



PUCPR
GRUPO MARISTA

Concorrência nas operações

Concorrência de transações

Em um sistema multiusuário, a probabilidade de acesso simultâneo aos mesmos dados é grande. Assim, a seguir, são mostrados os pontos mais importantes dessa teoria.

Técnicas de controle de concorrência

Controle de concorrência

- Assegurar a serialização de planos de execução usando protocolos.
- Executado automaticamente pelo gerenciador do banco de dados, de acordo com protocolos.

Protocolos

- Conjuntos de regras de controle de concorrência, que garantem a serialização no processamento de transações.

Técnicas de controle de concorrência

- Métodos baseados em bloqueio binário, compartilhado/exclusivo e em duas fases.
- Bloqueio em duas fases é usado por gerenciadores comerciais.

Protocolos

Protocolos de bloqueio de itens de dados

- Controlam o acesso ao banco de dados, bloqueando os dados que estão sendo usados e evitando que múltiplas transações acessem os itens de dados concorrentemente.

Protocolos com base em pré-ordenação (*timestamps*)

- Geram um identificador único (*timestamp*) associado às operações de leitura e escrita para cada transação, de forma que fiquem ordenadas, isto é, se T_i começa antes de T_j , então $T_s(T_i) < T_s(T_j)$.

Protocolos de validação ou otimistas

- Baseados no conceito de validação (certificação) de uma transação, efetuam testes de validação dos dados antes das operações de escrita.

Multiversão

- Utiliza múltiplas versões dos itens de dados (ambiente distribuído).

Técnica de bloqueio dos itens de dados

Bloqueio dos
itens de dados



Conjunto de itens de dados, como campo, registro, bloco do disco ou arquivo do banco de dados inteiro.

Objetivo: impedir que múltiplas transações accessem os itens concorrentemente.



SE a transação precisa de um item bloqueado,
ENTÃO é forçada a esperar até que o item seja liberado.

Técnica de bloqueio dos itens de dados – funcionamento

Bloqueio (**lock**)
do item de
dados



Variável associada ao item de dados, como campo, registro, bloco do disco ou arquivo do banco de dados inteiro.

Bloqueio para
cada item de
dados



Sincronizar o acesso por transações concorrentes aos itens do banco de dados.

Bloqueio binário

$\text{LOCK}(X) = 2 \text{ estados}$

X – item de dado

(1) Bloqueado
(2) Desbloqueado

- Um bloqueio é associado a um item X do banco de dados.
- Operações:
 - $\text{LOCK_ITEM}(X)$ e $\text{UNLOCK_ITEM}(X)$
- Se o **valor do bloqueio** de X é **1**, então X **não pode ser acessado** (leitura e escrita).
- Se o **valor do bloqueio** de X é **0**, então X **pode ser lido/escrito**.

Granularidade

Conjunto de itens de dados, que podem ser:

- Campo.
- Registro.
- Bloco do disco.
- Arquivo do banco de dados inteiro.

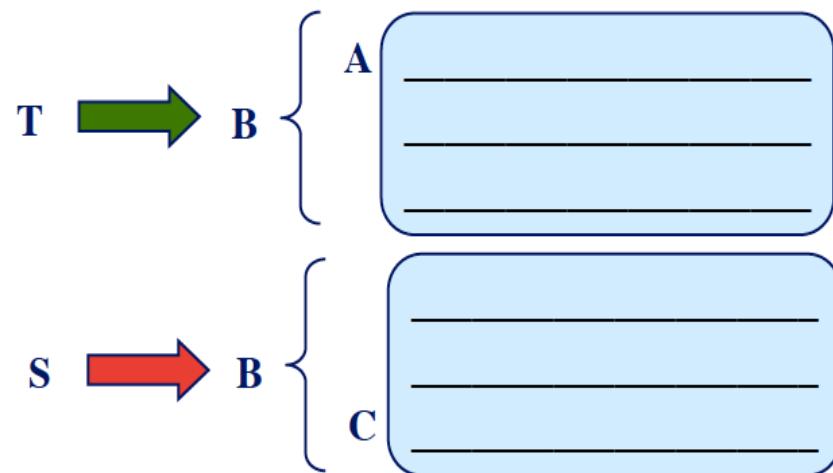
Definição da granularidade do bloqueio

- A escolha da granularidade (grossa ou fina) pode afetar a execução do controle de concorrência.

Granularidade – exemplo

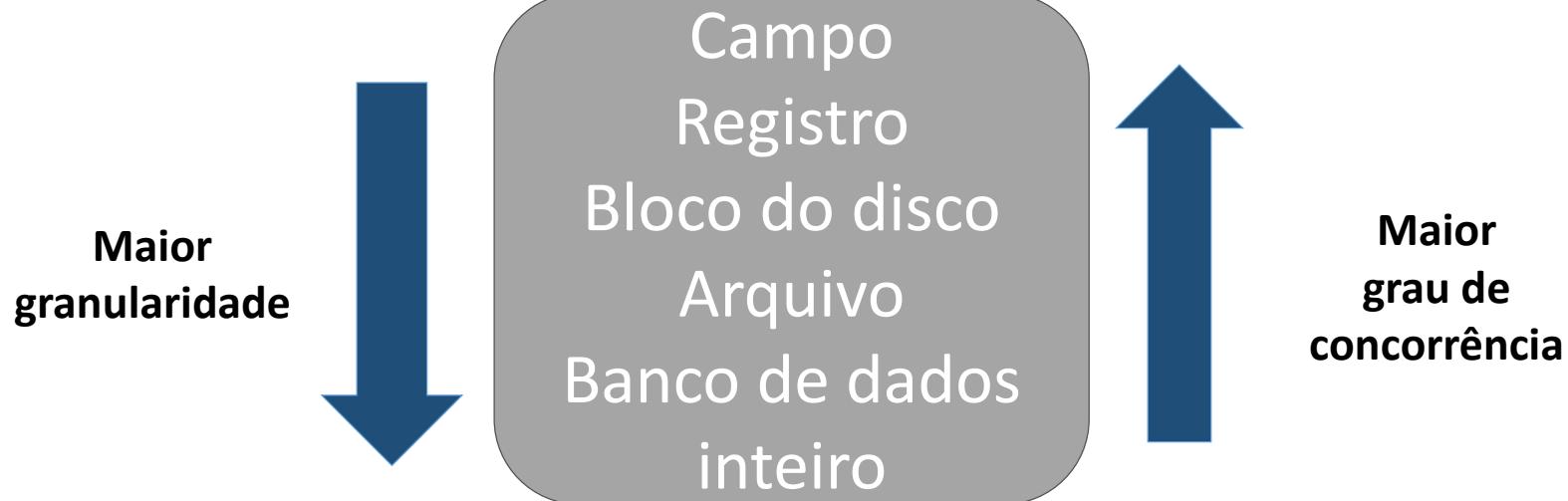
Tamanho do item de dados = bloco do disco

- (a) Se uma transação **T** precisa bloquear um **registro A**, deve bloquear todo o **bloco B**, que contém **A**.
- (b) Outra transação **S**, que quer bloquear um **registro C** armazenado no **bloco B**, é forçada a esperar.



Granularidade de item de dados

Quanto maior for o tamanho do item de dados,
mais baixo será o grau de concorrência permitido.



Bloqueio compartilhado/exclusivo

Bloqueio compartilhado

- Diversas transações poderão acessar o mesmo item X se todas tiverem o propósito de **leitura**.

Bloqueio exclusivo

- Quando uma transação altera (**escreve**) um item X.

Também chamado bloqueio de leitura/escrita.

Bloqueio compartilhado/exclusivo

$LOCK(X) = 3$ estados

X – item de dado

- (1) Bloqueio compartilhado
- (2) Bloqueio exclusivo
- (3) Desbloqueado

- Read_locked ou compartilhado (shared_lock)
 - Permite que outras operações leiam o item.
- Write_locked ou exclusivo (exclusive_lock)
 - Uma única transação controla o bloqueio no item.
- Unlocked ou desbloqueado

Operações

READ_LOCK_ITEM(X)
WRITE_LOCK_ITEM(X)
UNLOCK_ITEM(X)

Bloqueio compartilhado/exclusivo – exemplo

T1: Transação T₁

```
write_lock_item(X);
A=ler_item (X);
A = A - N;
escrever_item (X,A);
unlock_item(X);
write_lock_item(Y);
B=ler_item (Y);
B = B + N;
escrever_item (Y,B);
unlock_item(Y);
```

T2: Transação T₂

```
write_lock_item(X);
C=ler_item (X);
C = C + M;
escrever_item (X,C);
unlock_item(X);
```

Bloqueio em duas fases (*Two-Phase Locking – 2PL*)

Condição para que os bloqueios apresentados sejam serializáveis

- Todas as operações de bloqueio devem preceder a primeira operação de desbloqueio na transação.

Uma transação pode ser dividida em duas fases:

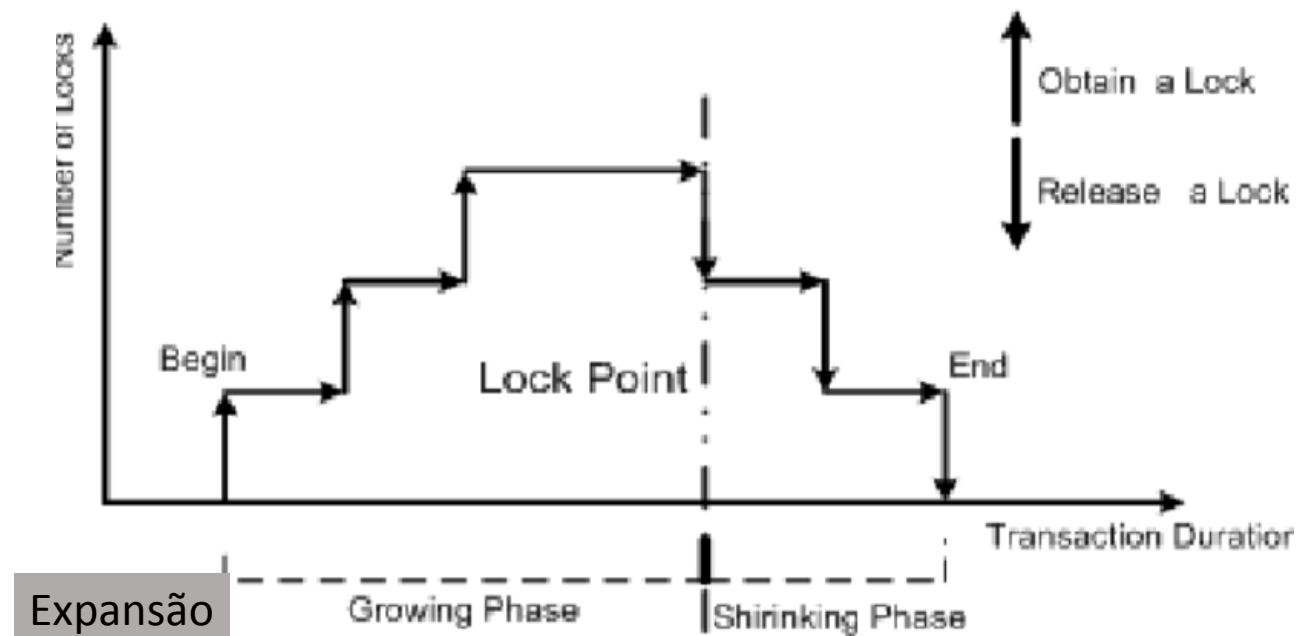
- Fase de expansão ou crescimento (1^a fase): quando os bloqueios são adquiridos, mas nenhum pode ser liberado.
- Fase de contração ou encolhimento (2^a fase): quando os bloqueios existentes são liberados, mas nenhum pode ser obtido.

Bloqueio em duas fases

Se a conversão de bloqueios é permitida, então:

As promoções de bloqueios vêm na fase de expansão.

Os rebaixamentos de bloqueios vêm na fase de contração.



Bloqueio em duas fases

Teorema:

- Se toda transação de um plano segue o bloqueio em duas fases, então o plano é serializável.
 - Não é necessário testar se o plano de execução é ou não serializável.

O bloqueio em duas fases limita a quantidade de concorrência em um plano de execuções.

- Exemplo: uma variável X já escrita deve esperar outras variáveis serem bloqueadas para depois ser desbloqueada.

Bloqueio em duas fases

Liberar o bloqueio logo após o uso:

- Aumenta a concorrência.
- Diminui a espera.
- Não garante isolamento.

Liberar no fim da transação:

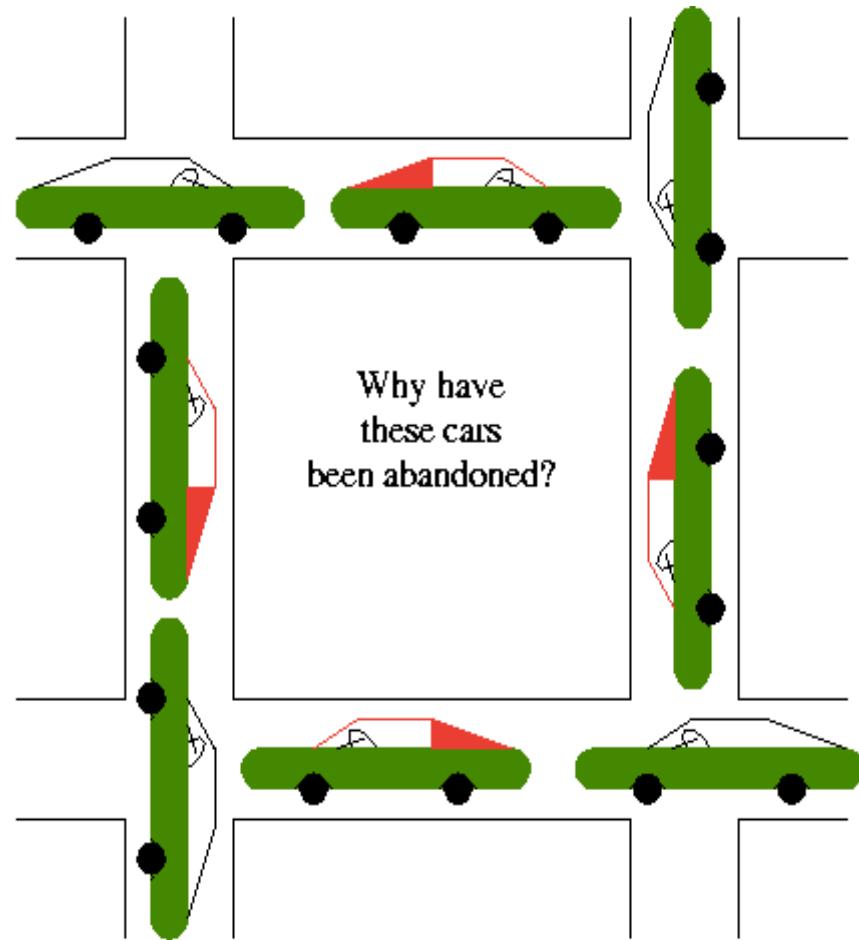
- Garante o isolamento.
- Aumenta a espera.

Deadlock:

- Ocorre quando há uma espera circular por itens bloqueados.

Deadlock

Qual é a saída?



Deadlock

Exemplo: T1 espera por T2 que espera por T1.

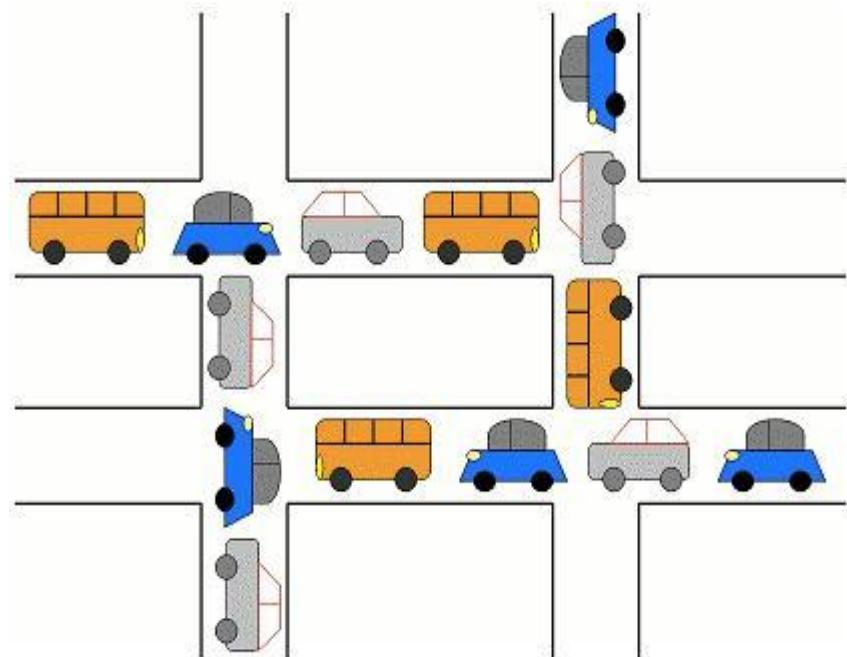
T1: Transação T₁
read_lock_item(Y);
ler_item (Y);

write_lock_item(X);

T2: Transação T₂

read_lock_item(X);
ler_item (X);

write_lock_item(Y);



Deadlock – técnicas para controle

Prevenção

- Evita os *deadlocks* antes que estes ocorram.
- Preferível se a probabilidade de ocorrerem *deadlocks* for muito alta.

Detecção e recuperação

- Não evitam os *deadlocks*, mas os detectam e impedem o bloqueio indefinido das transações envolvidas.
- Mais eficientes se ocorrerem poucos *deadlocks*.

Rollback

- Pode ser necessário, independentemente da técnica utilizada.

Deadlock – prevenção

2PL conservador: bloqueia todos os itens usados na transação antes de ela começar.

- Se algum item já está bloqueado, espera um pouco e tenta novamente.
- Limita a concorrência.

Ordenação de itens: ordena todos os itens do banco de dados e garante que as transações que os bloqueiam respeitem essa ordem.

- Se X vem antes de Y na ordem, Y só pode ser bloqueado se X está desbloqueado.
- Limita a concorrência e não é prático.

Com *timestamp* (marca de tempo): o *timestamp* de uma transação T, denotado por TS(T), é um número baseado na ordem em que T iniciou.

- Exemplo: se T1 começou antes que T2, então $TS(T1) < TS(T2)$.

Deadlock – detecção

Uma maneira mais prática de lidar com *deadlocks*, em vez de prevenir, é checar se já existe um *deadlock* e **desfazê-lo**.

Estratégia simples: grafos de espera.

- Um nodo é criado no grafo para cada transação.
- Se T_i espera o desbloqueio de X , que está bloqueado por T_j , então adicionamos a aresta direcionada ($T_i \rightarrow T_j$).
- Se T_j desbloqueia a variável, então retiramos a aresta do grafo.
- Se o grafo tem ciclo, então tem *deadlock*.

Deadlock – detecção

Outro método de detecção de *deadlocks* é o **timeout**.

- Cada transação tem um tempo máximo de execução.
- Após esse tempo, ela aborta (mesmo sem certeza de haver *deadlock*).
- Este método é simples e de baixa sobrecarga no SGBD.

Starvation

Ocorre quando uma transação não consegue continuar por um período indefinido de tempo, enquanto as outras são executadas normalmente.

Dá-se em esquemas em que a espera por itens bloqueados não é justa com todas as transações.

- Exemplo: quando uma transação que irá abortar é sempre escolhida para evitar *deadlock*.

Solução **FIFO**: alguma hora, a transação que entrar na fila será atendida.

Prioridades dinâmicas:

- Quanto mais tempo passa, maior é a prioridade. Alguma hora, a transação terá prioridade máxima e irá executar.
- Cada vez que uma transação é abortada, maior é a prioridade.

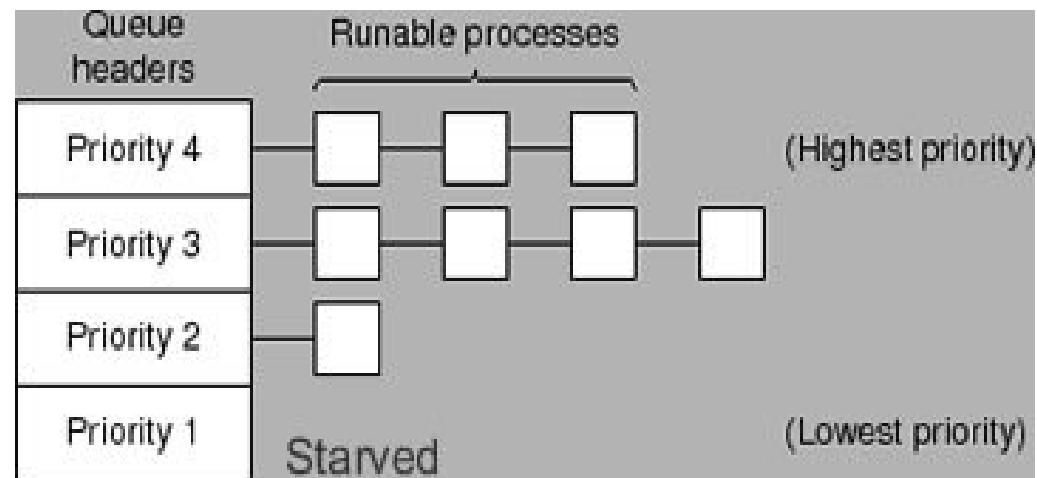
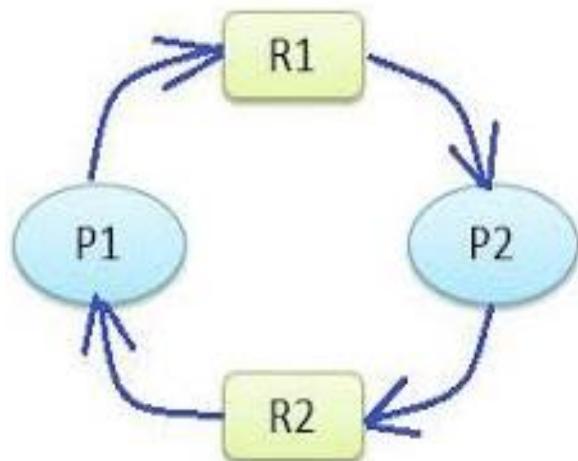
Starvation x Deadlock

Deadlock

- Espera circular por um recurso.

Starvation

- Não recebe prioridade, enquanto outros executam.





PUCPR

GRUPO MARISTA

Professor convidado
Mauro Augusto Borchrdt

© 2017 – PUCPR. Todos os direitos reservados.

Nenhum texto pode ser reproduzido sem prévia autorização.