Sistemas de Banco de Dados

Fundamentos em Bancos de Dados Relacionais

Wladmir Cardoso Brandão
www.wladmirbrandao.com

Fevereiro, 2020

Seção 22

Controle de Concorrência

Técnicas de controle de concorrência

- As técnicas de controle de concorrência são usadas para garantir a não interferência ou isolamento das transações executas simultaneamente.
- Garantem a **serialização de schedules** usando protocolos de controle de concorrência.
- Tais **protocolos de bloqueio** são utilizados na maioria dos SGBDs.

TÉCNICAS DE CONTROLE DE CONCORRÊNCIA:

TÉCNICAS DE BLOQUEIO EM DUAS FASES PARA CONTROLE DE CONCORRÊNCIA

Técnicas de bloqueio em duas fases

Natureza e os tipos de bloqueios

Bloqueios binários:

- Restritivos para fins de controle de concorrência.
- Possuem dois **estados** ou **valores**: *bloqueado* e *desbloqueado*.
- Um bloqueio distinto é associado a cada item do banco de dados.
  - Se **valor do bloqueio é 1**, o item não pode ser acessado por uma operação.
  - Se **valor do bloqueio é 0**, o item pode ser acessado quanto requisitado.

    E o valor do bloqueio é mudado para 1.

Técnicas de bloqueio em duas fases

Natureza e os tipos de bloqueios

Bloqueios binários:

- LOCK(X): Valor atual (ou estado) do bloqueio associado ao item X.
  - Se LOCK(X) = **1**, a transição é forçada a esperar.
  - Se LOCK(X) = **0**, a transição bloqueia o item (é configurada como **1**)

    e passa a possuir permissão para acessar o item X.

```
Nome_item_dado, LOCK, Bloqueio_de_transacao
```

Técnicas de bloqueio em duas fases

Natureza e os tipos de bloqueios

Bloqueios binários:

- Impõe uma **exclusão mútua** no item de dados:
  - Se uma transação requisita acesso a um item, emite uma operação **lock_item**(X).
    - Se **LOCK**(X) = **1**, a transação deve esperar.
    - Se **LOCK**(X) = **0**, ela é configurada como 1 (o item é **bloqueado**)
      e a transação possui permissão para acessar o item.
  - Se a transação termina de usar um item, emite uma operação **unlock_item**(X),
    - que define LOCK(X) para 0 - ou seja, **desbloqueia** o item, que pode ser acessado por outras transações.

Técnicas de bloqueio em duas fases

Natureza e os tipos de bloqueios

Bloqueios binários:

- Regras impostas pelo gerenciador de bloqueio do SGBD:

1. Uma transação precisa emitir a operação *lock_item(X)* antes de operações *read_item(X)* ou *write_item(X)* serem realizadas.

2. Uma transação precisa emitir a operação *unlock_item(X)* após todas as operações *read_item(X)* ou *write_item(X)* serem completadas.

3. Uma transação não emitirá uma operação *lock_item(X)* se já mantiver o bloqueio no item *X*.

4. Uma transação não emitirá uma operação *unlock_item(X)* a menos que ela já mantenha o bloqueio no item *X*.

Técnicas de bloqueio em duas fases

Natureza e os tipos de bloqueios

Bloqueios binários:

- Nenhuma intercalação deve ser permitida se uma operação de bloqueio/desbloqueio é iniciada, até que a operação termine ou a transação espere.
- Entre as operações *lock_item(X)* e *unlock_item(X)* na transação, diz-se que ela mantém o **bloqueio no item *X***.
- No máximo uma transação pode manter o bloqueio de um item em particular.

  Assim, duas transações não podem acessar o mesmo item simultaneamente.

Técnicas de bloqueio em duas fases

Natureza e os tipos de bloqueios

Bloqueios binários:

- **Tabela de bloqueio**: registros para os itens que o sistema mantém e que estão atualmente bloqueados.
- Os itens que não estão na tabela de bloqueio são considerados desbloqueados.
- O SGBD possui um **subsistema de gerenciador de bloqueio** para registrar e controlar o acesso aos bloqueios.

Técnicas de bloqueio em duas fases

Natureza e os tipos de bloqueios

Bloqueios compartilhados/exclusivos:

- Também conhecidos como bloqueios de **modo múltiplo** ou de **leitura/gravação**.
- Utilizados em esquemas de bloqueio de BD práticos.
- Permite que várias transações acessem o mesmo item se todas acessam apenas para *fins de leitura*.

Técnicas de bloqueio em duas fases

Natureza e os tipos de bloqueios

Bloqueios compartilhados/exclusivos:

- Se uma transação tiver de gravar um item, ela precisa ter acesso exclusivo.
- *LOCK(X)* possui três estados possíveis:
  - bloqueado para leitura (bloqueado p/ompartilhamento)
  - bloqueado para gravação (bloqueado exclusivo)
  - desbloqueado

Técnicas de bloqueio em duas fases

Natureza e os tipos de bloqueios

Bloqueios compartilhados/exclusivos:

- Deve-se registrar o número de transações que mantêm um bloqueio compartilhado (leitura) na tabela de bloqueio.
- O sistema mantém registros de bloqueio somente para os itens bloqueados.
  - Se **LOCK(X) = bloqueado para gravação**, o valor de *Bloqueio_de_transação* é uma única transação.
  - Se **LOCK(X) = bloqueado para leitura**, o valor de *Bloqueio_de_transação* é uma lista de uma ou mais transações.

Técnicas de bloqueio em duas fases

Natureza e os tipos de bloqueios

Bloqueios compartilhados/exclusivos:

- Nenhuma intercalação deve ser permitida depois que uma das operações for iniciada até que termine.

```
Nome_item_dado,LOCK,Num_de_leituras, Bloqueio_transacao
```

Técnicas de bloqueio em duas fases

Natureza e os tipos de bloqueios

Bloqueios compartilhados/exclusivos:

- Regras impostas pelo gerenciador de bloqueio do SGBD:

1. Uma transação precisa emitir a operação *read_lock(X)* ou *write_lock(X)* antes que qualquer operação *read_item(X)* seja realizada.

2. Uma transação precisa emitir a operação *write_lock(X)* antes que qualquer operação *write_item(X)* seja realizada.

3. Uma transação precisa emitir a operação *unlock(X)* após todas as operações *read_item(X)* e *write_item(X)* serem completadas.

Técnicas de bloqueio em duas fases

Natureza e os tipos de bloqueios

Bloqueios compartilhados/exclusivos:

- Regras impostas pelo gerenciador de bloqueio do SGBD:

4. Uma transação não emitirá uma operação *read_lock(X)* se ela já mantiver um bloqueio de leitura (compartilhado) ou um bloqueio de gravação (exclusivo) no item *X*.

5. Uma transação não emitirá uma operação *write_lock(X)* se ela já mantiver um bloqueio de leitura (compartilhado) ou um bloqueio de gravação (exclusivo) no item *X*.

6. Uma transação não emitirá uma operação *unlock(X)* a menos que mantenha um bloqueio de leitura (compartilhado) ou um bloqueio de gravação (exclusivo) no item *X*.

Técnicas de bloqueio em duas fases

Natureza e os tipos de bloqueios

Bloqueio de certificação:

- Uma transação que já mantém um bloqueio no item tem permissão, sob certas condições de **converter** o bloqueio de um estado bloqueado para outro.
  - **Upgrade** (*read-locked* para *write-locked*)
  - **Downgrade** (*write-locked* para *read-locked*)
- A tabela de bloqueios precisa incluir identificadores de transação no registro para cada bloqueio para armazenar a informação sobre quais transações mantém bloqueios no item.

Técnicas de bloqueio em duas fases

Protocolo de bloqueio em duas fases básico (2PL básico)

Bloqueio de certificação

- Todas as operações de bloqueio (*read_lock, write_lock*) precedem a primeira operação de desbloqueio na transação.
  - **Fase de expansão ou crescimento (primeira)**: novos bloqueios em itens podem ser adquiridos, mas nenhum pode ser liberado.
  - **Fase de encolhimento (segunda)**: bloqueios existentes podem ser liberados, mas nenhum novo bloqueio pode ser adquirido.

Técnicas de bloqueio em duas fases

Protocolo de bloqueio em duas fases básico (2PL básico)

Bloqueio de certificação

- Se a conversão de bloqueio for permitida:
  - O *upgrading* de bloqueios ocorre na fase de expansão.
  - O *downgrading* de bloqueios ocorre na fase de encolhimento.

Técnicas de bloqueio em duas fases

Protocolo de bloqueio em duas fases básico (2PL básico)

- Resultados de possíveis schedules seriais de T_1 e T_2:
  - Valores iniciais: X=20, Y=30
  - Schedule serial resultante T_1 seguido por T_2: X=50, Y=80
  - Schedule serial resultante T_2 seguido por T_1: X=70, Y=50

T_1	T_2

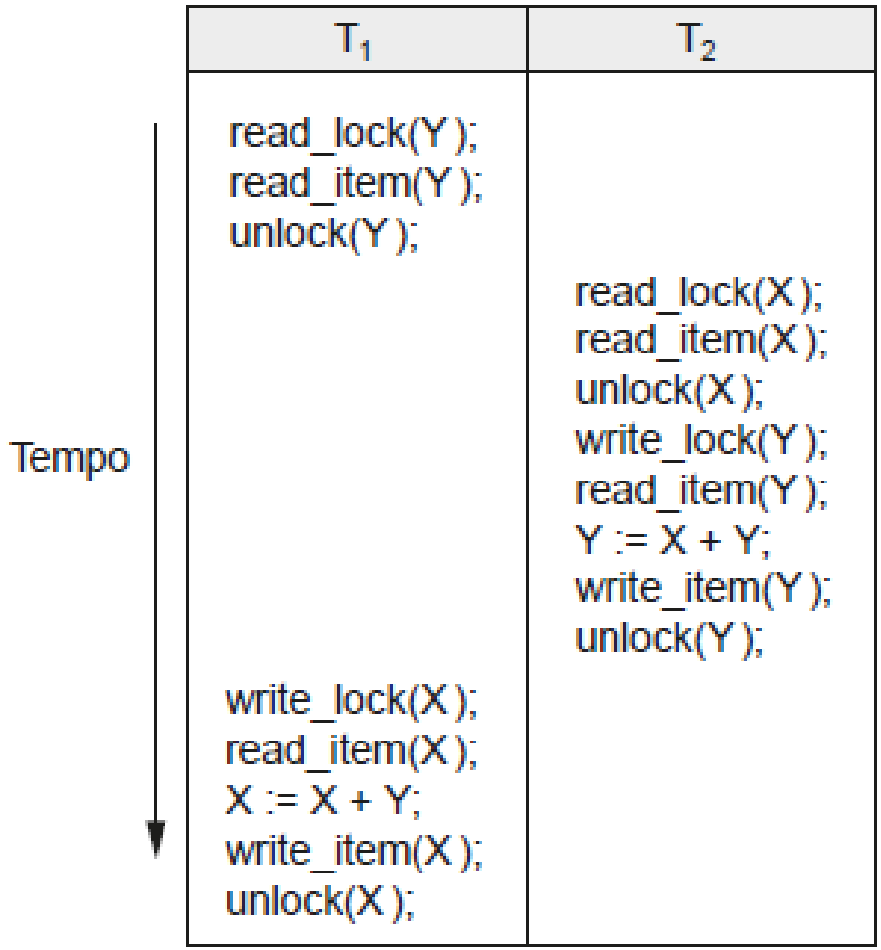
read_lock(Y);
read_item(Y);
unlock(Y);
write_lock(X);
read_item(X);
X := X + Y;
write_item(X);
unlock(X);
	read_lock(X);
	read_item(X);
	unlock(X);
	write_lock(Y);
	read_item(Y);
	Y := X + Y;
	write_item(Y);
	unlock(Y);

Transações que não obedecem ao bloqueio em duas fases.

Técnicas de bloqueio em duas fases

Protocolo de bloqueio em duas fases básico (2PL básico)

- Um schedule não serializável S que usa bloqueios.
  - Resultado de schedule S: X=50, Y=50 (não serializável)



Técnicas de bloqueio em duas fases

Protocolo de bloqueio em duas fases básico (2PL básico)

- O protocolo de bloqueio, ao impor as regras de bloqueio em duas fases, também impõe a serialização.
- Se ***cada*** transação em um schedule seguir o protocolo de bloqueio em duas fases, o schedule é ***garantidamente serializável***.
- O bloqueio em duas fases pode limitar a quantidade de concorrência passível de ocorrer em um schedule.

Técnicas de bloqueio em duas fases

Protocolo de bloqueio em duas fases básico (2PL básico)

- Embora o protocolo de bloqueio em duas fases garanta a serialização, ele não permite ***todos os schedules serializáveis*** possíveis.
- Ou seja, alguns schedules serializáveis serão proibidos pelo protocolo.

Técnicas de bloqueio em duas fases

Protocolo de bloqueio em duas fases conservador

- Variação conhecida como **2PL conservador** (ou **2PL estático**).
- Requer que uma transação bloqueie todos os itens que ela acessa *antes que a transação inicie a execução*, **pré-declarando** seu *conjunto de leitura e de gravação*.
- A transação começa em sua fase de encolhimento.
- Se um dos itens pré-declarados necessários não puder ser bloqueado, a transação espera até que os itens estejam disponíveis para bloqueio.
- É um protocolo livre de deadlock.

Técnicas de bloqueio em duas fases

Protocolo de bloqueio em duas fases estrito (2PL estrito)

- Garante schedules estritos.
- Uma transação não libera nenhum de seus bloqueios exclusivos (gravação) até ***depois*** de confirmar ou abortar.
- Levando a um schedule estrito para facilidade de recuperação.
- O 2PL estrito não é livre de deadlock.

Técnicas de bloqueio em duas fases

Protocolo de bloqueio em duas fases rigoroso (2PL rigoroso)

- Garante schedules estritos.
- Uma transação não libera nenhum de seus bloqueios (exclusivo ou compartilhado) até depois de confirmar ou abortar.
- A transação está em sua fase de expansão até que termine.
- É mais fácil de implementar do que o 2PL estrito.

Técnicas de bloqueio em duas fases

- Em muitos casos, o próprio subsistema de controle de concorrência é responsável por gerar as solicitações *read_lock* e *write_lock*.
- O uso de bloqueios pode causar dois problemas:

  deadlock e **inanição** (starvation)

Deadlock (impasse)

- Conjunto de transações esperando por algum item bloqueado por outra transação do conjunto.
- Ou seja, transações em fila de espera.
- Para lidar com deadlock:
  - Protocolos de prevenção
  - Detecção
  - *Timeouts*

Técnicas de controle de concorrência

VARIAÇÕES DE PROTOCOLOS DE PREVENÇÃO DE DEADLOCK:

1. Requer que cada transação bloqueie **todos os itens que precisar com antecedência**. Se qualquer um dos itens não puder ser obtido, a transação espera e tenta novamente.

2. **Ordena todos os itens** no BD e garante que as transações bloqueiem os itens de acordo com essa ordem.

3. **Rótulo de tempo (*timestamp*) de transação TS(T)**: O rótulo de tempo é um identificador exclusivo para cada transação. É gerado pelo sistema na mesma ordem que os tempos de início de transação.

Se a transação T_1 iniciar antes da transação T_2,
então $TS(T_1) < TS(T_2)$

Protocolos de prevenção de deadlock

- **Protocolos que incluem rótulos de tempo:**
  - **Esperar-morrer (*wait-die*):** Se $TS(T_i) < TS(T_j)$, T_i tem permissão para esperar.

    Caso contrário, aborta T_i e o reinicia posteriormente com o mesmo rótulo de tempo.
  - **Ferir-esperar (*wound-wait*):** Se $TS(T_i) < TS(T_j)$, aborta T_i e o reinicia posteriormente com o mesmo rótulo de tempo.

    Caso contrário, T_i tem permissão para esperar.

Protocolos de prevenção de deadlock

- **Protocolos que incluem algoritmos sem espera e espera cuidadosa:**
  - **Algoritmo sem espera:** Se uma transação for incapaz de obter um bloqueio, ela é abortada e reiniciada após certo atraso de tempo sem verificar se um deadlock ocorrerá ou não.
  - **Algoritmo espera cuidadosa:** Se T_j não estiver bloqueada, então T_i está bloqueada e tem permissão para esperar; caso contrário, aborte T_i.

Detecção de deadlock

Sistema verifica se um estado de deadlock realmente existe.

- *Soluções:*
  - **Grafo de espera**, modo de detectar um estado de deadlock.
  - Um nó é criado para cada transação que está sendo executada
  - Uma aresta é direcionada caso uma transação esteja esperando para bloquear um item que está atualmente bloqueado por uma outra transação.
  - Caso esta transação libere o bloqueio no item, a aresta é removida.
  - Temos um estado de deadlock, se o grafo de espera tiver um ciclo.

Detecção de deadlock

Sistema verifica se um estado de deadlock realmente existe.

Se o sistema estiver em deadlock, algumas das transações que causam este estado precisam ser abortadas.

- *Soluções:*
  - **Algoritmo seleção de vítima**, escolha de quais transações abortar.
  - Evita a seleção de transações que estiveram em execução por muito tempo e que realizam muitas atualizações.
  - Seleciona transações mais novas.

Timeouts (tempo-limite)

- Se uma transação esperar por um período maior que o *timeout,* o sistema pressupõe que a transação pode entrar em deadlock e a aborta.
- Baixo overhead e simples.

Inanição (starvation)

Acontece quando uma transação não pode prosseguir por um período indefinido, enquanto outras continuam normalmente (bloqueio).

- *Soluções:*
  - Possuir um esquema de espera justo, como o uso de uma fila **primeiro-a-chegar-primeiro-a-ser-atendido**;

    as transações bloqueiam um item na ordem em que solicitaram o bloqueio originalmente.

  - Aumentar a prioridade de uma transação quanto mais tempo ela esperar.

Inanição (starvation)

A inanição também pode ocorrer por causa da seleção de vítima se o algoritmo selecionar a mesma transação como vítima repetidamente.

- *Soluções:*
  - O algoritmo pode usar prioridades maiores para transações que tiverem sido abordadas várias vezes.
  - Os esquemas **esperar-morrer** e **ferir-esperar** evitam a inanição.

TÉCNICAS DE CONTROLE DE CONCORRÊNCIA:

Controle de concorrência por ordenação de rótulo de tempo

Controle de Concorrência - Rótulo de tempo

- O uso de bloqueios, combinado com o protocolo 2PL, garante e serialização de schedules;
- Schedules serializáveis (2PL) têm seus schedules seriais equivalentes com base na ordem em que as transações em execução bloqueiam os itens adquiridos;
- Se uma transação precisar de um item que está bloqueado.
  - É forçada a esperar até que o item seja liberado.

Controle de Concorrência - Rótulo de tempo

- Por conta de deadlocks, algumas transações podem ser abortadas e reiniciadas;
- Outra técnica de serialização:
  - Envolve o uso de rótulos de tempo de transação para ordenar a execução da transação para um schedule serial equivalente.

Controle de Concorrência - Rótulo de tempo

Rótulos de tempo (timestamp)

TS(T) - Rótulo de tempo de uma transação

- Identificador exclusivo criado pelo SGBD para **identificar uma transação**;
- Os valores são atribuídos na ordem em que as transações são submetidas ao sistema;
  - Hora de início da transação.
- As técnicas de controle de concorrência baseadas na ordenação por rótulo de tempo **não usam bloqueios**;
  - Deadlocks não podem ocorrer.

Rótulos de tempo (timestamp)

- Podem ser gerados de várias maneiras. Abaixo, duas alternativas:

  1. Utilizar um contador que é incrementado toda vez que seu valor é atribuído a uma transação;
     - Tem um valor máximo finito - o sistema precisa reiniciar o contador periodicamente.
  2. Usar o valor atual de data/hora do clock do sistema;
     - Deve-se garantir que dois valores de rótulo de tempo sejam gerados durante a mesma batida do clock.

Rótulos de tempo (timestamp) - algoritmo de ordenação

- Ordena as transações com base em seus rótulos de tempo;
- **Ordenação de rótulo de tempo (TO):**
  - Um schedule em que as transações participam é serializável e o **único schedule serial equivalete permitido** tem as transações na ordem de seus valores de rótulo de tempo;
  - O schedule é equivalente à **ordem serial em particular** correspondente à ordem dos rótulos de tempo da transação.

Rótulos de tempo (timestamp) - algoritmo de ordenação

- Para cada item acessado pelas **operações em conflito** no schedule, a ordem em que o item é acessado não deve violar a ordem do rótulo de tempo;
- O algoritmo associa a cada item X do banco de dados dois valores de rótulo de tempo (TS):
  1. **Read_TS(X)**
     - O rótulo de tempo de leitura do **item X** é o maior entre todos os rótulos de tempo das transações que leram com sucesso o **item X**;
     - **read_TS(X) = TS(T)**, onde T é a transação mais recente que leu X com sucesso.

Rótulos de tempo (timestamp) - algoritmo de ordenação

- O algoritmo associa a cada item X do banco de dados dois valores de rótulo de tempo (TS):

  2 **Write_TS(X)**

  - O rótulo de tempo de gravação do **item X** é o maior de todos os rótulos de tempo das transações que gravarem com sucesso o **item X**;
  - **write_TS(X) = TS(T)**, onde T é a transação mais recente que gravou X com sucesso.

Rótulos de tempo (timestamp) - algoritmo de ordenação

- Sempre que alguma transação T tenta emitir uma operação **read_item(X)** ou **write_item(X)**:
  - O algoritmo **TO básico** compara o rótulo de tempo de T com **read_TS(X)** e **write_TS(X)** para garantir que a ordem do rótulo não seja violada;
- Se a ordem for violada:
  - A transação T é abortada e submetida novamente com um novo rótulo de tempo.

Rótulos de tempo (timestamp) - algoritmo de ordenação

- Se T for abortada e revertida:
  - Qualquer transação T_1 que possa ter usado um valor gravado por T também precisa ser revertida.
- O efeito acima é chamado de **Propagação de cancelamento** ou **Rollback em cascata**;
  - Os schedules produzidos não têm garantias de serem recuperáveis.
- Nessário adicionar um **protocolo adicional** para garantir que os schedules sejam recuperáveis, sem cascata ou estritos.

Rótulos de tempo (timestamp) - algoritmo de ordenação

- O algoritmo de controle de concorrência deve verificar se as operações em conflito violam a ordenação em rótulo de tempo nos dois casos a seguir:

1. Sempre que uma transação T emitir uma operação **write_item(X)**, deve ser verificado:
   - Se **read_TS(X)** > **TS(T)** ou se **write_TS(X)** > **TS(T)**, aborte e reverta T e rejeite a operação;
   - Se a condição acima não ocorrer, execute a operação **write_item(X)** de T e defina **write_TS(X)** como **TS(T)**.

Rótulos de tempo (timestamp) - algoritmo de ordenação

- O algoritmo de controle de concorrência deve verificar se as operações em conflito violam a ordenação em rótulo de tempo nos dois casos a seguir:

  2. Sempre que uma transação T emitir a operação **read_item(X)**, o seguinte deve ser verificado:

     - Se **write_TS(X) > TS(T)**, então aborte e reverta T e rejeite a operação;
     - Se **write_TS(X) <= TS(T)**, então execute a operação **read_item(X)** de T e defina **read_TS(X)** como o **maior** de **TS(T)** e o **read_TS(X)** atual.

Rótulos de tempo (timestamp) - algoritmo de ordenação

- Sempre que o algoritmo de **TO básico** detectar duas operações em conflito que ocorrem na ordem incorreta:
  - Rejeita uma das duas, abortando a transação que a emitiu.
- Os schedules gerados pela **TO básica** têm garantias de serem **serializáveis por conflito**;
- Alguns schedules são possíveis sub um protocolo que não são permitidos sob o outro.
  - Nenhum protocolo permite todos os schedules serializáveis possíveis

Rótulos de tempo (timestamp) - algoritmo de ordenação

- **TO estrita**
  - Garante que os schedules sejam tanto **estritos** (maior facilidade de recuperação) quanto serializáveis (conflito);
  - Uma transação T emite um **read_item(X)** ou **write_item(X)** tal que **TS(T) > write_TS(X)** tenha sua operação adiada até que a transação T′ que gravou o valor de X (portanto, **TS(T′) = write_TS(X)**) tenha sido confirmada ou abortada;
  - Não causa deadlock, pois T espera por T′ somente se **TS(T) > TS(T′)**.

Rótulos de tempo (timestamp) - algoritmo de ordenação

- **Regras da gravação de Thomas**
  - Modificação do **TO básico**;
  - Não impõe a serialização por conflito, mas rejeita menos operações de gravação ao modificar as verificações para a operação **write_item(X)** da seguinte forma:
    a. Se **read_TS(X)** > **TS(T)**, então aborte e reverta **T**, e rejeite a operação;
    b. Se **write_TS(X)** > **TS(T)**, então não execute a operação de gravação, mas continue processando;
    c. Se nenhuma condição em **(a)** nem em **(b)** acontecer, então execute a operação **write_item(X)** de **T** e defina **write_TS(X)** para **TS(T)**.

TÉCNICAS DE CONTROLE DE CONCORRÊNCIA:

Técnicas para controle de concorrência multiversão

Controle de concorrência multiversão

- Protocolos para controle de concorrência que mantêm os valores antigos de um item de dados quando este é atualizado;
- Quando uma transação requer acesso a um item, uma versão apropriada é escolhida para manter a serialização do schedule atualmente em execução;
- Necessidade de mais armazenamento para manter várias versões dos itens no BD;
  - As versões mais antigas podem ser mantidas de qualquer forma.

Controle de concorrência multiversão

- Algumas aplicações de BD exigem que versões mais antigas sejam mantidas como histórico da evolução de valores;
- Banco de dados temporal (caso extremo)
  - Registra todas as mudanças e os momentos em que elas ocorreram;
  - Não existe penalidade de armazenamento adicional.
- Além dos esquemas de controle de concorrência, o **método de controle de concorrência de validação** também mantém múltiplas versões.

Multiversão baseada na ordenação de rótulo de tempo

- Para cada versão, o valor da versão X_i e os dois rótulos de tempo são mantidos:

1. **Read_TS(X_i)**
   - O rótulo de tempo de leitura de X_i é o maior de todos os rótulos de tempo de transações que leram a versão X_i com sucesso.

2. **Write_TS(X_i)**
   - O rótulo de tempo de gravação de X_i é o rótulo de tempo da transação que gravou o valor da versão X_i.

Controle de concorrência multiversão

Multiversão baseada na ordenação de rótulo de tempo

- Sempre que uma transação T tem permissão para executar uma operação **write_item**(X), uma nova versão X_{k+1} do item X é criada, e tanto **write_TS**(X_{k+1}) quanto **read_TS**(X_{k+1}) são definidos como TS(T);

- Regras usadas para garantir a serialização:

1) Se a transação T emitir uma operação **write_item**(X) e a versão **i** de X tiver o **write_TS**(X_i) mais alto de todas as versões de X <= TS(T) e **read_TS**(X_i) > TS(T):
    - Aborte e retroceda a transação T.

Senão Crie uma nova versão X_i de X com **read_TS**(X_i) = **write_TS**(X_i) = TS(T).

Multiversão baseada na ordenação de rótulo de tempo

- Regras usadas para garantir a serialização:

  2) Se a transação T emitir uma operação **read_item**(X):

     - Determine a versão **i** de X que tem o **write_TS**(X_i) mais alto de todas as versões de X <= **a** TS(T); depois, retorne o valor de X_i à transação T e defina o valor de **read_TS**(X_i) ao maior de TS(T) e o **read_TS**(X_i) atual.

- Para garantir a facilidade de recuperação:

  - Uma transação T não deve ter permissão para confirmar até que todas as transações que gravam alguma versão que T leu tenha sido confirmadas.

Controle de concorrência multiversão

Multiversão baseada na ordenação de rótulo de tempo

- Em modo múltiplo, existem três tipos de bloqueio para um item:
  - **Leitura, gravação** e **certificação**.
- O estado de LOCK(X) para um item X pode ser um dentre bloqueado para leitura, bloqueado para gravação, bloqueado para certificação ou desbloqueado;
- No esquema de **bloqueio padrão**, com apenas bloqueios de leitura e gravação, um bloqueio de gravação é um bloqueio exclusivo.

Multiversão baseada na ordenação de rótulo de tempo

- Tabela de compatibilidade de bloqueio:

	Leitura	Gravação
Leitura	Sim	Não
Gravação	Não	Não

Tabela de compatibilidade para o esquema de bloqueio leitura/gravação.

Multiversão baseada na ordenação de rótulo de tempo

- A ideia do **2PL** multiversão é permitir que outras transações T leiam um item X enquanto uma única transação T mantém o bloqueio de gravação em X.
  - Custo: uma transação pode ter de esperar sua confirmação até que obtenha bloqueios de certificação exclusivos em todos os itens que atualizou;
  - Evita a propagação de abortos;
  - Podem ocorrer deadlocks se o upgrading de um bloqueio de leitura para um bloqueio de gravação for permitido.

Multiversão baseada na ordenação de rótulo de tempo

- Tabela de compatibilidade de bloqueio:

	Leitura	Gravação	Certificação
Leitura	Sim	Sim	Não
Gravação	Sim	Não	Não
Certificação	Não	Não	Não

Tabela de compatibilidade para o esquema de bloqueio leitura/gravação/certificação.

TÉCNICAS DE CONTROLE DE CONCORRÊNCIA:

Técnicas de controle de concorrência de validação (otimista)

Controle de concorrência de validação (otimista)

- Também conhecidas como **técnicas de validação** ou **certificação**;
- **Nenhuma verificação** é feita enquanto a transação está executando;
- Vários métodos teóricos são baseados nesta técnica de validação - o esquema será apresentado a seguir;
- As atualizações na transação **não são** aplicadas diretamente aos itens do BD até que a transação alcance seu final;
  - Cópias locais dos itens de dados

Controle de concorrência de validação (otimista)

- Ideia geral do método:
  - Realizar todas as verificações ao mesmo tempo;
  - A execução da transação prossegue com um mínimo de overhead até que a fase de validação seja alcançada.
- Utiliza rótulos de tempo;
- Requer que os **write_sets** e **reads_sets** das transações sejam mantidas pelo sistema;
- Tempos de **início** e **fim** para algumas das três fases precisam ser mantidas para cada transação.

Controle de concorrência de validação (otimista)

- Fases para este protocolo de controle de concorrência:
  1. Fase de leitura:
     - Uma transação pode ler valores dos itens de dados confirmados com base no BD;
     - As atualizações são aplicadas a cópias locais.
  2. Fase de validação:
     - Verificação realizada para garantir que a serialização não será violada;
     - Confere se as atualizações da transação foram aplicadas ao BD.
  3. Fase de gravação:
     - Somente se a fase da validação for bem-sucedida;
     - Atualizações da transação são aplicadas ao BD.

Controle de concorrência de validação (otimista)

- A fase de validação para T_i verifica se, para **cada** transação T_j que é confirmada ou está em sua fase de validação, **uma** das seguintes condições é mantida:

  1. A transação T_j completa sua fase de gravação antes que T_i inicie sua fase de leitura;

  2. T_i inicia sua fase de gravação depois que T_j completa sua fase de gravação, e o **read_set** de T_i não tem itens em comum com o **write_set** de T_j;

  3. Tanto o **read_set** quanto o **write_set** de T_i não têm itens em comum com o **write_set** de T_j, e T_j completa sua fase de leitura antes que T_i o faça.

TÉCNICAS DE CONTROLE DE CONCORRÊNCIA:

Granularidade de itens e bloqueio de granularidade múltiplo

Granularidade de itens e bloqueio múltiplo

- Todas as técnicas de concorrência consideram que o banco de dados é formado por uma série de itens de dados nomeados;
- Um item de BD poderia ser escolhido como sendo um dos seguintes:
  - Um registro de BD;
  - Um valor de campo de um registro do BD;
  - Um bloco de disco;
  - Um arquivo inteiro;
  - Um banco de dados inteiro.
- A granularidade pode afetar o desempenho do controle de concorrência e recuperação.

Granularidade de itens e bloqueio múltiplo

Granularidade do item de dados - bloqueio

- Tamanho dos itens de dados;
- **Granularidade fina** refere-se a tamanhos de item pequenos;
- **Granularidade grossa** refere-se a tamanhos de itens grandes.

A seguir, será apresentada a discussão do tamanho do item de dados no contexto do bloqueio:

- Quanto maior o tamanho do item de dados, menor o grau de concorrência permitido, e vice-versa;

Granularidade de itens e bloqueio múltiplo

Granularidade do item de dados - bloqueio

- O sistema terá um grande número de bloqueios ativos para serem tratados pelo gerenciador de bloqueios
  - Cada transação está associada a um bloqueio.
- Quanto mais operações de bloqueio/desbloqueio, maior o overhead.
  - Também necessário mais espaço de armazenamento para a tabela de bloqueio.

Granularidade de itens e bloqueio múltiplo

Granularidade do item de dados - bloqueio

- Para os rótulos de tempo, o armazenamento é exigido para o **read_TS** e **write_TS** para cada item de dados
  - Haverá um overhead semelhante para o tratamento de um grande número de itens.

Qual o melhor tamanho de item?

Depende dos tipos de transações envolvidas.

- Se uma transação típica acessar um pequeno número de registros, é vantajoso ter uma granularidade com um registro.

Granularidade de itens e bloqueio múltiplo

Granularidade do item de dados - bloqueio

Qual o melhor tamanho de item?

Depende dos tipos de transações envolvidas.

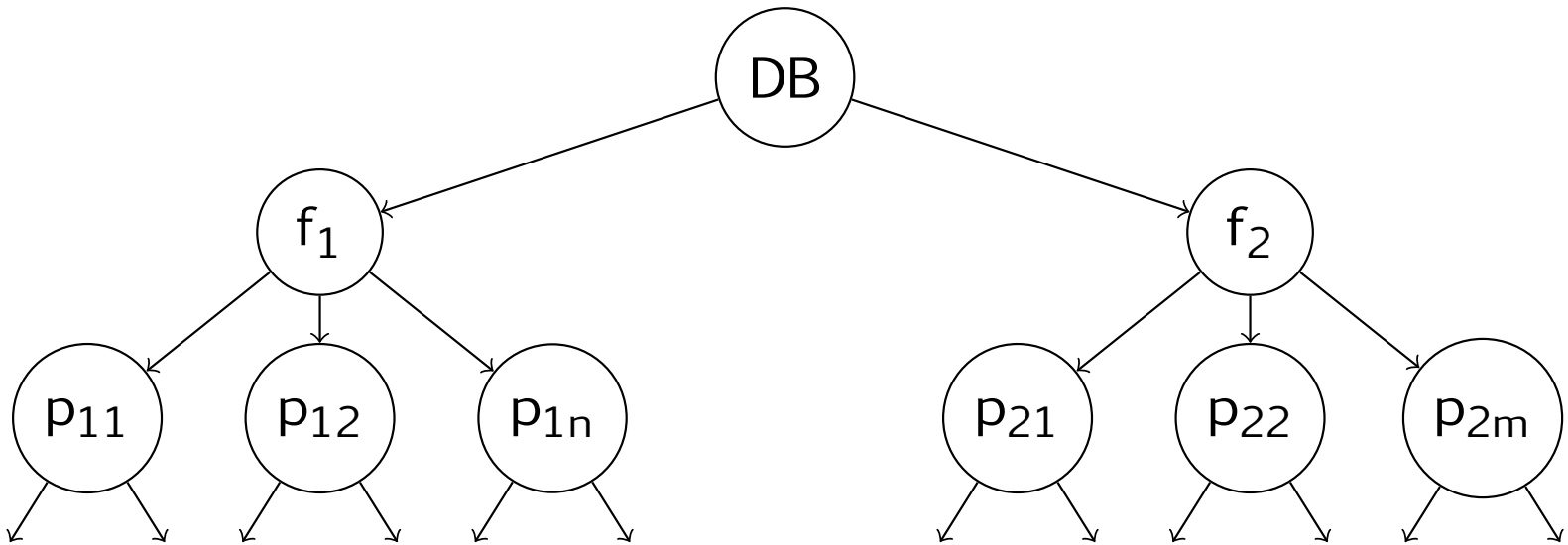
- Se uma transação acessar muitos registros em um mesmo arquivo, é vantajoso ter uma granularidade de bloco ou arquivo de modo que a transação considerará todos os registros como um (ou alguns) item(ns) de dados.

Bloqueio com nível de granularidade múltiplo

- Como o melhor tamanho de granularidade depende da transação dada, é apropriado que um sistema de BD admita múltiplos níveis de granularidade.

Bloqueio com nível de granularidade múltiplo

Hierarquia de granularidade simples com um banco de dados que contém dois arquivos - cada arquivo com várias páginas de disco, e cada página contendo vários registros:



(. . .)

Granularidade de itens e bloqueio múltiplo

Bloqueio com nível de granularidade múltiplo

- O esquema apresentado anteriormente ilustra o protocolo 2PL com **nível de granularidade múltiplo**.
  - O bloqueio pode ser solicitado em qualquer nível;
  - Tipos adicionais de bloqueio serão necessários para dar suporte a tal protocolo com eficiência.
- Para o tornar mais prático, são necessários os bloqueios de intenção.

Granularidade de itens e bloqueio múltiplo

Bloqueios de intenção

- Uma transação indica, junto com o caminho da raíz até o nó desejado, qual o tipo de bloqueio ele exigirá de um dos descendentes do nó;
- Três tipos de bloqueios de intenção:
  1. **Intention-shared (IS)**
     - Um ou mais bloqueios compartilhados serão solicitados em algum ou alguns nós descendentes.

Bloqueios de intenção

- Três tipos de bloqueios de intenção:

2. **Intention-exclusive (IX)**
   - Um ou mais bloqueios exclusivos serão solicitados em algum ou alguns nós descendentes.

3. **Shared-intention-exclusive (SIX)**
   - O nó atual está bloqueado de modo compartilhado, mas que um ou mais bloqueios exclusivos serão solicitados em algum ou alguns nós descendentes.

Bloqueios de intenção

Tabela de compatibilidade dos três bloqueios de intenção:

	IS	IX	S	SIX	X
IS	Sim	Sim	Sim	Sim	Não
IX	Sim	Sim	Não	Não	Não
S	Sim	Não	Sim	Não	Não
SIX	Sim	Não	Não	Não	Não
X	Não	Não	Não	Não	Não

Granularidade de itens e bloqueio múltiplo

- Protocolo de **bloqueio com granularidade múltipla**, regras:

  1. A compatibilidade de bloqueio deve ser aderida;
  2. A raíz da árvore precisa ser bloqueada primeiro, em qualquer modo;
  3. Um nó **N** pode ser bloqueado por uma transação **T** no modo **S** ou **IS** somente se o nó pai **N** já estiver bloqueado pela transação **T** no modo **IS** ou **IX**;

Granularidade de itens e bloqueio múltiplo

- Protocolo de **bloqueio com granularidade múltipla**, regras:

  4. Um no **N** só pode ser bloqueado por uma transação **T** no modo **X**, **IX** ou **SIX** se o pai do nó **N** já estiver bloqueado pela transação **T** no modo **IX** ou **SIX**;

  5. Uma transação **T** só pode bloquear um nó se não tiver desbloqueado qualquer nó;

  6. Uma transação **T** só pode desbloquear um nó, **N**, se nenhum dos filhos do nó **N** estiver atualmente bloqueado por **T**.

TÉCNICAS DE CONTROLE DE CONCORRÊNCIA:

Bloqueios para controle de concorrência de índices

Bloqueios p/ controle de concorrência de índices

- O bloqueio em duas fases também pode ser aplicado a índices:
  - Os nós de índice correspondem a páginas de disco.
- A estrutura de árvore do índice pode ser aproveitada **quando se desenvolve um esquema de controle de concorrência.**
- **Técnica conservadora para inserções:**
  - Bloquear o nó raíz no modo exclusivo e depois acessar o nó filho apropriado da raíz;
  - Se o nó filho não estiver cheio, o bloqueio no nó raíz pode ser liberado;

Bloqueios p/ controle de concorrência de índices

- **Técnica conservadora para inserções:**
  - Aplicável a toda a árvore (até a folha);
  - Embora os bloqueios exclusivos sejam mantidos, eles são logo liberados.
- **Técnica alternativa (I) - otimista:**
  - Manter **bloqueios compartilhados** nos nós que levam ao nó folha, com um **bloqueio exclusivo** na folha;
  - Se a inserção fizer que a folha seja divida, esta se propagará para os nós de maiores níveis;
  - Ao fim, os bloqueios nos nós de nível mais alto podem receber um upgrade para o modo exclusivo.

Bloqueios p/ controle de concorrência de índices

- **Técnica alternativa (II):**
  - Utiliza uma variante B^+-tree - **Árvore B-link**;
  - Os nós irmãos no mesmo nível são ligados em cada nível;
  - Permite que os bloqueios compartilhados sejam usados quando se solicita uma página e exige que o bloqueio seja liberado antes de acessar o nó filho;
  - Se a inserção fizer que a folha seja divida, esta se propagará para os nós de maiores níveis;
  - **Para inserção:** o bloqueio compartilhado em um nó receberia um upgrade para o modo exclusivo;

Bloqueios p/ controle de concorrência de índices

- **Técnica alternativa (II):**
  - **Para divisão:** o nó pai precisa se bloqueado novamente no modo exclusivo;
  - Complicação nas operações de pesquisa executadas simultâneamente com a atualização;
  - Para exclusão (em que dois ou mais nós da árvore são mesclados): faz parte do protocolo de concorrência da árvore B-link.
    - Os bloqueios nos nós a serem mesclados, e em seus "pais"são mantidos.

TÉCNICAS DE CONTROLE DE CONCORRÊNCIA:

Outras questões

Outras questões

Inserção, exclusão e registros fantasmas

- Quando um item é **inserido**, ele não pode ser acessado antes que a operação seja concluída.
  - Um bloqueio para o item pode ser criado e definido como exclusivo (gravação);
  - O bloqueio pode ser liberado ao mesmo tempo dos demais bloqueios;
  - Para um protocolo de rótulo de tempo, os rótulos de leitura e gravação são definidos como o **rótulo de tempo da transação de criação**.

Outras questões

Inserção, exclusão e registros fantasmas

- Quando um item é **excluído**:
  - Um bloqueio exclusivo deve ser obtido antes que a transação possa excluir o item;
  - Para a ordenação do rótulo do tempo, o protocolo precisa garantir que nenhuma gravação posterior tenha lido ou gravado o item antes da permissão para exclusão.

Outras questões

Inserção, exclusão e registros fantasmas

- **Problema do fantasma:**
  - Ocorre quando um novo registro que está sendo inserido por uma transação T satisfaz uma condição que um conjunto de registros acessados por outra transação T′ precisa satisfazer;
  - Se outras operações nas duas transações estiverem em conflito, o conflito do registro fantasma pode não ser reconhecido pelo protocolo.
  - **Solução:** Bloqueio de índice
  - **Solução alternativa:** Bloqueio de predicado
    - Bloqueia o acesso a todos os registros que satisfazem um **predicado** (condição) qualquer de maneira semelhante;

Outras questões

Latches

- Bloqueios mantidos por uma curta duração;
- Não seguem o protocolo de controle de concorrência normal
- Exemplo:
  - Pode ser usado para garantir a integridade física de uma página quando ela está sendo gravada do buffer para o disco;
  - Um latch seria adquirido para a página, a página seria gravada no disco e, depois, o latch seria liberado.

OBRIGADO

Wladmir Cardoso Brandão
www.wladmirbrandao.com

"Science is more than a body of knowledge. It is a way of thinking."
Carl Sagan