Sistemas de Banco de Dados

Fundamentos em Bancos de Dados Relacionais

Wladmir Cardoso Brandão
www.wladmirbrandao.com

CONTROLE DE CONCORRÊNCIA

Controle de Concorrência

Emprego de técnicas para garantia de isolamento entre transações

- Protocolos → garantem a serialização de escalonamentos
  - Bloqueio → aplica regras para bloqueio e desbloqueio de itens
  - Timestamp → ordena transações usando seus rótulos de tempo (*timestamp*)
  - Multiversão → armazena múltiplas versões de um item, escolhendo a mais apropriada para ser usada por cada transação em execução
  - Validação → checa *a posteriori* se execução ocorreu de forma isolada
  - Granularidade Múltipla → variação do bloqueio com ajuste de granularidade

Técnicas baseadas em protocolos de bloqueio são tipicamente adotadas por SGBDs comerciais

Protocolo de Bloqueio

Serialização por aplicação de regras para bloqueio e desbloqueio de itens

- Binário → duas operações de bloqueio
  - Lock $l(x)$ → altera estado do item para bloqueado pela transação, com outras transações sendo forçadas a esperar pelo desbloqueio
  - Unlock $u(x)$ → altera estado do item para livre, desbloqueando-o
  - Exclusividade → transações não acessam o mesmo item simultaneamente
- Ternário → três operações de bloqueio
  - Read Lock $rl(x)$ → altera estado para bloqueado para leitura, outras transações que demandarem escrita são forçadas a esperar desbloqueio
  - Write Lock $wl(x)$ → altera estado para bloqueado para escrita, outras transações são forçadas a esperar desbloqueio
  - Unlock $u(x)$ → altera estado do item para livre
  - Compartilhamento → transações podem realizar leitura simultaneamente

Protocolo de Bloqueio

Tabela de bloqueio registra emissão de solicitação de bloqueios para itens

- Regras → impostas pelo gerenciador de bloqueio
  - Transação precisa emitir bloqueio antes de operar sobre item
  - Transação precisa emitir desbloqueio após completar operações sobre o item
  - Transação não emite bloqueio se já tiver bloqueio sobre item
    - Promoção (*upgrade*) → bloqueio $rl(x)$ convertido para $wl(x)$
    - Descenso (*downgrade*) → bloqueio $wl(x)$ convertido para $rl(x)$
  - Transação não emite desbloqueio se não tiver bloqueio sobre item
  - Solicitação de bloqueio não atendida entra em fila de espera por desbloqueio

Aplicação de regras de bloqueio e desbloqueio não garante serialização

Regras adicionais para bloqueios e desbloqueios em fases são necessárias

Protocolo de Bloqueio

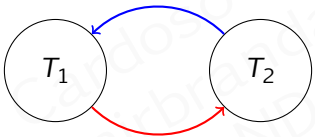
2PL → protocolo de bloqueio em duas fases (*2 Phase Locking*)

- Todas as solicitações de bloqueio de uma transação precedem sua primeira solicitação de desbloqueio
  - Espansão → bloqueios adquiridos, nenhuma liberação
    - Possibilidade de promoção de bloqueios
  - Retração → bloqueios liberados, nenhum bloqueio adquirido
    - Possibilidade de descenso de bloqueios
- Pode limitar concorrência
- Escalonamentos 2PL garantidamente serializáveis (ou seriais)

Protocolo de Bloqueio

Considere o escalonamento não serializável

$$S_a \rightarrow r_1(x), r_2(x), w_1(x), r_1(y), w_2(x), w_1(y)$$



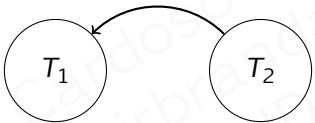
Aplicação do protocolo 2PL

$$S_a \rightarrow l_1(x), r_1(x), l_2(x), r_2(x), w_1(x), l_1(y), r_1(y), w_2(x), u_2(x), w_1(y), u_1(x), u_1(y)$$

Protocolo de Bloqueio

Escalonamento $S_{a'}$ efetivamente executado a partir da aplicação 2PL sobre S_a

$$S_{a'} \rightarrow r_1(x), w_1(x), r_1(y), w_1(y), r_2(x), w_2(x)$$



Note que T_2 parou sua execução ao tentar bloquear x, já bloqueado por T_1

$$S_a \rightarrow l_1(x), r_1(x), l_2(x), r_2(x), w_1(x), l_1(y), r_1(y), w_2(x), u_2(x), w_1(y), u_1(x), u_1(y)$$

Protocolo de Bloqueio

DEADLOCK → situação de travamento, impasse

$$S_a \rightarrow r_1(x), r_2(x), w_1(x), r_1(y), w_2(x), w_1(y)$$

Aplicação do protocolo 2PL com bloqueios ternários

$$S_a \rightarrow rl_1(x), r_1(x), rl_2(x), r_2(x), wl_1(x), w_1(x), rl_1(y), r_1(y), wl_2(x), w_2(x), u_2(x), wl_1(y), w_1(y), u_1(x), u_1(y)$$

Escalonamento $S_{a'}$ efetivamente executado com DEADLOCK

$$S_{a'} \rightarrow r_1(x), r_2(x), w_1(x), w_2(x)$$

Note que T_1 paralisa ao tentar $wl_1(x)$, aguardando T_2 liberar o $rl_2(x)$, e T_2 paralisa ao tentar $wl_2(x)$, aguardando T_1 liberar o $rl_1(x)$

Protocolo de Bloqueio

Resolução de DEADLOCK envolve prevenção, detecção ou presunção

- PREVENÇÃO → 2PL conservador ou estático
  - Transação emite solicitação de bloqueios que necessita antes de iniciar

$$S_a \rightarrow r_1(x), r_2(x), w_1(x), r_1(y), w_2(x), w_1(y)$$

Aplicação de protocolo 2PL estático com bloqueios ternários

$$S_a \rightarrow wl_1(x), wl_1(y), r_1(x), wl_2(x), r_2(x), w_1(x), r_1(y), w_2(x), u_2(x), w_1(y), u_1(x), u_1(y)$$

Escalonamento $S_{a'}$ efetivamente executado

$$S_{a'} \rightarrow r_1(x), w_1(x), r_1(y), w_1(y), r_2(x), w_2(x)$$

Protocolo de Bloqueio

Resolução de DEADLOCK envolve prevenção, detecção ou presunção

- PREVENÇÃO → reescalonamento por *timestamp* (TS)
  - Transação mais nova, bloqueando item necessário para outra mais antiga, aborta e é reescalonada
  - Se T_i inicia antes de T_j, então $TS(T_i) < TS(T_j)$

$$S_a \rightarrow r_1(x), r_2(x), w_1(x), r_1(y), w_2(x), w_1(y)$$

$$TS(T_1) < TS(T_2)$$

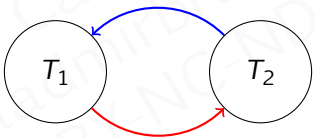
T_1 não consegue $wl_1(x)$ por $rl_2(x)$ de T_2, T_2 aborta e é reescalonada

$$S_{a'} \rightarrow r_1(x), w_1(x), r_1(y), w_1(y)$$

Protocolo de Bloqueio

Resolução de DEADLOCK envolve prevenção, detecção ou presunção

- DETECÇÃO → transações em impasse, mais nova aborta e é reescalonada
  - Transação mais nova, provavelmente com menos operações executadas
  - GRAFO DE ESPERA → nós são transações, arestas direcionadas são bloqueios em espera, ciclo indica impasse



$$S_a \rightarrow r_1(x), r_2(x), w_1(x), r_1(y), w_2(x), w_1(y)$$

Ciclo entre T_1 e T_2, T_2 aborta e é reescalonada por ser a mais nova

Protocolo de Bloqueio

Resolução de DEADLOCK envolve prevenção, detecção ou presunção

- PRESUNÇÃO → transação demorada aborta e é reescalonada
  - TEMPO LIMITE (*timeout*) → registro de tempo
  - Presume-se impasse na transação se seu tempo de execução exceder *timeout*
  - Simples, evita sobrecarga

Protocolo de Bloqueio

STARVATION → situação de espera indefinida, inanição

- Transação constantemente abortada e reescalonada
- No pior caso, nunca executada
- Solução
  - Priorização proporcional ao tempo de espera
  - Manutenção de *timestamp* original, pré-reescalonamento

Protocolo de Timestamp

Baseando-se no princípio da equivalência por conflito, ordem de acesso a um item em operações em conflito não pode violar ordem de *timestamp*

- Dois registros de *timestamp* para cada item
  - Leitura $rTS(x) \rightarrow$ *timestamp* da transação mais nova a ler x
  - Escrita $wTS(x) \rightarrow$ *timestamp* da transação mais nova a escrever x
- Regras $\rightarrow$ garantem ordenação por *timestamp*
  - T emite solicitação $r(x)$
    - Se $wTS(x) > TS(T)$, rejeite $r(x)$, aborte e reverta T
    - Senão, aceite $r(x)$, $rTS(x) = max(rTS(x), TS(T))$
  - T emite solicitação $w(x)$
    - Se $(rTS(x) > TS(T) \vee wTS(x) > TS(T))$, rejeite $w(x)$, aborte e reverta T
    - Senão, aceite $w(x)$, $wTS(x) = TS(T)$

Não gera deadlock, mas tipicamente implica em menor concorrência

Protocolo de Timestamp

Não pode haver conflito entre operação corrente de transação mais antiga e operação anterior de transação mais nova

$$S_a \rightarrow r_1(x), r_2(x), w_1(x), r_1(y), w_2(x), w_1(y)$$

$$TS(T_1) = 1 < TS(T_2) = 2$$

Aplicando-se o protocolo de *timestamp*

$$rTS(x) \rightarrow 0\ \ 1\ \ 2 \qquad wTS(x) \rightarrow 0\ \ 2 \qquad rTS(y) \rightarrow 0 \qquad wTS(y) \rightarrow 0$$

Escalonamento $S_{a'}$ efetivamente executado

$$S_{a'} \rightarrow r_2(x), w_2(x)$$

Referências Bibliográficas

[1] Elmasri, Ramez; Navathe, Sham. *Fundamentals of Database Systems*. 7ed. Pearson, 2016.

[2] Silberschatz, Abraham; Korth, Henry F.; Sudarshan, S. *Database System Concepts*. 6ed. McGraw-Hill, 2011.

[3] Date, Christopher J. *An Introduction to Database Systems*. 8ed. Pearson, 2004.