Sistema de recuperação
Cap 17

Prof. Heloise Manica P. Teixeira

Introdução

- **Mecanismos de recuperação de falhas:**
  - **garantem a consistência do BD após falhas do sistema**
  - **garantem as propriedades de atomicidade e durabilidade das transações**

- **Recuperação:**
  - **implica que o BD será restaurado para algum estado (correto) do passado**

Classificação de falha

- **Falha de transação**:
  - **Erros lógicos**: devido a alguma condição interna como uma entrada inadequada ou dados não encontrados.
  - **Erros do sistema**: o sistema entrou em um estado indesejável (por exemplo, impasse)
- **Falha do sistema:** devido a um defeito ou falha de hardware, falta de energia, falha de software (SGBD, SO, etc.).
  - Suposição falhar-parar: conteúdo do armazenamento não volátil é considerado como **não sendo** corrompido por falha do sistema
- **Falha do disco:** uma falha de cabeça ou falha de disco destrói todo ou parte do armazenamento de disco

Algoritmos de recuperação

- **Algoritmos de recuperação são técnicas para garantir a consistência do banco de dados e a atomicidade e durabilidade da transação apesar das falhas.**
- **Duas partes:**
  - **Ações tomadas durante o processamento normal da transação para garantir que existem informações suficiente para recuperação de falhas**
  - **Ações tomadas após uma falha para recuperar o conteúdo do banco de dados a um estado que garante atomicidade, consistência e durabilidade**

Estrutura de armazenamento

- **Armazenamento volátil (ex. memória principal, memória cache, etc.) não sobrevive a falhas do sistema**
- **Armazenamento não volátil (disco, fita, etc.) sobrevive a falhas do sistema**
- **Armazenamento estável:**
  - **forma de armazenamento que sobrevive a "todas" as falhas**
  - **replica a informação em vários meios de armazenamento nao-volateis**

Implementação do armazenamento estável

- **Para implementar o armazenamento estável, precisamos replicar a informação em várias cópias em meios de armazenamento não voláteis (normalmente discos) separados**
- **As cópias podem estar em sites remotos (para proteger contra desastres como incêndio ou inundação)**
- **A informação deve ser atualizada de maneira controlada para garantir que uma falha durante a transferência de dados não danifique os dados ou resulte em cópias inconsistentes**

Implementação do armazenamento estável

- **A transferência em bloco (entre memoria e disco) pode resultar em**
  - **Término bem-sucedido: a informação chegou ao destino com segurança;**
  - **Falha parcial: a falha ocorreu no meio da transferência, de modo que bloco de destino possui informações incorretas;**
  - **Falha total: a falha ocorreu no inicio da transferência, de modo que o bloco de destino nunca foi atualizado e permanece intacto.**

Implementação do armazenamento estável

- **Se houver uma falha de transferência de dados, o sistema deve a detectar e invocar um procedimento de recuperação para restaurar o bloco a um estado consistente.**
  - **Para isso o sistema deve manter dois blocos físicos para cada bloco lógico do BD**
  - **No caso de discos espelhados, os dois blocos estão no mesmo local**
  - **No caso de backup remoto, um dos blocos é o local, enquanto o outro está em uma instalação remota.**

Implementação do armazenamento estável

- **Proteção do meio de armazenamento contra falha durante a transferência de dados (uma solução):**
  - **Executar a operação de saída da seguinte forma (considerando duas cópias de cada bloco):**
    1. **Escreva a informação no primeiro bloco físico.**
    2. **Quando a primeira escrita terminar com sucesso, escreva a mesma informação no segundo bloco físico.**
    3. **A saída só é completada depois que a segunda escrita for completada com sucesso.**

Implementação de armazenamento estável (cont.)

- **Durante a recuperação o sistema examina cada par de blocos físicos. Se ambos forem iguais e não houver um erro detectável, então nenhuma outra ação é necessária.**
- **As cópias de um bloco podem diferir devido a falhas durante a operação de saída. Para recuperar-se :**
  1. Primeiro, encontre blocos inconsistentes
  2. Se uma cópia de um bloco inconsistente for detectada com um erro, escreva a outra cópia em cima dela.
  3. Se ambos não têm erro, mas forem diferentes, então o sistema substitui o conteúdo do primeiros bloco pelo valor do segundo.

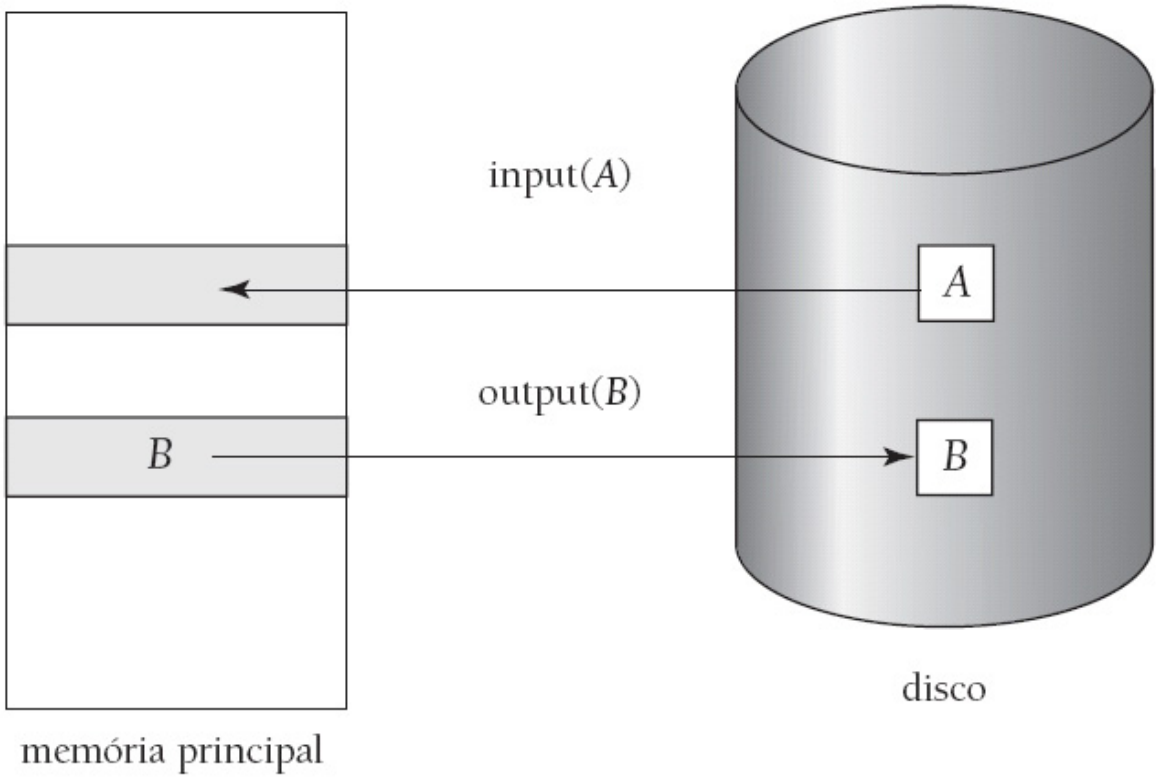
  → **Esse procedimento garante que uma escrita seja completa (atualiza todos os blocos) ou resulte em nenhuma mudança.**

Acesso aos dados

- O SBD residente em armazenamento não volátil e particionado em unidades de comprimento fixo chamadas **blocos**
- **Blocos físicos** são aqueles blocos que residem no disco.
- **Blocos de buffer** são os blocos que residem temporariamente na memória principal (buffer de disco)
- Os blocos são **unidades de transferência** de dados e podem conter vários itens de dados
  - Assumimos que nenhum item de dado abrange dois blocos

Acesso aos dados

- **Input(A) - transfere o bloco físico *A* para a memória principal.**
- **output(*B*) transfere o bloco de buffer *B* para o disco, e substitui o bloco físico apropriado.**



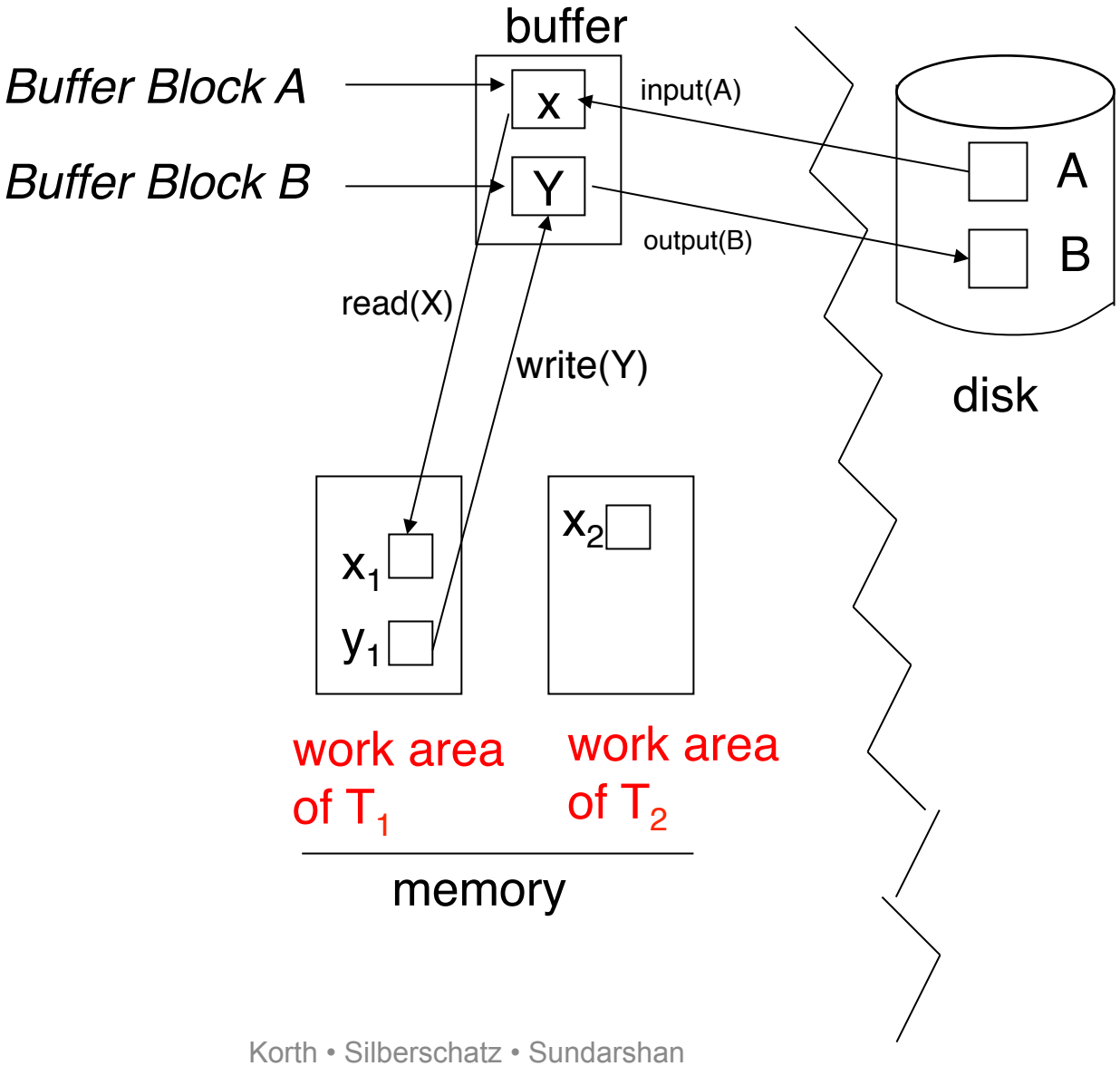
Acesso aos dados

- Cada transação T_i possui sua área de trabalho privada, onde são mantidas as cópias locais de todos os itens de dados acessados e atualizados por ela.
  - A cópia local de T_i de um item de dados X é chamada de x_i.
- Consideramos, por simplicidade, que cada item de dados cabe e é armazenado dentro de um único bloco.

Acesso aos dados (cont.)

- A transação transfere itens de dados entre os blocos de buffer do sistema e sua área de trabalho privada usando as seguintes operações:
  - read(*X*) atribui o valor do item de dado *X* à variável local x_i.
  - write(*X*) atribui o valor da variável local x_i ao item de dados {*X*} no bloco de buffer.
- Transações
  - Execute read(*X*) enquanto acessa *X* pela primeira vez;
  - Todos os acessos subseqüentes são para a cópia local.
  - Após o último acesso, a transação executa write(*X*).

Exemplo de acesso aos dados



Recuperação e atomicidade

- Modificar o BD sem garantir que a transação será confirmada pode leva-lo a um estado inconsistente.
- Considere a transação T_i que transfere \$50 da conta *A* para a conta *B*; o objetivo é realizar todas as modificações do banco de dados feitas por T_i ou nenhuma delas.
- Para garantir a atomicidade apesar das falhas, primeiro geramos informações descrevendo as modificações no armazenamento estável sem modificar o próprio BD.
- Duas técnicas: recuperação baseada em log e paginação de sombra
- Consideramos (inicialmente) que as transações serão executadas em série, ou seja, uma após a outra.

Recuperação baseada em log

- **Um log é mantido no armazenamento estável.**
  - **O log é uma sequencia de registros de log, com todas as atividades de atualização no BD.**
- **Quando a transação T_i inicia, ela se registra escrevendo um registro de log <T_i start>**
- ***Antes que* T_i execute** write(***X***), **um registro de log <T_i, X, V_1, V_2> é escrito, onde V_1 é o valor de X antes do write, e V_2 é o valor a ser escrito em X.**
- **Quando T_i termina sua última instrução, o registro de log <T_i** commit**> é escrito. Se abortada <T_i** abort**>**
  - **Consideramos, por enquanto, que os registros de log são escritos diretamente no armazenamento estável (ou seja, eles não são mantidos em buffer)**

Modificação de banco de dados adiada

- **A técnica de modificação de BD adiada registra todas as modificações no log, mas adia todas as escritas (WRITE) para depois da confirmação parcial (quando a ação final de T estiver sendo executada).**
- **Quando uma transação está parcialmente confirmada, a informação no log associado a transação é usada na execução de escrita das adiadas.**
- **Se o sistema falhar antes que a transação termine sua execução, ou se a transação abortar, então a informação no log é simplesmente ignorada.**

Modificação de banco de dados adiada

- **A execução de T_i procede da seguinte maneira:**
  - **Antes que T_i *inicie sua execução, o* registro <T_i *start*> *é escrito* no log.**
  - **Uma operação** write(***X***) **resulta em um novo registro <T_i, *X, V*>, onde *V* é o novo valor para *X***
    - **Nota: o valor antigo não é necessário nesta tecnica**
  - **A escrita não é realizada em *X* nesse momento, mas é adiada.**
  - **Finalmente, quando T_i confirma parcialmente, um registro <T_i** commit**> é escrito no log**
- **Se não ocorrer nenhuma falha, os registros de log são usados para executar as escritas previamente adiadas.**

Modificação de banco de dados adiada (cont.)

- **Durante a recuperação após uma falha, uma transação precisa ser refeita se e somente se tanto <T_i start> quanto <T_i commit> existirem no log.**
- **Refazer uma transação T_i (** redo T_i**) define o valor de todos os itens de dados atualizado pela transação como os novos valores.**
- **Falhas podem ocorrer enquanto**
  - **a transação estiver executando as atualizações originais, ou**
  - **enquanto a ação de recuperação estiver sendo tomada**

Modificação de banco de dados adiada (cont.)

- **A seguir mostramos o log conforme aparece em três instâncias de tempo.**

(a)	(b)	(c)
<T_0 start>	<T_0 start>	<T_0 start>
<T_0, A, 950>	<T_0, A, 950>	<T_0, A, 950>
<T_0, B, 2050>	<T_0, B, 2050>	<T_0, B, 2050>
	<T_0 commit>	<T_0 commit>
	<T_1 start>	<T_1 start>
	<T_1, C, 600>	<T_1, C, 600>
		<T_1 commit>

- **Se o log no armazenamento estável no momento da falha for um dos casos**
  - **(a) Nenhuma ação de rede precisa ser tomada**
  - **(b) redo(T_0) precisa ser realizado, pois <T_0 commit> está presente**
  - **(c) redo(T_0) precisa ser realizado seguido por redo(T_1), pois <T_0 commit> e <T_i commit> estão presentes**

Modificação de banco de dados imediata

- **Esta técnica permite que as modificações sejam enviadas ao BD enquanto uma transação ainda está no estado ativo.**
- **Como pode ser preciso desfazer, os logs de atualização precisam ter tanto o valor antigo como o valor novo**
- **O registro de log de atualização precisa ser escrito *antes* que o item do banco de dados seja escrito**

Modificação de banco de dados imediata (cont.)

- **O procedimento de recuperação possui duas operações:**
  - undo(T_i) **restaura o valor de todos os itens de dados atualizados por T_i aos seus valores antigos, indo para trás a partir do último registro para T_i**
  - redo(T_i) **define o valor de todos os itens de dados atualizados por T_i aos novos valores, indo para frente a partir do primeiro registro para T_i**
- **As duas operações precisam ser idempotentes**
  - **Ou seja, mesmo que a operação seja executada várias vezes, o efeito é o mesmo que se fosse executada uma vez**
    - **Necessário porque as operações podem ser novamente executadas durante a recuperação**

Modificação de banco de dados imediata (cont.)

- **Ao recuperar-se após a falha:**
  - **A transação T_i precisa ser desfeita se o log tiver o registro <T_i start>, mas não contém o registro <T_i commit>.**
  - **A transação T_i precisa ser refeita se o log tiver o registro <T_i start> e o registro <T_i commit>.**
- **Operações de undo são realizadas primeiro, depois as operações de redo.**

Exemplo de recuperação de modificação de BD imediata

Estado do log em três instâncias de tempo:

(a)	(b)	(c)
<T_0 start>	<T_0 start>	<T_0 start>
<T_0, A, 1000, 950>	<T_0, A, 1000, 950>	<T_0, A, 1000, 950>
<T_0, B, 2000, 2050>	<T_0, B, 2000, 2050>	<T_0, B, 2000, 2050>
	<T_0 commit>	<T_0 commit>
	<T_1 start>	<T_1 start>
	<T_1, C, 700, 600>	<T_1, C, 700, 600>
		<T_1 commit>

As ações de recuperação em cada um destes casos são:

(a) undo (T_0): B é restaurado para 2000 e A para 1000.

(b) undo (T_1) e redo (T_0): C é restaurado para 700, *A* e *B* são definidos para 950 e 2050.

(c) redo (T_0) e redo (T_1): A e B são definidos para 950 e 2050 respectivamente. Depois, *C* é definido para 600

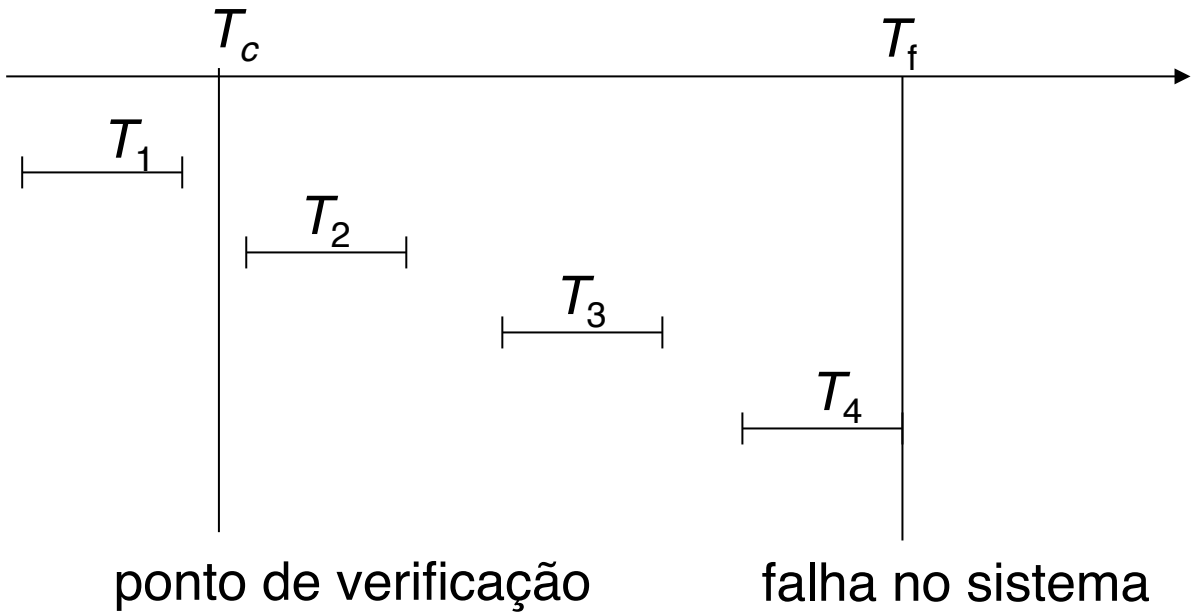
Pontos de verificação (Check points)

- **Quando ocorre uma falha, é preciso consultar o log para determinar as transações que precisam ser refeitas e as que precisam ser desfeitas.**
- **Problemas no procedimento de recuperação:**
  - **pesquisar o log inteiro é demorado**
  - **poderíamos desnecessariamente refazer transações que já enviaram atualizações para o BD.**

Pontos de verificação (Check points)

- **Para reduzir esses tipos de sobrecarga, introduzimos os pontos de verificação (check points), que exigem as seguintes ações:**
  - **Enviar todos os registros de log atualmente residindo na memória principal para o armazenamento estável.**
  - **Envie todos os blocos de buffer modificados para o disco.**
  - **Escreva um registro de log <**checkpoint**> no armazenamento estável.**

Exemplo de pontos de verificação



T_1 pode ser ignorada (atualizações já enviadas ao disco devido ao ponto de verificação)

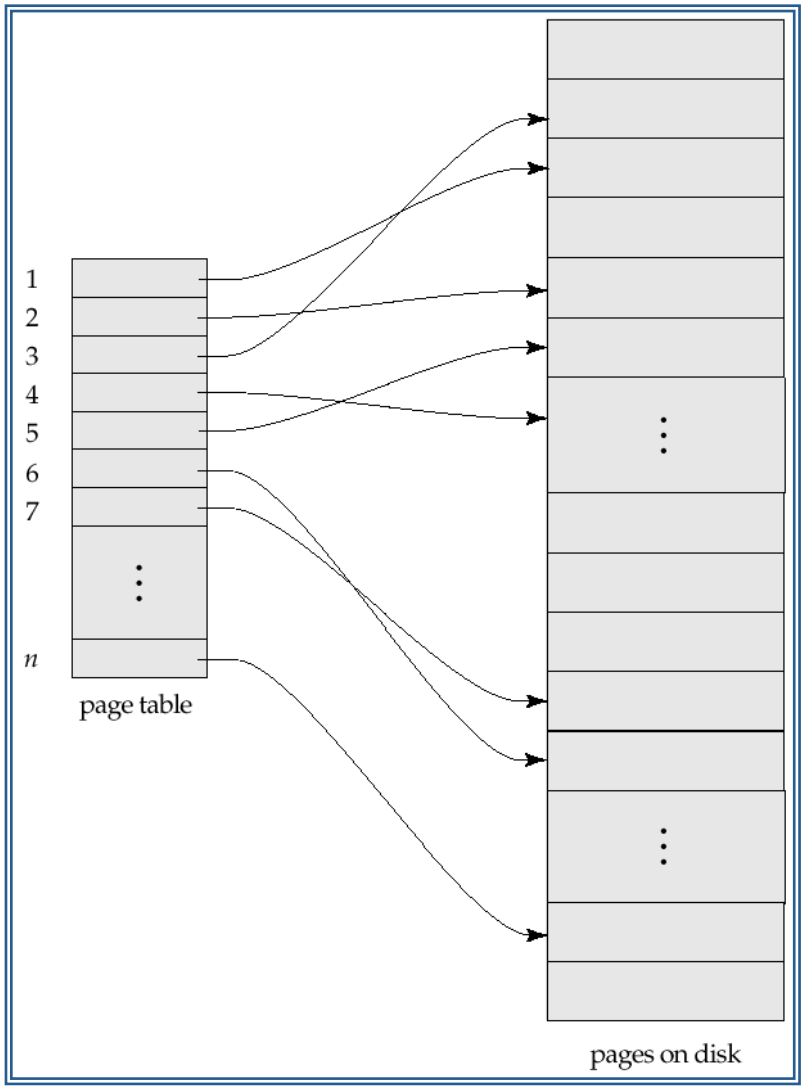
- **T_2 T_3 T_4 refeitos**

Paginação de sombra

- O BD é particionado em um número de blocos de comprimento fixo, chamado de páginas
- A paginação de sombra é uma alternativa à recuperação baseada em log; útil se as transações forem executadas em série
- Essas páginas não necessitam ser armazenadas e alguma ordem no disco

Exemplo de tabela de página

- A tabela de página tem n entradas, uma para cada página no BD
- Cada entrada contém um **ponteiro** para uma página no disco
- Conforme a figura, a ordem lógica das páginas não necessitam corresponder a ordem física no disco



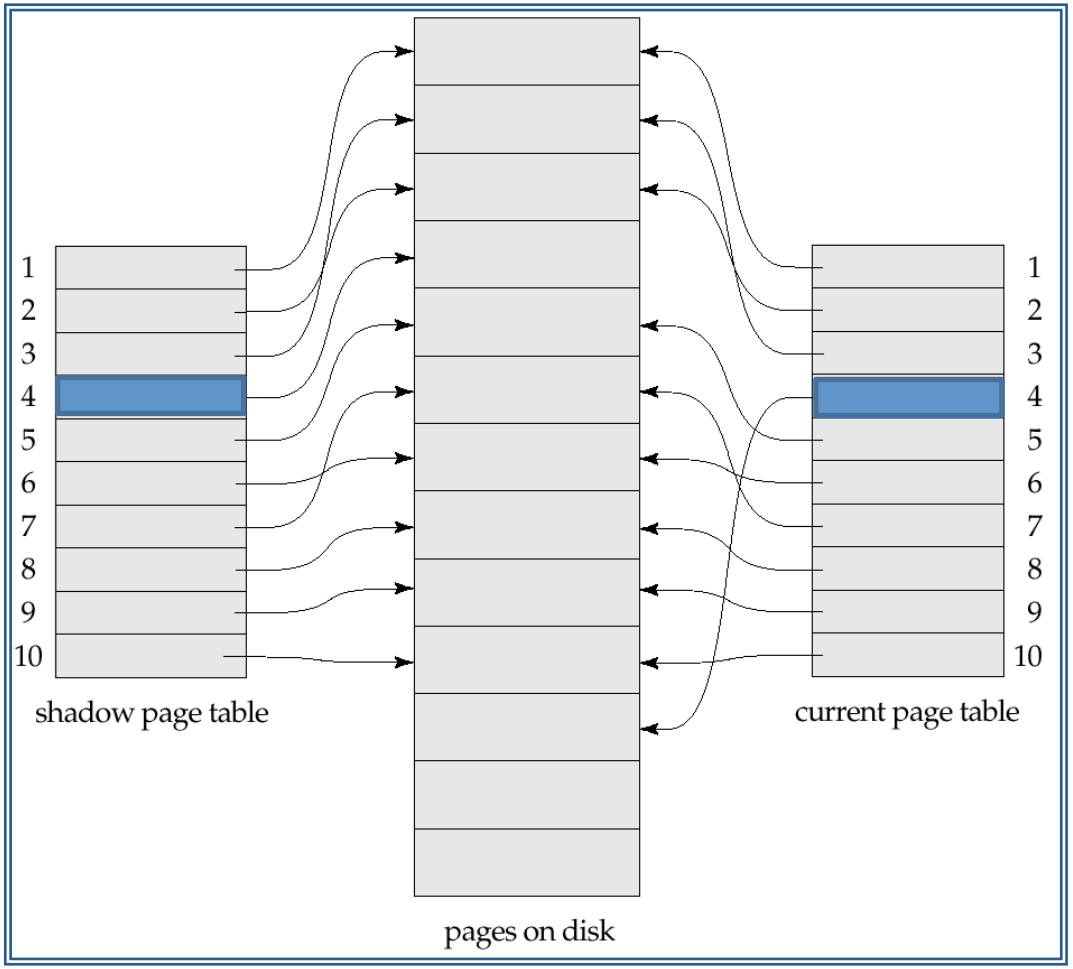
Paginação de sombra

- **Idéia: manter *duas* tabelas de página durante o tempo de vida de uma transação - a tabela de página atual, e a tabela de página de sombra**
- **A tabela de páginas atuais pode ser mantida na memória principal (armazenamento volátil)**
- **A tabela de página de sombra é armazenada (armazenamento não volátil) de modo que o estado do BD antes da execução da transação possa ser recuperado.**
  - **A tabela de página de sombra nunca é modificada durante a execução**

Exemplo de paginação de sombra

Tabelas de sombra e página atual após escrita na página 4

- **Inicialmente, as duas tabelas de página são idênticas.**
- **Somente a tabela de página atual é usada para os acessos ao item de dados durante a execução da transação**
- **Quando a transação é parcialmente efetivada, a tabela shadow é descartada e a corrente torna-se a nova tabela de páginas**
- **Se a transação é abortada, a tabela de páginas atuais é simplesmente descartada**



Paginação de sombra

- **Vantagens em relação aos esquemas baseados em log**
  - a recuperação é trivial
  - nenhuma sobrecarga de escrita de registros de log
- **Desvantagens:**
  - Copiar a tabela de página inteira é muito dispendioso
  - Os dados são fragmentados (páginas relacionadas ficam separadas no disco)
  - Após o término de cada transação, as páginas do banco de dados contendo versões antigas de dados modificados precisam passar pela coleta de lixo
  - Difícil de estender o algoritmo para permitir que transações sejam executadas simultaneamente
    - Mais fácil para estender esquemas baseados em log

Recuperação com transações concorrentes

- **Modificamos os esquemas de recuperação baseados em log para permitir que várias transações sejam executadas simultaneamente.**
  - **Todas as transações compartilham um único buffer de disco e um único log**
  - **Um bloco de buffer pode ter itens de dados atualizados por uma ou mais transações**
- **O esquema de recuperação depende muito do esquema de controle de concorrência que é usado**
- **Consideramos o controle de concorrência usando um bloqueio estrito em duas fases;**
  - **ou seja, as atualizações de transações não confirmadas não devem ser visíveis a outras transações**

Recuperação com transações concorrentes

- **O logging é feito conforme descrevemos anteriormente.**
  - **Os registros de log de diferentes transações podem ser intercalados no log.**
- **A técnica de ponto de verificação e as ações tomadas na recuperação precisam ser alteradas**
  - **pois várias transações podem estar ativas quando um ponto de verificação é realizado.**
- **Os pontos de verificação são realizados como antes, exceto que o registro de log do ponto de verificação agora tem a forma < checkpoint L>**
  onde L é a lista de transações ativas no momento do ponto de verificação.

Recuperação com transações concorrentes (cont.)

- **Quando o sistema se recupera de uma falha:**
  1. **Inicializa a *lista de undo* e *lista de redo* para vazio**
  2. **Varre o log para trás a partir do fim, parando quando o primeiro registro <**checkpoint ***L*****> for encontrado.**

  – **Para cada registro encontrado durante a varredura:**
    - ★ **se o registro for <T_i** commit**>, acrescenta T_i à *lista de redo***
    - ★ **se o registro for <T_i** start**>, então se T_i não está na *lista de redo*, acrescenta T_i à *lista de undo***

  → ***lista de redo*** **consiste em transações acabadas, que precisam ser refeitas.**
  → ***lista de undo*** **consiste em transações incompletas, que precisam ser desfeitas.**
  → **L, lista de transações ativas no momento do ponto de verificação**

  3. Verifica a lista L: para cada T_i em *L*, se T_i não estiver na *lista de redo*, acrescenta T_i à *lista de undo*

Recuperação com transações concorrentes (cont.)

- **Quando a lista de redo e undo tiverem sido construídas, a recuperação continua da seguinte forma:**

  1. **O sistema varre novamente o log para trás a partir do registro mais recente e realiza undo para cada registro de log que pertence a uma transação na lista de undo.**
     - » **Os registros de logs das transações na lista redo são ignorados nesta fase.**
     - » **A varredura termina quando os registros <Ti start> tiverem sido encontrados para cada transação Ti na lista de undo**
  2. **Localize o registro <checkpoint L> mais recente.**
  3. **O sistema varre o log para frente a partir do registro <checkpoint L> até o final do log. Durante a varredura, realize redo para cada registro de log que pertence a uma transação na lista de redo.**

Buffering de registro de log

- **Em muitos casos, um registro de logo é muito menor que um bloco;**
- **O custo da saída de bloco para armazenamento estável é alto. Portanto é melhor que saiam diversos registros de log de uma só vez.**
- **Buffering de registro de log: os registros de log são mantidos na memória principal, onde permanecem temporariamente, até serem enviados para o armazenamento estável.**
- **Registros de log são enviados ao armazenamento estável quando um bloco de registros de log no buffer estiver cheio, ou uma operação de log forçado for executada.**

Buffering de registro de log

- **Vários registros de log, portanto, podem ser enviados por meio de uma única operação de saída, reduzindo o custo da E/S.**
- **Como os registros de log ficam na memoria principal (volátil) por um tempo, se o sistema cair tais registros podem ser perdidos.**
- **O log forçado é realizado para confirmar uma transação forçando todos os seus registros de log (incluindo o registro de commit) para o armazenamento estável.**

Buffering de registro de log (cont.)

- **Para garantir a atomicidade das transações, as regras a seguir precisam ser seguidas:**
  - **Os registros de log são enviados para o armazenamento estável na ordem em que são criados.**
  - **A transação T_i só entra no estado de commit quando o registro de log <T_i commit> tiver sido enviado ao armazenamento estável.**
  - **Antes que um bloco de dados na memória principal seja enviado ao banco de dados, todos os registros de log pertencentes aos dados nesse bloco precisam ter sido enviados ao armazenamento estável.**
    - **Essa regra é chamada logging de escrita antecipada ou regra WAL (Write Ahead Logging)**

Buffering de banco de dados

- O BD mantém um buffer na memória dos blocos de dados
  - Quando um novo bloco é necessário, se o buffer estiver cheio, um bloco existente precisa ser removido do buffer
  - Se o bloco escolhido para remoção tiver sido atualizado, ele terá que ser enviado para o disco
- Como resultado da regra de logging de escrita antecipada, se um bloco com atualizações não confirmadas for enviado ao disco, os registros de log devem ser enviados ao log no armazenamento estável primeiro.

Buffering de banco de dados

- **Considere que um bloco B2 precisa ser carregado na MP e não tem mais espaço.**
  - **O Bloco B1 terá que ser enviado para armazenamento estavel para abrir espaço para o bloco B2.**
- **A sequência de ações do sistema é como segue:**
  - **Enviar todos os registros de log pertencentes a um bloco B1 para armazenamento estável**
  - **Enviar o bloco B1 para saída no disco**
  - **Enviar o bloco B2 do disco para a entrada na memória principal**

→ **É importante que nenhuma gravação no bloco B1 esteja em processo enquanto esta sequencia de ações estiverem em execução. Esta condição é satisfeita quando se usa um meio especial de bloqueio, como segue.**

Buffering de banco de dados

- **Para o bloqueio especial, a sequência de ações do sistema é como segue:**
  - **Antes de escrever um item de dados, a transação adquire bloqueio exclusivo sobre o bloco contendo o item de dados**
  - **O bloqueio pode ser liberado quando a escrita terminar.**
    - **Tais bloqueios mantidos por uma curta duração software chamados de latches (trancas).**
  - **Antes que um bloco seja enviado ao disco, o sistema adquire um latch exclusivo sobre o bloco**
    - **Garante que nenhuma atualização pode estar em andamento no bloco**

Gerenciamento de buffer (cont.)

- **O buffer de BD pode ser implementado:**
  1. **Em uma área de memória principal real reservada para o BD gerenciar (não pelo SO)**
     - **Desvantagem: Limita a flexibilidade do uso da MO - A memória é particionada de antemão entre o buffer de BD e as aplicações, mesmo quando não tiver aplicações usando a MP esta não poderá ser utilizada pelo BD**
  2. **Na memória virtual do sistema operacional**
     - **Desvantagem: o BD perde o controle da saída de blocos do buffer. Normalmente SO atuais não são projetados para aceitar exigências de gerenciamento de log do banco de dados.**

Falha com perda de armazenamento não volátil

- **Embora falhas no armazenamento não volátil sejam raras, estas devem ser consideradas.**
- **Esquema básico:**
  - **Periodicamente descarregar (dump) todo o conteúdo inteiro do BD para armazenamento estável (uma vez por dia)**
  - **Nenhuma transação pode estar ativa durante o procedimento de dump;**
  - **Necessário procedimento similar para checkpoints**
    - **Enviar todos os registros de log atualmente residindo na memória principal para o armazenamento estável.**
    - **Enviar todos os blocos de buffer para o disco.**
    - **Copie o conteúdo do banco de dados para o armazenamento estável.**
    - **Envie um registro <**dump**> para log no armazenamento estável.**

Falha com perda de armazenamento não volátil

- **O procedimento de descarga simples descrito anteriormente é oneroso pelos seguintes razões:**
  - **Todo BD deverá ser copiado em armazenamento estável, resultando em considerável transferência de dados**
  - **Já que o processamento de transações é suspenso durante a descarga, siclos de CPU são perdidos**

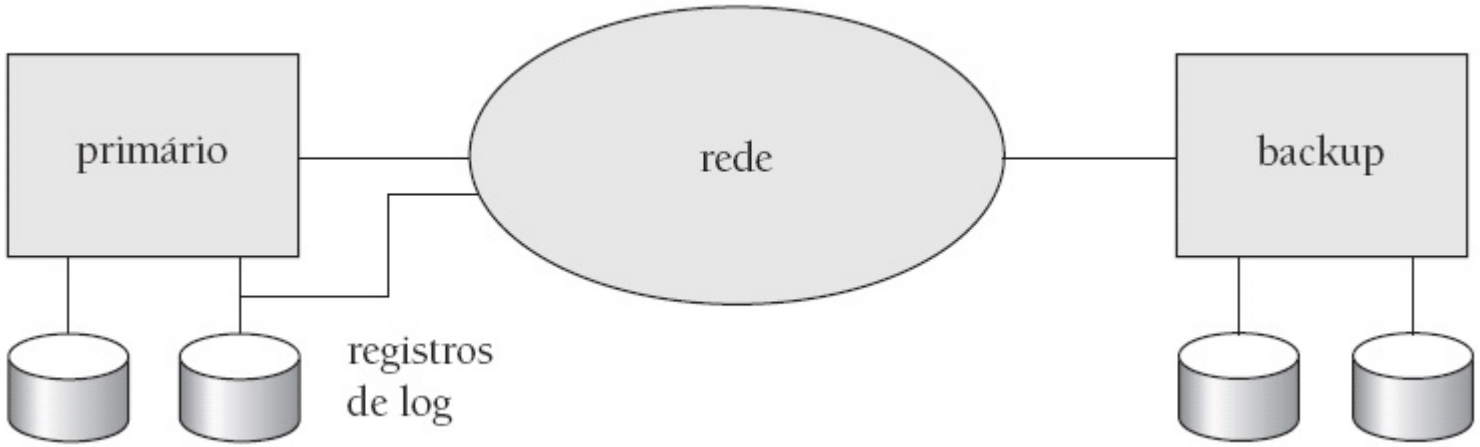
- **Existem esquemas donominados fuzzy dmp (descarga indistinta) que permitem a existencia de transações ativas enquanto a descarga está em progresso.**

Técnicas de recuperação avançadas

- **Admite técnicas de bloqueio que aumentam a concorrência, que utilizam algoritmos para controle de concorrência de árvore B^+, permitindo que os bloqueios sejam liberados mais cedo, sem usar duas fases.**
- **Operações como inserções e exclusões de árvore B^+ liberam bloqueios mais cedo.**
- **ARIES é um método de recuperação de última geração**
  - **Incorpora diversas otimizações para reduzir sobrecargas durante o processamento normal e agilizar a recuperação**

Sistemas de backup remoto

- **Os sistemas de backup remoto oferecem alta disponibilidade, permitindo que o processamento de transação continue mesmo que o site primário seja destruído.**



Sistemas de backup remoto (cont.)

- **Detecção de falha: o site de backup precisa detectar quando o site primário falhou**
  - **para distinguir a falha do site primário da falha do enlace, mantenha vários links de comunicação entre o site primário e o backup remoto.**
- **Transferência de controle:**
  - **Para assumir o controle, o site de backup primeiro realiza a recuperação usando sua cópia do banco de dados e todos os registros de log que recebeu do site primário.**
    - **Assim, transações completadas são refeitas e transações incompletas são descartadas.**
  - **Quando um site de backup assume o processamento, ele se torna o novo site primário**
  - **Para transferir o controle de volta ao antigo primário quando se recuperar, o antigo primário precisa receber logs de redo do backup antigo e aplicar todas as atualizações localmente.**

Referência

- KORTH, H. F.; SILBERSCHATZ, A.; SUDARSHAN, S. Sistema de Banco de Dados, Tradução da 5ª Edição, Campus, 2006.