Aspectos Operacionais: Controle de Concorrência

Valéria Times

Controle de Concorrência

- Definição

  Concorrência é a propriedade de uma transação ser executada simultaneamente com outras transações

Controle de Concorrência

- Justificativa de uso

  Com a execução de várias transações ao mesmo tempo, o processador pode ser compartilhado entre essas transações, melhorando a eficiência global do computador dado que uma maior quantidade de trabalho é executada em menos tempo

Controle de Concorrência

- Justificativa de um controle

  É necessário que o sistema monitore a interação entre transações concorrentes, de modo a evitar que elas destruam a consistência do BD

Controle de Concorrência

- Técnicas de Bloqueio (Locks)
- LOCK
  - Variável associada a um item de dados no BD que descreve o *status* desse item com respeito a possíveis operações a serem aplicadas a ele
- LOCK BINÁRIO
  - Dois estados ou valores: locked (1) ou unlocked (0)

Controle de Concorrência

- LOCK BINÁRIO (Cont.)
  - Um lock distinto é associado a cada item do BD - referenciado como lock (x) para o item x
  - Operações incluídas nas transações: lock_item e unlock_item, implementadas como operações indivisíveis
  - Um gerenciador é mantido pelo SGBD para registrar e controlar o acesso a *locks*

Controle de Concorrência

- LOCK BINÁRIO (continuação)
  - Registro de lock:

    <nome-do-item, LOCK, trans-ID>
  - Tabela de locks: mantém esses registros
  - Regras:
    - Uma transação T tem que executar uma operação lock_item(x) antes de qualquer leitura ou gravação executada por T

Controle de Concorrência

- Regras (Cont.)
  - Uma transação T tem que executar a operação unlock_item(x) após todas leituras/gravações completadas em T
  - Uma transação não pode executar outra operação lock_item(x) se já tem um bloqueio sobre x

Controle de Concorrência

- Regras (Cont.)
  - Uma transação T não pode executar um unlock_item(x) a menos que tenha um bloqueio sobre x
  - Apenas uma transação pode ter um bloqueio sobre um dado item

Controle de Concorrência

- LOCK DE MODO MÚLTIPLO
  - Três operações (indivisíveis)
    - *read_lock(x)* - lock compartilhado
    - *write_lock(x)* - lock exclusivo
    - *unlock(x)*
  - Registro de Lock

    <nome-do-item, LOCK, número-de-leituras, ID>

Controle de Concorrência

- Regras:
  - Uma transação T tem que executar uma operação *read_lock(x)* ou *write_lock(x)* antes de qualquer *read-item(x)* em T
  - Uma transação T tem que executar uma operação *write_lock(x)* antes de qualquer *write-item(x)* em T
  - Uma transação tem que executar uma operação *unlock(x)* após todas as operações *read_item(x)* e *write_item(x)* completadas em T

Controle de Concorrência

- Regras (Cont.)
  - Uma transação T não executará um *read_lock(x)* se já tem um bloqueio compartilhado ou um bloqueio exclusivo em **x** (pode ser relaxado)
  - Uma transação T não executará outro *write_lock(x)* se já tem um bloqueio exclusivo em **x** ou um bloqueio compartilhado (pode ser promovido)
  - Uma transação T não executará um *unlock(x)* a menos que já tenha um bloqueio compartilhado ou exclusivo em **x**.

Controle de Concorrência

- INCREMENTO DE LOCK

  Uma transação após ter um read_lock(x) e sendo a única que detém x, pode posteriormente executar um write_lock(x) sobre x
- DECREMENTO DE LOCK

  Após ter um bloqueio exclusivo sobre um item x, uma transação T pode decrementar o lock, executando um read_lock(x)

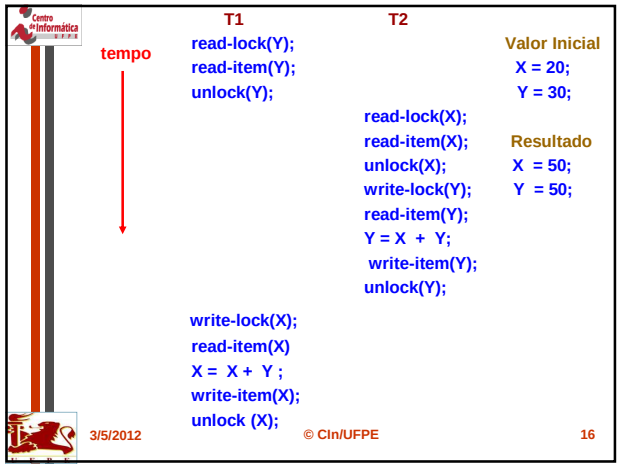
Controle de Concorrência

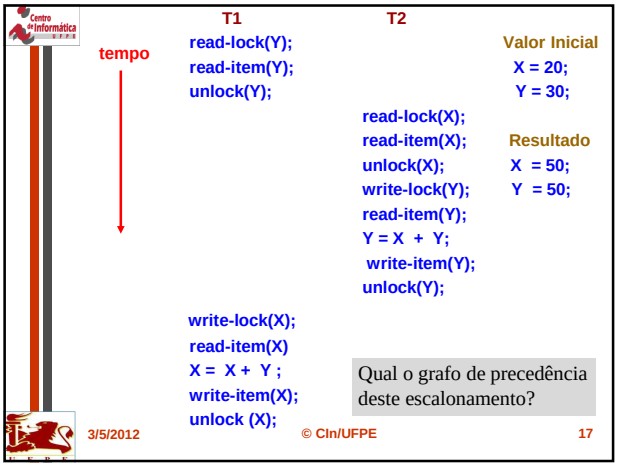
- Locks binários ou múltiplos não garantem a serializabilidade de escalonamentos nos quais as transações participam.
- É necessário um protocolo adicional que contemple o posicionamento de *locks* e *unlocks* em cada transação.

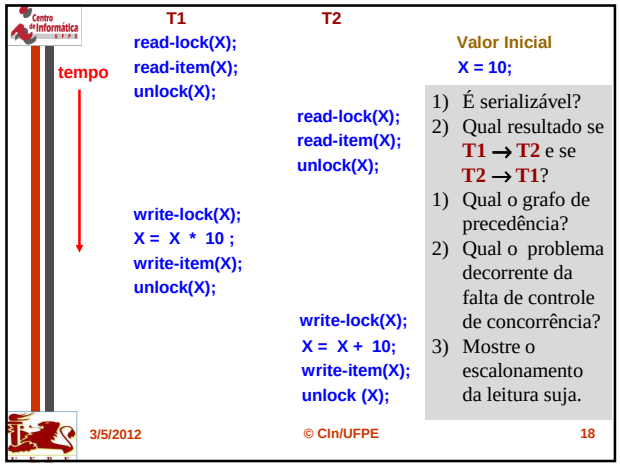
Controle de Concorrência

- EXEMPLO:

T1	T2	Valores Iniciais
read-lock(Y);	read-lock(X);	X = 20; Y = 30
read-item(Y);	read-item(X);	
unlock(Y);	unlock(X)	T1 → T2
write-lock(X);	write-lock(Y);	X = 50, Y=80
read-item(X);	read-item(Y);	
X = X + Y;	Y = X + Y;	T2 → T1
write-item(X);	write-item(Y)	X = 70, Y=50
unlock(X);	unlock(Y);	







Controle de Concorrência

- BLOQUEIOS EXCLUSIVOS

  Se uma transação T contiver um bloqueio exclusivo em algum objeto, então nenhuma transação distinta T' pode fazer bloqueio daquele objeto até que T libere o seu bloqueio

Controle de Concorrência

- BLOQUEIOS COMPARTILHADOS

  Se uma transação T mantiver um bloqueio compartilhado em algum objeto, então uma transação T' distinta também pode fazer um bloqueio compartilhado sobre aquele objeto.

  Mas, nenhuma transação T' distinta pode fazer um bloqueio exclusivo sobre aquele objeto até que todos os bloqueios compartilhados do objeto tenham sido liberados.

Controle de Concorrência

T' \ T	Excl	Comp	–
Excl	N	N	C
Comp	N	C	C
–	C	C	C

Controle de Concorrência

- Protocolo

  Qualquer transação que pretenda atualizar um registro R, primeiro terá que acessar aquele registro e bloqueá-lo exclusivamente. Se o bloqueio não puder ser feito, a transação entra em um estado de espera e apenas continuará o processamento quando o registro se tornar disponível e o bloqueio puder ser concedido

Controle de Concorrência

- Protocolo (Cont.)
  - O sistema terá que garantir que uma transação forçada a esperar por causa desse protocolo, eventualmente sairá do estado de espera
  - Para evitar a possibilidade de perda de atualizações, nenhuma transação poderá ter acesso aos dados de uma atualização não efetivada

Controle de Concorrência

- BLOQUEIOS DE DUAS FASES

  Se todas as transações obedecerem às seguintes regras:

  - Antes de operar sobre qualquer objeto, a transação primeiro adquire um bloqueio sobre aquele objeto;
  - Após liberar o bloqueio, a transação não adquire mais bloqueios.

Controle de Concorrência

- Todas as execuções intercaladas dessas transações são serializáveis.
- Uma transação que obedeça as regras 1 e 2 anteriores é dito que satisfaz o bloqueio de duas fases.
- Fases:
  - Crescente:quando os bloqueios são feitos
  - Encolhimento: quando os bloqueios são liberados

Controle de Concorrência

- Se o decréscimo de lock for permitido, ele tem que ser feito na fase de encolhimento.
- Pode limitar o acesso concorrente

  Uma transação tem que manter um bloqueio sobre **x** mesmo que não precise dele, se tiver que exercer um bloqueio sobre outro objeto **y**. Por outro lado, para poder liberar **x** mais cedo, a transação terá que bloquear **y** mais cedo.

Controle de Concorrência

- BLOQUEIOS DE DUAS FASES (cont.)
  - Variação do protocolo:
    - BÁSICO (visto anteriormente)
    - CONSERVADOR OU ESTÁTICO: bloqueia todos os itens que acessará, antes de iniciar a execução da transação
    - ESTRITO (mais popular): não libera os bloqueios até o commit ou abort
  - Podem causar Deadlock ou Livelock ou Starvation

Controle de Concorrência

- DEADLOCK

  É uma situação em que duas ou mais transações estão em estado simultâneo de espera, cada uma aguardando que uma das demais libere um bloqueio para ela poder prosseguir.
- Principais métodos para solucionar o impasse podem resultar na escolha repetida da transação para:
  - Permanecer em estado de espera (livelock)
  - Ser abortada e reiniciada (starvation)
  - Solução: prevenir o impasse

Controle de Concorrência

- LIVELOCK
  - A transação não pode prosseguir por um período indefinido de tempo enquanto outras transações continuam normalmente
  - Esquema injusto de espera por bloqueio
  - Soluções possíveis
    - uso de uma fila (esquema FIFO)
    - incremento de prioridade com o tempo

Controle de Concorrência

- STARVATION

  Problema similar ao livelock, ocorre se o algoritmo seleciona a mesma transação como vítima repetidamente, causando cancelamentos repetidos e nunca acabando a execução

Controle de Concorrência

- Principais Métodos de Prevenção de Deadlock
  - Tipos de Protocolos
    - Conservador: se algum dos itens não pode ser bloqueado, nenhum item será bloqueado
    - Ordenado: tenta por uma ordenação em todas as transações e os bloqueios só podem ocorrer segundo esta ordem.

Controle de Concorrência

- Técnicas Baseadas em Ordenação (*Timestamping*)
  - Identificador único assinalado a cada transação
  - Ordenados segundo a ordem em que as transações começaram
  - Principal vantagem: não se baseiam em bloqueios, logo *deadlock* é impossível

Controle de Concorrência

- Protocolos

  Quando uma transação solicita o acesso de um registro que já está em uso por outra transação, então um dos dois procedimentos é seguido:
  - Wait-die - se a transação que solicitou o acesso é a mais antiga, pode aguardar. Se for a mais nova, sofre rollback e recomeça mais tarde com mesmo timestamping.

Controle de Concorrência

- Protocolos (Cont.)
  - Wound-wait - se for a transação mais nova, pode aguardar. Se for a mais antiga, interrompe a mais nova, a qual sofre rollback. Recomeça mais tarde com mesmo timestamping.
- Os protocolos wait-die e wound-wait apresentados evitam o deadlock

Controle de Concorrência

- Outro controle de concorrência sem bloqueios
  - TIMESTAMPS

    Criados pelo SGBD para identificar uma transação:
    - São assinalados na ordem em que as transações são submetidas ao sistema [TS(T)]
    - Geração: contador ou clock do sistema

Controle de Concorrência

- Ordenação de transações baseadas em Timestamp:
  - executam livremente
  - asseguram que para cada item acessado por mais de uma transação, a ordem dos acessos não viola a serializabilidade do escalonamento.

Controle de Concorrência

Ordenação de transações baseadas em Timestamp (cont.):

- Para tanto são associados dois timestamps para cada item do BD
  - read_TS(x) - timestamp de leitura de x
    O maior timestamp de todas as transações que leram x com sucesso.
  - write_TS(x) - timestamp de escrita de x
    O maior timestamp de todas as transações que atualizaram x

Controle de Concorrência

ORDENAÇÃO (Cont.)

- Para uma transação realizar uma leitura ou gravação em x, o timestamp de T é comparado com o read_TS ou write_TS de x para garantir a ordenação da execução.

Controle de Concorrência

- Algoritmo para write

a . Se read_TS(x) > TS(T)
/* transação mais nova leu antes desta atualizar */
Então: abortar T e dar rollback em T

b. Se write_TS(x) > TS(T)
Então: não executar a gravação mas prosseguir o processamento para não perder a última atualização feita

c . Não ocorrendo "a" nem "b"
write_item(x) e write_TS(x) = TS(T)

Controle de Concorrência

- Algoritmo para read

a . Se write_TS(x) > TS(T)
/* transação mais nova escreveu o item antes desta ler */
Então: abortar T e dar rollback em T

b. Se write_TS(x) < = TS(T)
Então: read_item(x) e
read_TS(x) = maior(TS(T),atual read_TS(x))

Controle de Concorrência

- OBS: Starvation pode ocorrer se T é continuamente abortada e recomeçada (Recomeço Cíclico)

Controle de Concorrência

- Protocolo de Detecção e Recuperação
  - Verificação da existência de deadlock:
    - sempre que uma solicitação de bloqueio causa "espera"
      - permite detecção imediata do impasse
      - acarreta um overhead
    - em alguma base periódica
      - reduz o overhead
      - alguns deadlocks serão detectados tardiamente

Controle de Concorrência

- Quando Detectar e Recuperar?
  - Condições ideais
    - Pequena interferência entre as transações
    - Transações curtas
    - Bloqueios de poucos itens
    - Carga da transação leve.

Controle de Concorrência

- Quebra de deadlock consiste na escolha de uma das transações para forçá-la a um rollback:
  - a que foi iniciada mais recentemente
  - a que tiver feito o menor número de bloqueios
  - a que tiver feito o menor número de atualizações
- Situação oposta:
  - usar um esquema de prevenção de deadlock.

Controle de Concorrência

Outros Métodos

- Técnicas de controle de concorrência com multi-versão
  - Várias versões X1, X2,..., Xn de cada item de dado X são guardadas pelo sistema
    - Baseada na ordem dos timestamping
    - Bloqueio de duas fases

Controle de Concorrência

- Técnica de Validação (otimista)
  - As atualizações efetuadas não são efetivadas no BD até o final da transação
    - Fase de leitura: leitura e atualizações não efetivadas
    - Fase de validação: verifica se tudo ocorreu em ordem
    - Fase de escrita: se validação OK, atualiza o BD

Controle de Concorrência

- O ambiente de BD difere de outros ambientes no que se refere aos problemas de bloqueio devido a:
  - O conjunto de objetos bloqueáveis não só é muito grande como também muda muito rápido
  - Objetos bloqueáveis são tipicamente endereçados pelo conteúdo
  - O escopo preciso do bloqueio para determinada transação normalmente é determinado com muita rapidez.

Controle de Concorrência

Granularidade de Bloqueio no BD

< concorrência ↓ **campo específico, registro, arquivo, BD** ↑ **> número de bloqueios**

Quanto maior número de *bloqueios* a ser gerenciado:
- **maior *overhead***
- **mais espaço para a tabela de *bloqueios***

OBS: **1. os bloqueios em todo o BD são liberados no final do programa e não no *commit***
2. os bloqueios de atualização só fazem sentido na granularidade mais fina