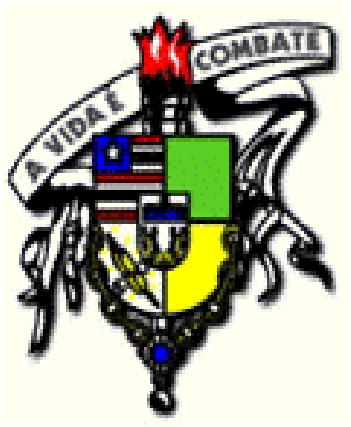


UNIVERSIDADE FEDERAL DO MARANHÃO - UFMA

Banco de Dados II

Recuperação

Carlos Eduardo **Portela** Serra de Castro

*

Sumário

1 – Conceitos

2 – Atualização adiada

3 – Atualização imediata

4 – Paginação *shadow*

5 – Aries

6 – BDs múltiplos

7 – Falhas catastróficas

Conceitos

1 – Algoritmos de recuperação

2 – *Caching* de blocos de disco

3 – Registro adiantado em *Log*

4 – *Checkpoints*

5 – *Rollback* de uma transação

Algoritmos de recuperação

- A recuperação de falhas existe para garantir as propriedades de atomicidade e durabilidade de transações.

- O sistema de recuperação (restauração) de falhas é responsável pela restauração do banco de dados para um estado consistente – que havia antes da ocorrência da falha.

- O Sistema deve manter informações sobre as atualizações do BD em separado (LOG).

Estratégia típica

- Perda por falha catastrófica: reconstrução (REDO)

  Backup ----------> estado consistente mais próximo da falha

  LOG ----------> até o instante da falha

- O BD tornou-se inconsistente: reverter mudanças (UNDO)

  BD inconsistente -------> BD consistente

Técnicas para recuperação de falhas não catastróficas

- Atualização adiada

  Algoritmo NO-UNDO/REDO

- Atualização imediata

  Algoritmo UNDO/REDO

Atualização adiada

- Somente atualiza o banco de dados físico depois que a transação alcança o ponto de efetivação:

  1 – antes do *commit* os registros são atualizados nos *buffers*;

  2 – durante *commit* os registros são gravados primeiro nos LOGS e depois no banco de dados.

  3 - não é necessário Undo e o Redo é necessário em alguns casos: No-Undo/Redo

Atualização imediata

- O banco de dados é atualizado antes do *commit*

  1 – gravação no *log* antes do BD.

  2 – se uma transação falhar antes do ponto de efetivação a transação deverão ser revertidas (usando os registros do *log*).

  3 – atualização imediata Undo e Redo necessários: Undo/Redo

  4 – variação: UNDO/NO-REDO

Caching de blocos de disco

- *Cache* de SGBD: buffer na memória principal sob controle do SGBD
- Catálogo da *Cache*: controla que itens do BD estão em qual buffer. Quando é requerido um item que não está na "*cache*", provoca a paginação.
- *Dirty Bit*: indica se um item de dado na *cache* foi
- atualizado (1) ou não (0).
- *In-place updting*
- *Shadowing*
  - Imagem anterior (BFIM): valor antigo do dado
  - Imagem posterior (AFIM): valor novo do dado

Registro adiantado em log

- *(Write-ahead log – WAL)*
- *Quando as atualizações são "realizadas no local", é necessário preservar os valores anteriores dos dados caso seja necessário reverter as operações.*
- *O mecanismo de recuperação deve garantir que a BFIM do item de dado seja registrada em uma entrada de log e que essa entrada seja trasnferida para o disco antes que a BFIM seja sobrescrita pela AFIM.*

Técnicas de Gerência de *Buffer*

- *NO-STEAL*
  - um bloco na *cache* utilizado por uma transação T_x não pode ser gravado no BD antes do *commit* de T_x
    - vantagem
      - processo de *recovery* mais simples (NO-UNDO)
      - evita dados de transações inacabadas sendo gravadas no BD
- *STEAL*
  - um bloco na *cache* utilizado por uma transação T_x pode ser gravado no BD antes do *commit* de T_x
    - necessário se algum dado é requisitado do BD por outra transação e não há blocos disponíveis na *cache*
    - vantagem
      - evita a necessidade de um espaço grande em buffer para o armazenamento em memória principal de todas páginas atualizadas

Técnicas de Gerência de *Buffer*

- *FORCE*
  - os blocos que mantêm dados atualizados por uma transação T_x são imediatamente gravados no BD quando T_x alcança o *commit*
  - vantagem
    - garante a durabilidade de T_x o mais cedo possível
- *NO-FORCE*
  - os blocos que mantêm dados atualizados por T_x não são imediatamente gravados no BD quando T_x alcança o *commit*
  - vantagem:
    - blocos atualizados podem permanecer na *cache* e serem utilizados por outras transações, após o *commit* de T_x (reduz custo de acesso a disco)

Checkpoints

- Momento em que o SGBD grava no BD todas as atualizações feitas por transações
  - inclusão de um registro de checkpoint no *Log*

- Periodicidade:
  - Em tempo
  - Em número de transações

Procedimento de execução de *checkpoint*

1. Suspensão de todas as transações;
2. Gravação (forçada) dos blocos atualizados da *cache* no BD;
3. Inserção de um registro *checkpoint* no *Log* e sua gravação em disco;
4. Retomada da execução das transações.

- Vantagem da técnica de *checkpoint*
  - Transações *committed* antes do *checkpoint* não precisam sofrer REDO em caso de falha
    - Elas já estão garantidamente no BD

Rollback

- Rollback de Transações (Undo): É necessário se uma falha ocorrer depois do BD ter sido atualizado.
    - Qualquer item de dado atualizado pela transação deve voltar a seu valor anterior.

- Rollback em Cascata: Se uma transação T é desfeita e uma transação S leu algum dado atualizado por T, S também tem que ser desfeita e assim por diante.

Ilustrando a reversão (*rollback*) em cascata (um processo que nunca ocorre em planos restritos ou livres de cascata – *cascadeless*).

(a) As operações *read* (ler) e *write* (gravar) das três transações.

(b) *Log* (histórico) do sistema no ponto da queda (*crash*).

(c) Operações antes da queda.

(a)

T_1	T_2	T_3
ler_item(A)	ler_item(B)	ler_item(C)
ler_item(D)	escrever_item(B)	escrever_item(B)
escrever_item(D)	ler_item(D)	ler_item(A)
	escrever_item(D)	escrever_item(A)

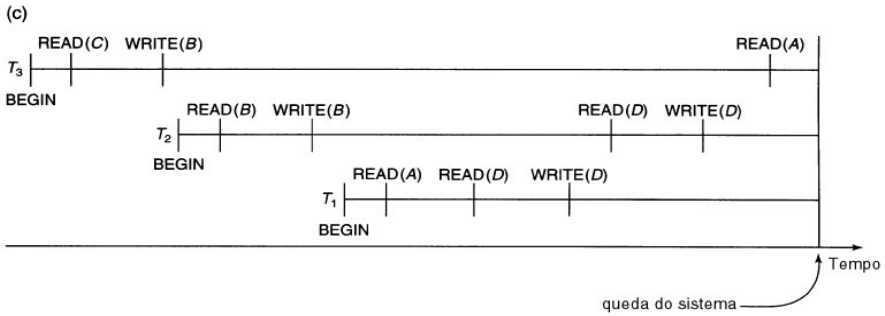
(b)

		A	B	C	D
		30	15	40	20
	[start_transaction, T_3]				
	[ler_item, T_3, C]				
*	[escrever_item, T_3, B, 15, 12]		12		
	[start_transaction, T_2]				
	[ler_item, T_2, B]				
**	[escrever_item, T_2, B, 12, 18]		18		
	[start_transaction, T_1]				
	[ler_item, T_1, A]				
	[ler_item, T_1, D]				
	[escrever_item, T_1, D, 20, 25]				25
	[ler_item, T_2, D]				
**	[escrever_item, T_2, D, 25, 26]				26
	[ler_item, T_3, A]				

← queda do sistema

* T_1 é revertida porque não alcançou seu ponto de efetivação.
** T_2 é revertida porque lê o valor do item B escrito por T_3.

(c)



Recuperação com Atualização Adiada

Ambiente Monousuário

- Protocolo
  - Uma transação não pode modificar o BD até seu commit
  - Uma transação não pode chegar ao commit até que todas as operações de atualização sejam gravadas no log e o log no disco
- Procedimento
  - Usar duas listas de transações: as transações acabadas desde o último checkpoint e as transações ativas
  - Aplicar a operação Redo para todas as operações write_item das transações acabadas no log, na ordem na qual elas foram gravadas
  - Recomeçar as transações ativas

Recuperação com Atualização Adiada

Ambiente Multi-usuário

- Protocolo
  - Depende do protocolo usado no controle de concorrência
  - Assumir protocolo de duas fases com bloqueios antes do início da execução, guardando-os até o commit
- Procedimento
  - Usar duas listas de transações: as transações acabadas desde o último checkpoint e as transações ativas
  - Aplicar a operação Redo para todas as operações write_item das transações acabadas no log, na ordem na qual elas foram gravadas
  - As transações ativas e não acabadas são canceladas e devem ser re-submetidas

Recuperação com Atualização Adiada

Desvantagem

- Limita a execução concorrente das transações porque itens ficam bloqueados até o commit das transações.

Vantagens

- Uma transação não grava as modificações no BD até o commit. Logo, uma transação nunca é desfeita por causa de falha
- Uma transação nunca vai ler o valor de um item gravado por outra não acabada, porque os itens estão bloqueados. Logo, não ocorrerá rollback em cascata

Recuperação com Atualização Imediata

- Dois tipos de Algoritmos
  - Undo/No-Redo: Se a técnica de recuperação garante que todas as atualizações são gravadas no BD (disco) antes do commit da transação, não é necessário Redo
  - Undo/Redo: Se as modificações são gravada no BD (disco) depois do commit da transação
    - Caso mais geral e mais complexo

Recuperação com Atualização Imediata

Ambiente Monousuário

- Procedimento
  - Usar duas listas de transações: as transações acabadas desde o último checkpoint e a transação ativa (máximo 1)
  - Aplicar o procedimento Undo: para desfazer todas as operações write_item da transação ativa no log
  - Aplicar a operação Redo: para todas as operações write_item das transações acabadas na ordem na qual elas foram gravadas no log

Recuperação com Atualização Imediata

Ambiente Monousuário

- Undo
  - Desfazer uma operação write_item consiste em examinar sua entrada no log: [write_item, T, X, valor_ant, valor_novo] e colocar o valor do item X no BD como“valor_ant” (ImAn)
  - Para desfazer as operações write_item de uma ou mais transações do log, deve-se proceder na ordem inversa de gravação no log

Recuperação com Atualização Imediata

Ambiente Multiusuário

- Protocolo
  - Assume-se que o log inclui checkpoints e o protocolo de concorrência como o de duas fases
- Procedimento
  - Usar duas listas de transações: as transações acabadas desde o último checkpoint e as transações ativas
  - Aplicar a operação Undo para todas as operações write_item das transações ativas, na ordem inversa de suas gravações no log
  - Aplicar a operação Redo para todas as operações write_item das transações acabadas, na ordem na qual elas foram gravadas no log

Paginação Shadow

- Ambiente Mono-usuário: não é necessário o log
- Ambiente Multi-usuário: pode-se usar o log se o método de controle de concorrência usar.
- Pressupõe-se que o BD seja composto por "n" páginas de tamanho fixo.
- Uma tabela de páginas com "n" entradas é construída onde pag_tabi = pag_bdi

Begin_Transaction =>tab_pag_corrente -->tab_pag_imagem

↑

(Nunca é modificada durante a transação)

Paginação Shadow

- Vantagem
  - Não há necessidade de Undo ou Redo de operações
- Desvantagens
  - Páginas atualizadas mudam de localização no disco, impedindo de manter juntas páginas relacionadas
  - Se a tabela de páginas é grande, o tempo para gravar as tabelas de páginas imagem no commit é significativo
  - Garbage Collection (liberação de páginas antigas) é necessário após o commit

ARIES

- **Algorithms for Recovery and Isolation Exploiting Semantics**, ou **ARIES,** é um algoritmo de recuperação projetada para funcionar com uma abordagem no-force, steal; Ele é usado pelo IBM DB2 e pelo Microsoft SQL Server entre outros sistemas de banco de dados.

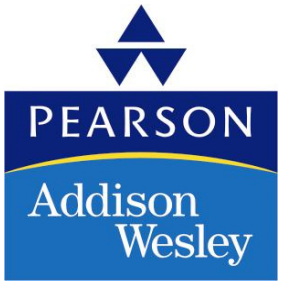
- Três grande princípios norteiam o ARIES:
  - Write ahead logging: Qualquer alteração em um objeto é o primeiro registrada no log, e o log deve ser escrito no armazenamento persistente antes que as alterações no objeto sejam gravados no disco..
  - Repeating history during Redo: No restart depois de um acidente, o ARIES reconstitui as ações de um banco de dados antes do acidente e traz o sistema volta ao estado exato em que estava antes do acidente. Em seguida, ele desfaz as operações ainda ativas no momento do acidente.
  - Logging changes during Undo: As alterações feitas no banco de dados durante operações undo são registrados no log para garantir tal ação não seja repetido em caso de reinicializações repetidas.

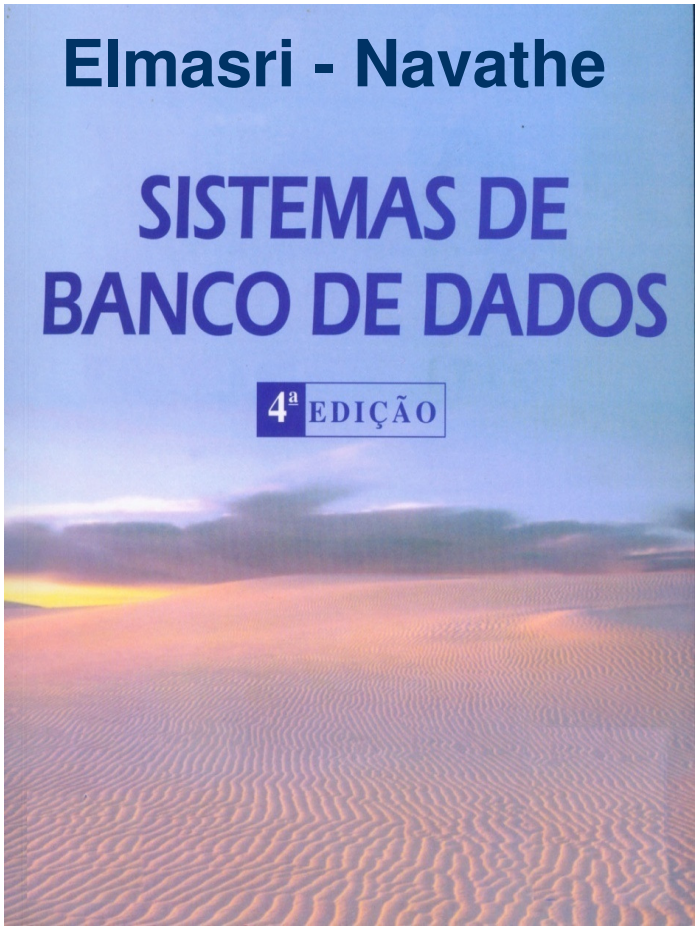
BDs Múltiplos

- Gerenciador de recuperação global.
- Protocolo de efetivação em duas fases.

Backup e Recuperação após Falhas Catastróficas

- Exemplo: pane no disco
  - Principal Técnica: backup do BD
    - Cópia periódica do BD e do log em outro meio de armazenamento (fitas)
    - É necessário backup do log para não perder as transações efetuadas desde o último checkpoint
  - Um log é inicializado após cada operação de backup. Logo, para recuperar após uma falha no disco:
    - O BD é recriado a partir do último backup
    - Os efeitos de todas as transações committed, cujas operações foram gravadas no log, são reconstruídos





Atenção:

Leitura do

Capítulo 19: Técnicas de Recuperação de Banco de Dados