SISTEMAS DE RECUPERAÇÃO RECUPERAÇÃO BASEADA EM LOGS

Sérgio Mergen

Versão modificada de Database System Concepts, 5th Ed.

Sistemas de Recuperação

- Na transação ao lado, ocorreu uma falha antes que a transação comitasse
- Essa falha impede que a transação prossiga
- Para garantir a atomicidade, é necessário
  - Reverter a transação, desfazendo instruções já executadas (roll back) ou
  - Concluir a transação, executando as instruções pendentes

- Essas ações ficam a cargo do sistema de recuperação de um banco de dados

T1
read (A)
A := A - 50
write *(A)*
Falha
read *(B)*
B := B + 50
write *(B)*
commit

Classificação de Falhas em SGBDs

- As falhas que podem levar a interrupção de uma transação são:
  - Falhas da transação
  - Crashs no sistema
  - Falhas no disco

Classificação de Falhas em SGBDs

- **Falha da Transação**:
  - **Erros Lógicos**:
    - transação não pode continuar devido a algum erro interno
  - **Erros de Sistema**:
    - o sistema deve interromper uma transação devido a algum problema detectado
      - ex. Deadlock

Classificação de Falhas em SGBDs

- **Crash no Sistema**
  - uma falha de energia ou falha de hardware ou software.

- **Falhas no disco**
  - uma quebra de cabeçote e outras falhas do tipo destroem todo o disco ou parte dele

Suposição Fail-stop

- **assume-se que o conteúdo não-volátil não seja corrompido por falhas**
- **Sistemas de banco de dados usam diversas verificações de integridade para *prevenir erros* em dados no disco**

Algoritmos de Recuperação

- Algoritmos de Recuperação são *técnicas* que garantem consistência, atomicidade e durabilidade das transações
  - apesar das possíveis falhas

- Os algoritmos dividem-se em duas partes*:*
  1. Ações tomadas **durante** o processamento normal das transações

     para garantir que existam informações suficientes para a recuperação em caso de falha

  2. Ações tomadas **depois** de uma falha

     para trazer o conteúdo do banco a um estado que garante atomicidade, consistência e durabilidade

Acesso aos Dados: Input e Output

- Os itens de dados que se deseja acessar estão em blocos físicos no disco

- Para serem usados, os blocos precisam ser carregados para a memória primária (buffer).

- Transferências de bloco
  - Entre o disco e a memória
  - Composto por duas operações
    - **input**(*B*) transfere o bloco físico B para a memória.
    - **output**(*B*) transfere o bloco B em memória para o disco

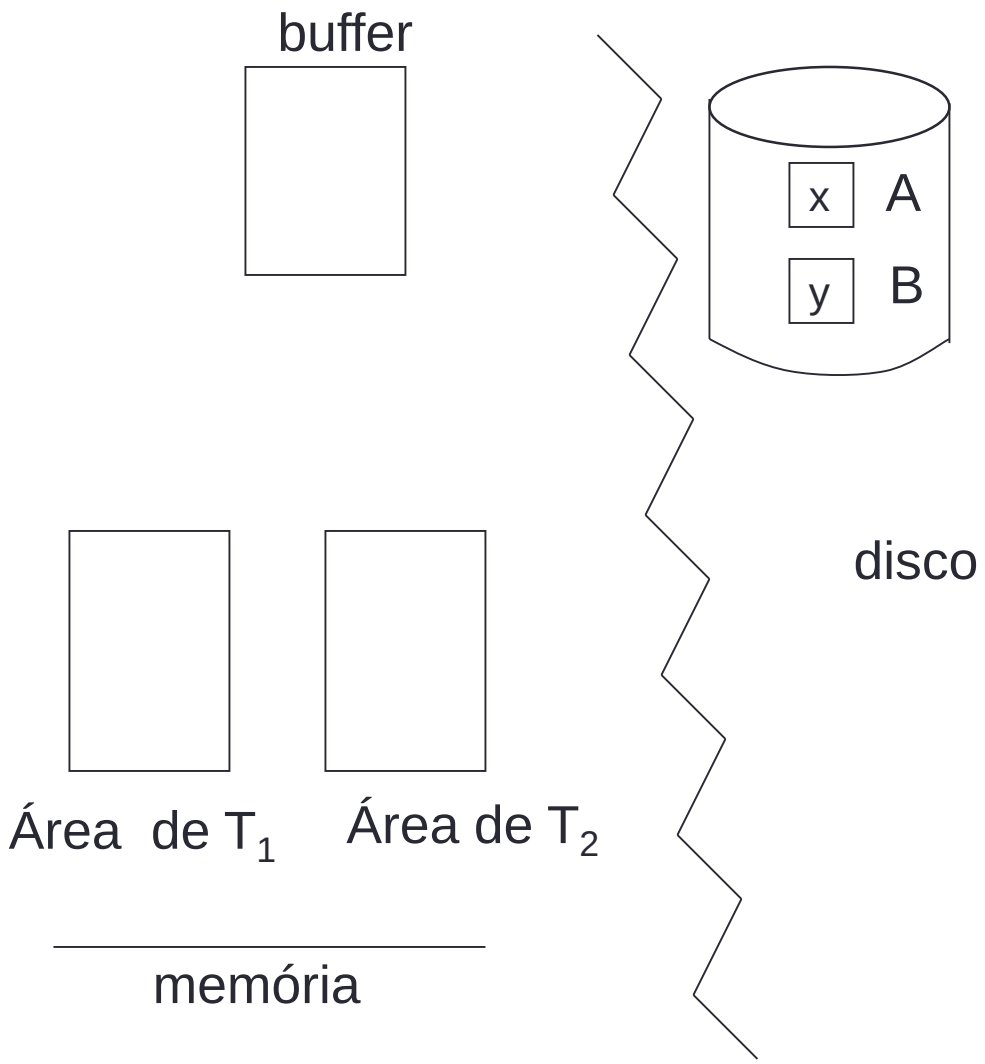
Acesso aos Dados: Read e Write

- Cada transação $\mathbf{T_i}$ tem sua área de trabalho privada
  - onde cópias locais de todos itens de dados acessados e atualizados são mantidos
  - A cópia local do item X usado por $\mathbf{T_i}$ é chamada $\mathbf{x_i}$.

- Uma transação transfere itens de dados entre a memória e sua área privada usando as seguintes operações:
  - read(X) atribui o item em memória X a uma variável local x_i.
  - write(X) atribui a variável local x_i à memória.

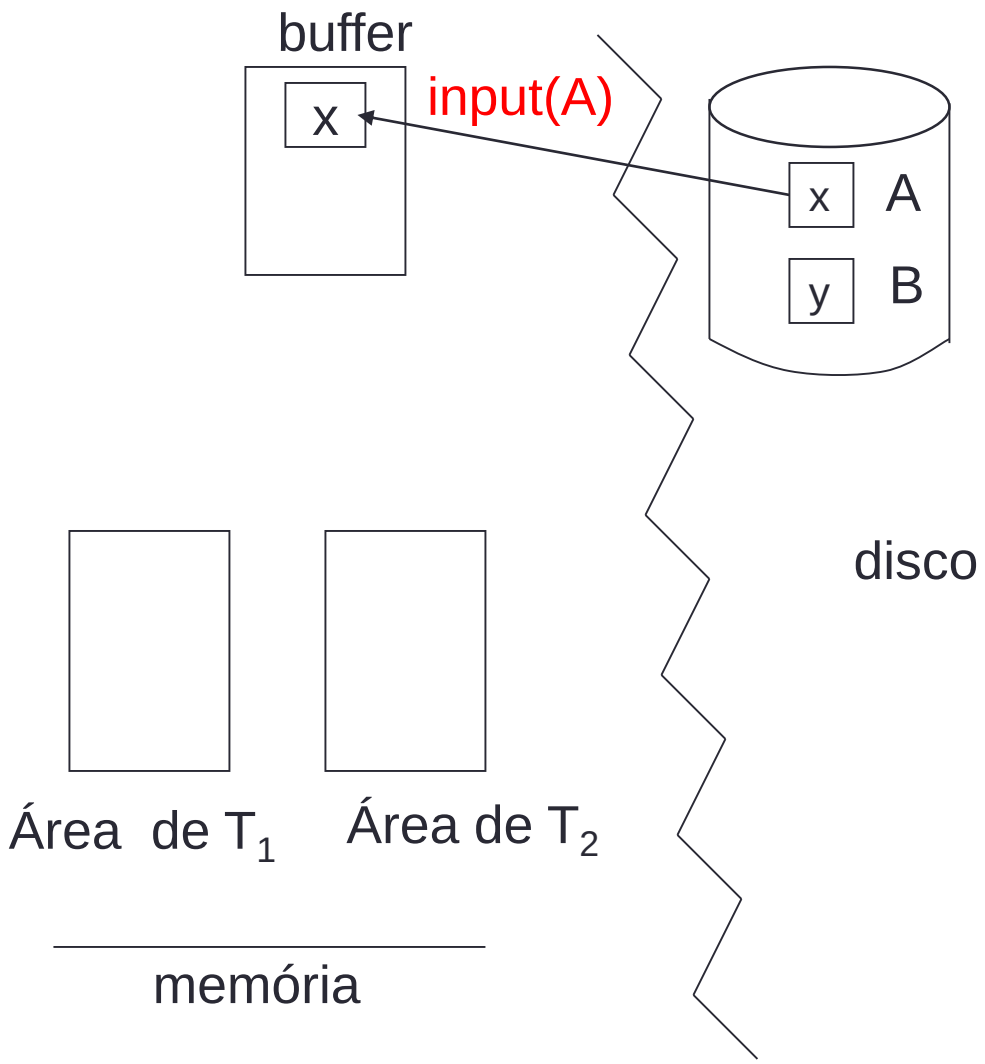
Exemplo de Acesso aos Dados

T1	T2
read (X) read(Y) Write(Y) . . .	 read(X) . . .



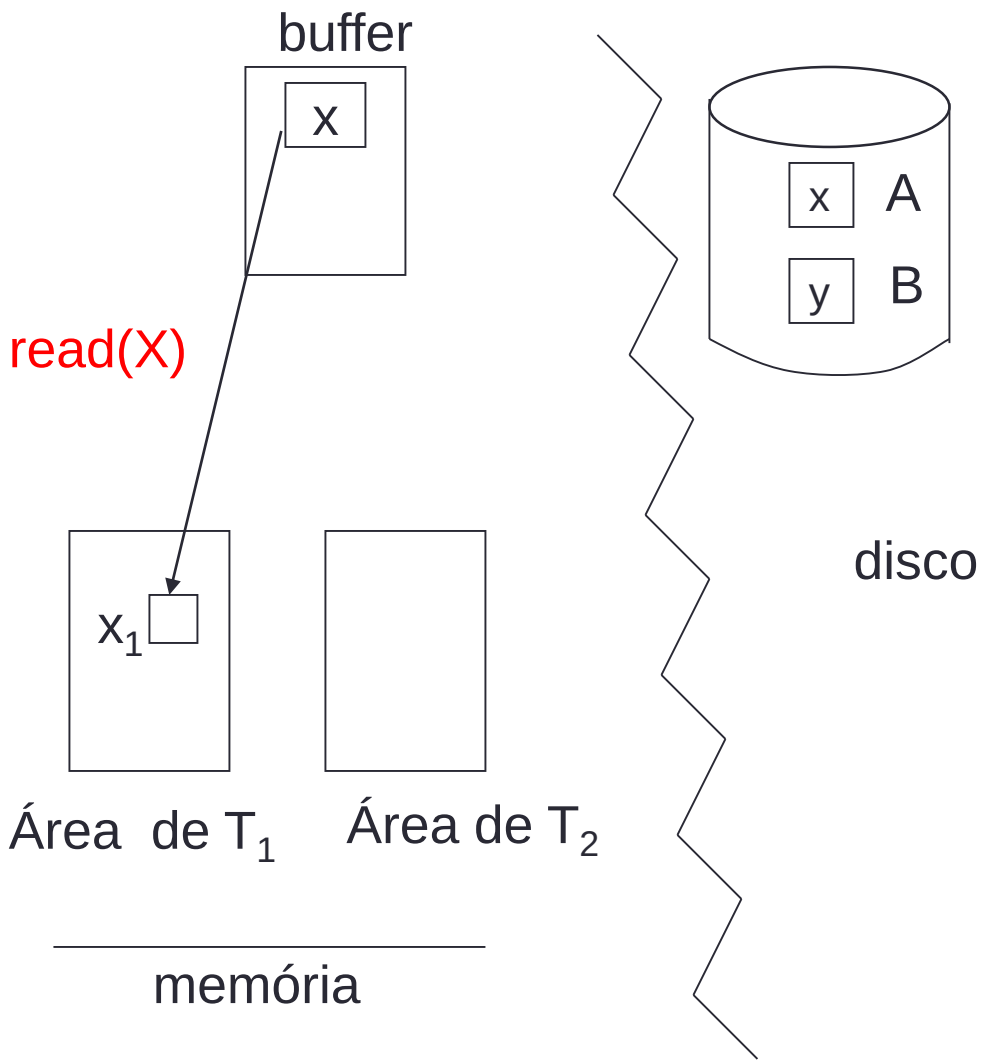
Exemplo de Acesso aos Dados

T1	T2
read (X) read(Y) Write(Y) . . .	 read(X) . . .



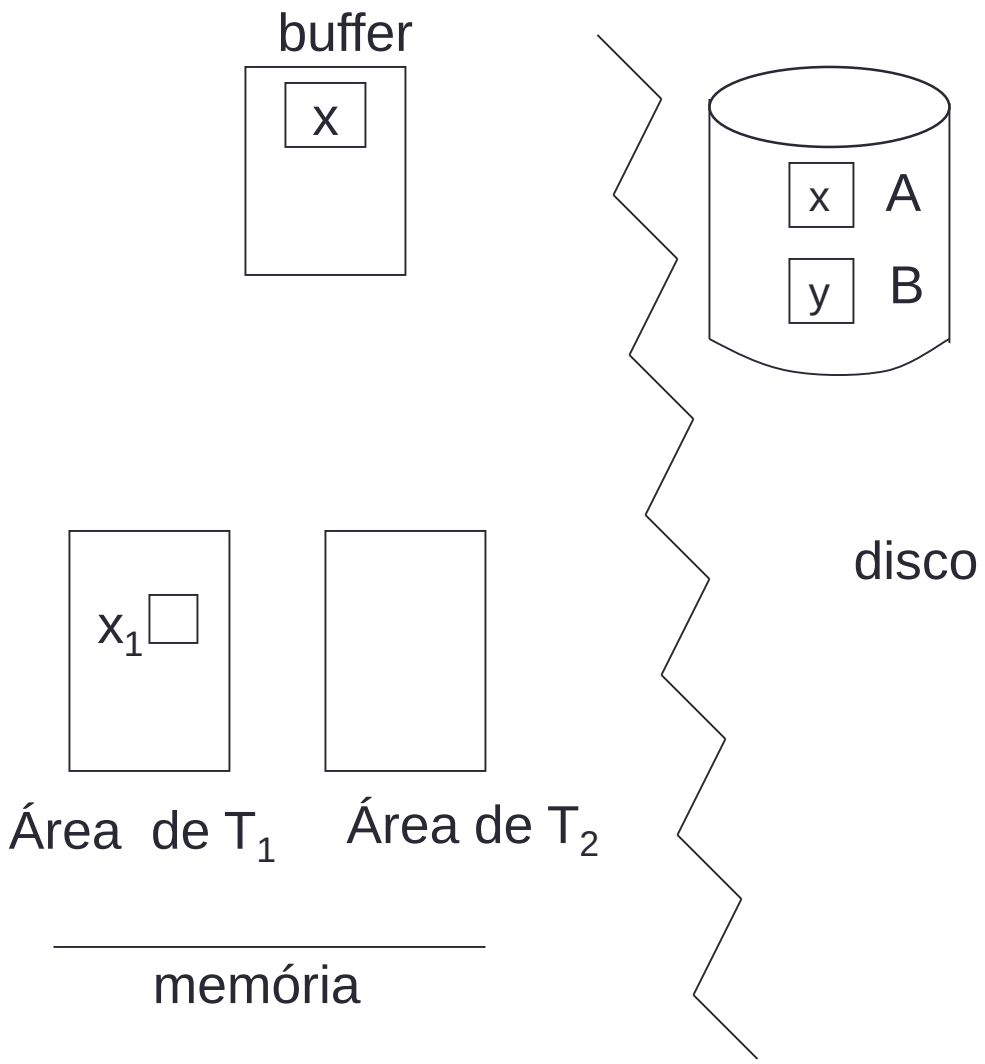
Exemplo de Acesso aos Dados

T1	T2
read (X) read(Y) Write(Y) . . .	 read(X) . . .



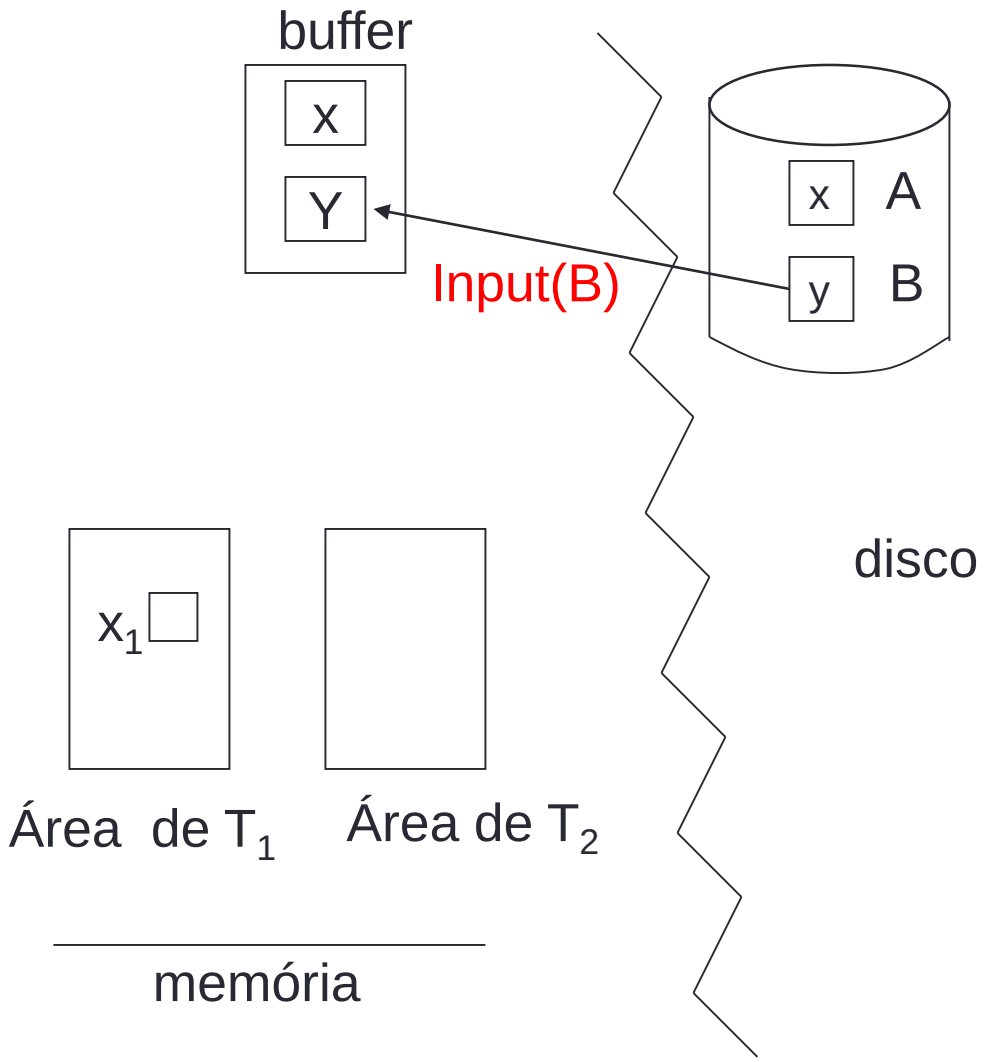
Exemplo de Acesso aos Dados

T1	T2
read (X) **read(Y)** **Write(Y)** .	 **read(X)** .



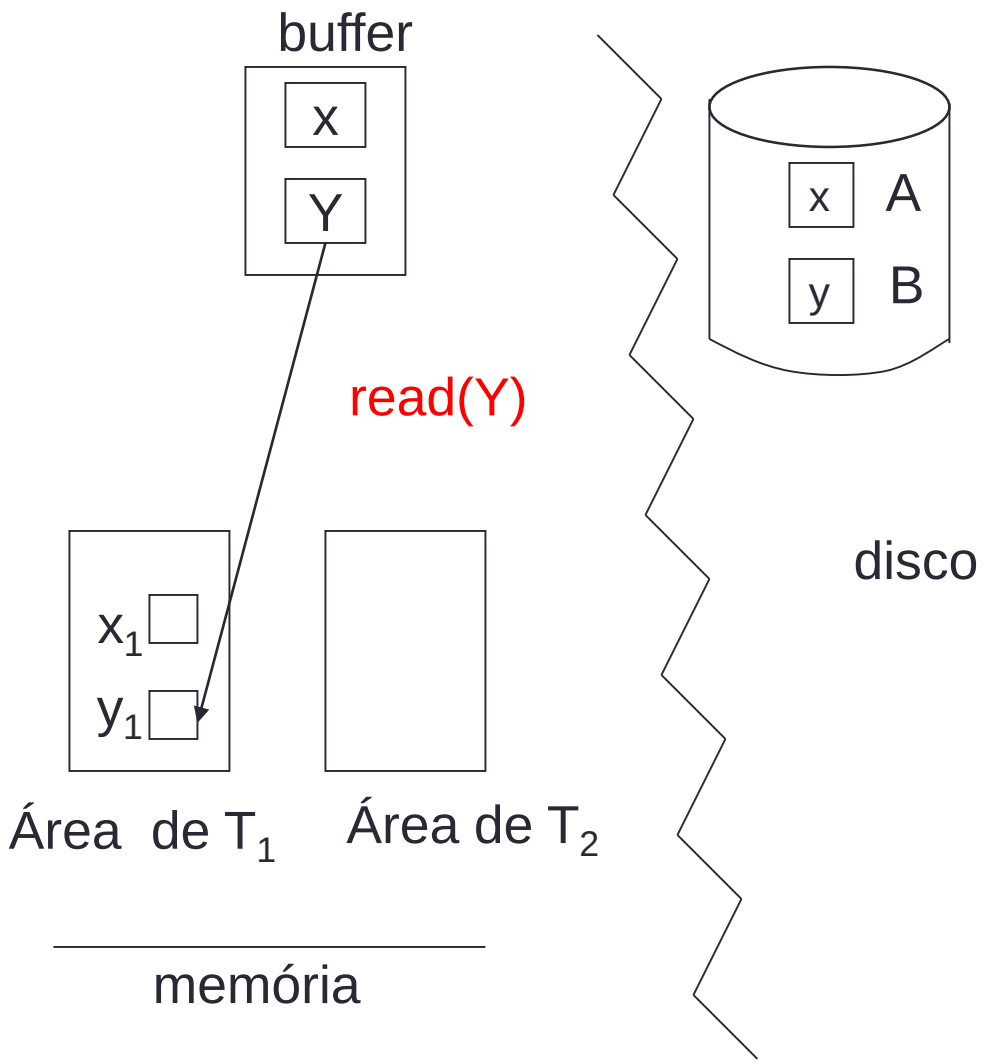
Exemplo de Acesso aos Dados

T1	T2
read (X) **read(Y)** **Write(Y)** .	 **read(X)** .



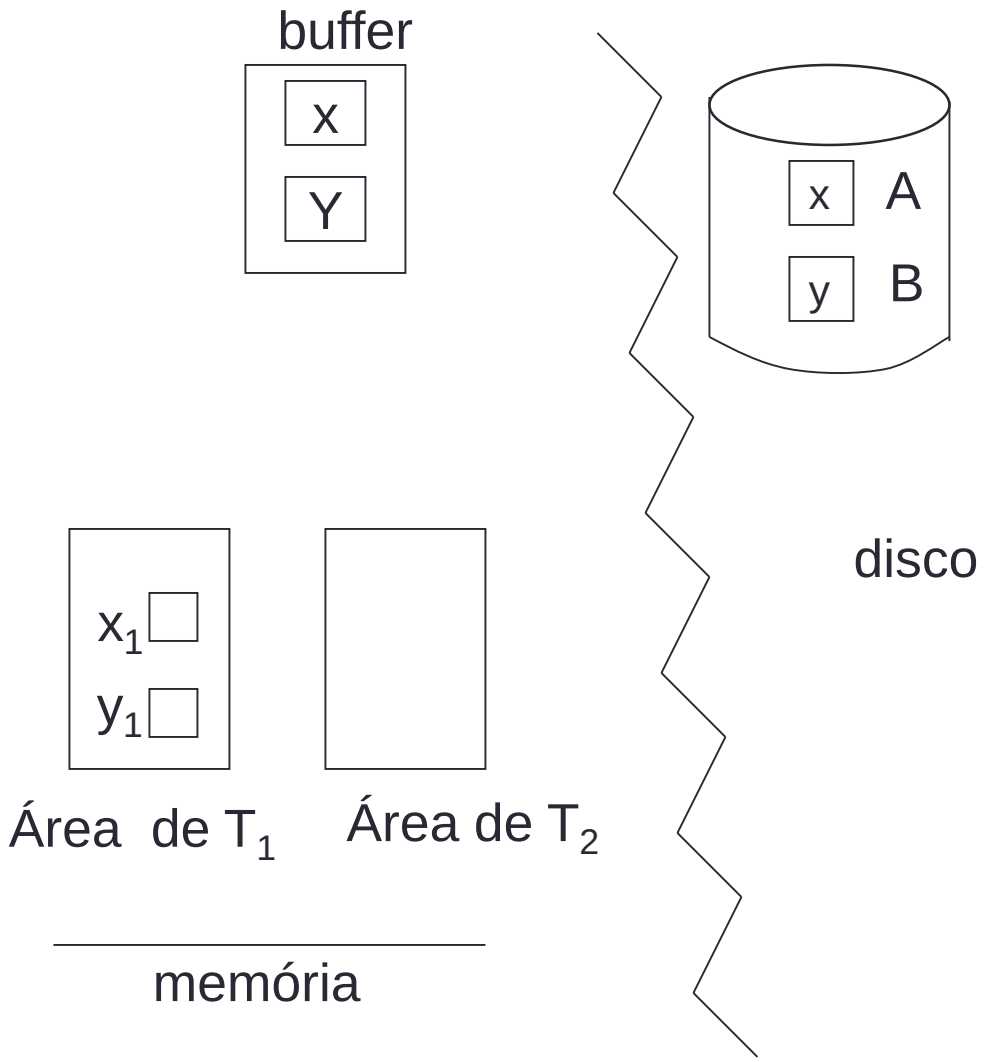
Exemplo de Acesso aos Dados

T1	T2
read (X) **read(Y)** **Write(Y)** . . .	 **read(X)** . . .



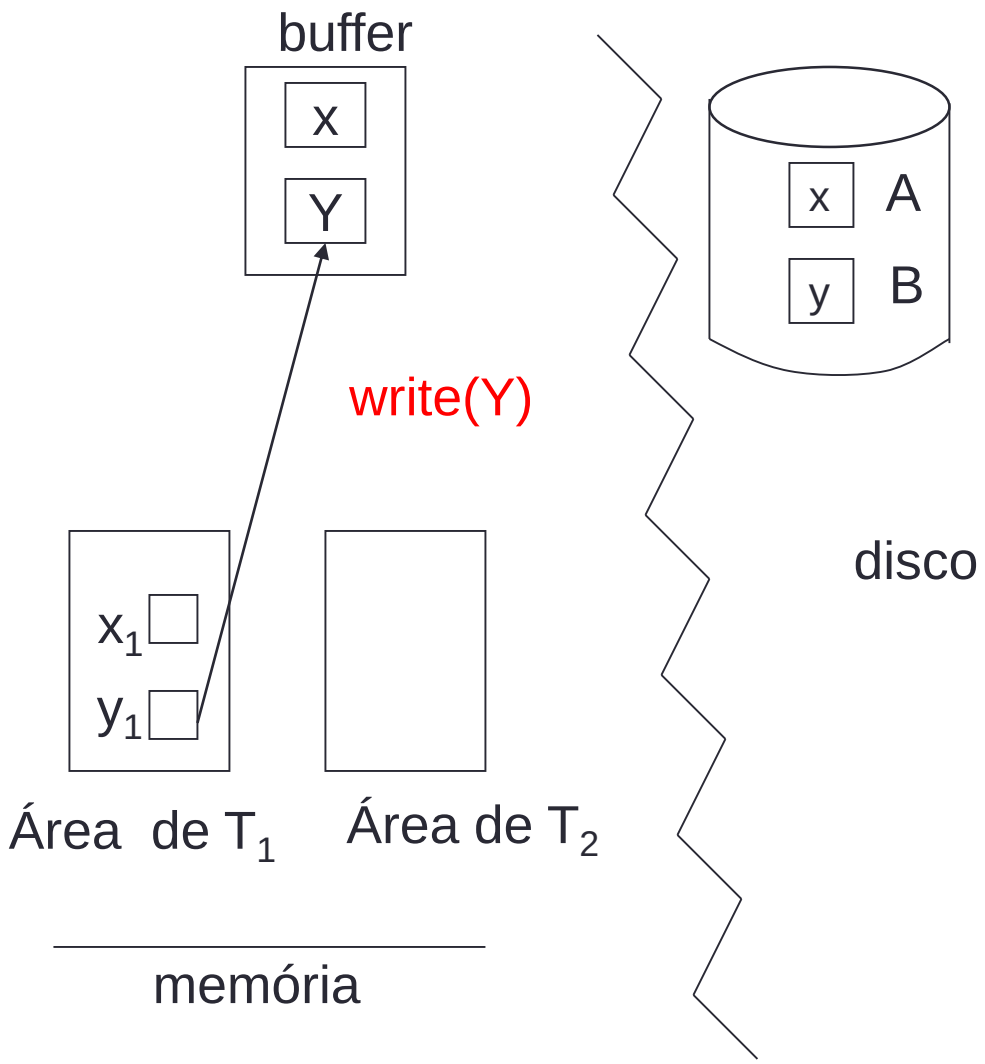
Exemplo de Acesso aos Dados

T1	T2
read (X) **read(Y)** **Write(Y)** . . .	read(X) . . .



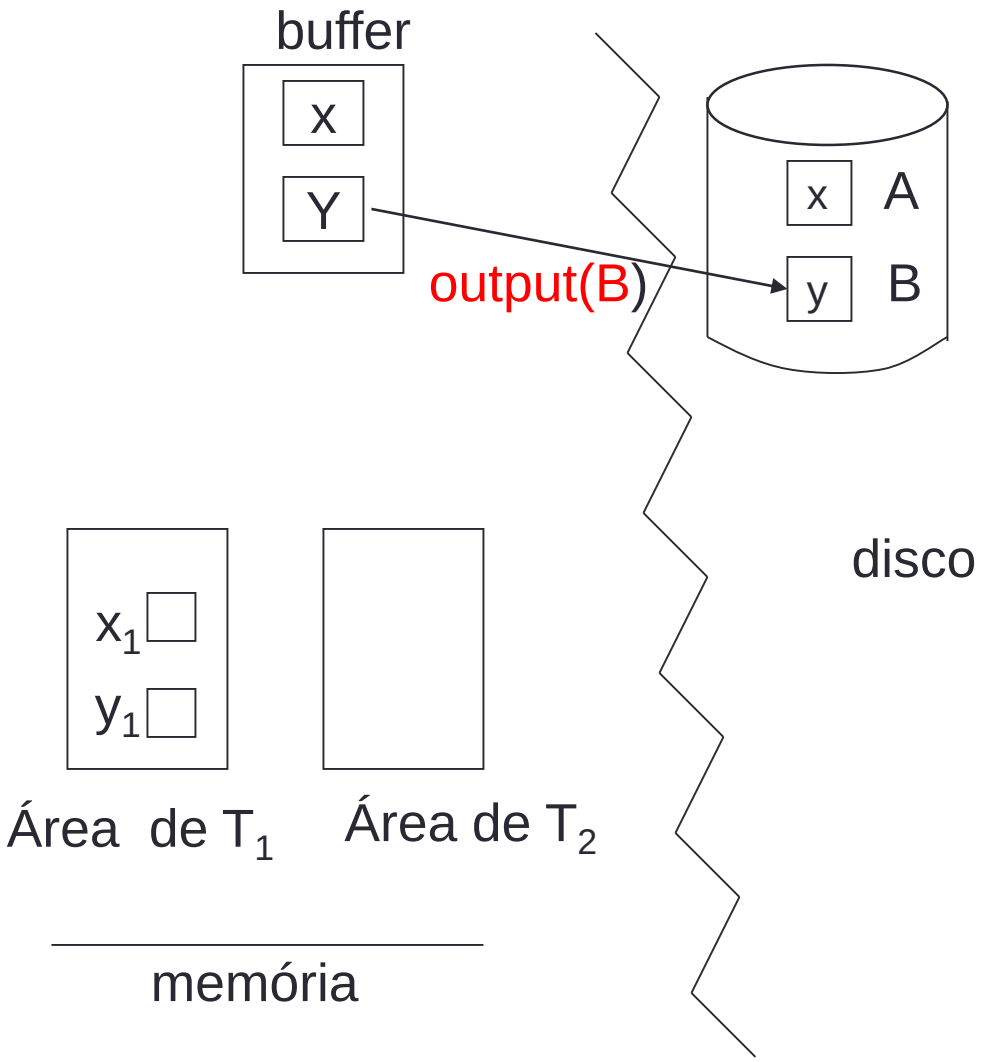
Exemplo de Acesso aos Dados

T1	T2
read (X) **read(Y)** **Write(Y)** . . .	read(X) . . .



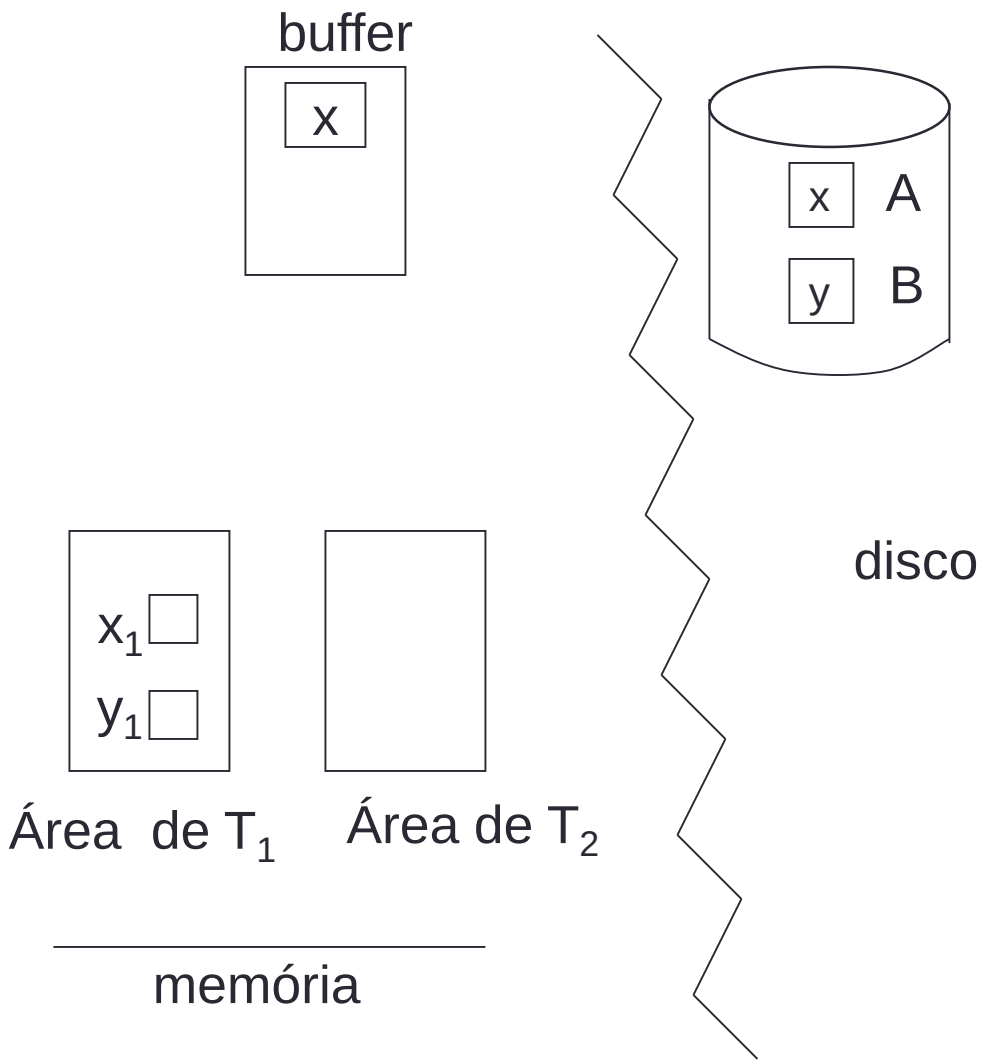
Exemplo de Acesso aos Dados

T1	T2
read (X) **read(Y)** **Write(Y)** . . .	read(X) . . .



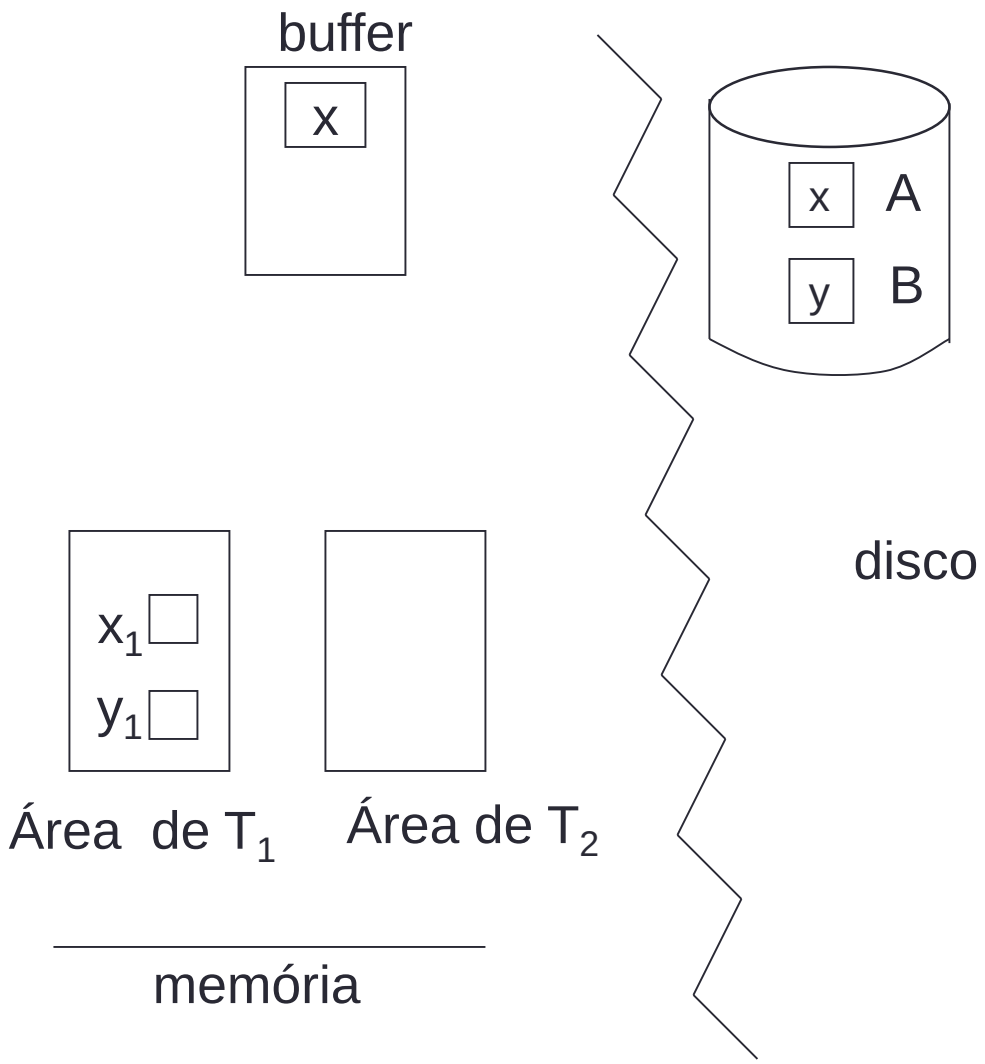
Exemplo de Acesso aos Dados

T1	T2
read (X) **read(Y)** **Write(Y)** . . .	read(X) . . .



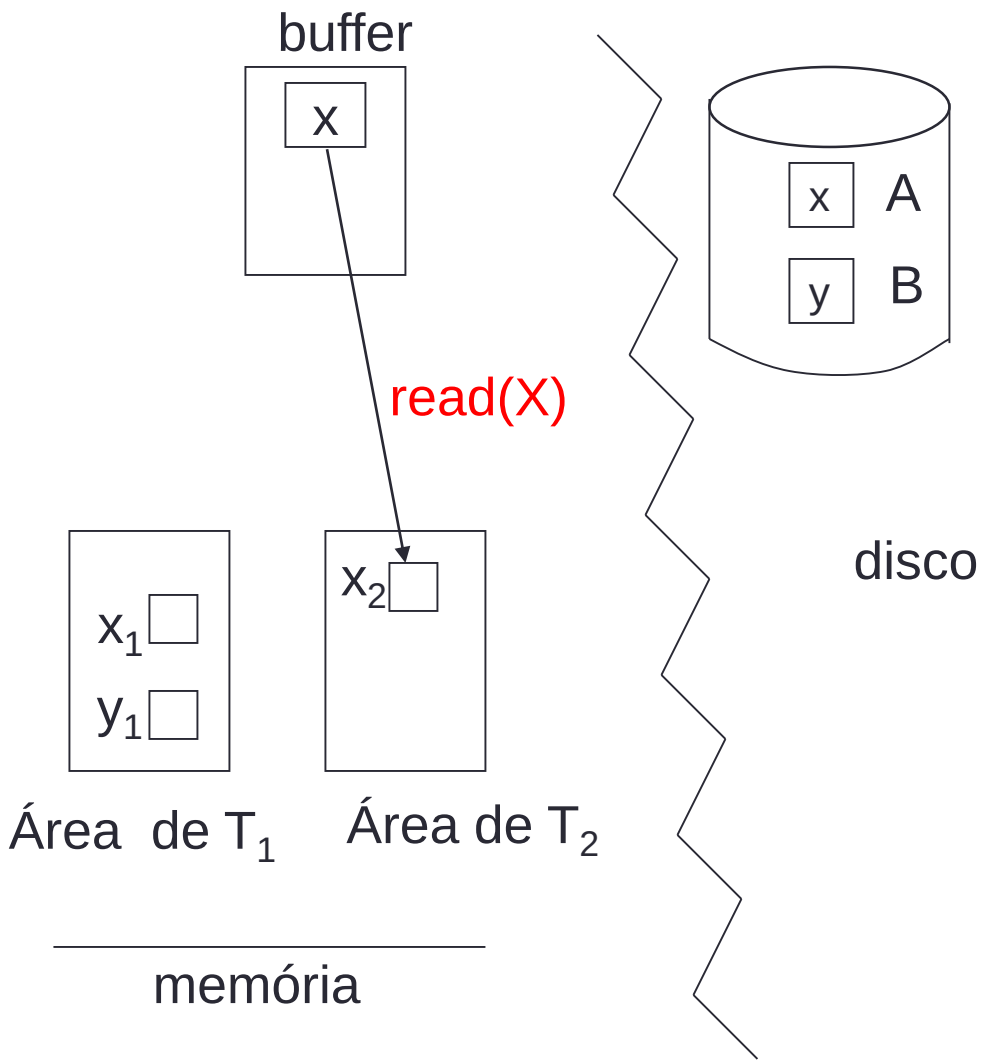
Exemplo de Acesso aos Dados

T1	T2
read (X) **read(Y)** **Write(Y)** . . .	**read(X)** . . .



Exemplo de Acesso aos Dados

T1	T2
read (X) **read(Y)** **Write(Y)** .	**read(X)**



Acesso aos Dados

- Read(X) / Input(B)
  - Uma transação que realiza read(X) resgata X da memória
  - Se X ainda não estiver em memória
    - É feito um input(X).
- Write(X) / Output (B)
  - Uma transação manipula a cópia local de X
  - Um escrita (write) é enviada para a memória.
  - output(B_X) não precisa vir logo após o write(X).
    - O SGBD podem realizar a operação de output quando achar conveniente.

Recuperação e Atomicidade

- Na transação ao lado
  - Considere que o item de dados A fica no bloco B
- O banco ficará em um estado inconsistente se
  - For enviado um output(B) antes da falha
    - E esse bloco ficar disponível para outras transações
  - Nesse caso, a transação deixou de ser atômica

T1
read (A) / input (B)
A := A - 50
write (A) / output(B)
Falha
read (B)
B := B + 50
write (B)
Commit

Recuperação Baseada em Log

- Para garantir atomicidade em caso de falhas, usa-se estratégias de recuperação baseadas em log

- O log
  - é mantido em meio físico.
  - é uma sequência de registros
    - Que contém as operações de atualização do banco.

- Deve-se gravar informações no log
  - **Antes** de modificar o banco
  - Regra Write Ahead Log (WAL)

Recuperação Baseada em Log

- Quando a transação T_i inicia
  - É escrito o registro de log **<T_i start>**
- Quando T_i passa pela sua última instrução
  - É escrito o registro de log **<Ti commit>**.
- Caso T_i aborte
  - É escrito o registro de log **<Ti abort>**.
- ***Antes*** *que* T_i execute **write**(*X*)
  - é escrito um registro de log contendo a atualização (regra WAL)

Recuperação Baseada em Log

- Duas abordagens usam log
  - Modificação postergada
    - O banco é atualizado após o commit parcial
  - Modificação imediata
    - O banco pode ser atualizado após cada instrução de write

Modificação de Banco Postergada

- O esquema de modificação postergada
  - Registra todas operações no log
  - Mas posterga as escritas(outputs) para após o estado de commit parcial.

Modificação de Banco Postergada

- Uma operação de **write**(*X*) resulta na gravação do seguinte registro de log
  - ***<Ti, X, V>***
    - V é o novo valor de X
- A escrita (output) de X é postergada.
  - Só pode ser feita após o commit estar no log

Modificação de Banco Postergada

- Durante a recuperação após falha , a transação T_i precisa ser refeita (redo T_i) , caso
  - o registro <T_i start> estiver no log.
  - o registro <T_i commit> estiver no log.

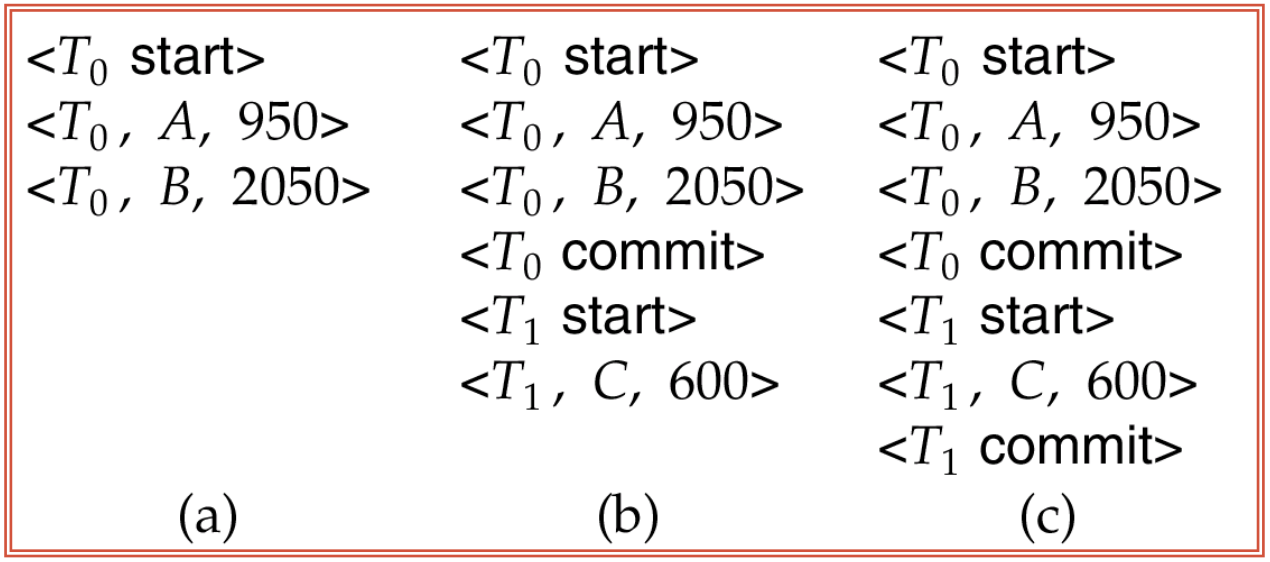
Modificação de Banco Postergada

- Ex: transações T_0 e T_1 (***T_0*** executa antes de ***T_1***):

T0	T1
read (A) **A := A - 50** **write *(A)*** **read *(B)*** ***B := B + 50*** **write *(B)*** ***commit***	**read(C)** **C := C - 100** **write(C)** **commit**

Modificação de Banco Postergada

- O exemplo abaixo mostra o progresso no tempo (a,b,c) de um arquivo de log:



- Se uma falha ocorrer no instante de tempo:
  - (a) nenhuma ação de redo é necessária

Modificação de Banco Postergada

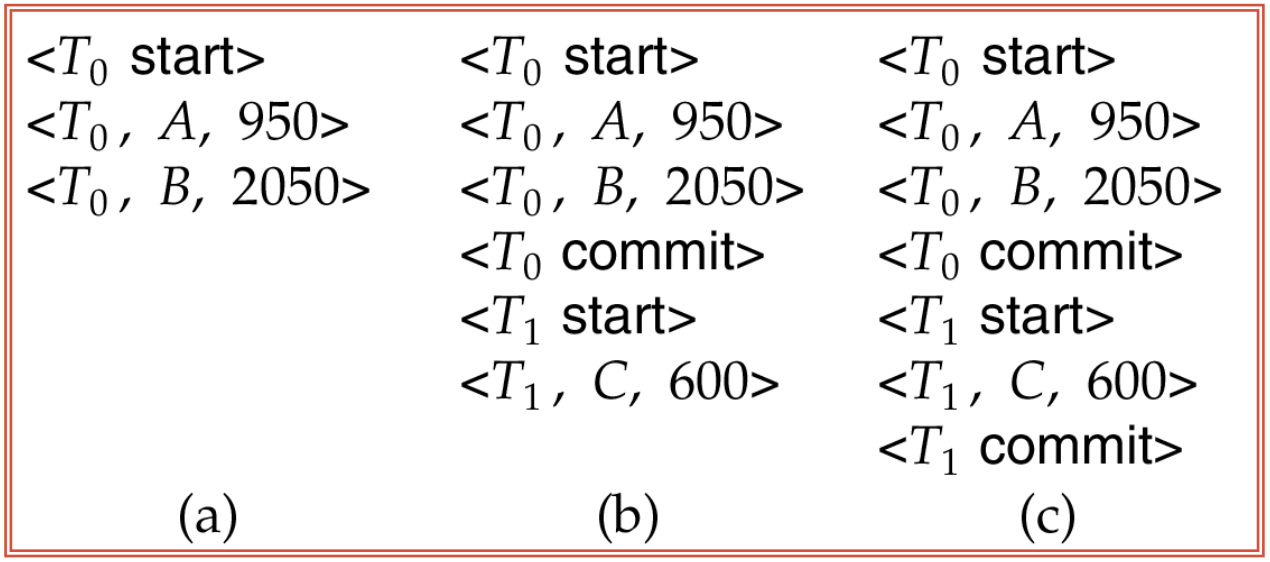
- O exemplo abaixo mostra o progresso no tempo (a,b,c) de um arquivo de log:

(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 uma falha ocorrer no instante de tempo:
  - (b) deve-se realizar **redo(T_0)**
    - já que **$<T_0$ commit>** está presente

Modificação de Banco Postergada

- O exemplo abaixo mostra o progresso no tempo (a,b,c) de um arquivo de log:



- Se uma falha ocorrer no instante de tempo:
  - (c) deve-se realizar **redo(T_0)** seguido de **redo(T_1)**
    - já que **<T_0 commit>** e **<T_i commit>** estão presentes

Modificação de Banco Imediata

- O esquema de modificação imediata permite realizar
  - Atualizações (outputs) de uma transação ainda não comitada
    - na medida que os writes acontecem

- Pode ser necessário desfazer os outputs(undo) se a transação abortar
  - Por isso, uma operação de **write**(*X*) resulta na gravação do seguinte registro de log
    - ***<Ti, X, V1, V2>***
      - V1 é o valor antigo de X
      - V2 é o valor novo de X

Modificação de Banco Imediata

Log	Write	Output
<T_0 **start**>		
<T_0, A, 1000, 950>		
	A = 950	
<T_0, B, 2000, 2050>		
	B = 2050	
<T_0 **commit**>		
<T_1 **start**>		
<T_1, C, 700, 600>		
	C = 600	
		B_B, B_C
<T_1 **commit**>		
		B_A

O output pode ocorrer a qualquer momento

- Nota: B_X denota bloco contendo X.

Modificação de Banco Imediata

- O procedimento de recuperação tem dois procedimentos em vez de um só:
  - undo(T_i)
    - Restaura o valor de todos itens atualizados por T_i aos seus valores antigos
    - voltando a partir do último registro no log de T_i
  - redo(T_i)
    - Seta o valor de todos os itens atualizados por T_i para os novos valores
    - indo adiante a partir do primeiro registro no log de T_i

Modificação de Banco Imediata

- Ao recuperar depois de uma falha:
  - A transação T_i precisa ser desfeita (undone) se
    - o log tiver o registro <T_i start>,
    - Mas não tiver o registro <T_i commit>.
  - A transação T_i precisa ser refeita (redone) se o log tiver:
    - o registro <T_i start>
    - e o registro <T_i commit>.
- Ordem de execução
  - Primeiro operações de undo
  - Seguidas pelas operações de redo

Recuperação Usando Modificação de Banco Imediata

- O exemplo abaixo mostra o progresso no tempo (a,b,c) de um arquivo de log:

<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>
(a)	(b)	(c)

- Se uma falha ocorrer no instante de tempo
  - (a) **undo (T_0)**:
    - B é restaurado para 2000 e A para 1000.

Recuperação Usando Modificação de Banco Imediata

- O exemplo abaixo mostra o progresso no tempo (a,b,c) de um arquivo de log:

(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>

- Se uma falha ocorrer no instante de tempo
  - (b) **undo (T1)** e **redo (T0):**
    - C é restaurado para 700, e
    - A e B recebem 950 e 2050 respectivamente.

Recuperação Usando Modificação de Banco Imediata

- O exemplo abaixo mostra o progresso no tempo (a,b,c) de um arquivo de log:

<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>
(a)	(b)	(c)

- Se uma falha ocorrer no instante de tempo
  - (c) **redo (T0)** e **redo (T1):**
    - A e B recebem 950 e 2050 respectivamente.
    - *C* recebe 600

Modificação Imediata vs postergada

- **Modificação imediata**

- Vantagem
  - Os outputs podem ser feitos em momentos oportunos, sem ter que esperar a transação acabar
    - Leva a otimização na gravação
  - O buffer pode ser melhor gerenciado, removendo blocos pouco acessados
- Desvantagens
  - ações de recuperação são mais custosas
    - Devido ao undo
  - Falhas do sistema levariam à reversão de muitas transações ativas

Modificação Imediata vs postergada

- **Modificação postergada**

- Desvantagem
  - Os outputs só ocorrem quando a transação termina
  - Transações que alteram muitos itens podem encher o buffer
    - Isso impediria a carga de novos blocos
    - E torne o output mais custoso

- Vantagens
  - A ação de recuperação é mais barata
    - Apenas redos são necessários
  - Além disso, ações de recuperações são menos frequentes
    - Necessárias apenas quando ocorre alguma falha depois do commit parcial

Atividade Individual

- Considere que as transações do próximo slide sejam executadas usando o Protocolo de lock de duas fases.

- Para efeito de simulação, as transações executam uma instrução de cada vez.

Atividade Individual

- Estado Inicial dos itens de dados
  - A = 0;
  - B = 100;
  - C = 200;

T1:	T2:	T3:
Read (B); Write (A=10);	Read (B); Read (C);	Write (C=40); Read (A); Write (A=20);

Atividade Individual – Parte 1

- Mostre o arquivo de log que seria gerado
  - usando modificação de banco postergada.

- Além disso, mostre as ações de recuperação que seriam executadas
  - caso uma falha ocorra após o write(A) da transação T3.

Atividade Individual – Parte 2

- Mostre o arquivo de log que seria gerado
  - usando modificação de banco imediata.

- Além disso, mostre as ações de recuperação que seriam executadas
  - caso uma falha ocorra após o write(A) da transação T3.