aos impasses.
- O gerenciador de controle de concorrência pode ser projetado para impedir a inanição.

Tratamento de impasse

- **Considere as duas transações a seguir:**

  T_1: write (X) **T_2: write(Y)**

  write(Y) **write(X)**

- **Plano de execução com impasse**

T_1	T_2
lock-X on X	
write (X)	
	lock-X on Y
	write (Y)
	wait for **lock-X** on X
wait for **lock-X** on Y	

Tratamento de impasse

- O sistema está em impasse se houver um conjunto de transações de modo que cada transação no conjunto esteja esperando por outra transação no conjunto.
- **Protocolos de *prevenção de impasse*** garantem que o sistema *nunca* entrará em um estado de impasse. Algumas estratégias de prevenção:
  - Exigem que cada transação bloqueie todos os seus itens de dados antes de iniciar a execução (pré-declaração) → ex.: 2PL conservador é um protocolo *deadlock-free*
  - Impõem a ordenação parcial de todos os itens de dados e exigem que uma transação possa bloquear itens de dados somente na ordem especificada pela ordem parcial (protocolo baseado em gráfico).

Mais estratégias de prevenção de impasse

- Os seguintes esquemas usam **estampas de tempo** de transação para fins de prevenção de impasse apenas
- **Esquema esperar-morrer — não preemptivo**
  - a transação mais antiga pode esperar que a mais recente libere o item de dados. As transações mais recentes nunca esperam pelas mais antigas; em vez disso, elas são revertidas.
  - Se T_i solicita item de dados mantido por T_j, T_i tem permissão para esperar somente se tiver uma estampa de tempo menor do que o de T_j. Caso contrário, T_i é revertida
  - uma transação pode morrer várias vezes antes de adquirir o item de dados necessário

Mais estratégias de prevenção de impasse

- Os seguintes esquemas usam **estampas de tempo** de transação para fins de prevenção de impasse apenas
- **Esquema ferir-esperar — preemptivo**
  - a transação mais antiga *fere* (força o rollback) a transação mais nova em vez de esperar por ela. Transações mais novas podem esperar pelas mais antigas.
  - Se T_i solicita item de dados mantido por T_j, T_i tem permissão para esperar somente se tiver uma estampa de tempo maior do que o de T_j. Caso contrário, T_j é revertida
  - pode ter menos rollbacks que o esquema *esperar-morrer*.

Prevenção de impasse

- Nos esquemas *esperar-morrer* e *ferir-esperar*, uma transação revertida é reiniciada com sua estampa de tempo original. Transações mais antigas, assim, têm precedência em relação às mais novas, e a inanição é evitada
- **Esquemas baseados em tempo limite**:
  - uma transação só espera por um bloqueio por um certo tempo especificado. Depois disso, a espera esgota o tempo limite e a transação é revertida.
  - assim, os impasses não são possíveis
  - simples de implementar; mas a inanição é possível. Também difícil de determinar um bom valor do intervalo de tempo limite.

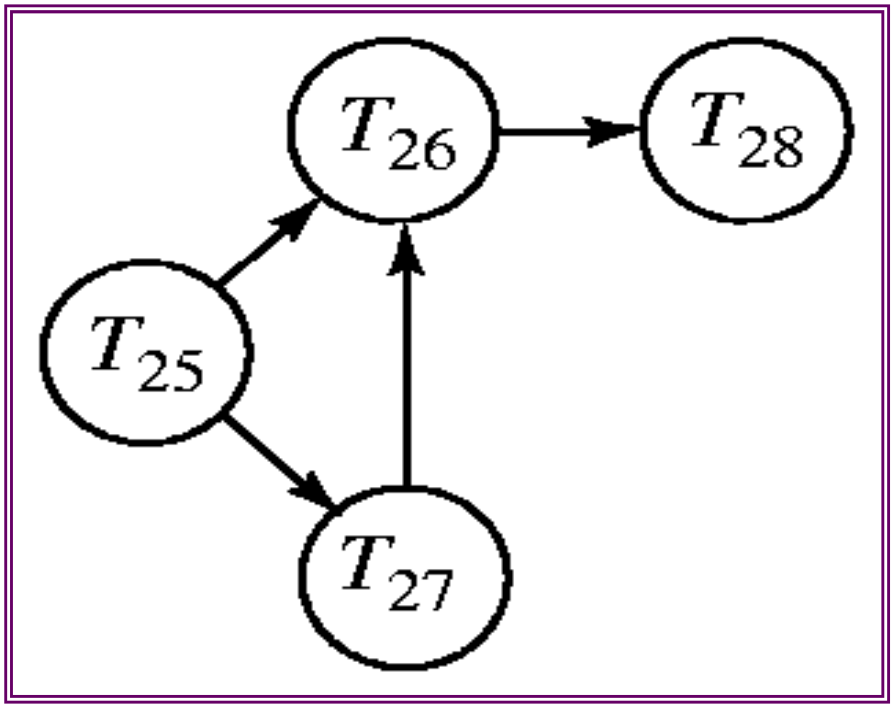
Detecção de impasse

- Abordagem mais prática para lidar com impasses
- O sistema verifica se um estado de deadlock realmente existe
- Solução atraente se soubermos que existem poucas interferências entre as transações
  - Transações raramente acessam os mesmos itens ao mesmo tempo
  - Transações curtas que bloqueiam poucos itens

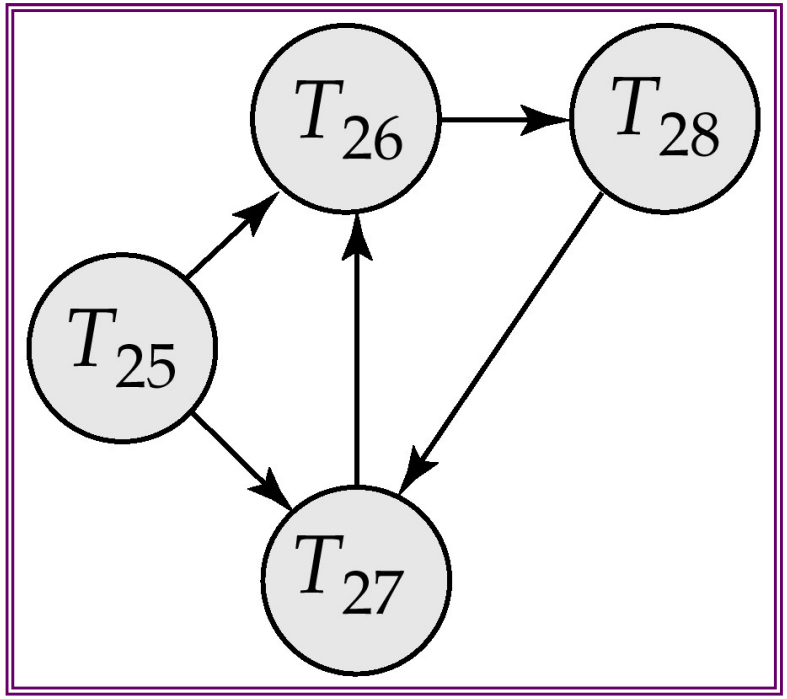
Detecção de impasse

- Os impasses podem ser descritos como um grafo de *espera*, que consiste em um par $G = (V,E)$,
  - V é um conjunto de vértices (todas as transações no sistema)
  - E é um conjunto de arestas; cada elemento é um par ordenado $T_i \rightarrow T_j$.
- Se $T_i \rightarrow T_j$ está em E, então existe uma aresta dedicada de T_i para T_j, implicando que T_i está esperando que T_j libere um item de dados.
- Quando T_i solicita um item de dados atualmente mantido por T_j, então a aresta T_i T_j é inserida no grafo de espera. Essa aresta só é removida quando T_j não está mais mantendo um item de dados necessário por T_i.
- O sistema está em um estado de impasse se e somente se o grafo de espera tiver um ciclo. Precisa invocar um algoritmo de detecção de impasse periodicamente para procurar ciclos.

Detecção de impasse



Grafo de espera sem um ciclo



Grafo de espera com um ciclo

Recuperação de impasse

- Quando o impasse for detectado:
  - Alguma transação terá que ser revertida (uma vítima) para romper o impasse. Selecione essa transação como a vítima que terá o menor custo.
  - Rollback - determine até onde reverter a transação
    - Rollback total: Aborte a transação e depois reinicie-a.
    - Mais eficiente reverter a transação somente até o ponto necessário para romper o impasse.
  - A **inanição** (*starvation*) acontece se a mesma transação sempre for escolhida como vítima. Inclua o número de rollbacks no fator de custo para evitar inanição.

Leitura complementar para casa

- Capítulo 18 do livro: Elmasri, Ramez; Navathe, Shamkant B. Sistemas de banco de dados.
- Capítulo 16 do livro: Silberschatz, A; Korth, H. F.; Sudarshan, S. Sistema de banco de dados.

Exercícios complementares

1. O que é o protocolo de bloqueio em duas fases? Como ele garante a serialização? Utilize exemplos para ilustrar o seu funcionamento.
2. Discuta os problemas de deadlock(impasse) e starvation (inanição) e as diferentes abordagens para a manipulação desses problemas. Dê exemplos.
3. Explique o protocolo de ordenação por timestamp para controle de concorrência. Dê exemplos para ilustrar o seu funcionamento.