# Técnica UNDO/REDO

- Quando se percorre o *Log forward* para fazer REDO, é possível que um dado X tenha sido atualizado por mais de uma transação *committed*
- Variante da técnica UNDO/REDO
  - detectar que X é atualizado mais de uma vez e realizar apenas a última atualização
    - técnica UNDO/REDO com REDO único para cada dado
  - estratégia
    - na varredura *backward* do *Log* para fazer UNDO, quando for encontrada a primeira atualização de um dado X por uma transação *committed*, inclui-se X e sua *afterImage* na lista-REDO-dados
      - novas atualizações de X feitas por transações *committed* que forem encontradas são ignoradas
    - após, varre-se a lista-REDO-dados, atualizando os dados

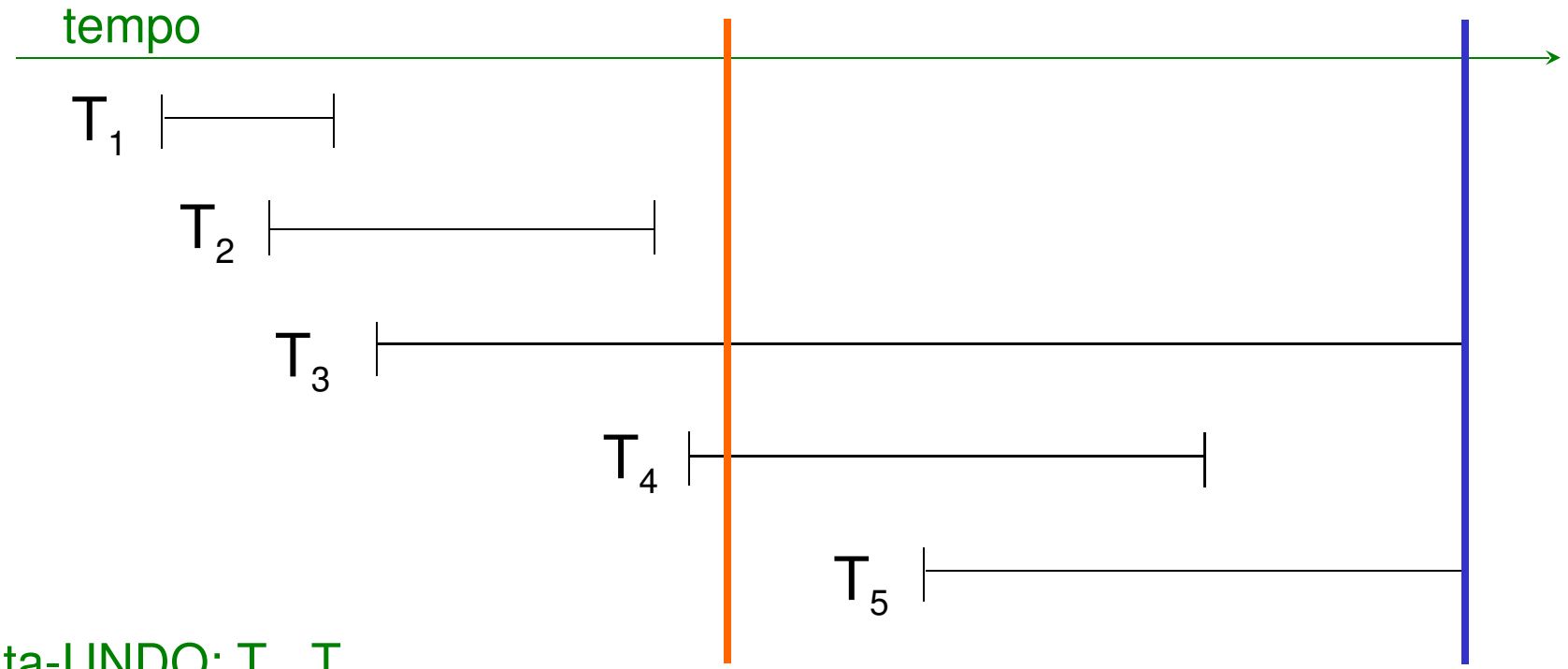
# *Checkpoint*

- SGBD com alta demanda de transações
  - *Log* de tamanho grande
    - *recovery* demorado
- *Checkpoint*
  - momento em que o SGBD grava no BD todas as atualizações feitas por transações
  - disparo manual ou automático
  - inclusão de um registro de *checkpoint* no *Log*
    - <checkpoint $T_1, T_2, \ldots, T_n$>

# *Checkpoint*

- Procedimento de execução de *checkpoint*
  1. suspensão de todas as transações
  2. descarga do *buffer* de *Log* em disco
     - FORCE do *Log*
  3. gravação dos blocos atualizados da *cache* no BD
  4. inserção de um registro *checkpoint* no *Log* e sua gravação em disco
  5. retomada da execução das transações
- Vantagem da técnica de *checkpoint*
  - transações *committed* antes do *checkpoint* não precisam sofrer REDO em caso de falha
    - elas já estão garantidamente no BD

# Técnica UNDO/REDO c/ *Checkpoint*




lista-UNDO: $T_3$, $T_5$

lista-REDO: $T_4$

- $T_1$ e $T_2$ concluíram e estão garantidamente no BD $\Rightarrow$ não sofrem REDO
- $T_4$ concluiu, mas suas atualizações não necessariamente estão no BD (supondo NOT-FORCE) $\Rightarrow$ sofre REDO
- $T_3$ e $T_5$ não concluíram $\Rightarrow$ sofrem UNDO

# Técnica UNDO/REDO c/ *Checkpoint*

- Procedimento de *Recovery* – **Passo 1**

  Percorre-se o *Log backward* até alcançar um registro *Checkpoint*

  – se achou <commit $T_x$>, insere $T_x$ na lista-REDO

  – se achou <start $T_x$> e $T_x$ não está na lista-REDO, insere $T_x$ na lista-UNDO

# Exemplo de *Recovery* com *Checkpoint*

```
<start T1>
<write T1,A,5,10>
<start T2>
<write T2,C,30,45>
<write T2,E,7,17>
<commit T2>
<write T1,C,45,55>
<start T3>
<write T3,B,15,20>
<commit T1>
<start T4>
<write T4,C,55,65>
<checkpoint T3,T4>
<start T5>
<write T5,D,31,39>
<start T6>
<write T3,A,5,25>
<write T6,F,1,2>
<write T3,E,17,28>
<commit T3>
<write T6,A,25,32>
<start T7>
<write T7,B,20,30>
<commit T7>
<write T4,E,28,34>
Crash!
```

**Log**

Lista UNDO:

Lista REDO:

Situação no BD após a falha:

A = 25
B = 20
C = 65
D = 39
E = 17
F = 2

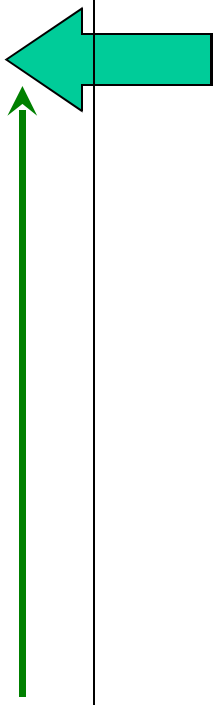
(não necessariamente todos os *writes* foram persistidos no BD)

Desejável:

A = 25
B = 30
C = 55
D = 31
E = 28
F = 1

# Após a execução do Passo 1

<start T1>
<write T1,A,5,10>
<start T2>
<write T2,C,30,45>
<write T2,E,7,17>
<commit T2>
<write T1,C,45,55>
<start T3>
<write T3,B,15,20>
<commit T1>
<start T4>
<write T4,C,55,65>
**<checkpoint T3,T4>**
**<start T5>**
**<write T5,D,31,39>**
**<start T6>**
**<write T3,A,5,25>**
**<write T6,F,1,2>**
**<write T3,E,17,28>**
**<commit T3>**
**<write T6,A,25,32>**
**<start T7>**
**<write T7,B,20,30>**
**<commit T7>**
**<write T4,E,28,34>**
***Crash!***



***Log***

Lista UNDO: **T6, T5**

Lista REDO: **T7, T3**

Situação no BD após a falha:
A = 25
B = 20
C = 65
D = 39
E = 17
F = 2

Desejável:
A = 25
B = 30
C = 55
D = 31
E = 28
F = 1

# Técnica UNDO/REDO c/ *Checkpoint*

- Procedimento de *Recovery* – **Passo 2**

Analisa-se cada transação $T_x$ no registro *checkpoint*

– se $T_x$ não estiver na lista-REDO, insere $T_x$ na lista-UNDO

<start T1>
<write T1,A,5,10>
<start T2>
<write T2,C,30,45>
<write T2,E,7,17>
<commit T2>
<write T1,C,45,55>
<start T3>
<write T3,B,15,20>
<commit T1>
<start T4>
<write T4,C,55,65>
**<checkpoint T3, T4>**
<start T5>
<write T5,D,31,39>
<start T6>
<write T3,A,5,25>
<write T6,F,1,2>
<write T3,E,17,28>
<commit T3>
<write T6,A,25,32>
<start T7>
<write T7,B,20,30>
<commit T7>
<write T4,E,28,34>
***Crash!***

***Log***

# Após a execução do Passo 2

Lista UNDO: T6, T5, **T4**

Lista REDO: T7, T3

→ **neste ponto já se conhece todas as transações que devem sofrer UNDO ou REDO !**

Situação no BD após a falha:

A = 25
B = 20
C = 65
D = 39
E = 17
F = 2

Desejável:

A = 25
B = 30
C = 55
D = 31
E = 28
F = 1

# Técnica UNDO/REDO c/ *Checkpoint*

- Procedimento de *Recovery* – **Passo 3**

  Percorre-se de novo o *Log backward*, até que todas as transações em lista-UNDO tenham sofrido UNDO

  – marca-se na lista-REDO as transações $T_x$ cujos registros <start $T_x$> estão sendo encontrados nessa varredura

<start T1>
<write T1,A,5,10>
<start T2>
<write T2,C,30,45>
<write T2,E,7,17>
<commit T2>
<write T1,C,45,55>
<start T3>
<write T3,B,15,20>
<commit T1>
<start T4>
<write T4,C,55,65>
<checkpoint T3,T4>
<start T5>
<write T5,D,31,39>
<start T6>
<write T3,A,5,25>
<write T6,F,1,2>
<write T3,E,17,28>
<commit T3>
<write T6,A,25,32>
<start T7>
<write T7,B,20,30>
<commit T7>
<write T4,E,28,34>
Crash!

Log

# Após a execução do Passo 3

Lista UNDO: ~~T6~~, ~~T5~~, ~~T4~~

Lista REDO: T7, T3 √

Situação no BD após a falha:

A = 25
B = 20
C = ~~65~~ 55
D = ~~39~~ 31
E = ~~17~~ 28
F = ~~2~~ 1

Desejável:

A = 25
B = 30
C = 55
D = 31
E = 28
F = 1

# Técnica UNDO/REDO c/ *Checkpoint*

- Procedimento de *Recovery* – **Passo 4**

  Caso existam transações não marcadas na lista-REDO ao final da varredura *backward*

  – continua-se a varredura *backward* até que todas as transações na lista-REDO tenham sido marcadas

<start T1>
<write T1,A,5,10>
<start T2>
<write T2,C,30,45>
<write T2,E,7,17>
<commit T2>
<write T1,C,45,55>
<start T3>
<write T3,B,15,20>
<commit T1>
<start T4>
<write T4,C,55,65>
<checkpoint T3,T4>
<start T5>
<write T5,D,31,39>
<start T6>
<write T3,A,5,25>
<write T6,F,1,2>
<write T3,E,17,28>
<commit T3>
<write T6,A,25,32>
<start T7>
<write T7,B,20,30>
<commit T7>
<write T4,E,28,34>
*Crash!*

*Log*

# Após a execução do Passo 4

Lista UNDO: ~~T6~~, ~~T5~~, ~~T4~~

Lista REDO: T7, T3

√ √

Situação no BD após a falha:

A = 25
B = 20
C = ~~65~~ 55
D = ~~39~~ 31
E = ~~17~~ 28
F = ~~2~~ 1

Desejável:

A = 25
B = 30
C = 55
D = 31
E = 28
F = 1

# Técnica UNDO/REDO c/ *Checkpoint*

- Procedimento de *Recovery* – **Passo 5**

  Percorre-se o *Log forward* do ponto de parada, realizado REDO das transações na lista-REDO

<start T1>
<write T1,A,5,10>
<start T2>
<write T2,C,30,45>
<write T2,E,7,17>
<commit T2>
<write T1,C,45,55>
<start T3>
<write T3,B,15,20>
<commit T1>
<start T4>
<write T4,C,55,65>
<checkpoint T3,T4>
<start T5>
<write T5,D,31,39>
<start T6>
<write T3,A,5,25>
<write T6,F,1,2>
<write T3,E,17,28>
<commit T3>
<write T6,A,25,32>
<start T7>
<write T7,B,20,30>
<commit T7>
<write T4,E,28,34>
Crash!

Log

# Após a execução do Passo 5

Lista UNDO: ~~T6~~, ~~T5~~, ~~T4~~

Lista REDO: ~~T7~~, ~~T3~~

Situação no BD após a falha:

A = 25 √

B = 20 √ ~~20~~ 30

C = ~~65~~ 55

D = ~~39~~ 31

E = ~~17~~ 28 √

F = ~~2~~ 1

Desejável:

A = 25

B = 30

C = 55

D = 31

E = 28

F = 1

Vantagem da técnica:

**Não é necessário varrer todo o *Log* para realizar o *Recovery*!**

# Exercício 3

Apresente um arquivo de *Log* (um arquivo para cada item abaixo) em que o uso de *checkpoint*:

a) mesmo assim requer uma varredura completa do *Log*;

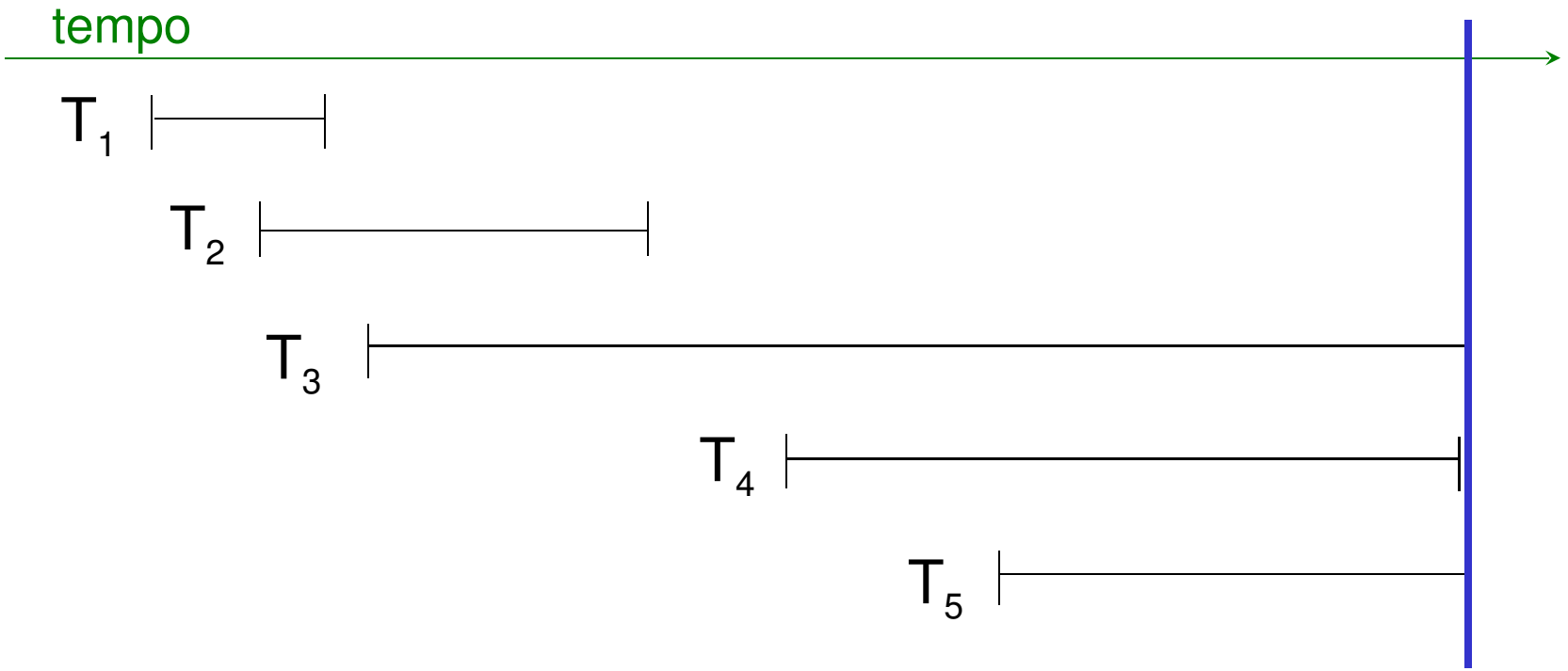
b) indica que nenhuma operação de UNDO e REDO precisa ser realizada;

c) não requer a realização de nenhuma operação de REDO

# Técnica UNDO/NO-REDO

- Outra técnica de modificação imediata do BD
- Grava o *commit* de $T_x$ no *Log* depois de todas as atualizações de $T_x$ terem sido gravadas no *Log*, e depois delas terem sido gravadas no BD
  - assim, se <commit $T_x$> está no *Log*, $T_x$ está garantidamente efetivada no BD
    - vantagem: não há necessidade de fazer REDO
    - desvantagem: pode-se fazer UNDO de uma transação que foi gravada com sucesso no BD, porém não foi gravado a tempo o seu *commit* no Log
- Requer um *Log* de UNDO
- Procedimento *default*
  - faz uma varredura *backward* do *Log*, realizando UNDO das transações na lista-UNDO (transações ativas)

# Técnica UNDO/NO-REDO - Exemplo




- $T_1$ e $T_2$ concluíram e tem *commit* no *Log* $\Rightarrow$ não sofrem REDO
- $T_4$ concluiu, mas não tem *commit* no *Log* $\Rightarrow$ sofre UNDO
- $T_3$ e $T_5$ não concluíram $\Rightarrow$ sofrem UNDO

# Modificação Postergada do BD

- Abordagem na qual dados atualizados por uma transação $T_x$ não podem ser gravados no BD antes do *commit* de $T_x$
- Gerenciamento de *buffer* mais complexo
  - utiliza técnica NOT-STEAL
    - blocos atualizados por $T_x$ não podem ser "roubados" enquanto $T_x$ não realizar *commit*
  - por outro lado, o *recovery* é mais simples
    - transações inacabadas não precisam sofrer UNDO
- Técnica
  - NO-UNDO/REDO

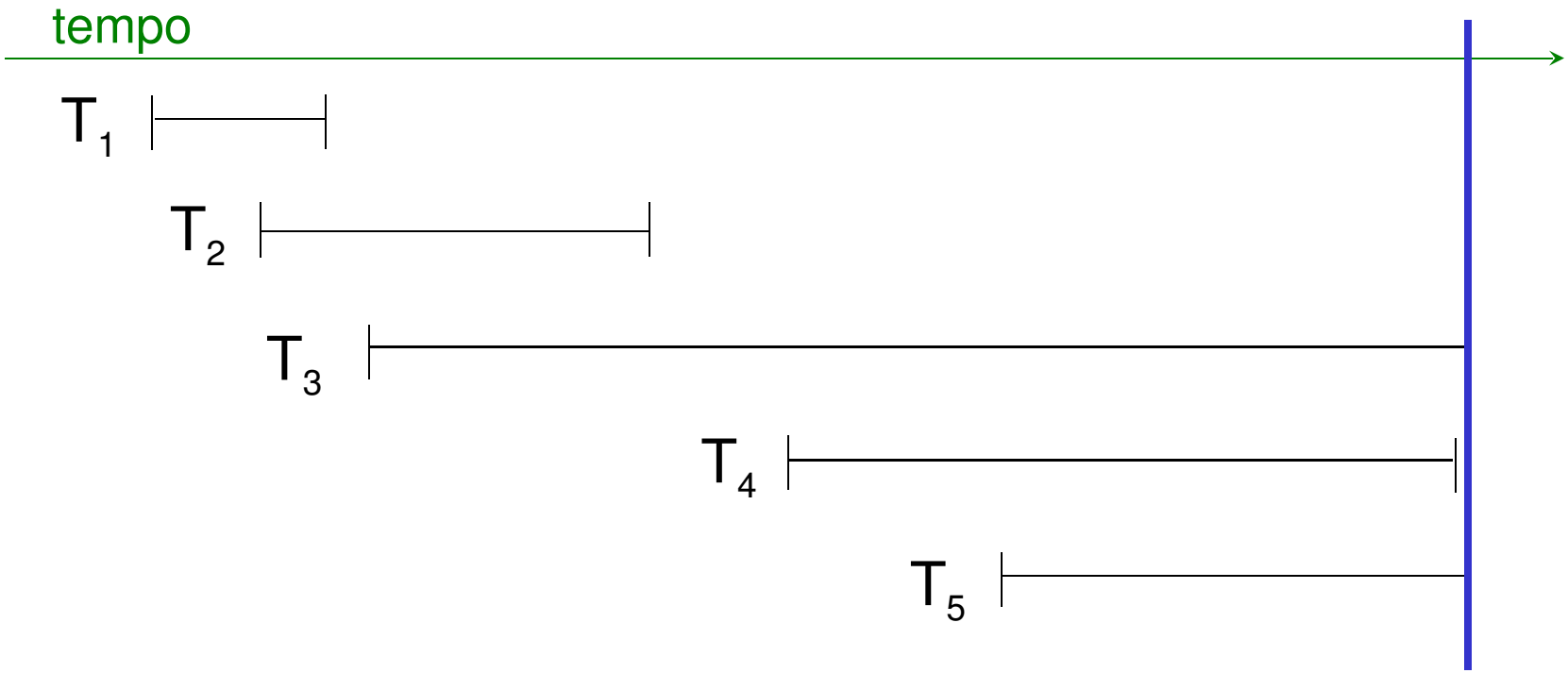
# Técnica NO-UNDO/REDO

- Quando $T_x$ conclui suas atualizações, força-se a gravação do *Log* em disco (com `<commit T`$_x$`>` )
  - FORCE no *Log*
- Vantagem
  - se $T_x$ falha antes de alcançar o *commit*, não é necessário realizar UNDO de $T_x$
    - nenhuma atualização de $T_x$ foi gravada no BD
  - requer apenas um *Log* de REDO
- Desvantagem
  - *overhead* no tempo de processamento (NOT-STEAL)
    - um bloco da *cache* pode permanecer em memória por muito tempo
      - dependente do *commit* de uma ou mais transações que atualizaram dados nele
      - se a *cache* fica cheia, é possível que algumas transações requisitando dados do BD tenham que esperar pela liberação de blocos

# Técnica NO-UNDO/REDO

- Procedimento *default* de *recovery*
  - faz uma varredura *forward* do *Log*, realizando REDO das transações na lista-REDO (transações *committed*)
- Transações ativas após o *recovery*
  - seus registros podem ser excluídos do *Log*
    - reduz o tamanho do *Log*
    - pode-se realizar também essa exclusão em técnicas que fazem UNDO de transações
      - após a conclusão do UNDO dessas transações
- A técnica NO-UNDO/REDO com REDO único para cada dado pode ser aplicada
  - exige varredura *backward* no *Log*
    - para definir inicialmente a lista-REDO-dados

# Técnica NO-UNDO/REDO - Exemplo



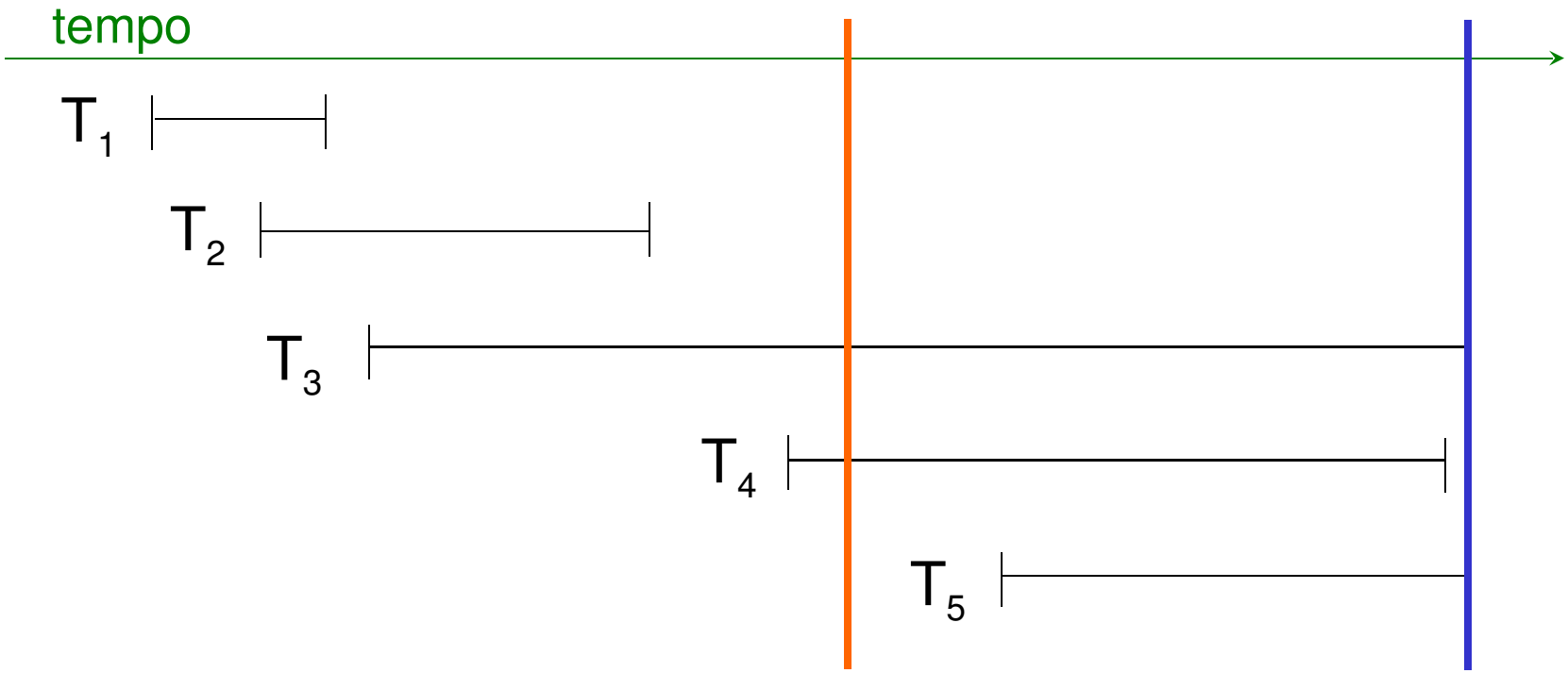

lista-REDO: $T_1$, $T_2$ e $T_4$

- $T_1$ e $T_2$ concluíram e atualizaram o BD $\Rightarrow$ sofrem REDO
- $T_4$ concluiu, mas não chegou a atualizar o BD $\Rightarrow$ sofre REDO
- $T_3$ e $T_5$ não concluíram e portanto não atualizaram o BD $\Rightarrow$ não sofrem UNDO

# NO-UNDO/REDO c/ *Checkpoint*

- No exemplo anterior, $T_1$ e $T_2$ não precisavam sofrer REDO...
  - técnica de *checkpoint* poderia ser utilizada para minimizar a quantidade de REDOs
- Técnica de *checkpoint* em uma abordagem de modificação postergada do BD
  - procedimento mais complexo
  - somente blocos de transações *committed* (na lista-REDO) devem ser descarregados no BD
    - e se não for possível descarregar todos esses blocos?
      - uma solução pode ser postergar a aplicação do *checkpoint* para um momento no qual todos os blocos de transações *committed* possam ser descarregados (não haja interferência de outras transações ativas)

# NO-UNDO/REDO c/ *Checkpoint*




lista-REDO: $T_4$

*checkpoint (supondo que T3 e T4 não atualizam os blocos atualizados por T1 e T2)*

falha (*crash*)

- $T_1$ e $T_2$ concluíram antes do *checkpoint* ⇒ não sofrem REDO
- $T_4$ concluiu depois do *checkpoint* ⇒ sofre REDO
- $T_3$ e $T_5$ não concluíram e portanto não atualizaram o BD ⇒ não sofrem UNDO

# Exercício 4

Suponha que o SGBD é monousuário, ou seja, uma nova transação só é executada após uma transação anterior ter concluído. Qual o impacto desta restrição sobre as 3 técnicas de *recovery* apresentadas anteriormente (UNDO/REDO, UNDO/NO-REDO e NO-UNDO/REDO), sem considerar *checkpoints*?

# Técnica ARCHIVE/DUMP/REDO

- Técnica baseada em *Log* para recuperação de falha de meio de armazenamento
- Operação ARCHIVE
  - ocorre durante o funcionamento normal do SGBD
  - gravação de uma ou mais cópias *backup* do BD em dispositivos diferentes de memória secundária
  - disparo manual ou automático (periódico)
  - deve-se suspender o início de novas transações
  - nenhuma transação pode estar ativa
    - se existem transações nesse estado, deve-se aguardar até elas encerrarem com sucesso
  - o *Log* corrente é "descartado" (excluído ou mantido associado ao *backup* anterior do BD) e um novo *Log* ("zerado") é iniciado

# Técnica ARCHIVE/DUMP/REDO

- Operações DUMP + REDO
  - realizam o *recovery* de uma falha no BD
  - procedimento
    - restaura o BD a partir do último *backup* (DUMP)
    - realiza uma varredura *forward* do *Log*, realizando REDO das transações *committed*
    - as transações ativas no momento da falha podem ser re-submetidas à execução pelo SGBD
      - já que não houve perda de dados na memória principal
- Técnicas baseadas em *Log* requerem um *Log* seguro
  - *archive* do *Log* também deve ser realizado
    - com freqüência igual ou superior ao *archive* do BD

# ARCHIVE/DUMP/REDO - Exemplo