Controle de Concorrência

Banco de Dados: Teoria e Prática

André Santanchè e Luiz Celso Gomes Jr
Instituto de Computação – UNICAMP
Setembro 2013

Picture by Ben Collins [http://www.flickr.com/photos/graylight/]

Controle de Concorrência

- Propósitos
  - Garantir a propriedade de isolamento em transações concorrentes
  - Preservar a consistência do banco pela preservação da consistência na execução das transações
  - Resolver conflitos leitura-gravação e gravação-gravação

Exercício 1

- Um sistema de gerenciamento arquivos deve definir a granularidade de acesso concorrente permitido.
  - São exemplos de opções de baixa granularidade: controle de acesso por disco e por diretório.
  - São exemplos de alta granularidade: controle por arquivo e por byte.
- Um SGBD também deve definir um nível de granularidade de acesso aos dados. Dê exemplos de opções de alta e baixa granularidade e mencione brevemente suas vantagens e desvantagens.

Bloqueio

- Um bloqueio (*lock*) é uma variável associada a um item de dados
  - Descreve a condição do item em relação às possíveis operações que podem ser aplicadas a ele
- Geralmente há um bloqueio para cada item no BD
- Tipos analisados:
  - bloqueio binário
  - bloqueios compartilhados/exclusivos

Bloqueio Binário

- Dois estados:
  - bloqueado (*locked*) – o item não pode ser acessado quando solicitado
  - desbloqueado (*unlocked*) – o item pode ser acessado quando solicitado
- Operações atômicas de bloqueio binário:
  - lock_item(X)
  - unlock_item(X)

Bloqueio Binário
Operação lock_item(X)

```
B: if (LOCK(X) = 0) then
      LOCK(X) ← 1
   else {
          wait (until LOCK(X)=0 and
                <the lock manager wakes up
                 the transaction>)
          goto B
    }
```

(Elmasri, 2010)

Bloqueio Binário

Operação unlock_item(X)

```
LOCK(X) ← 0
if <any transactions are waiting> then
    wake up one of the waiting the
        transactions;
```

(Elmasri, 2010)

Bloqueio Binário
Controle de Concorrência

- Para cada transação
  - lock(X)
    - antes de ler(X)/gravar(X)
    - se X ainda não tiver lock
  - unlock(X)
    - depois de todas as operações ler(X)/gravar(X)
    - apenas se tiver o lock de X

Limitações?

Exercício 2

- Qual o principal problema associado ao uso de bloqueio binário?

Limitações?

- Duas ações que apenas leem registros precisam se bloquear mutuamente?

Limitações?

- Duas ações que apenas leem registros precisam se bloquear mutuamente?
  - não

Bloqueio Compartilhado/Exclusivo

- Bloqueio compartilhado (*shared lock*)
  - utilizado para leitura (*read lock*)
  - mais de uma transação pode empregá-lo
  - impede a requisição de um bloqueio exclusivo
- Bloquei exclusivo (*exclusive lock*)
  - utilizado para gravação (*write lock*)
  - somente uma transação pode solicitá-lo
- Matriz de compatibilidade:

	shared	**exclusive**
shared	S	N
exclusive	N	N

Gerenciador de Bloqueio

- Gerencia o bloqueio de itens
- Mantém uma tabela de controle de bloqueio
  - Exemplo:
    - Controle(Transação, Item, Modo, Próximo_item)

(Elmasri, 2007)

Bloqueio Compartilhado/Exclusivo
Operação rlock(X)

```
B: if LOCK(X) = "unlocked" then {
       LOCK(X) ← "read-locked"
       no_of_reads(X) ← 1
   }
   else if LOCK(X)="read-locked" then
       no_of_reads(X)++
   else {
       wait (until LOCK(X) = "unlocked" and
             <the lock manager wakes up the
              transaction>)
        goto B
   }
```

(Elmasri, 2010)

Bloqueio Compartilhado/Exclusivo

Operação wlock(X)

```
B: if LOCK (X) = "unlocked" then
       LOCK (X) ← "write-locked"
   else {
       wait (until LOCK(X) = "unlocked" and
             <the lock manager wakes up the
              transaction>)
       goto B
   }
```

(Elmasri, 2010)

Bloqueio Compartilhado/Exclusivo
Operação unlock(X)

```
if LOCK(X) = "write-locked" then {
    LOCK(X) ← "unlocked"
    wakeup up one of the transactions,
       if any
}
else if LOCK(X) = "read-locked" then {
    no_of_reads(X)--
    if no_of_reads(X) = 0 then {
       LOCK (X) ← "unlocked";
       wakeup up one of the transactions,
          if any
    }
}
```

(Elmasri, 2010)

Bloqueio Compartilhado/Exclusivo
Controle de Concorrência

- Para cada transação
  - rlock(X)
    - antes de read(X)
    - se ainda não tiver rlock ou wlock
  - wlock(X)
    - antes de read(X) com intenção de write(X)
    - antes de write(X)
    - se ainda não tiver wlock
  - unlock(X)
    - depois de todas as operações ler(X)/gravar(X)
    - apenas se tiver o lock de X

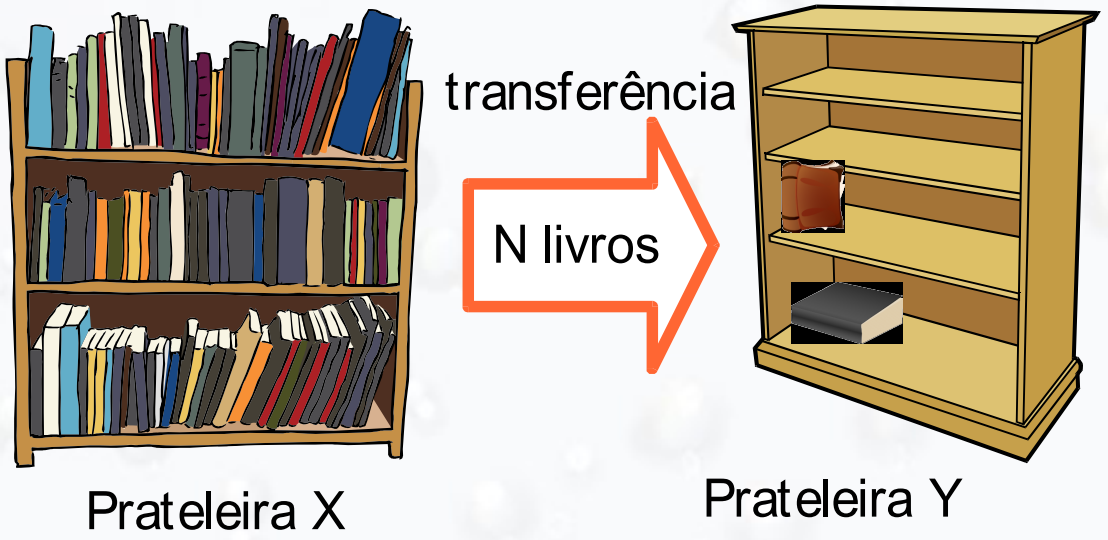
Bloqueio Compartilhado/Exclusivo
Upgrade e Downgrade

- Lock Upgrade
  - rlock(x) → wlock(x)
    - condição: não há outro rlock em X
- Lock Downgrade
  - wlock(x) → rlock(x)

Garantindo a Serialização
Transação 1: Transferência

T1

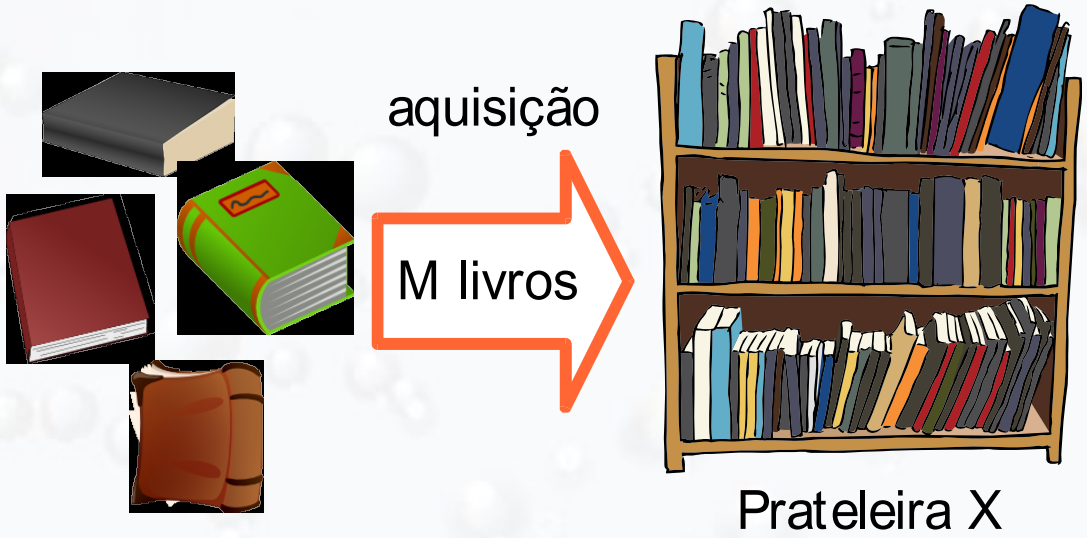
```
ler(X)
X = X - N
gravar(X)
ler(Y)
Y = Y + N
gravar(Y)
```



Garantindo a Serialização
Transação 2: Aquisição

T2

```
ler(X)
X = X + M
gravar(X)
```

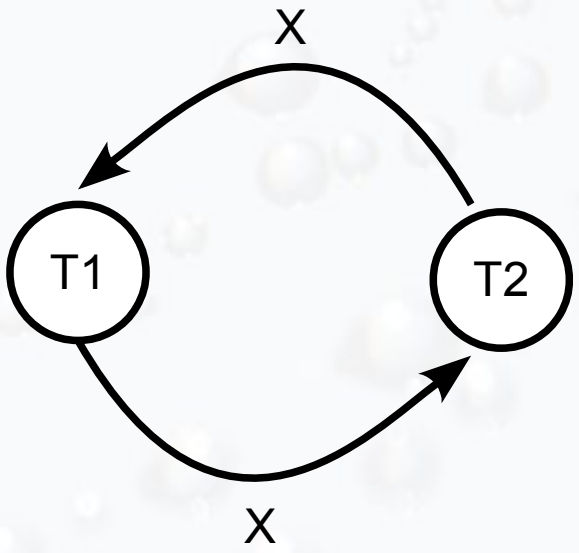


Garantindo a Serialização
Plano de Execução

T1	T2
ler(X)	**ler**(X)
X = X - N	X = X + M
gravar(X)	**gravar**(X)
ler(Y)	
Y = Y + N	
gravar(Y)	

Não Serializável

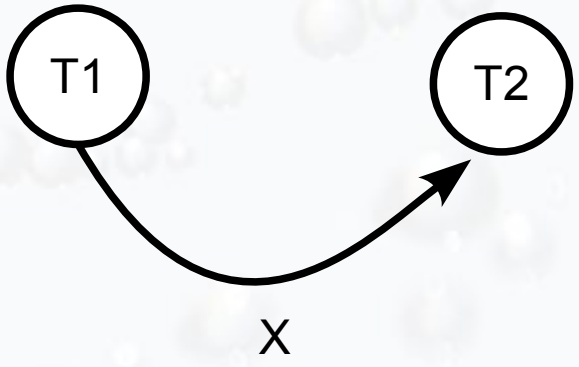
T1	T2
ler(X)	
X = X - N	
	ler(X)
	X = X + M
gravar(X)	
ler(Y)	
	gravar(X)
Y = Y + N	
gravar(Y)	



Plano Não Serializável

Serializável

T1	T2
ler(X)	
X = X - N	
gravar(X)	
	ler(X)
	X = X + M
	gravar(X)
ler(Y)	
Y = Y + N	
gravar(Y)	



Plano Serializável

Protocolo Two-phase (2PL) Locking

- Garante serialização
- Protocolo de bloqueio de duas fases
  - **Fase de crescimento:** Transação pode obter bloqueios, mas não pode liberar
  - **Fase de encolhimento:** Transação pode liberar bloqueios, mas não pode obter

2PL
Diagrama

Transações com Locks

T1	T2
wlock(X)	**wlock**(X)
ler(X)	**ler**(X)
X = X - N	X = X + M
gravar(X)	**gravar**(X)
unlock(X)	**unlock**(X)
wlock(Y)	
ler(Y)	
Y = Y + N	
gravar(Y)	
unlock(Y)	

Plano 2PL

T1	T2
wlock(X)	
ler(X)	
X = X - N	
gravar(X)	
wlock(Y)	
unlock(X)	
	wlock(X)
	ler(X)
	X = X + M
	gravar(X)
	unlock(X)
ler(Y)	
Y = Y + N	
gravar(Y)	
unlock(Y)	

Limites do 2PL

Limites do 2PL
Exemplo

- Cálculo de livros necessários de acordo com a média

T1
ler(X) Q = Media - X **gravar**(Q)

Limites do 2PL
Exemplo

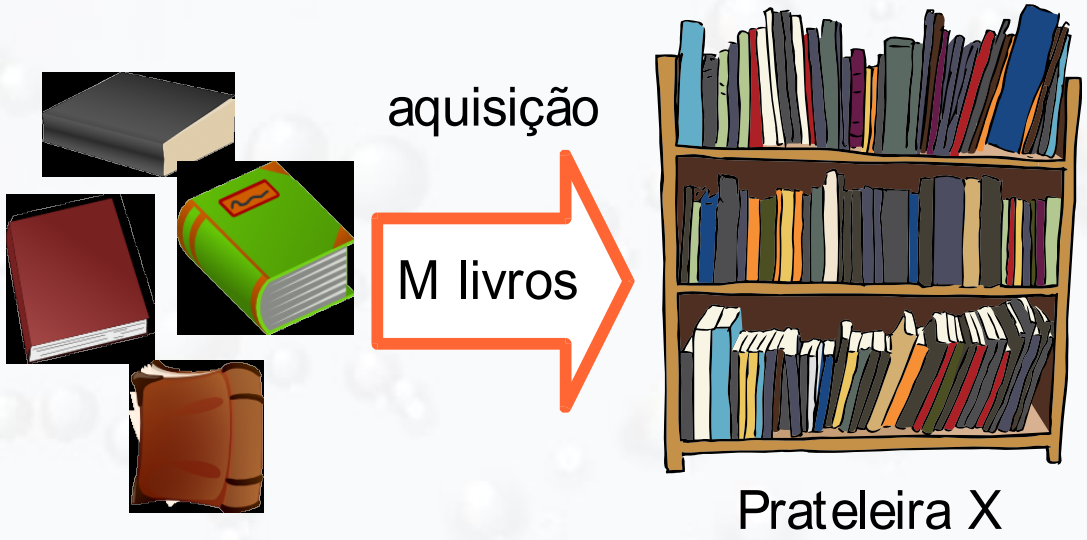
- Com locks

T1
rlock(X) **ler**(X) **unlock**(X) Q = Media - X **wlock**(Q) **gravar**(Q) **unlock**(Q)

Limites do 2PL
Exemplo

- Aquisição de Q livros

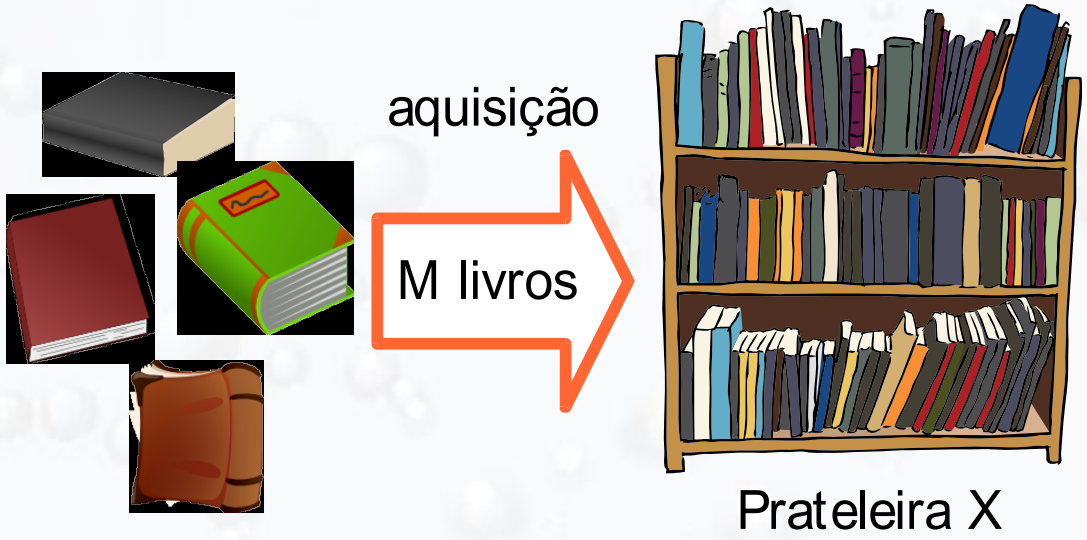
T2
ler(Q)
ler(X)
X = X + Q
gravar(X)



Limites do 2PL
Exemplo

- Com locks

T2
rlock(Q) **ler**(Q) **unlock**(Q) **wlock**(X) **ler**(X) X = X + Q **gravar**(X) **unlock**(X)



Limites do 2PL
Exemplo

T1	T2
	rlock(Q) **ler**(Q)

Limites do 2PL
Exemplo

T1	T2
	rlock(Q) **ler**(Q)
rlock(X) **ler**(X)	

Limites do 2PL
Exemplo

T1	T2
	rlock(Q)
	ler(Q)
rlock(X)	
ler(X)	
	wlock(X)
	** espera **

Limites do 2PL
Exemplo

- Pode haver problemas no 2PL?

T1	T2
	rlock(Q)
	ler(Q)
rlock(X)	
ler(X)	
	wlock(X)
	** espera **
wlock(Q)	
** espera **	

Deadlock

- Impasse
- “Ciclo de transações esperando mutuamente pela liberação de locks.”

Tradução livre (Ramakrishnan, 2003)

Grafo de Espera

Tratando Deadlocks

- Prevenção de deadlock
- Detecção de deadlock

2PL
Conservador ou Estático

- Bloqueia todos os itens a ser lidos/gravados antes de iniciar a transação
- Livre de deadlock
- Exige pré-declaração (leituras/gravações) no início da transação

2PL Conservador ou Estático
Diagrama

2PL
Estrito e Rigoroso

- 2PL Estrito
  - Não libera wlocks até o commit ou abort
  - Garante schedule estrito
    - T só lê e/ou grava valores que foram alterados por transações que já realizaram commit
- 2PL Rigoroso
  - Não libera rlocks/wlocks até o commit ou abort
  - Mais fácil de implementar que o Estrito

2PL Estrito e Rigoroso
Diagrama

Exercício 3

- Considere as seguintes transações:
  - T1 = r1(x), w1(y)
  - T2 = r2(x), r2(y), w2(x)

a) Encontre um plano de execução intercalado que poderia ser gerado por um algoritmo 2PL (com upgrade de locks).

b) Desenhe o grafo de espera para o plano encontrado em (a).

Prevenção de Deadlock

Rótulo de Tempo

- Transações com timestamps
- Considere Ti quer lock(X) e Tj tem lock(X)
- stamp(Ti) < stamp(Tj) (maior prioridade)
- Políticas:
  - Wait-die (Esperar-morrer): Ti espera por Tj; senão Ti aborta
  - Wound-wait (Ferir-esperar): Tj aborta; senão Ti espera

Detecção de Deadlock

- Atualiza e verifica grafo de espera
  - Aborta uma das transação em deadlock
  - Algoritmo de seleção da vítima – evitar transações executadas há muito tempo
- Timeout

Starvation

- Transação não pode prosseguir por um período indefinido (Elmasri, 2010)
- Soluções
  - Primeiro a chegar, primeiro a ser atendido
  - Prioridade aumenta com a espera

André Santanchè
http://www.ic.unicamp.br/~santanche

Referências

- Elmasri, Ramez; Navathe, Shamkant B. (2005) **Sistemas de Bancos de Dados**. Addison-Wesley, 4ª edição em português.
- Elmasri, Ramez; Navathe, Shamkant B. (2010) **Sistemas de Banco de Dados**. Pearson, 6ª edição em português.
- Ramakrishnan, Raghu; Gehrke, Johannes (2003) **Database Management Systems.** McGraw-Hill, 3rd edition.
- Ramakrishnan, Raghu; Gehrke, Johannes (2003b) Database Management Systems. McGraw-Hill, 3rd edition (companion slides).

Licença

- Agradecimentos: fotografia da capa e fundo por Ben Collins - http://www.flickr.com/photos/graylight/.
  Ver licença específica em
  http://www.flickr.com/photos/graylight/261480919/