Técnicas de Recuperação de falhas

Laboratório de Bancos de Dados

Introdução

- Serão discutidas algumas técnicas de recuperação de falhas → log
- Ênfase na descrição conceitual das abordagens. Técnicas específicas nos manuais. Vinculadas às técnicas de concorrência.

Agenda

- Conceitos de recuperação: categorização de algoritmos, bufferização, gravação adiantada, checkpoint e reversão de transações.
- Técnicas de recuperação baseadas na atualização adiada.
- Técnicas de recuperação baseadas na atualização imediata.
- Paginação Shadow.
- O algoritmo ARIES.
- Backup de BDs e recuperação de falhas catastróficas.

Conceitos de Recuperação: Categorização de algoritmos

- A recuperação de transações que falharem significa que o BD será restaurado para o estado de consistência mais recente → manter informação sobre as alterações → log.
- Estratégia típica de recuperação:
  - Se houver um dano extenso, falha catastrófica, restaura uma cópia anterior (em fita) e o reconstrói, refaz as transações armazenadas no log até o momento da falha.
  - No caso de falha não catastrófica, reverte mudanças que causaram inconsistências desfazendo algumas operações e refazendo outras para restaurar um estado consistente.

Conceitos de Recuperação: Categorização de algoritmos

- Duas técnicas principais de recuperação: atualização adiada; atualização imediata.
- Atualização adiada: somente se atualiza o BD (disco) depois da efetivação da transação. Antes da efetivação os registros são atualizados no buffer. Durante a efetivação, as atualizações são, primeiro registradas persistentemente no log e, então, gravadas no BD. UNDO não é necessário. Pode ser necessário REDO. Algoritmo NO-UNDO/REDO

Conceitos de Recuperação: Categorização de algoritmos

- Atualização imediata: O BD pode ser atualizado por algumas operações de uma transação antes do seu ponto de efetivação. Essas operações estarão registradas normalmente no log em disco (gravação forçada) antes que elas fossem aplicadas no BD para a recuperação. Se uma transação falhar antes da sua efetivação os efeitos de suas operações devem ser desfeitos. No caso geral as operações UNDO e REDO são necessárias. Algoritmo UNDO/REDO.
- Variação: todas as atualizações são registradas no BD antes que uma transação seja efetivada → requer apenas UNDO → Algoritmo UNDO/NO-REDO

Bufferização de Blocos de Disco

- O processo de recuperação fortemente entrelaçado com as funções do SO.
- O caching de páginas em disco é tradicionalmente função do SO, mas por questões de eficiência ela é manipulado pelo SGBD por meio de chamadas de rotinas de baixo nível do SO.
- Coleção de buffers sob controle do SGBD → cache do SGBD. Catálogo para o cache, controla quais itens do BD estão nos buffers. Cada entrada da forma: <endereço das páginas do disco, localização do buffer>
- Associado a cada buffer → dirty bit, para indicar se foi modificado (valor 1). Quando a página tiver que sair ela é gravado no disco dependendo do bit. Outro bit pin-unpin; define se a página não pode ser escrita de volta no disco.
- Duas estratégias empregadas para gravação de buffers no disco: atualização no local (uma única versão); e shadowing grava em um local diferente do disco (diferentes versões).

Registro Adiantado em Log, Roubado/Não-Roubado e Forçado/Não-Forçado

- Atualização no local → necessário log para recuperação. O mecanismo de recuperação deve garantir que a BFIM do item seja registrada no log e que essa entrada do log seja transferida (flushed) para o disco, antes que a BFIM seja sobrescrita pela AFIM no BD (disco) → registro adiantado em log.
- Dois tipos de entrada em log para um write: necessárias para UNDO; e necessárias para REDO.
- Uma entrada do tipo REDO inclui o novo valor (AFIM) do item gravado; e a do tipo UNDO inclui o valor antigo (BFIM). Um algoritmo UNDO/REDO ambos tipos de entradas devem ser combinadas.

Registro Adiantado em Log, Roubado/Não-Roubado e Forçado/Não-Forçado

- O cache do SGBD inclui não apenas blocos de dados, mas também blocos de índice e os blocos de log. Com a abordagem de **registro adiantado no log**, os blocos de log que contiverem registros de log associados a uma atualização em em um bloco de dados em particular deverão ser gravados primeiro no disco, antes que o bloco de dados.
- Quando uma página do BD poderá ser gravada em disco a partir do cache:
  - Abordagem não-roubada (no-steal): Uma página em cache atualizada não pode ser gravada antes que a transação se efetive. O bit pin-unpin estará ligado. Se o protocolo permitir o buffer atualizado antes da efetivação da transação → roubado.
  - Abordagem forçada: todas as páginas atualizadas são imediatamente escritas no disco quando a transação se efetivar. O contrário é não forçado.

Registro Adiantado em Log, Roubado/Não-Roubado e Forçado/Não-Forçado

- Esquema de recuperação de atualização adiado segue uma abordagem no-steal. Normalmente os SBDs empregam uma estratégia roubada/não-forçada. Roubada → evita a necessidade de um espaço muito grande de buffer. Não-forçada → uma página atualizada por uma transação efetivada deverá ainda estar no buffer quando outra transação necessitar de atualização → economia de E/S
- Recuperação usando atualização no local as entradas a serem recuperadas devem ser permanentemente registradas no disco de log, antes que as alterações sejam aplicadas no BD.

Registro Adiantado em Log, Roubado/Não-Roubado e Forçado/Não-Forçado

- Para facilitar o processo de recuperação → são mantidas listas relacionadas às transações: lista de transações ativas (iniciaram mas não foram efetivadas); lista de transações efetivadas e abortadas desde o último “checkpoint”.

Checkpoints no Log de Sistema

- Um registro de “checkpoint” é escrito periodicamente dentro do log, no ponto que o sistema grava no BD no disco todos os buffers do SGBD que tivessem sido modificadas. Todas as transações que tiverem um “commit” no log, antes de um “checkpoint”, não necessitarão ter suas operações WRITE refeitas no caso de falha.
- O gerenciador de recuperação deve decidir quando submeter um checkpoint.

Checkpoints no Log de Sistema

- Submeter um “checkpoint” consiste nas seguintes ações:
  1. Suspender a execução das transações temporariamente
  2. Forçar a gravação no disco de todos os buffers na memória principal que tenham sido alterados.
  3. Escrever um registro de “checkpoint” no log e forçar a gravação de log no disco.
  4. Reassumir a execução das transações.
- Conseqüência do passo 2, o registro de checkpoint pode incluir também informações, como: uma lista das transações ativas e as localizações de todos os registros, do mais antigo ao mais recente no log, de cada transação ativa.

Reverter uma Transação (Rollback)

- Se uma transação falhar, depois de atualizar o BD, pode ser necessário reverter a transação. As entradas de log do tipo desfazer (UNDO) são usadas para restaurar os valores antigos dos itens de dados que deverão ser revertidos.
- Pode ser gerada uma reversão em cascata, quando o protocolo de recuperação garantir planos recuperáveis, mas não garantir planos restritos ou livres de cascata.
- Reversão em cascata é custosa → a maioria dos sistemas evita este fenômeno.
- A Fig. 19.1, exemplo no qual a reversão em cascata é exigida.
- As operações de ler_item no log apenas para determinar se s reversão em cascata das transações adicionais

Figura 19.1 Ilustrando a reversão (*rollback*) em cascata (um processo que nunca ocorre em planos restritos ou livres de cascata – cas*cadeless*).

(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)

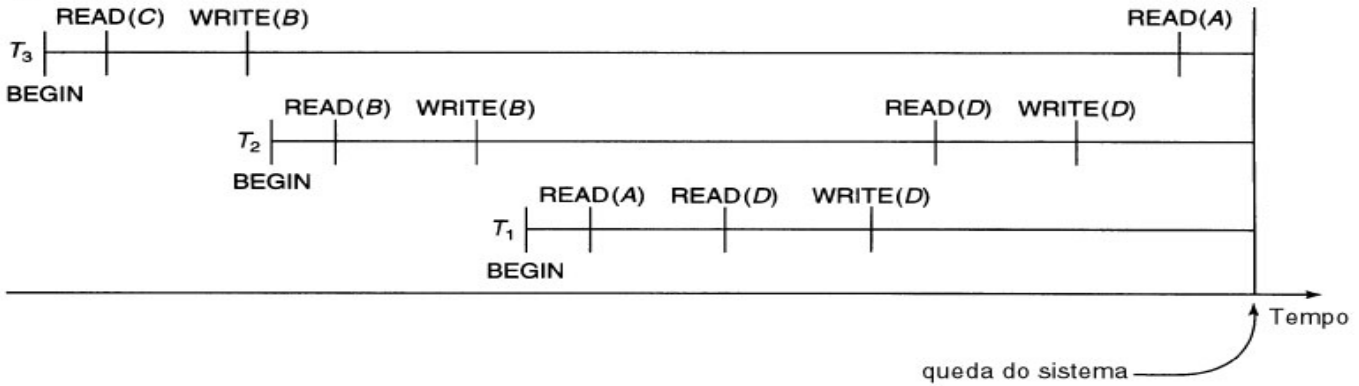
(b)

		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_3 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)



Técnicas de Recuperação Baseadas na Atualização Adiada

- A idéia destas técnicas é postergar qualquer atualização real no BD até que a transação complete sua execução com sucesso e alcance seu ponto de efetivação. Durante a execução, as atualizações são registradas apenas no log e nos buffers.
- Quando a transação alcançar seu ponto de efetivação e se forçar a gravação de log no disco, as atualizações serão registradas no BD.
- Falha antes da efetivação não --> desfazer nenhuma operação.

Técnicas de Recuperação Baseadas na Atualização Adiada

- Um protocolo deste tipo pode ser declarado:
  - Uma transação não pode mudar o BD em disco atá que alcance seu ponto de efetivação.
  - Uma transação não alcança seu ponto de efetivação até que todas as suas operações de atualização sejam registradas no log e até que seja forçada a gravação do log no disco. --> protocolo de registro adiantado no log (WAL)
- Algoritmo associado NO-UNDO/REDO.
- REDO será necessário no caso de o sistema falhar depois que uma transação for efetivada, mas antes que todas as mudanças sejam registradas no BD (disco).

Recuperação usando atualização adiada em um ambiente Monousuário

- Procedimento RDU_S: Usa duas listas de transações: as transações efetivadas desde o último checkpoint e as transações ativas (pelo menos uma transação). Aplica a operação REDO para todas as operações escrever_item das transações efetivadas no log, seguindo a seqüência em que elas foram escritas no log. Reiniciar as transações ativas.
- Procedimento REDO: Refazer uma operação de escrita consiste em examinar a entrada log[escrever_item, T, X, novo_valor] e apontar o valor de X para o novo_valor (AFIM).

Recuperação usando atualização adiada em um ambiente Monousuário

- A operação REDO deve ser idempotente.
- A Fig. 19.2 mostra um exemplo de recuperação em um ambiente monousuário.

Figura 19.2 Um exemplo de recuperação usando atualização em ambiente monousuário. (a) As operações READ e WRITE de duas transações. (b) O *log* do sistema no instante da queda.

(a)

T_1	T_2
ler_item(*A*)	ler_item(*B*)
ler_item(*D*)	escrever_item(*B*)
escrever_item(*D*)	ler_item(*D*)
	escrever_item(*D*)

(b)

[inicia_transacao, T_1]
[ler_item, T_1, *D*, 20]
[confirma, T_1]
[inicia_transacao, T_2]
[ler_item, T_2, B, 10]
[escrever_item, T_2, D, 25] ← queda do sistema

As operações [escrever_item, ...] de T_1 são refeitas.
As entradas no *log* de T_2 são ignoradas pelo processo de recuperação.

Atualização adiada com Execução Concorrente em um Ambiente Multiusuário

- O processo de recuperação pode ser mais complexo dependendo do protocolo usado para o controle de concorrência.
- Considere o controle de concorrência usa bloqueio estrito em duas fases → o bloqueio dos itens permanecerá em efeito até que a transação alcance seu ponto de efetivação
- Um possível algoritmo neste caso RDU_M

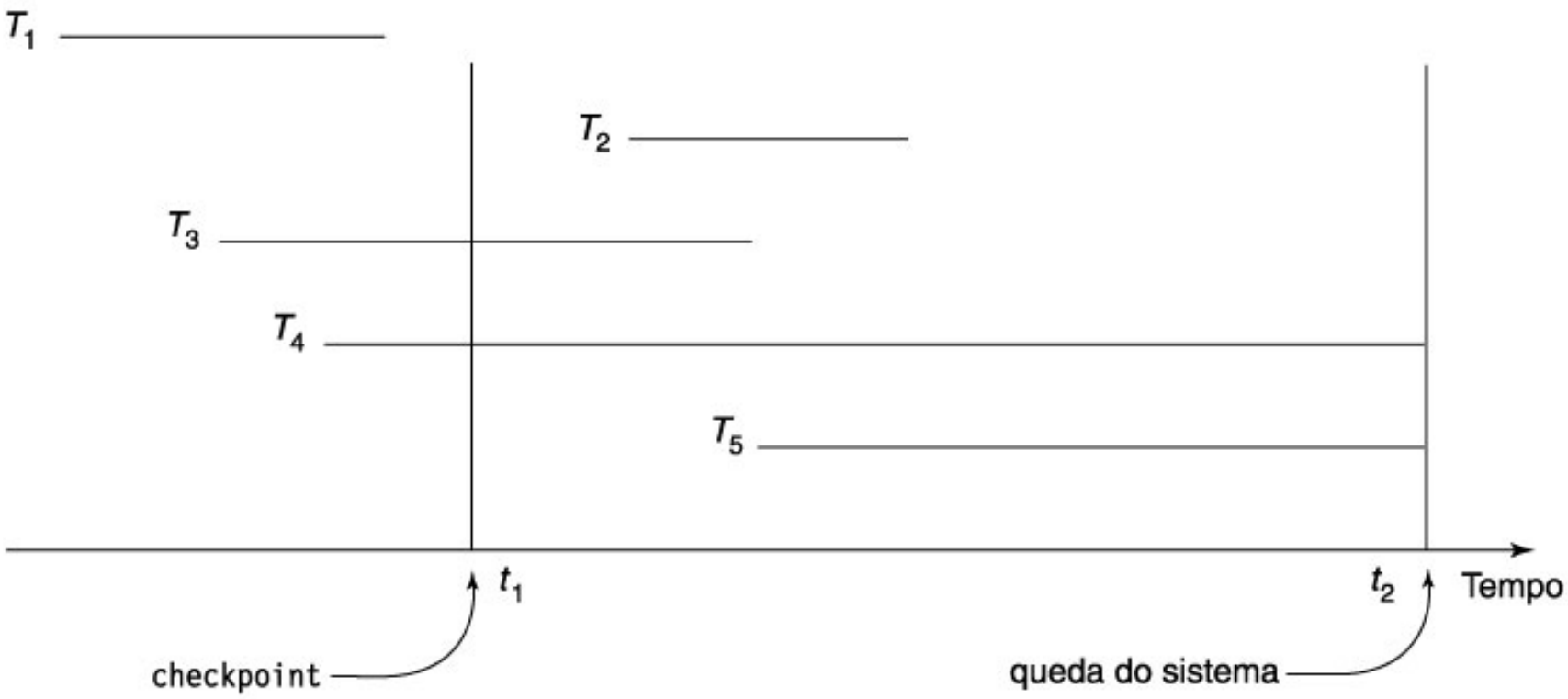
Atualização adiada com Execução Concorrente em um Ambiente Multiusuário

- RDU_M: Usa as duas listas: as transações T efetivadas desde o último checkpoint (lista de efetivação) e a transações T´ ativas. REDO (definido acima) todas as operações WRITE das transações efetivadas no log, na seqüência em que elas se encontraram no log. As transações que estiverem ativas, mas não efetivadas, serão canceladas e deverão ser novamente submetidas → Algoritmo NO-UNDO/REDO.

Atualização adiada com Execução Concorrente em um Ambiente Multiusuário

- Veja Fig. 19.3.
- Pode-se tornar o algoritmo mais eficiente. Se um item X tiver sido atualizado por mais de uma transação efetivada desde o último checkpoint, apenas será necessário REDO a última atualização de X do log durante a recuperação.
- Desvantagens: limita a concorrências das transações porque todos os itens permanecem bloqueados até que a transação alcance seu ponto de efetivação. Exige espaço excessivo de buffer.

Figura 19.3 Um exemplo de recuperação em ambiente multiusuário.



Atualização adiada com Execução Concorrente em um Ambiente Multiusuário

- Beneficio principal do método é que as operações de transação nunca terão de ser desfeitas por duas razões:
  - Uma transação não registrará nenhuma mudança no BD em disco até imediatamente após seu ponto de efetivação. Portanto, uma transação nunca será revertida em função de falhas durante a execução de transações.
  - Uma transação nunca lerá o valor de um item escrito por uma transação ainda não-efetivada, porque seus itens permanecerão bloqueados até a efetivação. Não há reversão em cascata.
- Veja um exemplo na Fig 19.4

Figura 19.4 Um exemplo de recuperação que usa atualização adiada com transações concorrentes. (a) As operações READ e WRITE de quatro transações. (b) *Log* do sistema no ponto de queda.

(a)

T_1	T_2	T_3	T_4
ler_item(A)	ler_item(B)	ler_item(A)	ler_item(B)
ler_item(D)	escrever_item(B)	escrever_item(A)	escrever_item(B)
escrever_item(D)	ler_item(D)	ler_item(C)	ler_item(A)
	escrever_item(D)	escrever_item(C)	escrever_item(A)

(b)

[inicia_transacao, T_1]
[escrever_item, T_1, D, 20]
[confirma, T_1]
[checkpoint]
[inicia_transacao, T_4]
[escrever_item, T_4, B, 15]
[escrever_item, T_4, B, 20]
[confirma, T_4]
[inicia_transacao, T_2]
[escrever_item, T_2, B, 12]
[inicia_transacao, T_3]
[escrever_item, T_3, A, 30]
[escrever_item, T_2, D, 25] ← queda do sistema

T_2 e T_3 são ignoradas porque não alcançaram seus pontos de efetivação.
T_4 é refeita porque seu ponto de efetivação está depois do último *checkpoint* do sistema.

Técnicas de Recuperação Baseadas em Atualização Imediata

- A atualização é feita sem esperar a efetivação.
- Entretanto deve ser respeitado o protocolo de registro adiantado no log (WAL).
- Devem existir condições para desfazer os efeitos de operações de atualização.
- Duas técnicas:
  - UNDO/NO-REDO: garante que todas as atualizações de uma transação seja registrada no BD (disco) antes que a transação efetivar.
  - Mas se for permitido que uma transação se efetive antes que todas as suas mudanças sejam escritas no BD, temos o caso mais geral : UNDO/REDO

Recuperação UNDO/REDO Baseada em Atualização Imediata em um Ambiente Monousuário

- Procedimento RIU_S:
  1. Usar duas listas de transações mantidas: transações efetivadas desde o último checkpoint e as transações ativas (pelo menos uma delas irá falhar → monousuário)
  2. Desfazer todas as operações *escrever_item* da transação ativa do log, com o procedimento UNDO.
  3. Refazer as operações escrever_item das transações efetivadas do log, na seqüência em que elas foram gravadas, usando o procedimento REDO.
- UNDO(WRITE_OP): Desfazer uma operação de escrita consiste em examinar a entrada log[escrever_item, T, X, valor_antigo, novo_valor] e apontar o valor de X para o valor_antigo (BFIM).

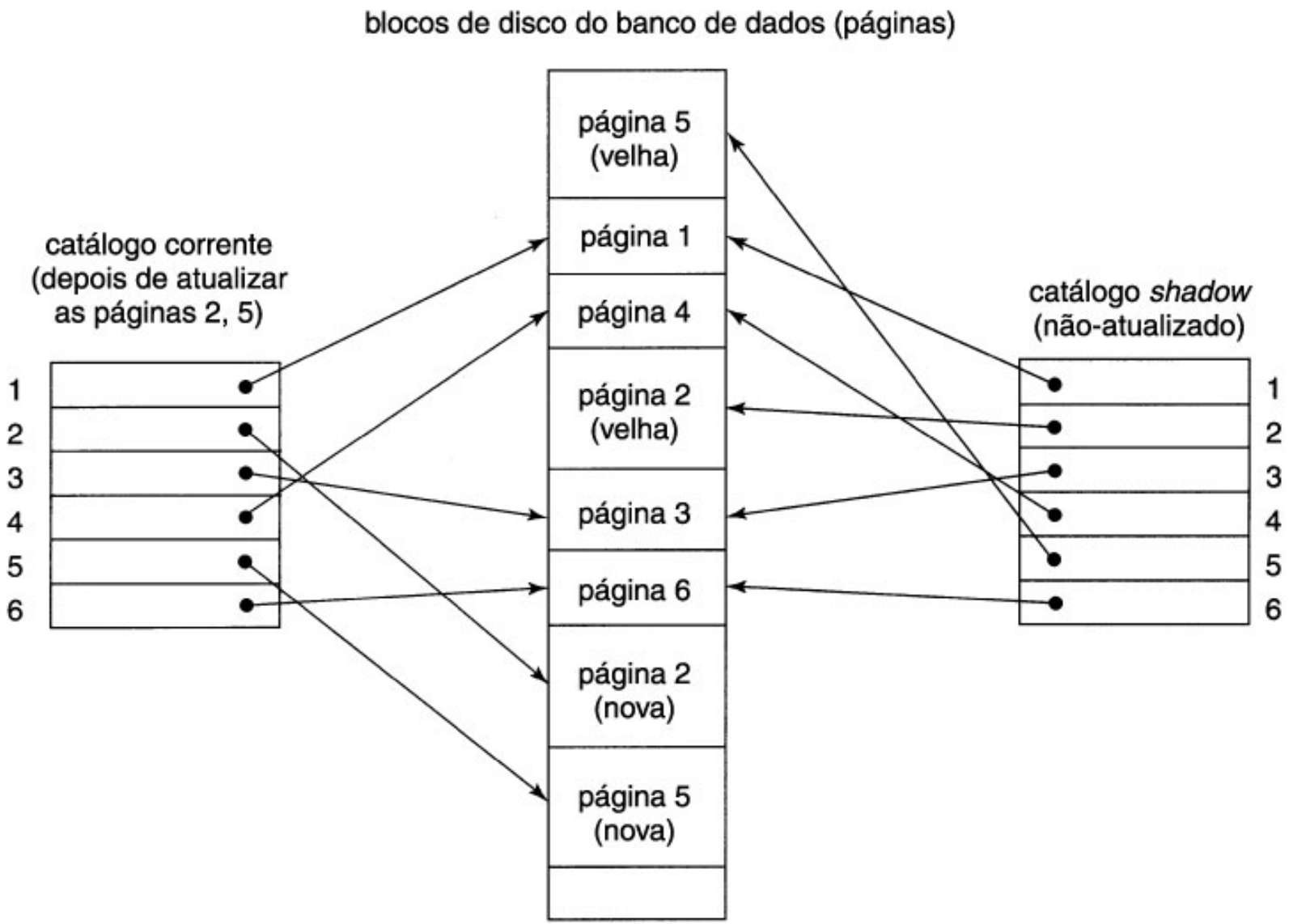
Recuperação UNDO/REDO Baseada em Atualização Imediata com Execução Concorrente

- Assumimos que o log inclui checkpoints e que o protocolo de concorrência produz plano restritos (ex: 2PL restrito). Que não gera reversão em cascata mas sim deadlocks → implica UNDO.
- Procedimento RIU_M
  1. Usar as duas listas (efetivadas e ativas)
  2. Desfazer todas as operações escrever_item feitas pelas transações ativas, usando o procedimento UNDO, na ordem inversa em que elas foram gravadas no log.
  3. Refazer todas as operações escrever_item das efetivadas do log na seqüência em que elas foram escritas no log.
- O passo 3 pode ser mais eficiente se for iniciado no fim do log e refizer apenas a última atualização de cada item X.

Paginação Shadow (Sombra)

- Não precisa de log em ambiente monousuário, mas em ambiente multiusuário sim (para o controle de concorrência). BD considerado como um número n de páginas de tamanho fixo. São usados dois catálogos: corrente e shadow.
- Quando se inicia uma transação, o catálogo corrente (aponta para as páginas de disco mais recentes ou correntes) é copiado em um shadow. Este último catálogo é salvo no disco, enquanto o catálogo corrente é usado pela transação.
- Uma operação de *escrever_item* gera uma nova cópia da página modificada e a cópia antiga não é sobrescrita. Veja Fig. 19.5. Páginas atualizadas pela transação, serão mantidas 2 versões. Versão antiga → catálogo shadow; nova versão → catálogo corrente.

Figura 19.5 Um exemplo de paginação *shadow*.



Paginação Shadow (Sombra)

- A recuperação de falhas → livra das páginas modificadas e descarta o catálogo corrente.
- O estado do BD antes da execução está totalmente disponível no catálogo shadow. Efetivar uma transação corresponde a descartar o catálogo shadow anterior. Esta recuperação não implica desfazer nem refazer itens → NO-UNDO/NO-REDO.

O Algoritmo de Recuperação ARIES

- Usa uma abordagem roubada/não-forçada para gravação e está baseado em três conceitos: 1) registro adiantado em log (WAL); 2)repetição do histórico durante o refazer; e 3) mudanças de log durante o desfazer.
- Repetição histórico: o sistema relê todas as ações tomadas pelo SBD antes da queda, para reconstruir o estado no momento da falha.
- Usando o log durante desfazer, evitará que torne a desfazer operações já desfeitas, falha durante recuperação e reinicio da recuperação.

Backup de BDs e Recuperação em Falhas Catastróficas

- Suposição até aqui que o log não seja perdido.
- Mas falhas catastróficas como quebra do disco?
- O log deve ser armazenado também em fita.
- O backup do BD é muito grande então a periodicidade é menor. O log deve ser gravado com maior freqüência.
- O BD deverá ser, primeiro, recriado a partir do último backup em fita. A seguir todas as transações efetivadas, cujas operações tenham sido registradas nas cópias backup do log do sistema, serão reconstruídas.

Figura 19.7 Um plano de execução exemplo e seu *log* correspondente.

[inicia_transacao, T_1]
[ler_item, T_1, A]
[ler_item, T_1,D]
[escrever_item, T_1,D, 20, 25]
[confirma, T_1]
[checkpoint]
[inicia_transacao, T_2]
[ler_item, T_2,B]
[escrever_item, T_2, B, 12, 18]
[inicia_transacao, T_4]
[ler_item, T_4,D]
[escrever_item, T_4, D, 25, 15]
[inicia_transacao, T_3]
[escrever_item, T_3, C, 30, 40]
[ler_item, T_4, A]
[escrever_item, T_4, A, 30, 20]
[confirma, T_4]
[ler_item, T_2, D]
[escrever_item, T_2, D, 15, 25] ← queda do sistema