UNIVERSIDADE FEDERAL DO MARANHAO - UFMA

Banco de Dados II

Controle de Concorrência

Carlos Eduardo **Portela** Serra de Castro

*

Sumário

1 – Boqueio de 2 fases

2 – Timestemp

3 – Multiversão

4 – Validação

5 – Granularidade

6 – Índices

7 – Outras opções

1 – Bloqueio de 2 fases

- Um bloqueio (lock) é uma variável associada a um item de dado que descreve o status do item com relação a possíveis operações que podem ser aplicadas ao mesmo:
  - Bloqueio Binário: locked (1) ou unlocked (0).
    - lock_item
    - read_item ou write_item
    - unlock_item
  - Bloqueios Compartilhados/Exclusivos: Read/Write.
    - read_lock (share-locked)
    - write_lock (exclusive-locked)
    - unlock

Granularidade de Itens de Dados

Todas as técnicas de controle de concorrência consideram que o banco de dados é formado a partir de uma série de itens de dados com nomes. Um item de dados pode ser escolhido como sendo entre os seguintes:

Itens de dados

- Um registro do banco de dados
- Um campo de um registro do banco de dados
- Um bloco de disco
- O arquivo inteiro
- O banco de dados completo

==========

- Um atributo de uma tupla
- Uma linha da tabela
- Uma pagina da tabela
- A tabela inteira
- O banco de dados inteiro

Tipos de bloqueio

Tipos de bloqueio:
1. **Compartilhado (*S*):** se uma transação Ti obteve esse bloqueio sobre um item, então Ti poderá ler, mas não poderá escrever nesse item.
2. **Exclusivo (*X*):** se uma transação Ti obteve esse bloqueio de um item, Ti poderá ler e escrever nesse item.

O gerenciador de controle concede ou não o bloqueio solicitado a depender da relação de compatibilidade entre os modos.

	S	*X*
S	VERDADEIRO	FALSO
X	FALSO	FALSO

Função de compatibilidade entre os modos S e X

Bloqueio em Duas Fases (2PL)

Diz-se que uma transação segue o protocolo de bloqueio em duas fases se todas as operações de bloqueio (read_lock ou write_lock) precedem a primeira operação de desbloqueio na transação. Essa transação pode ser dividida em duas fases: uma fase de expansão ou crescimento, durante a qual podem-se obter novos bloqueios em itens mas nenhum pode ser liberado; e uma fase de encolhimento ou retração (segunda fase), durante a qual bloqueios existentes podem ser liberados, mas nenhum bloqueio pode ser obtido.

T1

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

Bloqueio em Duas Fases (2PL)

- 2PL básico.
- 2PL conservador:
  - requer que uma transação bloqueie todos os itens que acessa antes que a transação inicie sua execução;
  - se qualquer um dos itens não puder ser bloqueado, a transação não bloqueia item algum;
  - é livre de deadlock;
  - é difícil de se utilizar na prática.
- 2PL estrito:
  - uma transação T não libera qualquer um de seus bloqueios exclusivos (de gravação) até que conclua (commit) ou seja abortada;
  - nenhuma outra transação pode ler ou gravar um item que esteja gravado por T até que T esteja concluída (commit);
- 2PL rigoroso:
  - uma transação T não libera qualquer um de seus bloqueios exclusivos ou compartilhados (de gravação ou leitura) até que conclua (commit) ou seja abortada;
  - nenhuma outra transação pode ler ou gravar um item que esteja gravado por T até que T esteja concluída (commit).

2 – Timestamping

Antes do início da execução de uma transação Ti, o SGBD associa um timestamp (marcação de tempo) fixo a transação Ti, que será denominado por TS(Ti), e que representará a ordem de serialização. As formas de atribuição dos valores de *timestamp* podem ser através do relógio do sistema (hora em que a transação foi solicitada) ou através do uso de um contador lógico que é incrementado a cada nova solicitação de execução de uma transação.

Se uma transação Tj entra no sistema depois de Ti, temos então que TS(Ti) < TS(Tj) e o sistema precisará garantir que a escala de execução produzida seja equivalente a uma escala serial em que a transação Ti aparece antes da Tj.

2 – Timestamping

Os valores de *timestamp* são associados também a itens de dados, para serem usados no controle de conflitos, através de protocolos que utilizam marcação de tempo. Dado um item Q qualquer, são descritas duas funções para a atribuição do *timestamp:*

– **W-timestamp(Q)**: atribuirá o maior *timestamp* da transação que execute uma função write(Q) com sucesso;

– **R-timestamp(Q):** idem ao anterior, mas a função será read(Q).

2 – Timestamping

PROTOCOLO COM BASE EM TIMESTAMP

Protocolo de Ordenação:

- Não utiliza o conceito de bloqueio;
- Trabalha com tempos associados a cada transação;
- Serialização através da ordenação;
- Transações são desfeitas caso não possa executar uma operação.

- Ti emite um read(Q):

- TS(Ti) < W-timestamp (Q) – negado
- Se TS(Ti) >= W-timestamp(Q) – permitido
- Para qualquer R-timestamp(Q) – permitido
- Atualização de R-timestamp(Q)

2 – Timestamping

- Ti emite uma operação de escrita em Q

- TS(Ti) < R-timestamp(Q) – negado

Dado foi lido por outra transação e atualização foi considerada como não produzida

2 – Timestamping

- **Ti emite uma operação de escrita em Q**
  - TS(Ti) < W-timestamp(Q) – negado

Ti tentando escrever “dado obsoleto”

2 – Timestamping

Caso o W-timestamp(Q) ou R-timestamp(Q) seja menor que o *timestamp* de Ti, então a operação é permitida e Ti segue o seu fluxo normalmente.

A tabela abaixo resume os possíveis casos:

	TS(Ti) < TS(Tj)		
Ação de Ti	***W-timestamp(Q) = Tj***	***R-timestamp(Q) = Tj***	***(W-R)-Timestamp(Q) < Ti***
Leitura	FALSO	FALSO	VERDADEIRO
Escrita	FALSO	FALSO	VERDADEIRO

3 – Multiversão

-Diversas versões do item são mantidas

-Quando uma transação exige acesso a um item de dado uma versão apropriada é escolhida.

- Diminuição de necessidade de reexecução

- Ausência de deadlock – sem bloqueios

- Não permite estados inconsistentes

- A cada operação de escrita é criada uma nova versão de um item de dados Q

- Na leitura é selecionada uma das versões

3 – Multiversão

MULTIVERSÃO COM ORDENAÇÃO DE *TIMESTAMP*

Para cada item de dado Q existe uma seqüência de versões representadas por <Q1, Q2, ...,Qn> e contém os seguintes elementos:

– **content:** é o dado propriamente dito;

– **W-timestamp(Qk):** é o *timestamp* da transação que criou a versão Qk;

– **R-timestamp(Qk):** é o *timestamp* mais alto de uma transação que tenha lido com sucesso a versão Qk.

Quando ocorre uma operação de escrita de uma transação Ti, uma versão Qk é criada, cujo conteúdo será o valor a ser escrito por Ti, os *timestamp* de escrita e leitura serão o de Ti.

3 – Multiversão

Seja uma transação Ti e um versão do item de dados Qk, onde o W-timestamp(Qk) é mais alto e menor ou igual *timestamp que* TS(Ti). O funcionamento deste protocolo é o seguinte:

– Se a transação Ti emite uma operação de leitura, então esta será realizada com sucesso;

– Se a transação Ti emite uma operação de escrita, existem duas situações:

- Se Qk foi lido por uma transação com *timestamp* maior que o de Ti, então a operação é abortada;
- Se foi Ti quem criou o dado Qk, então o conteúdo será sobrescrito e a operação é realizada;
- Se quem criou a versão Qk, foi outra transação então uma nova versão Qk+1 é criada.

A eliminação de versões é feita pelo sistema, eliminando as versões mais velhas de item Q (deixando a mais nova) que o último valor do *timestamp* de uma transação.

4 – Validação

- Nenhuma verificação é realizada enquanto a transação está sendo executada.
- As atualizações na transação não são aplicadas diretamente aos itens de dados até que a transação atinja seu final.
- Durante a execução da transação, todas as atualizações são aplicadas a cópias locais dos itens de dados que são mantidos para a transação.
- Ao final da execução da transação, a fase de validação verifica se qualquer uma das atualizações da transação viola a serialização.
- Se a serialização não for violada, a transação é concluída; caso contrário, a transação é abortada e posteriormente reiniciada.

4 – Validação

Foram definidas três fases diferentes nas quais uma transação pode se encontrar em um determinando momento:

– **Fase de leitura:** toda transação (Ti) inicia nesta fase, onde são lidos os valores dos itens de dados necessários e armazenados em variáveis locais de Ti. Não há alteração no banco de dados;

– **Fase de validação:** a transação Ti inicia um teste de validação para determinar se pode atualizar os itens no banco de dados, sem causar inconsistência;

– **Fase de escrita:** caso obtenha sucesso na fase de validação, a atualização é aplicada no banco de dados. Caso contrário, a transação é desfeita.

4 – Validação

No protocolo com base em validação é necessário saber quando ocorreram cada uma das fases da transações Ti. São utilizados três valores de *timestamp*, chamados, respectivamente, **Start(Ti), Validation(Ti)** e **Finish(Ti).**

A ordem de serialização é determinada usando o protocolo de ordenação por timestamp, usando o valor de Validation(Ti), em vez de Start(Ti), buscando a diminuição do tempo de resposta na resolução do conflito.

O teste de validação para Ti exige que, para todas as transações Tj com Validation(Ti) < Validation(Tj), uma das duas condições a seguir sejam realizadas:

5 - Granularidade

- Transações diferentes podem acessar itens de dados com granularidades diferentes.
- Modelo de hierarquização dos dados, representada graficamente em árvores de quatro níveis: na raiz é representado o banco de dados como um todo; no segundo nível, o BD é dividido em áreas; no terceiro, os nós são do tipo arquivo e por último, os nós representam registros.
- Cada nó da árvore pode ter bloqueio individual nos modos exclusivo e compartilhado.
- Quando um nó é bloqueado de um tipo por uma transação, imediatamente todos os seus filhos serão bloqueados com o mesmo tipo de forma implícita.
- Uma nova classe de bloqueio é introduzida, conhecida como modo de bloqueio intencional, que será aplicado em um nó que possui um descendente que está bloqueado com algum outro modo de bloqueio de forma explicita.

5 - Granularidade

O modo intencional é dividido em três outros tipos, para indicar qual o tipo de bloqueio explícito aplicado. São eles:

– IS: compartilhado intencional;
– IX: exclusivo intencional;
– SIX: compartilhado e exclusivo intencional, indica que o nó em questão está bloqueado no tipo compartilhado e o bloqueio exclusivo é feito explicitamente em um dos seus nós filhos.

5 - Granularidade

- Utiliza o protocolo de duas fases e obedece a matriz de compatibilidade abaixo:

	IS	***IX***	***S***	***SIX***	***X***
IS	VERDADEIRO	VERDADEIRO	VERDADEIRO	VERDADEIRO	FALSO
IX	VERDADEIRO	VERDADEIRO	FALSO	FALSO	FALSO
S	VERDADEIRO	FALSO	VERDADEIRO	FALSO	FALSO
SIX	VERDADEIRO	FALSO	FALSO	FALSO	FALSO
X	FALSO	FALSO	FALSO	FALSO	FALSO

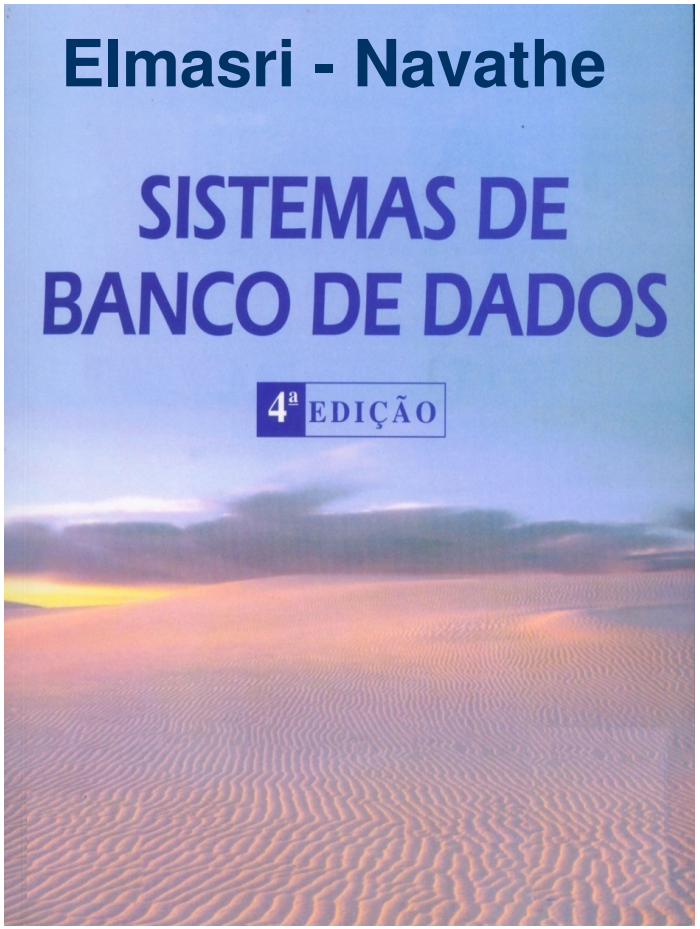
Desvantagem: é possível a ocorrência de deadlock.

6 - Índices

- Utilização do bloqueio de 2 fases para índices.
- Índices correspondentes às páginas dos discos.
- Bloqueio da raiz -> bloqueio da página toda
  - Bloqueio no filho leitura -> bloqueio nó pai liberado

7 – Outras opções

- Inserção, remoção e registro fantasma
  - Inserção de um novo ítem completa
  - Remoção de um item bloqueado
  - Registro fantasma: um item sendo incluido em uma transação satisfaz a a condição que um conjunto de registros acessados por outra transação deveria satisfazer.
- Uma linha da tabela
  - Transações interativas leelm de uma entrada ou escrevem uma saida em um dispositivo interativo, tal como um monotir de tela, antes que elas sejam efetivadas.
- Travas
  - Garantir a integridade fisica de uma página enquanto ela esta sendo escrita do buffer para o disco .



Atenção:

Leitura do

Capítulo 18:

Técnicas de Controle de Concorrência