TÉCNICAS DE CONTROLE DE CONCORRÊNCIA - parte 2 -

Profa. Dra. Maria Madalena Dias

TÉCNICAS DE CONTROLE DE CONCORRÊNCIA - parte 2 -

- Protocolo *Two Phase Locking* (2PL)
- Variações do Protocolo 2PL
- Problemas Adicionais
- Tratamento de *Deadlock*
- *Livelock*
- *Starvation*
- Granularidade dos Itens de Dados

Protocolo Two Phase Locking (2PL)

- Indica o posicionamento das operações de *lock* e *unlock* em cada transação
- Uma transação T segue o 2PL se
  - todas as operações *read_lock* e *write_lock* de T precedem a primeira operação de *unlock* de T
  - todas as travas requisitadas por uma transação devem ser obtidas antes que qualquer uma delas seja liberada

Protocolo Two Phase Locking (2PL)

- Uma transação que segue 2PL tem duas fases:
  - **fase de crescimento ou expansão**
    - a transação apenas pode requisitar novas travas, mas não pode liberá-las
  - **fase de encolhimento**
    - a transação apenas pode liberar suas travas, mas não pode requisitar novas travas
- Garante escalonamentos serializáveis
  - de acordo com a equivalência de conflito

Protocolo Two Phase Locking (2PL)

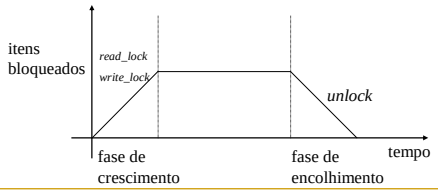
- Limita concorrência
  - uma transação T pode não ser capaz de liberar a trava associada ao item de dado x depois que x foi utilizado se T deve obter a trava de um outro item de dado y posteriormente
  - qualquer outra transação tentando acessar x será obrigada a esperar, mesmo que T já tenha utilizado x

Protocolo Two Phase Locking (2PL)

- Limita concorrência
  - uma transação T deveria obter a trava associada ao item de dado y antes de sua utilização para poder liberar a trava associada a x mais rapidamente
  - se a trava de y é obtida antes do necessário, qualquer outra transação tentando utilizar y é forçada a esperar, mesmo que T não esteja utilizando y

Variações do Protocolo 2PL

- 2PL Básico
  - características apresentadas anteriormente
  - pode causar *deadlock*

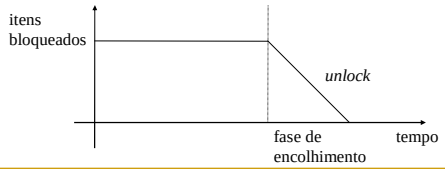


Variações do Protocolo 2PL

- 2PL Conservativo ou Estático
  - uma transação tem que requisitar todas as travas associadas aos dados a serem acessados por suas operações antes de iniciar sua execução
  - pré-declaração do conjunto de operações *read* e *write* das transações
  - se uma das travas solicitadas não puder ser obtida então nenhuma trava é realizada, todas as travas já obtidas são liberadas, a transação deve esperar e tentar requisitar todas as travas novamente

Variações do Protocolo 2PL

- 2PL Conservativo ou Estático
  - não causa *deadlock*
  - limita a concorrência

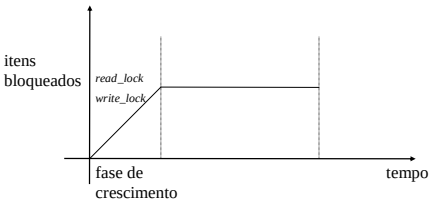


Variações do Protocolo 2PL

- 2PL Restrito ou Dinâmico
  - variação mais popular do protocolo 2PL
  - uma transação T não pode liberar qualquer trava até que seja finalizada (com sucesso ou sem sucesso)
  - nenhuma outra transação pode ler ou escrever um item de dado que é escrito por T até que T tenha finalizado
  - pode causar *deadlock*
  - não causa *deadlock* se associada ao 2PL conservativo

Variações do Protocolo 2PL

- 2PL Restrito ou Dinâmico



Problemas Adicionais

- Protoloco 2PL
  - garante a serialização dos escalonamentos
  - introduz dois problemas adicionais
    - *deadlock*
    - *livelock*
  - ***Deadlock***
    - quando uma transação T_1 espera por uma trava sendo utilizada por T_2, T_2 espera por uma trava sendo utilizada por T_n e T_n espera por uma trava sendo utilizada por T_1

Exercício

- As transações a seguir estão em *deadlock*

transação 1	**transação 2**
read_lock (y)	
read_item (y)	
	read_lock (x)
	read_item (x)
write_lock (x)	
	write_lock (y)

13

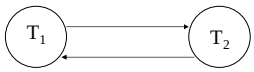
Tratamento de *Deadlock*

- Protocolos de prevenção
  - 2PL conservativo
  - utilização de *timestamp* associado a transações
    - **wait_die**
      - se TS(Ti) < TS(Tj) (Ti é mais velho do que Tj)
      - então Ti espera
      - senão abortar Ti e reiniciá-lo mais tarde
    - **wound_wait**
      - se TS(Ti) < TS(Tj) (Ti é mais velho do que Tj)
      - então abortar Tj e reiniciá-lo mais tarde
      - senão Ti espera

14

Tratamento de *Deadlock*

- Protocolos de detecção
  - verificam periodicamente o sistema para ver se este se encontra em deadlock
  - grafo wait_for



15

Livelock

- Situação
  - transação T
    - não consegue ser executada por um período indefinido de tempo
  - outras transações
    - continuam suas execuções normalmente
- Ocorrência
  - quando o algoritmo de espera por itens de dados bloqueados é injusto
  - outras transações sempre possuem maior prioridade do que T

16

Starvation

- Pode ocorrer nos algoritmos que tratam *deadlock*
- Ocorre se os algoritmos selecionam a mesma transação como vítima repetidamente, causando sempre o seu aborto e nunca finalizando sua execução
- *wait_die* e *wound_wait* evitam *starvation*

17

Granularidade dos Itens de Dados

- Tamanho dos itens de dados
- Granularidade fina
  - pequenos tamanhos de itens de dados
- Granularidade grossa
  - grandes tamanhos de itens de dados

18

Granularidade dos Itens de Dados

- Um item do banco de dados poderia ser:
  - um registro
  - um campo de um registro
  - um bloco de disco
  - um arquivo inteiro
  - um banco de dados inteiro

Granularidade - Discussão

- Quanto maior o grão, menor o grau de concorrência
  - tamanho do item de dados: bloco de disco
  - transação T
    - deseja utilizar o registro R_1 localizado no bloco de disco B, então trava o bloco de disco B
  - transação S
    - deseja utilizar o registro R_2 localizado no bloco de disco B
    - obrigada a esperar até que T libere a trava associada a B

Granularidade - Discussão

- Quanto menor o grão, maior armazenamento é necessário
  - o número de travas mantidas pelo sistema é elevado
  - mais operações de *lock* e *unclock* são necessárias
  - aumento no *overhead* do sistema
  - maior espaço de armazenamento é necessário para a tabela de travas

Granularidade - Discussão

- Qual o melhor tamanho de item de dado?
  - Depende dos tipos de transações envolvidas
  - Se as transações basicamente acessam um número pequeno de registros

    então granularidade do dado: registro
  - Se as transações basicamente acessam muitos registros do mesmo arquivo

    então granularidade do dado: arquivo ou bloco

Técnicas Controle de Concorrência x Granularidade

- Apresentam tamanho de item de dado uniforme
  - técnicas mais comuns
- Permitem tamanhos variáveis de item de dado
  - tamanho do item de dado pode ser alterado para melhor se adequar às transações que estão executando concorrentemente no sistema

Técnicas Controle de Concorrência Multiversão

- Quando uma transação requer acesso a um item, uma versão apropriada é escolhida para manter a serialização da execução concorrente, se possível.
- A ideia é que algumas operações de leitura que seriam rejeitadas em outras técnicas poderiam ser aceitas por ler uma versão mais antiga do item para manter a serialização.
- Quando uma transação escreve um item, ela escreve uma nova versão e a velha versão do item é retida.