# Aspectos Operacionais: Recuperação após Falhas


**Valéria Times**

## Recuperação após Falhas

- Recuperação de BD
  - Processo de restaurar o BD para um estado correto na ocorrência de uma falha
    - Consistente
    - Confiável



## Recuperação após Falhas

- Causas de Falhas
  - Problemas no Sistema
    - Erros de software/hardware, resultando em perdas de dados da memória principal
  - Falhas na mídia
    - Perdas de dados do armazenamento secundário
  - Erros da Aplicação
    - Problemas na lógica do programa acessando o BD, causando falhas de transação



## Recuperação após Falhas

- Causas de Falhas (Cont.)
  - Desastres Físicos Naturais
    - Incêndios, Enchentes, Terremotos.
  - Falta de cuidado
    - Destruição sem intenção dos dados ou dispositivos
  - Sabotagem
    - Corrupção intencional dos dados ou dispositivos

**Prevenir tais erros está fora do escopo de problemas tratados pelo mecanismo de recuperação do SGBD!**



## Recuperação após Falhas

- O que o SGBD faz?
  - Mecanismos de Recuperação são projetados para lidar com algumas das conseqüências destes erros
    - Perda de dados
- Para recuperar, SGBD deve prover:
  - Mecanismo de Backup
  - Gerenciador de Recuperação
  - Gerenciador de Log
  - Mecanismo de Checkpoint



## Recuperação após Falhas

- Mecanismo de Backup
  - Funcionalidades para permitir a geração de cópias de backup do BD e do Log
    - Em intervalos de tempo regulares
    - Sem interromper a execução do SGBD
    - De forma completa ou incremental
  - Estado consistente mais próximo da falha
- Gerenciador de Recuperação
  - Componente do SGBD responsável pela restauração do BD
    - De modo a reverter ou refazer mudanças

## Recuperação após Falhas

- Gerenciador de Log (Journal)
  - Rastrear o estado atual das transações e das atualizações do BD
  - Contém registros de transação e de checkpoint

| TID | Tempo | Operação | Objeto | Antes | Depois | pAnt | pProx |
|---|---|---|---|---|---|---|---|
| T1 | 10 : 12 | START | | | | 0 | 2 |
| T1 | 10 : 13 | UPDATE | X | OLDV | NEWV | 1 | 8 |
| T2 | 10 : 14 | START | | | | 0 | 4 |
| T2 | 10 : 16 | INSERT | Y | | NEWV | 3 | 5 |
| T2 | 10 : 17 | DELETE | Z | OLDV | | 4 | 6 |
| T2 | 10 : 17 | UPDATE | W | OLDV | NEWV | 5 | 9 |
| T3 | 10 : 18 | START | | | | 0 | 11 |
| T1 | 10 : 18 | COMMIT | | | | 2 | 0 |
| | 10 : 19 | CHECKP | T2, T3 | | | | |
| T2 | 10 : 19 | COMMIT | | | | 6 | 0 |
| T3 | 10 : 20 | INSERT | R | | NEWV | 7 | 12 |
| T3 | 10 : 21 | COMMIT | | | | 11 | 0 |



## Recuperação após Falhas

- Gerenciador de Log (Cont.)
  - Contém uma representação do histórico de execução das transações de BD
  - Mantido em separado do BD cujo conteúdo e formato depende do algoritmo de recuperação usado
  - Tipos de Log
    - Log Físico
      - Contém valores de item de dados sendo gravados por transações. É mais extenso.
    - Log Lógico
      - Contém descrições de operações de mais alto nível de abstração. É de difícil interpretação.
        - Registro R foi inserido no Arquivo A



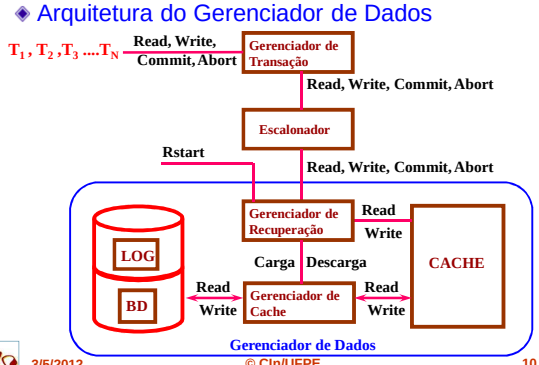
## Recuperação após Falhas

- Mecanismo de Checkpoint
  - Objetiva limitar a busca e o processamento subseqüente feito no Log
  - Representa o ponto de sincronização entre o BD e o Log
  - Operações de Checkpoint
    - Escrita do Log da memória no disco
    - Escrita de partições do buffer do BD, que foram modificadas, no disco
    - Inserção de um registro do tipo Checkpoint no Log do disco.



## Recuperação após Falhas

- Arquitetura do Gerenciador de Dados






## Recuperação após Falhas

- Abordagens para o Gerenciador de Dados
  - *In-place Updating*
    - Mantém exatamente uma única cópia de cada item de dados em disco
    - Grava o item de dado no mesmo local do disco, apagando o valor anterior
    - Log deve ser usado e gravado antes do BD
  - *Shadowing*
    - Mantém mais de uma cópia para cada item de dados (*shadow copies*)
    - Grava novo valor em uma posição diferente
    - Não precisa de Log



## Recuperação após Falhas

- Gerenciador de Cache
  - Provê operações para transferir dados da(o):
    - *Cache* do SGBD (buffer na memória principal sob controle do SGBD) para disco
    - Disco para uma partição da *cache*
  - Possui um diretório que identifica quais itens do BD estão em quais partições

Cache

| Partição | Dirty Bit | Valor Item |
|---|---|---|
| 1 | 1 | Informática |
| 2 | 0 | 3.1415 |

Diretório da Cache

| Nome Item | Partição |
|---|---|
| X | 2 |
| Y | 1 |

## Recuperação após Falhas

- Gerenciador de Recuperação
  - Possui 5 operações atômicas
    - RM_Read ( $T_i$ , x )
    - RM_Write ( $T_i$ , x , v )
    - RM_Commit ( $T_i$ )
    - RM_Abort ( $T_i$ )
    - RM_Restart
  - Mantém as seguintes listas em disco
    - Ativas: iniciadas mas não efetivadas
    - Consolidadas: desde o último checkpoint
    - Abortadas: desde o último checkpoint



## Recuperação após Falhas

- Gerenciador de Recuperação (Cont.)
  - *Garbage Collection*
    - Para limitar o crescimento do Log, SGBD deve liberar e reusar o espaço contendo dados inúteis para a recuperação do BD
  - Regras: A entrada [$T_i$ , x , v ] pode ser removida do Log se e somente se:

1. $T_i$ abortou
2. $T_i$ *commitou* mas outra transação efetivada já atualizou "x" depois de $T_i$



## Recuperação após Falhas

- Regras do *Garbage Collection* (Cont.):

3. [$T_i$ , x , v ] pode ser removida do Log se:
   a) UNDO não é feito,
   b) v é o último valor efetivado de x,
   c) v é o valor de x no BD,
   d) [$T_i$ , x , v ] é a única entrada no Log para x.

- Informações antigas e desnecessárias também são removidas das listas de transações ativas, efetivadas e abortadas.



## Recuperação após Falhas

- Gerenciador de Recuperação (Cont.)
  - *Read*, *Write*, *Commit*, *Abort* executam atomicamente, de forma indivisível
  - *Rstart* pode interromper qualquer uma delas a qualquer momento
  - Rstart pode na verdade interromper sua própria execução
    - *Rstart* é idempotente
    - *Rstart* (*Rstart* ( *Rstart* () ) ) = *Rstart* ()
    - Reinício de *Rstart* produz o mesmo efeito da execução completa do 1º. *Rstart*



## Recuperação após Falhas

- Gerenciador de Recuperação (Cont.)
  - Na ocorrência de uma falha, o BD pode:
    - Conter valores atualizados por transações não efetivadas
    - Não conter todos os valores atualizados por transações já efetivadas.
    - Estado Inconsistente ➡ Estado Consistente
    - Requerer UNDO se permite que transações atualizem o BD e depois se consolidem
    - Requerer REDO se permite que transações se consolidem antes de atualizar o BD.



## Recuperação após Falhas

- Gerenciador de Recuperação (Cont.)
  - Duas técnicas usadas para restaurar o BD
    - Atualização adiada: BD atualizado depois do *commit* de transações (REDO)
    - Atualização imediata: BD atualizado antes do *commit* de transações (UNDO)
  - Algoritmos de Recuperação são classificados:
    - UNDO e REDO
    - UNDO sem REDO
    - REDO sem UNDO
    - Sem UNDO e sem REDO

## Recuperação após Falhas

- Atualização adiada: BD é atualizado depois do *commit* de transações
  - antes do *commit*: atualizações são gravadas no Log mantido na *cache* do SGBD
  - durante o *commit*: atualizações são gravadas no Log do disco e depois no BD
  - Operação UNDO não é necessária. REDO é necessária em alguns casos.
    - NO-UNDO / REDO algoritmo
    - BD contém atualizações faltantes
    - Requer mais espaço na cache e um volume pequeno de transações



## Recuperação após Falhas

- Atualização imediata: BD é atualizado antes do *commit* de transações
  - atualizações são gravadas no Log do disco antes de serem aplicadas ao BD
    - para que o processo de recuperação ocorra
  - Operações UNDO e REDO podem ser necessárias. Possui duas abordagens.
    - UNDO / NO-REDO algoritmo
    - UNDO / REDO algoritmo
    - BD contém atualizações inapropriadas
    - Realiza mais operações de E/S



## Recuperação após Falhas

- Ambas seguem protocolo *Write-Ahead Logging*
  - Realiza atualização *in-place*
  - Garante que a ImAn seja gravada no Log do disco antes da gravação da ImDp no BD
- Entradas no Log de Recuperação
  - Do tipo *Redo*: usada na gravação do novo valor do item de dado (ImDp)
  - Do tipo *Undo*: usada na gravação do antigo valor do item de dado (ImAn)



## Recuperação após Falhas

- Protocolo *Write-Ahead Logging* / UNDO e REDO
  - Regra UNDO
    - ImAn do BD não pode ser apagada pela sua ImDp no disco, até que todas as entradas do tipo UNDO daquela transação (até aquele ponto) tenham sido gravadas no disco.
    - Garante que o último valor consolidado de cada item de dado é salvo em disco antes de ser sobrescrito por um valor não efetivado.



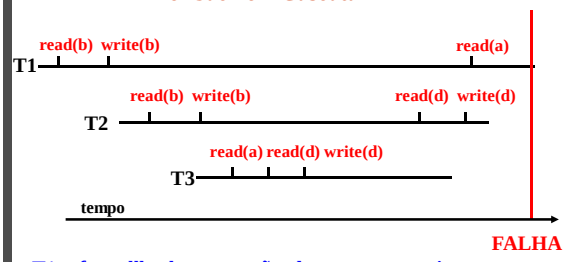
## Recuperação após Falhas

- Protocolo *Write-Ahead Logging* / UNDO e REDO
  - Regra REDO
    - Operação *Commit* de uma transação não pode ser completada até que todas as entradas do tipo REDO tenham sido gravadas no disco
    - Garante que o novo valor atualizado por uma transação está salvo em disco no momento de sua consolidação



## Recuperação após Falhas

***Rollback em Cascata***




**T1 sofre *rollback* porque não alcançou o *commit***

**T2 também porque leu o valor de "b" gravado por T1**

## Recuperação após Falhas

- Gerenciador de Recuperação (Cont.)
  - Processo de restauração do BD depende do protocolo de controle de concorrência usado
  - Para fins de estudo, assume-se que:
    - Escalonador invoca a execução de operações ao mecanismo de recuperação em uma ordem que:
      - ➡ Produz um escalonamento estrito e serializável
    - *Rollback em cascata* de transações não é considerado nos protocolos de recuperação



## Recuperação após Falhas

- Recuperação com Atualização Adiada (RAA)
  - Protocolo
    - Transações não podem modificar o BD até chegarem ao ponto de *commit*
    - Transações não podem chegar ao *commit* até que todas as operações de atualização sejam gravadas no Log e o Log no disco
    - Usa duas listas de transações:
      - consolidadas desde o último checkpoint
      - abortadas desde o último checkpoint



## Recuperação após Falhas

- Recuperação com Atualização Adiada (RAA)
  - Protocolo (Cont.)
    - Aplicar a operação REDO para todas as operações *write_item* das transações efetivadas no Log, na ordem na qual elas foram gravadas
      - Pode-se otimizar este processo, refazendo apenas a última atualização do item de dado x
    - Transações ativas e não acabadas são canceladas e devem ser re-submetidas



## Recuperação após Falhas

- Exemplo de Recuperação com Atualização Adiada (RAA):
  - Examinar a entrada no log [ write_item, T, X, novo-valor ] e atualizar o valor do item X no BD com o novo-valor (ImDp)

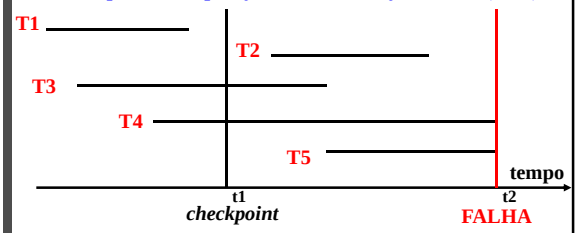
| T1 | T2 | Log |
|---|---|---|
| read_item(a) | read_item(b) | [start_transaction, T1] |
| read_item(d) | write_item(b) | [write_item, T1, d, 20] |
| write_item(d) | read_item(d) | [commit, T1] |
| | write_item(d) | [start_transaction, T2] |
| | Falha ➡ | [write_item, T2, b, 10] |
| | | [write_item, T2, d, 25] |

**A operação *write_item* de T1 é refeita**
**As entradas no log de T2 são ignoradas**



## Recuperação após Falhas

***Exemplo de Recuperação com Atualização Adiada (RAA)***




**T1: OK**
**T2 e T3: operações *write_item* devem ser refeitas**
**T4 e T5: canceladas e re-submetidas**



## Recuperação após Falhas

- Recuperação com Atualização Adiada (RAA)
  - Vantagens
    - Uma transação não grava as modificações no BD até o *commit*. Logo, uma transação nunca é desfeita por causa de falhas.
    - 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 após Falhas

- Recuperação com Atualização Adiada (RAA)
  - Desvantagens
    - Limita a execução concorrente das transações porque itens ficam bloqueados até o *commit* das transações
    - Requer mais espaço de *cache* para manter todas as mudanças em memória principal até que o ponto de consolidação ocorra
    - Requer transações curtas ou com um volume menor de atualizações



## Recuperação após Falhas

- Recuperação com Atualização Imediata (RAI)
  - Protocolo possui duas abordagens:
    - Undo / No-Redo:

      Garante que todas as atualizações são gravadas no BD antes do *commit* da transação. Logo, não é necessário REDO
    - Undo / Redo

      As modificações podem ser ou não gravadas no BD antes do *commit* da transação. A decisão é tomada pelo Gerenciador de *Cache* do SGBD.



## Recuperação após Falhas

- Recuperação com Atualização Imediata (RAI)
  - Protocolo
    - Usar as seguintes listas de transações:
      - consolidadas desde último checkpoint e
      - ativas
      - abortadas desde último checkpoint



## Recuperação após Falhas

- Recuperação com Atualização Imediata (RAI)
  - Protocolo (Cont.)
    - Aplicar a operação UNDO para todas as operações *write_item* das transações ativas, na ordem inversa de suas gravações no Log
    - Procedimento UNDO
      - Desfazer uma operação *write_item* consiste em examinar sua entrada no log: [write_item, T, X, valor_ant, valor_novo] e atualizar o valor do item X no BD com o valor_ant (ImAn)



## Recuperação após Falhas

- Recuperação com Atualização Imediata (RAI)
  - Protocolo (Cont.)
    - Aplicar a operação REDO para todas as operações *write_item* das transações acabadas, na ordem na qual elas foram gravadas no Log ou na ordem inversa (otimização)
    - Procedimento REDO
      - Refazer uma operação *write_item* consiste em examinar sua entrada no log: [ write_item, T, X, novo-valor ] e atualizar o valor do item X no BD com o novo-valor (ImDp)



## Recuperação após Falhas

- Recuperação com Atualização Imediata (RAI)
  - Protocolo (Cont.)
    - Sempre que um item de dado é recuperado, ele é adicionado à lista de itens refeitos ou de itens desfeitos.

## Recuperação após Falhas

- Algoritmo UNDO / REDO
  - Consiste na opção mais complexa
  - É mais flexível sobre quando transferir dados da *cache* para o disco, deixando a decisão para o gerenciador de *cache*
    - Evita transferências desnecessárias, minimizando E/S
    - Minimiza custo de armazenamento na cache
  - Maximiza desempenho em condições normais de processamento
    - Porém, tem custo maior de recuperação



## Recuperação após Falhas

- Algoritmo UNDO / REDO
  - Suponha que $T_i$ atualiza x com o valor v
    - GR interage com GC para ler x, se o mesmo não estiver em *cache*
      - Valor lido de x é transferido para Log da memória e para uma partição da *cache*
    - Valor de x é atualizado junto com o Log
    - GC apenas transfere x para disco quando necessita liberar espaço
    - Se GC salva x no disco e $T_i$ aborta ou uma falha ocorre antes de $T_i$ ser efetivada:
      - ➡ UNDO é necessário
    - Se Ti é efetivada e a falha ocorre antes do GC liberar a partição de x:
      - ➡ REDO é necessário



## Recuperação após Falhas

- Algoritmo UNDO / REDO
  - Procedimento de Gravação

RM_Write ( $T_i$ , x , v )

1. Adicione $T_i$ à lista de transações ativas, caso não esteja lá
2. Se x não estiver na *cache*, leia x
3. Adicione [ $T_i$ , x , valor_lido, v ] ao Log
4. Atualize a partição de x com o valor v
5. Informe o final do processamento ao escalonador.



## Recuperação após Falhas

- Algoritmo UNDO / REDO
  - Procedimento de Leitura

RM_Read ( $T_i$ , x )

1. Se x não estiver na *cache*, leia x
2. Retorne o valor lido e mantido na partição de x ao escalonador.



## Recuperação após Falhas

- Algoritmo UNDO / REDO
  - Procedimento de Consolidação

RM_Commit ( $T_i$ )

1. Adicione [ $T_i$ , commit ] ao Log do disco e $T_i$ à lista de transações consolidadas
2. Informe o *commit* de $T_i$ ao escalonador
3. Remova $T_i$ da lista de transações ativas.



## Recuperação após Falhas

- Algoritmo UNDO / REDO
  - Procedimento para Abortar transação

RM_Abort ( $T_i$ )

1. Para cada item de dado x atualizado por $T_i$ :
   a) Se x não estiver na *cache*, leia x.
   b) Atualize a partição de x com o valor ImAn.
2. Adicione $T_i$ à lista de transações abortadas
3. Informe que $T_i$ abortou ao escalonador
4. Remova Ti da lista de transações ativas.

## Recuperação após Falhas

- Algoritmo UNDO / REDO
  - Procedimento para recuperar

RM_Restart ( )

1. Considere limpas as partições da *cache*
2. Faça refeito = { } e desfeito = { }
3. Comece com a última entrada do Log e percorra o arquivo no sentido inverso (em direção ao início). Repita os passos abaixo até que: (i) refeito U desfeito = BD ou (ii) não existam mais entradas no Log
   a) Se x não estiver na *cache*, leia x.
   b) Se $T_i$ estiver na lista de transações efetivadas:
      i. Atualize a partição de x com o valor ImDp.
      ii. refeito = refeito U { x }



## Recuperação após Falhas

- Algoritmo UNDO / REDO
  - Procedimento para recuperar (Cont.)
   c) Caso contrário, se: (i) $T_i$ estiver na lista de transações abortadas ou (ii) na lista de transações ativas e não estiver na lista de transações efetivadas:
      i. Atualize a partição de x com o valor ImAn.
      ii. desfeito = desfeito U { x }
4. Para cada $T_i$ da lista de transações efetivadas:
   a) Se $T_i$ estiver na lista de transações ativas, remova $T_i$ desta lista.
5. Informe o final do processamento de restauração ao escalonador



## Recuperação após Falhas

- Algoritmo UNDO / NO-REDO
  - Registra todas as atualizações no BD antes da transação ser efetivada.
  - Transfere dados da *cache* para o disco mais frequentemente para garantir que o BD esteja atualizado antes do *commit* (mais E/S)
  - Requer menos tempo para recuperar (sem REDO)
  - Difere do algoritmo anterior nos seguintes procedimentos:
    - RM_Commit
    - RM_Restart



## Recuperação após Falhas

- Algoritmo UNDO / NO-REDO
  - Procedimento de Consolidação

RM_Commit ( Ti )

1. Para cada item de dado x atualizado por Ti, se x estiver em *cache*, salve Log e x no disco
2. Adicione [ Ti , commit ] ao Log do disco e Ti à lista de transações efetivadas
3. Informe o *commit* de Ti ao escalonador
4. Remova Ti da lista de transações ativas.



## Recuperação após Falhas

- Algoritmo UNDO / NO-REDO
  - Procedimento para recuperar

RM_Restart ( )

1. Considere limpas as partições da *cache*
2. Faça desfeito = { }
3. Comece com a última entrada do Log e percorra o arquivo no sentido inverso (em direção ao início). Repita os passos abaixo até que: (i) desfeito = BD ou (ii) não existam mais entradas no Log para examinar. Para cada [ $T_i$ , x , ImAn, ImDp ], faça:
   a) Se $T_i$ não estiver na lista de transações efetivadas e x não estiver em desfeito:
      i. Selecione uma partição da *cache* para x
      ii. Atualize a partição de x com o valor ImAn.
      iii. desfeito = desfeito U { x }



## Recuperação após Falhas

- Algoritmo UNDO / NO-REDO
  - Procedimento para recuperar (Cont.)

4. Para cada $T_i$ da lista de transações efetivadas:
   a) Se $T_i$ estiver na lista de transações ativas, remova $T_i$ desta lista
5. Informe o final do processamento de restauração ao escalonador.

## Recuperação após Falhas

- Algoritmo NO-UNDO / REDO
  - Registra no BD apenas as atualizações feitas por transações já efetivadas.
  - Requer mais espaço da *cache* para manter as atualizações até que o *commit* ocorra
  - Requer menos tempo para recuperar (sem UNDO)



## Recuperação após Falhas

- Algoritmo NO-UNDO / REDO
  - Procedimento de Gravação

RM_Write ( $T_i$ , x , v )

1. Adicione [ $T_i$ , x , v ] ao Log
2. Informe o final do processamento de escrita ao escalonador.



## Recuperação após Falhas

- Algoritmo NO-UNDO / REDO
  - Procedimento de Leitura

RM_Read ( $T_i$ , x )

1. Se $T_i$ atualizou antes x, retorne ao escalonador a ImDp de x escrito por $T_i$
2. Senão:
   a) Se x não estiver na *cache*, leia x; e
   b) Retorne o valor lido e mantido na partição de x ao escalonador.



## Recuperação após Falhas

- Algoritmo NO-UNDO / REDO
  - Procedimento de Consolidação

RM_Commit ( $T_i$ )

1. Adicione [ $T_i$ , commit ] ao Log do disco e $T_i$ à lista de transações consolidadas
2. Para cada item de dado x atualizado por $T_i$ :
   a) Se x não estiver na *cache*, leia x
   b) Atualize a partição de x com o valor ImDp escrito por $T_i$
3. Informe o *commit* de $T_i$ ao escalonador.



## Recuperação após Falhas

- Algoritmo NO-UNDO / REDO
  - Procedimento para Abortar transação

RM_Abort ( $T_i$ )

1. Adicione $T_i$ à lista de transações abortadas.
2. Informe que $T_i$ abortou ao escalonador.



## Recuperação após Falhas

- Algoritmo NO-UNDO / REDO
  - Procedimento para recuperar

RM_Restart ( )

1. Considere limpas as partições da *cache*
2. Faça refeito = { }
3. Comece com a última entrada do Log e percorra o arquivo no sentido inverso (em direção ao início). Repita os passos abaixo até que: (i) refeito = BD ou (ii) não existam mais entradas no Log para examinar. Para cada [ $T_i$ , x , ImAn, ImDp ], faça:
   a) Se $T_i$ estiver na lista de transações efetivadas e x não estiver em refeito:
      i. Selecione uma partição da *cache* para x
      ii. Atualize a partição de x com o valor ImDp.

## Recuperação após Falhas

- Algoritmo NO-UNDO / REDO
  - Procedimento para recuperar (Cont.)

    iii. refeito = refeito U { x }

  4. Informe final do processamento de restauração ao escalonador.



## Recuperação após Falhas

- Shadowing (Páginas sombreadas)
  - Considera que o BD é composto por *n* páginas de tamanho fixo
  - Mantém 2 tabelas de páginas durante o ciclo de execução de uma transação
    - Tabela de páginas atual
    - Tabela de páginas sombreada
  - No início da execução da transação, as 2 tabelas são idênticas
  - Durante a execução da transação:
    - Tabela sombreada nunca é atualizada
      - É usada para recuperar BD, se uma falha ocorre
    - Tabela atual é usada p/ registrar todas as mudanças



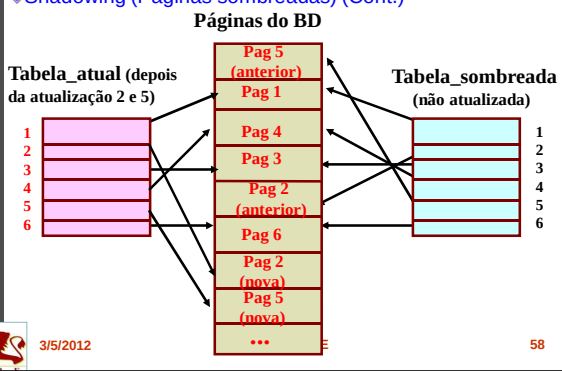
## Recuperação após Falhas

- Shadowing (Páginas sombreadas) (Cont.)
  - Quando a transação termina a execução, a tabela sombreada é atualizada a partir da atual
  - Não necessita de Log para recuperar
    - Log pode ser usado com outros objetivos
  - Vantagem
    - Não há necessidade de UNDO ou REDO de operações



## Recuperação após Falhas

- Shadowing (Páginas sombreadas) (Cont.)





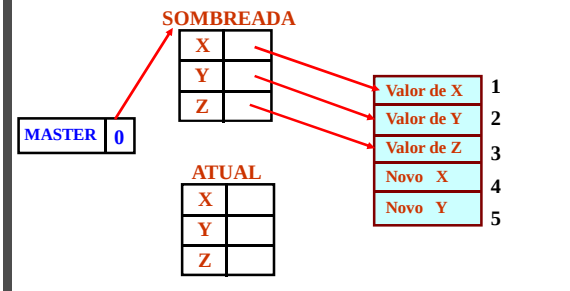

## Recuperação após Falhas

- Shadowing (Páginas sombreadas) (Cont.)
  - Desvantagens
    - Páginas atualizadas mudam de localização no disco, impedindo de manter juntas páginas relacionadas
    - Se a tabela de páginas é grande, o tempo para regravá-la, no momento de *commit,* é significativo
    - Garbage Collection (liberação de páginas antigas) é necessário após o *commit*



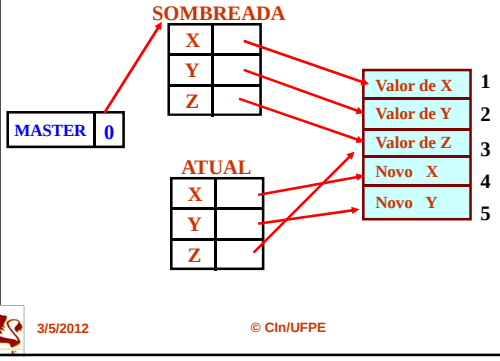
## Recuperação após Falhas

- Shadowing (Páginas sombreadas) (Cont.)



## Recuperação após Falhas

- Shadowing (Páginas sombreadas) (Cont.)






## Recuperação após Falhas

- Shadowing (Páginas sombreadas) (Cont.)