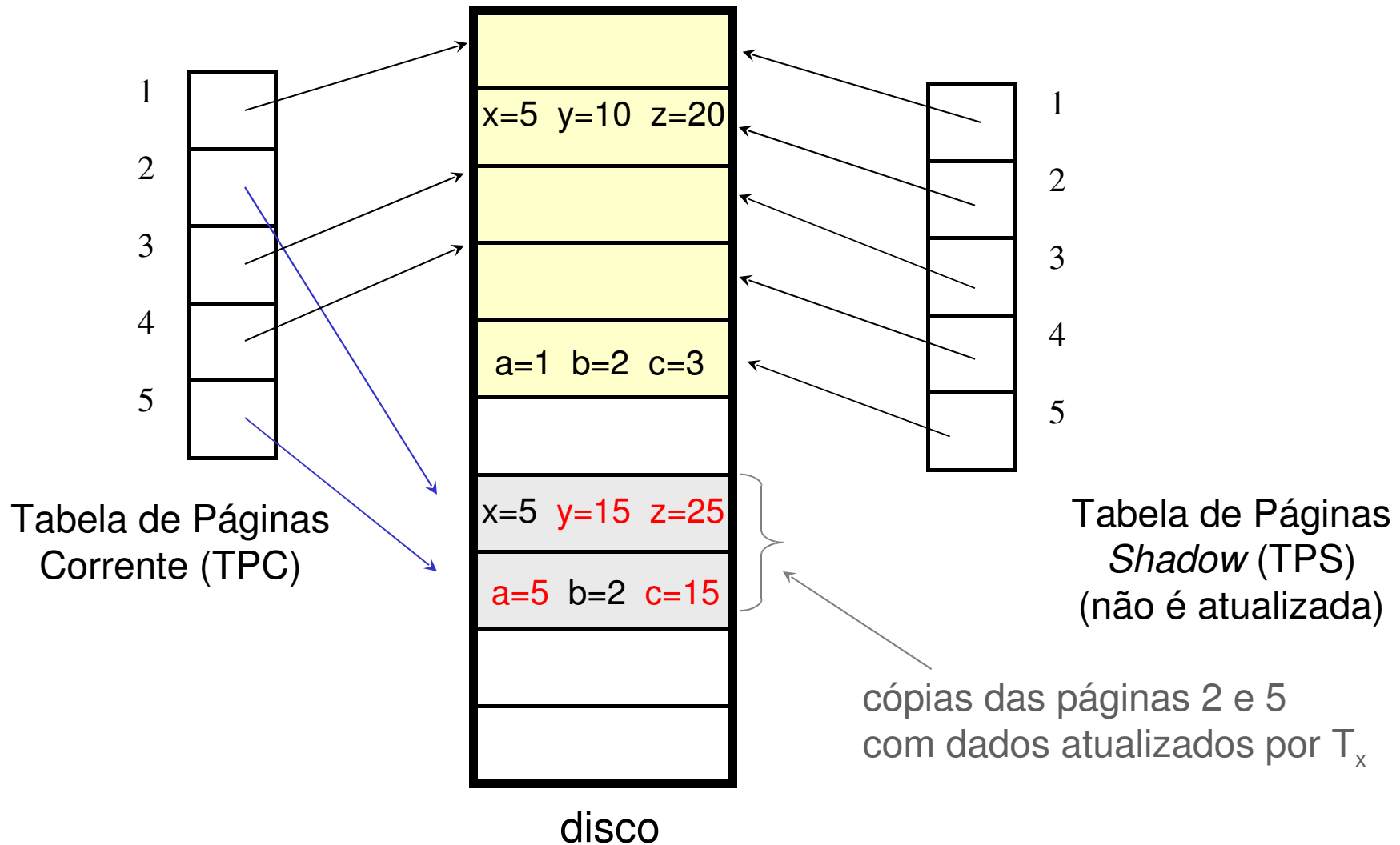


Técnica NO-UNDO/NO-REDO (técnica *Shadow Pages*)

- Supõe uma tabela de endereços de blocos (páginas) de disco que mantém dados do BD
 - Tabela de Páginas Corrente (TPC)
- A cada nova transação T_x
 - TPC é copiada para uma Tabela de Páginas *Shadow* (TPS)
 - páginas atualizadas por T_x são copiadas para novas páginas de disco e TPC é atualizada
 - TPS não é atualizada enquanto T_x está ativa
- Em caso de falha de T_x , TPC é descartada e TPS torna-se a TPC
 - não é preciso acessar o BD para fazer *recovery*

Técnica *Shadow Pages*



Shadow Pages - Procedimento

- 1) Quando uma transação T_x inicia
 - $TPS \leftarrow TPC$
 - FORCE TPS
- 2) Quando T_x atualiza dados de uma página P
 - se é a primeira atualização de T_x em P
 - se P não está na *cache* então busca P no disco
 - busca-se uma página livre P' na **Tabela de Páginas Livres (TPL)**
 - $P' \leftarrow P$ (grava P nessa página livre em disco)
 - apontador de P na TPC agora aponta para P'
 - atualiza-se os dados em P'

Shadow Pages – Procedimento (cont.)

3) Quando T_x solicita *commit*

- FORCE das páginas $P_1, \dots, P_n$ atualizadas por T_x que ainda não foram para disco
 - $P_1, \dots, P_n$ estão sendo gravadas em páginas diferentes no disco
- FORCE da TPC
 - endereço da TPC recebe o local onde TPC está persistida

• Vantagens

- Falha antes ou durante o passo 3
 - não é preciso realizar UNDO, pois TPS mantém as páginas do BD consistentes antes de T_x
 - basta fazer $TPC \leftarrow TPS$
- Falha após o passo 3
 - não é preciso realizar REDO, pois as atualizações de T_x estão garantidamente no BD

Técnica *Shadow Pages* - Desvantagens

- Adequada a SGBD monousuário
 - uma transação executando por vez
 - Pois cada transação controla a tabela de páginas inteira
 - SGBD multiusuário
 - gerenciamento complexo!
 - Não dá para controlar a TPC por completo (bloquear todos os blocos!)
 - Uma possível solução: Cada transação bloqueia inicialmente todas as páginas que vai atualizar, gerando páginas sombra delas
 - » overhead de I/O com persistência de tabelas e páginas atualizadas
 - Em caso de falha, descarta-se as páginas atualizadas e considera-se as páginas sombra como correntes
- Não mantém dados do BD *clusterizados*
- Requer coleta de lixo
 - quando T_x encerra, existem páginas obsoletas
 - páginas obsoletas devem ser incluídas na TPL

Escalonamentos *X Recovery*

- Considere o seguinte exemplo de escalonamento registrado em um *Log*:

LOG

```
<start T1>  
<write T1, A, 1, 2>  
<start T2>  
<write T2, A, 2, 5>  
<start T3>  
<write T1, B, 10, 2>  
<write T3, B, 2, 4>  
<commit T2>  
<commit T3>  
crash!
```

T1

```
r(A)  
A = A + 1  
w(A)  
r(B)  
B = B / A  
A = A + B  
w(A)
```

T2

```
r(A)  
A = A + 3  
w(A)
```

T3

```
r(B)  
B = B * B  
w(B)
```

Escalonamentos X *Recovery*

- Exemplo

LOG

```
<start T1>  
<write T1, A, 1, 2>  
<start T2>  
<write T2, A, 2, 5>  
<start T3>  
<write T1, B, 10, 2>  
<write T3, B, 2, 4>  
<commit T2>  
<commit T3>  
crash!
```

T1

```
r(A)  
A = A + 1  
w(A)  
r(B)  
B = B / A  
A = A + B  
w(A)
```

T2

```
r(A)  
A = A + 3  
w(A)
```

T3

```
r(B)  
B = B * B  
w(B)
```

Estados iniciais:

$A = 1$ $B = 10$

Deseja-se efetivar apenas os
Resultados de T2 e T3! Logo:

$A = 1 \Rightarrow A = 4$ $B = 10 \Rightarrow B = 100$

Escalonamentos X *Recovery*

- Exemplo

LOG

```
<start T1>  
<write T1, A, 1, 2>  
<start T2>  
<write T2, A, 2, 5>  
<start T3>  
<write T1, B, 10, 2>  
<write T3, B, 2, 4>  
<commit T2>  
<commit T3>  
crash!
```

T1

```
r(A)  
A = A + 1  
w(A)  
r(B)  
B = B / A  
A = A + B  
w(A)
```

T2

```
r(A)  
A = A + 3  
w(A)
```

T3

```
r(B)  
B = B * B  
w(B)
```

Estados iniciais:

$A = 1$ $B = 10$

Entretanto, após o *recovery*

UNDO/REDO, temos:

$A = 1 \Rightarrow A = 5$

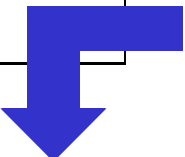
$B = 10 \Rightarrow B = 4$

Escalonamentos X *Recovery*

- Exemplo

LOG

```
<start T1>
<write T1, A, 1, 2>
<start T2>
<write T2, A, 2, 5>
<start T3>
<write T1, B, 10, 2>
<write T3, B, 2, 4>
<commit T2>
<commit T3>
crash!
```



T1

```
r(A)
A = A + 1
w(A)
r(B)
B = B / A
A = A + B
w(A)
```

T2

```
r(A)
A = A + 3
w(A)
```

T3

```
r(B)
B = B * B
w(B)
```

Estados iniciais:

$A = 1 \quad B = 10$

Entretanto, após o *recovery*

UNDO/REDO, temos:

$A = 1 \Rightarrow A = 5$

$B = 10 \Rightarrow B = 4$

Conclusão: nem todo escalonamento é recuperável!

Scheduler deve estar em sintonia com *recovery*, gerando escalonamentos válidos para recuperação de falhas!