Técnicas de Recuperação de Banco de Dados

André Luís Schwerz
Rafael Liberato Roberto

Tópicos

- Conceitos de Recuperação
- Técnicas de recuperação
  - atualização adiada
  - atualização imediata

Revisão

Log do sistema

```
[start_transaction, T]
[write_item, T, X, valor_antigo, valor_novo]
[read_item, T, X]
[commit, T]
[abort, T]
```

Schedules recuperáveis

- Um schedule S é recuperável se nenhuma T_i em S for concluída até que todas as transações que gravaram dados lidos por T_i tenham sido concluídas.

Recuperável

T1	T2
read(A)	
A = A – 20	
write(A)	
	read(A)
	A = A + 10
	write(A)
commit( )	
	commit( )

Não Recuperável

T1	T2
read(A)	
A = A – 20	
write(A)	
	read(A)
	A = A + 10
	write(A)
	commit( )
abort( )	

Conceitos de Recuperação

- Certas técnicas de recuperação são melhor usadas com métodos específicos de controle de concorrência.
- Nossa discussão é independente da técnica de controle de concorrência usada.

Conceito de Recuperação

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

- A recuperação a falhas de transação em geral significa que o banco de dados é **restaurado** ao estado consistente mais recente

- O sistema precisa manter dados sobre as mudanças realizadas por várias transações
  - Utiliza informações mantidas no **log do sistema**

- As estratégias de recuperação podem ser resumidas de acordo com o tipo de falha:
  - Falha Catastrófica
  - Falha Não Catastrófica

Falha catastrófica

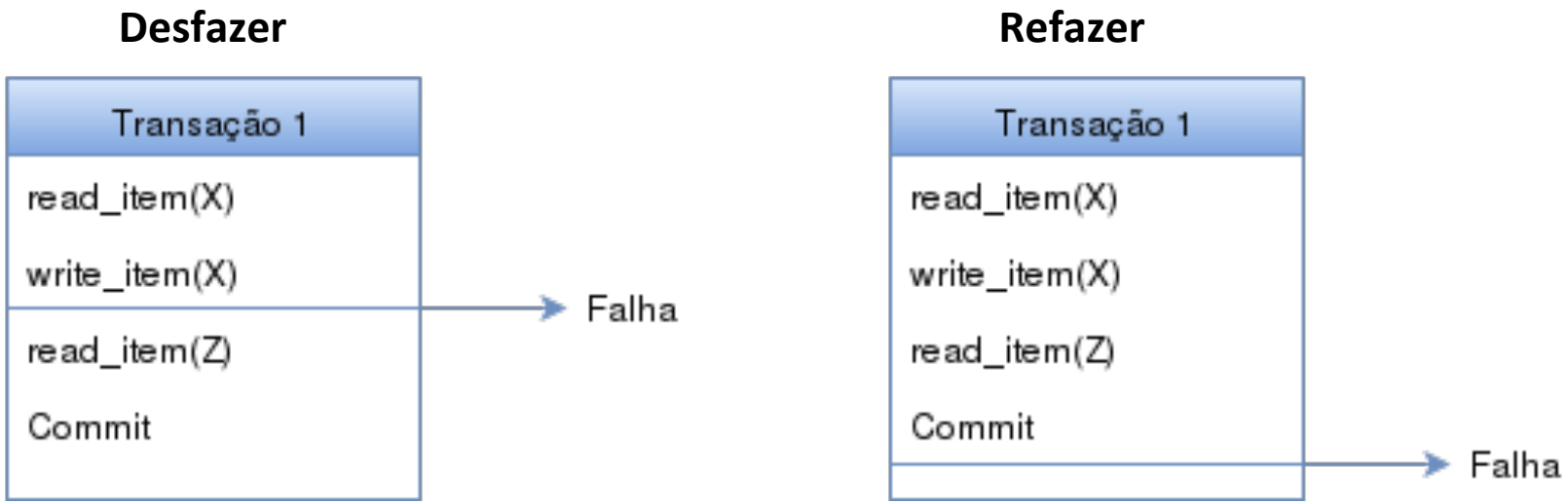
- Uma falha catastrófica é um dano extensivo a uma grande parte do banco de dados
  - Falhas no disco
  - Problemas físicos, roubos, incêndios, etc.
- O que fazer?
  - Restaurar uma cópia antiga do banco de dados
  - Reaplicar ou refazer as operações das transações confirmadas no log em backup até o momento da falha.

Falha não catastrófica

- Banco de dados **não** danificado fisicamente
  - Erro de transação ou do sistema
  - Erros locais ou condições de execução detectadas pela transação
  - Imposição de controle de concorrência
- Protocolo de recuperação não precisa de um backup completo do banco de dados
- O que fazer?
  - Entradas mantidas no log no disco são analisadas para determinar as ações apropriadas para recuperação.
- Exemplo:
  - Problema: suponha uma transação que atualizou alguns itens de dados no disco, mas não confirmou.
    - Suas alterações devem ser desfeitas ou revertidas.
  - Problema: suponha uma transação tiver sido confirmada, mas algumas operações de gravação ainda não tiveram sido gravadas em disco.
    - Suas operações devem ser refeitas para restaurar o estado consistente.

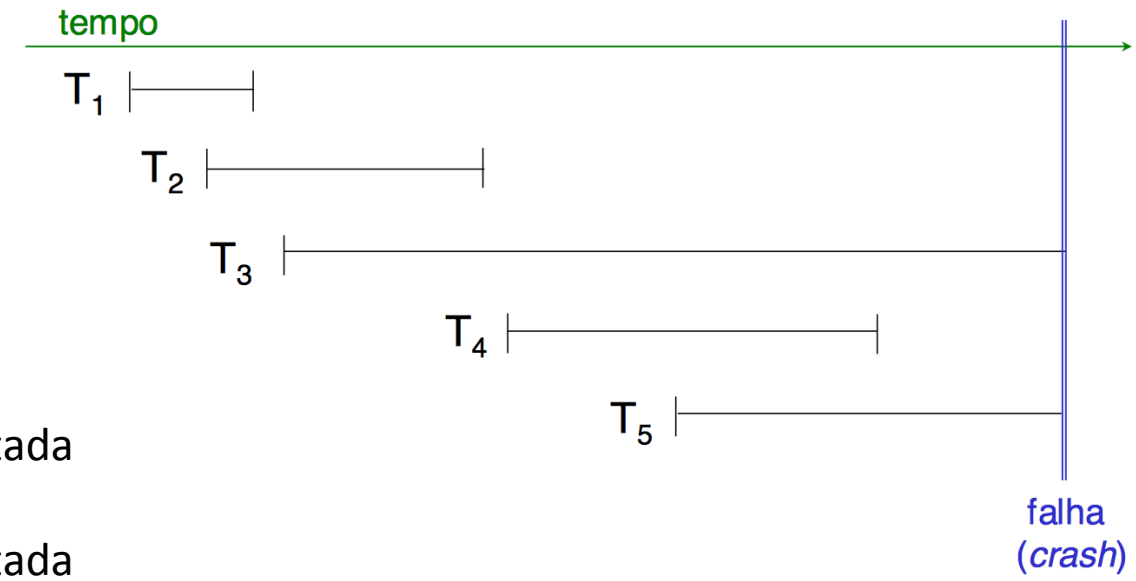
Técnicas para recuperação de falhas não catastróficas

- Atualização adiada:
  - Algoritmo NO-UNDO/REDO
- Atualização imediata:
  - Algoritmo UNDO/REDO



Atualização adiada

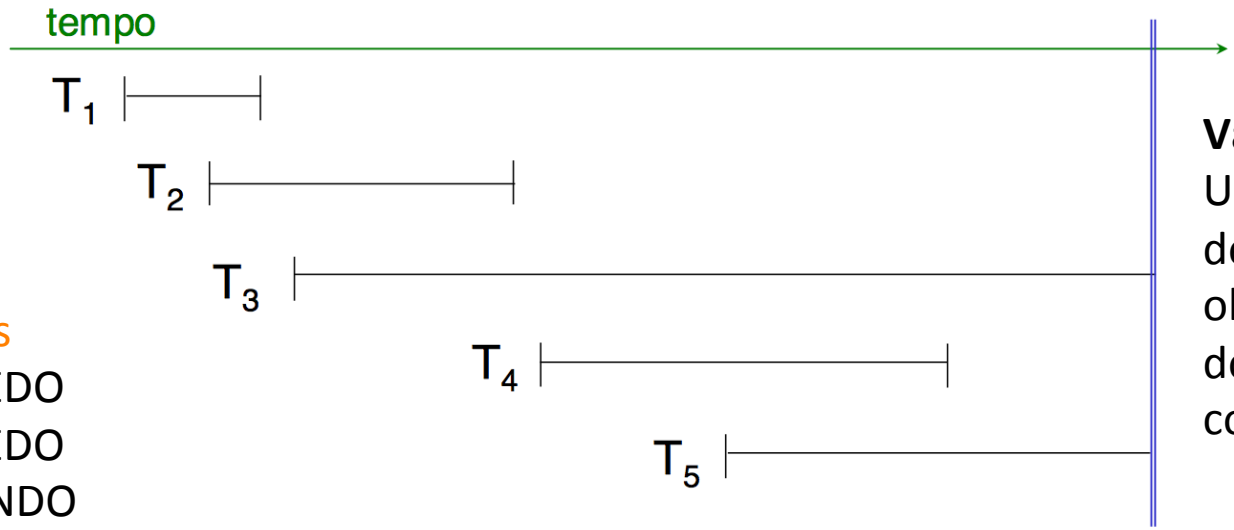
- Somente atualiza o banco de dados físico depois que a transação alcança o ponto de efetivação:
  1. Antes do *commit* os registros são atualizados nos *buffers*;
  2. Durante *commit* os registros são gravados primeiro nos *logs* e depois no banco de dados.
  3. Não é necessário UNDO e o REDO é necessário em alguns casos:
     - NO-UNDO/REDO



Ações
T1 REDO
T2 REDO
T3 Descartada
T4 REDO
T5 Descartada

Atualização imediata

- O banco de dados pode ser atualizado antes do ponto de confirmação:
  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



Ações
T1 REDO
T2 REDO
T3 UNDO
T4 REDO
T5 UNDO

Variação:
UNDO/NO-REDO (banco de dados é atualizado obrigatoriamente antes do ponto de confirmação)

Idempotentes

- Operações de UNDO e REDO devem ser idempotentes, ou seja, executá-las várias vezes é equivalente a executá-las uma vez.

- O resultado da recuperação de uma falha do sistema durante a recuperação deve ser igual ao resultado da recuperação quando não há falha durante esse processo.

Caching de blocos de disco

- *Cache* de SGBD
  - buffer de páginas do disco do banco de dados na memória principal sob controle do SGBD
- Catálogo da *cache*:
  - controla quais itens de dados estão em qual buffer.
    - `[Endereço página disco, Localização do Buffer, ...]`
  - Quando é requerido um item que não está na "*cache*", provoca a substituição de páginas.
    - Técnicas de substituição de páginas: MRU, FIFO, etc.
- *Bit sujo*:
  - indica se um item de dado na *cache* foi atualizado (1) ou não (0)
- *Bit preso-solto:*
  - indica se o dado pode ser gravado em disco (0) ou se ainda está preso a uma transação em execução (1).

Atualização de blocos de disco e o *log*

- Quando o buffer modificado deve ser atualizado no disco:
  - Atualização no local – solução prática:
    - grava o buffer modificado no mesmo local da versão anterior
    - BFIM e AFIM são mantidos no *log*
  - Atualização por sombreamento – solução teórica:
    - grava o buffer modificado em outro local e não será necessário uso de *log.*

- Versões do dado:
  - BFIM(*before image*) – valor do dado antes da atualização
  - AFIM(*after image*) – valor do dado após atualização.

Gerenciamento de Buffer

- *Logging write-ahead*
  - Quando as atualizações são "realizadas no local", é necessário preservar os valores anteriores dos dados caso seja necessário reverter (UNDO) 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 transferida para o disco antes que a BFIM seja sobrescrita pela AFIM.
  - Esse processo é chamado de *logging write-ahead*.

Técnicas de Gerência de *Buffer*

- *NOT-STEAL*
  - Um bloco na *cache* utilizado por uma transação *T* não pode ser gravado antes do ponto de confirmação de *T.*
  - Vantagem:
    - Recuperação mais simples – UNDO desnecessário.
    - Evita dados de transações inacabadas sendo gravadas no BD
- *STEAL*
  - Um bloco na *cache* utilizado por uma transação *T* pode ser gravado antes do ponto de confirmação de *T.*
    - Necessário se algum dado é requisitado do BD por outra transação e não há blocos disponíveis na *cache*
  - O bloco "vítima" é escolhido através de alguma técnica de SO – LRU, FIFO, ...
  - 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* são imediatamente gravados no BD quando *T* alcança o ponto de confirmação.
  - Vantagem:
    - garante a durabilidade de *T* o mais cedo possível

- *NO-FORCE*
  - Os blocos que mantêm dados atualizados por *T* não são imediatamente gravados no BD quando *T* alcança o ponto de confirmação.
  - Vantagem:
    - blocos atualizados podem permanecer na *cache* e serem utilizados por outras transações, após o ponto de confirmação de *T* (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*

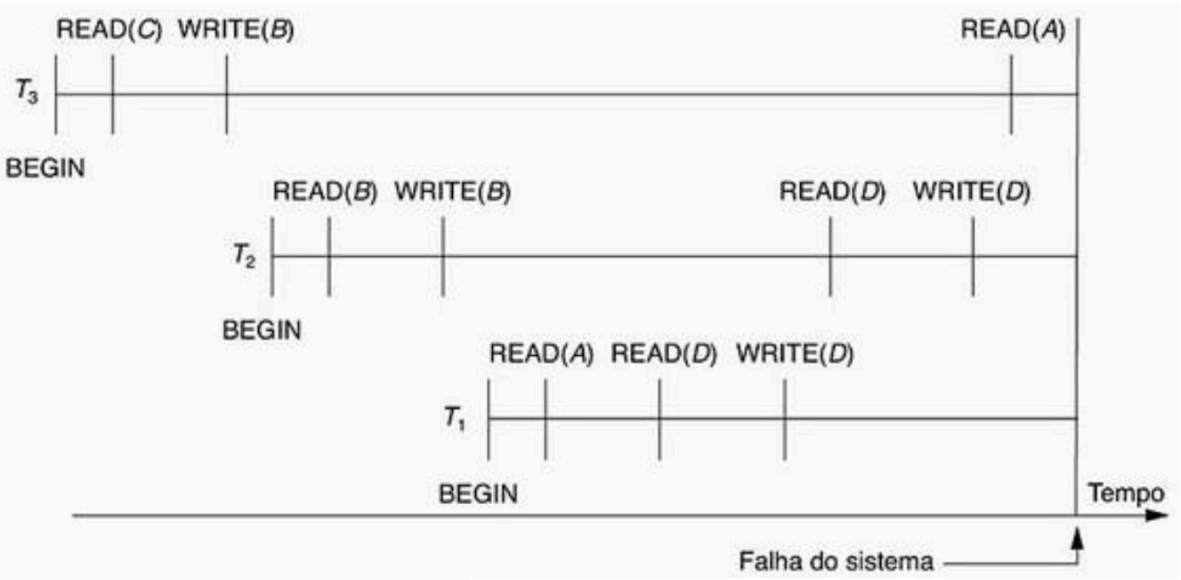
1. Suspensão de todas as transações;
2. Gravação (forçada) dos blocos atualizados da *cache* no BD;
3. Inserção de um registro *checkpoint* no *Log* e sua gravação em disco;
4. Retomada da execução das transações.

- Vantagem da técnica de *checkpoint*
  - Transações confirmadas antes do *checkpoint* não precisam sofrer REDO em caso de falha
    - Elas já estão garantidas 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.

Exemplo



(b)

		A	B	C	D
		30	15	40	20
	[start_transaction,T_3]				
	[read_item,T_3,C]				
*	[write_item,T_3,B,15,12]		12		
	[start_transaction,T_2]				
	[read_item,T_2,B]				
**	[write_item,T_2,B,12,18]		18		
	[start_transaction,T_1]				
	[read_item,T_1,A]				
	[read_item,T_1,D]				
	[write_item,T_1,D,20,25]				25
	[read_item,T_2,D]				
**	[write_item,T_2,D,25,26]				26
	[read_item,T_3,A]				
	← Falha do sistema				

* T_3 é revertido porque não atingiu seu ponto de confirmação.

** T_2 é revertido porque lê o valor do item B gravado por T_3.

Recuperação NO-UNDO/REDO

baseada em atualização adiada

Recuperação com Atualização Adiada

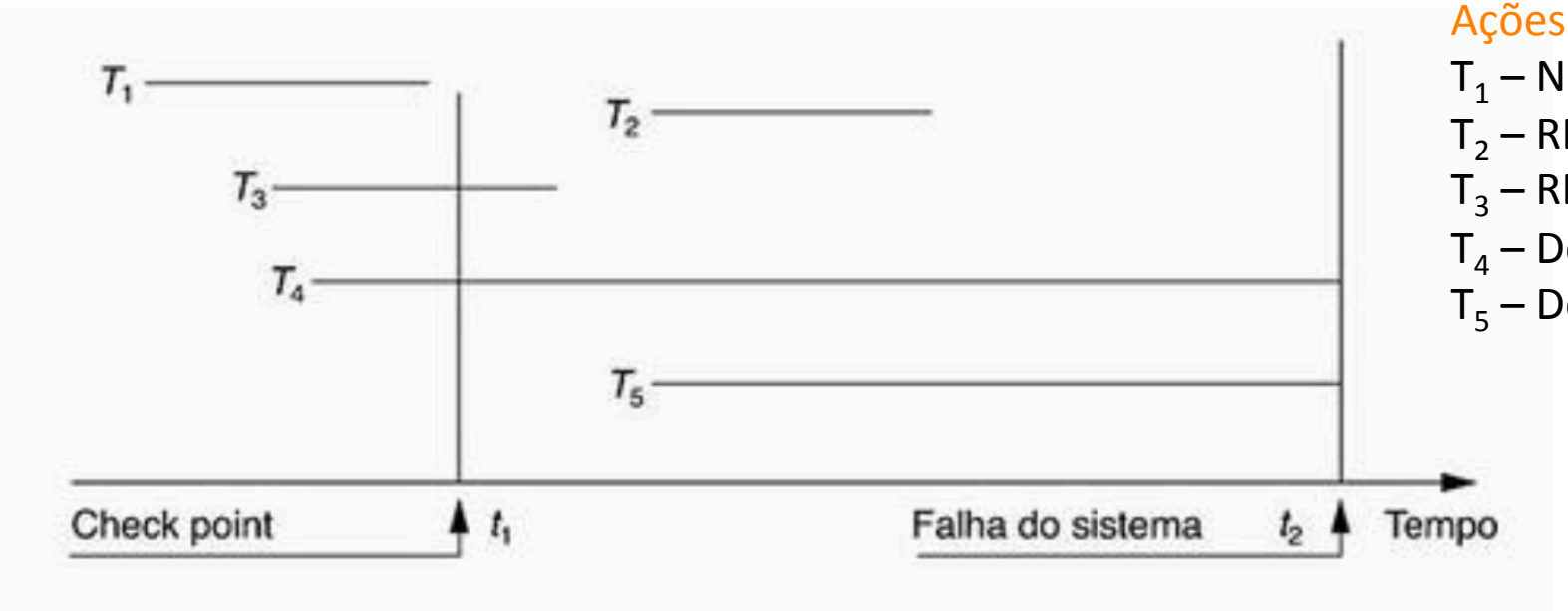
- Ambiente Monousuário
- Protocolo:
  - Uma transação não pode modificar o BD até seu ponto de confirmação
  - Uma transação não pode chegar ao ponto de confirmação até que todas as operações de atualização sejam gravadas no log e o log no disco
- Procedimento:
  - Usar duas listas de transações: as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

Recuperação com Atualização Adiada

- Postergar quaisquer atualizações para o disco (BD) até que a transação termine sua execução com sucesso e atinja o seu ponto de confirmação.
- Protocolo:
  - Depende do protocolo usado no controle de concorrência
  - Assumir o 2PL Estrito onde bloqueios são guardando-os até o ponto de confirmação.
  - Uma transação não pode chegar ao ponto de confirmação 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 ultimo checkpoint e as transações ativas
  - Aplicar a operação REDO para todas as operações write 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

- 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 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



Ações:

T_1 – Nada
T_2 – REDO
T_3 – REDO
T_4 – Descarta
T_5 – Descarta

Recuperação com Atualização Adiada

- Desvantagem:
  - Limita a execução concorrente das transações porque itens ficam bloqueados até o ponto de confirmação das transações.
  - Ocupa espaço do buffer.

- Vantagens:
  - Uma transação não grava as modificações no BD até o ponto de confirmação. 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 UNDO/NO-REDO UNDO/REDO

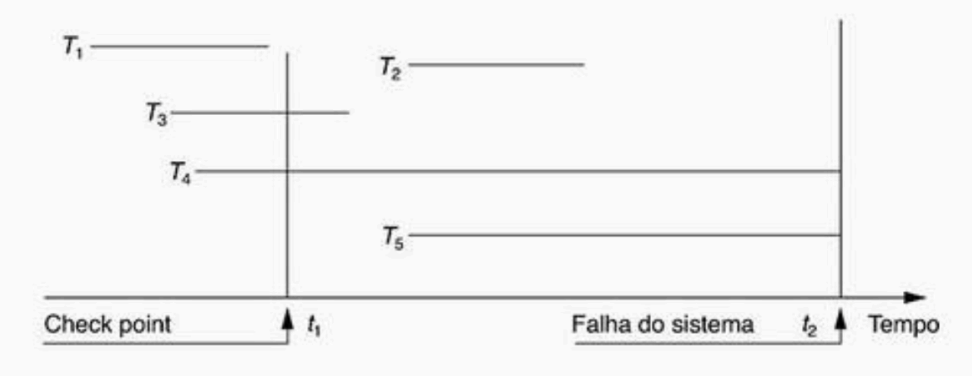
baseada em atualização imediata

Recuperação com Atualização Imediata

- Quando uma operação é executada, ela pode ser imediatamente atualizada no disco sem que a transação atinja o ponto de confirmação.
- Dois tipos de algoritmos:
  - UNDO/NO-REDO:
    - Se a técnica de recuperação garante que todas as atualizações são gravadas no BD (disco) antes do ponto de confirmação da transação, não é necessário REDO
    - Usa o método FORCE.
  - UNDO/REDO:
    - Se as modificações são gravada no BD (disco) depois do ponto de confirmação da transação
    - Usa o método NO-FORCE.
    - Caso mais geral e mais complexo

Recuperação com Atualização Imediata

- Protocolo:
  - Assume-se que o log inclui checkpoints e o protocolo de concorrência como o de duas fases
- Procedimento para (UNDO/REDO)
  - 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 das transações ativas, na ordem inversa de suas gravações no log
  - Aplicar a operação REDO para todas as operações write das transações acabadas, na ordem na qual elas foram gravadas no log



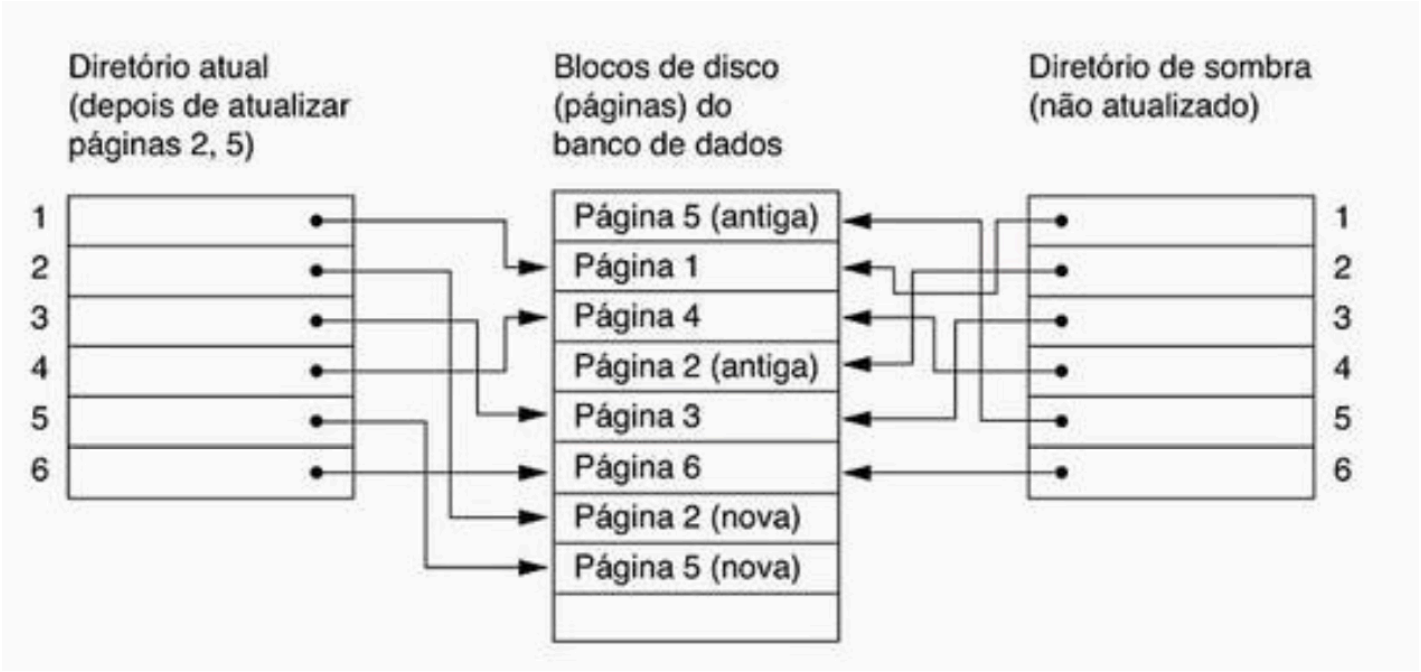
Ações:

T_1 – Nada
T_2 – REDO
T_3 – REDO
T_4 – UNDO
T_5 – UNDO

Paginação de Sombra

Paginação *Shadow*

- Pressupõe-se que o BD seja composto por "*n*" páginas de tamanho fixo.
- Solução factível para execuções seriais.
- Para as páginas atualizadas pela transação, duas versões são mantidas. A versão antiga é referenciada pelo diretório de sombra e a nova versão, pelo diretório atual.



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 ponto de confirmação é significativo
  - *Garbage Collection* (liberação de páginas antigas) é necessário após o ponto de confirmação.

Dúvidas