

Recovery

Recuperação de Falhas

- Garantia de **atomicidade** e **durabilidade** de Transações
 - requer um SGBD **tolerante a falhas**
- Tolerância a falhas em BDs
 - capacidade de conduzir o BD a um **estado passado consistente**, após a ocorrência de uma falha que o deixou em um estado inconsistente
 - baseia-se em **redundância** de dados
 - não é um mecanismo 100% seguro
 - responsabilidade do **subsistema de recovery** do SGBD

Subsistema de *Recovery*

- Controles
 - durante o funcionamento normal do SGBD
 - manter informações sobre o que foi atualizado no BD pelas transações
 - realizar cópias periódicas do BD
 - após a ocorrência de uma falha
 - executar ações para retornar o BD a um estado consistente
 - ações básicas
 - UNDO: desfazer uma atualização no BD
 - REDO: refazer uma atualização no BD
- Considerações sobre o seu projeto
 - tipos de falhas a tratar
 - técnica de *recovery* a aplicar

Ações Básicas de *Recovery*

- Transaction UNDO
 - uma transação não concluiu suas operações
 - as modificações realizadas por esta transação no BD são desfeitas
- Global UNDO
 - uma ou mais transações não concluíram as suas operações
 - as modificações realizadas por todas estas transações no BD são desfeitas
- Partial REDO
 - na ocorrência de uma falha, algumas transações podem ter concluído suas operações (*committed*), mas suas ações podem não ter se refletido no BD
 - as modificações realizadas por estas transações são refeitas no BD
- Global REDO
 - no caso de um comprometimento do BD, todas as transações *committed* no BD são perdidas
 - as modificações realizadas por todas estas transações no BD são refeitas

Tipos de Falhas

- Falha de Transação

- uma transação ativa termina de forma anormal
- causas
 - violação de RI, lógica da transação mal definida, *deadlock*, cancelamento pelo usuário, ...
- não compromete a memória principal e a memória secundária (disco, em geral)
- falha com maior probabilidade de ocorrência
- seu tempo de recuperação é pequeno
 - ação: Transaction UNDO

Tipos de Falhas

- Falha de sistema

- o SGBD encerra a sua execução de forma anormal
- causas
 - interrupção de energia, falha no SO, erro interno no SW do SGBD, falha de HW, ...
- compromete a memória principal e não compromete o disco
- falha com probabilidade média de ocorrência
- seu tempo de recuperação é médio
 - ações: Global UNDO e Partial REDO

Tipos de Falhas

- Falha de meio de armazenamento
 - o BD torna-se total ou parcialmente inacessível
 - causas
 - setores corrompidos no disco, falha no cabeçote de leitura/gravação, ...
 - não compromete a memória principal e compromete o disco
 - falha com menor probabilidade de ocorrência
 - seu tempo de recuperação é grande
 - ação: **Global REDO**

Técnicas de Recovery

- Baseadas em *Log*

- modificação imediata do BD

- técnica UNDO/REDO
 - técnica UNDO/NO-REDO

- modificação postergada do BD

- técnica NO-UNDO/REDO

- recuperação de meio de armazenamento

- técnica ARCHIVE/DUMP/REDO

recuperação de
falhas de transação
e de sistema

- Baseadas em *Shadow Pages*

- técnica NO-UNDO/NO-REDO

recuperação de
falhas de transação
e de sistema

Técnicas Baseadas em Log

- Técnicas mais comuns de *recovery*
- Utilizam um *arquivo de Log* (ou *Journal*)
 - registra seqüencialmente as atualizações feitas por transações no BD
 - é consultado em caso de falhas para a realização de UNDO e/ou REDO de transações
 - mantido em uma ou mais cópias em memória secundária (disco, fita, ...)
 - tipos de *log*
 - *log de UNDO*
 - mantém apenas o valor antigo do dado (*before image*)
 - *log de REDO*
 - mantém apenas o valor atualizado do dado (*after image*)
 - *log de UNDO/REDO* (mais comum)
 - mantém os valores antigo e atualizado do dado

Tipos de Registro no Log

- Supõe-se que toda transação possui um identificador único gerado pelo SGBD
- Para fins de recuperação de falhas, operações *read* não precisam ser gravadas
 - úteis apenas para outros fins (auditoria, estatísticas, ...)
- Principais tipos de registro
 - início de transação: `<start Tx>`
 - *commit* de transação: `<commit Tx>`
 - atualização: `<write Tx, X, beforeImage, afterImage>`
 - não é necessário em log REDO ←
 - não é necessário em log UNDO ←

Exemplo de Log

Log

```
<start T3>
<write T3, B, 15, 12>
<start T2>
<write T2, B, 12, 18>
<start T1>
<write T1, D, 20, 25>
<commit T1>
<write T2, D, 25, 26>
---e se ouver falha?
<write T3, A, 10, 19>
<commit T3>
<commit T2>
```

T₁

```
read(A)
read(D)
write(D)
```

T₂

```
read(B)
write(B)
read(D)
write(D)
```

T₃

```
read(C)
write(B)
read(A)
write(A)
```

...

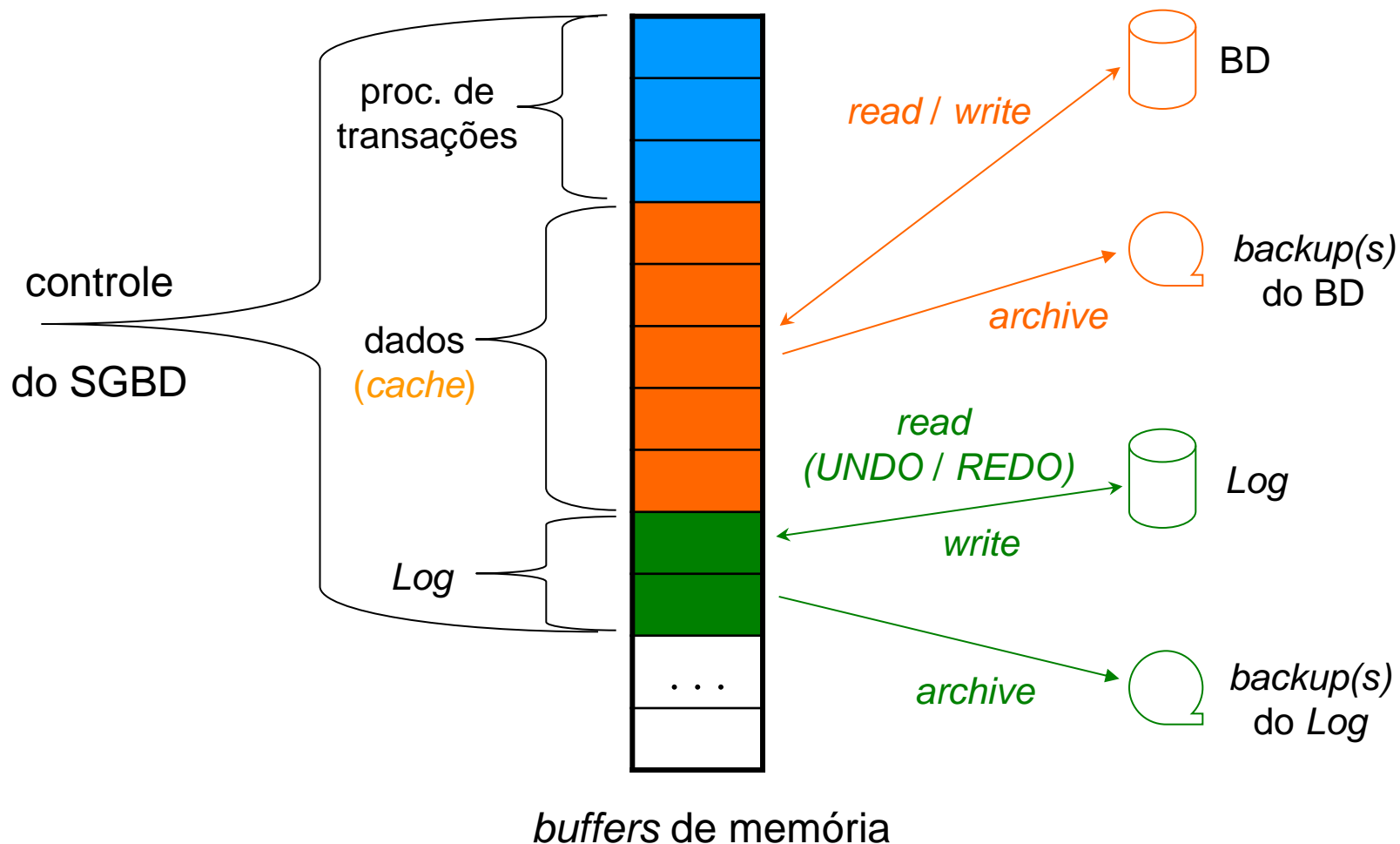
Tipos de Registro no Log

- Forma alternativa de representar atualizações
 - considera a operação DML feita no BD
 - **insert**: `<write Tx, X, INSERT, afterImage>`
 - **update**: `<write Tx, X, UPDATE, beforeImage, afterImage>`
 - **delete**: `<write Tx, X, DELETE, beforeImage>`
- A indicação do tipo de operação facilita o entendimento do que deve ser UNDO ou REDO no BD

Gerenciamento de Buffer

- *Buffer*
 - conjunto de blocos da memória principal
 - considera-se **bloco** e **página** conceitos sinônimos
- O SGBD é responsável pela gerência de alguns *buffers*
 - *buffers* para dados, para processamento de transações e para o *Log*
 - ele assume o controle desses *buffers*, ao invés do SO, requisitando apenas serviços de leitura/escrita de blocos ao SO

Gerenciamento de Buffer



Gerenciamento de Buffer

- Técnicas de *recovery* devem sincronizar os *buffers* de *log* e de dados
 - princípio básico
 - um bloco atualizado na *cache* só pode ser gravado no BD após o histórico dos dados atualizados neste bloco ter sido gravado no *Log* em disco
 - *Write-Ahead-Log (WAL)*
 - uma transação T_x só pode passar para o estado *efetivada (committed)* após todas as suas atualizações terem sido gravadas no BD segundo o princípio *WAL*
- O SGBD aplica técnicas de gerenciamento de *buffer*
 - estas técnicas influenciam as técnicas de *recovery*

Técnicas de Gerência de Buffer

- *NOT-STEAL*

- um bloco na *cache* utilizado por uma transação T_x **não pode** ser gravado antes do *commit* de T_x
 - bloco possui um *bit de status* indicando se foi (1) ou não (0) modificado
 - **vantagem**: processo de *recovery* mais simples - evita dados de transações inacabadas sendo gravadas no BD

- *STEAL*

- um bloco na *cache (dados)* utilizado por uma transação T_x **pode** ser gravado antes do *commit* de T_x
 - necessário se algum dado é requisitado do BD por outra transação e não há blocos disponíveis na *cache*
 - o bloco “vítima” é escolhido através de alguma técnica de SO
 - LRU, FIFO, ...
 - **vantagem**: não há necessidade de manter blocos bloqueados por transações

Técnicas de Gerência de Buffer

- *FORCE*

- os blocos que mantêm dados atualizados por uma transação T_x *são* imediatamente gravados no BD quando T_x alcança o *commit*
 - deve-se saber quais os blocos que T_x *atualizou dados*
- *vantagem*: garante a durabilidade de T_x o mais cedo possível - permite o REDO de T_x em caso de falha

- *NOT-FORCE*

- os blocos que mantêm dados atualizados por T_x *não são* imediatamente gravados no BD quando T_x alcança o *commit*
- *vantagem*: blocos atualizados podem permanecer na *cache* e serem utilizados por outras transações, após o *commit* de T_x (reduz custo de acesso a disco)

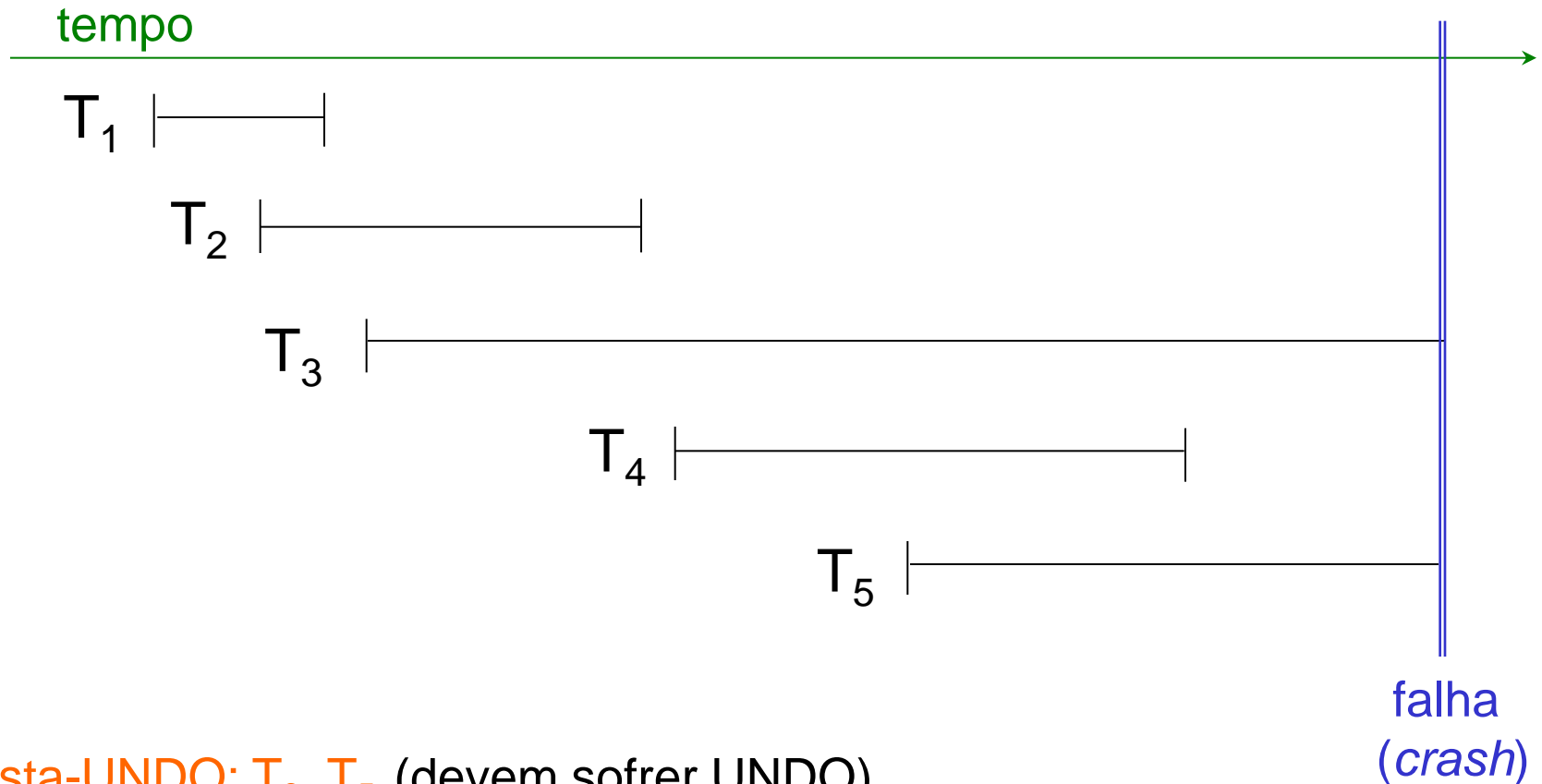
Modificação Imediata do BD

- Abordagem na qual dados atualizados por uma transação T_x podem ser gravados no BD antes do *commit* de T_x
- Abordagem mais comum de *recovery*
 - gerenciamento de *buffer* mais simples
 - utiliza técnica STEAL
- Duas técnicas
 - UNDO/REDO
 - técnica mais comum de *recovery*
 - UNDO/NO-REDO

Técnica UNDO/REDO

- Grava o *commit* de T_x no Log *depois de* todas as atualizações de T_x terem sido gravadas no Log, e antes dessas atualizações serem gravadas no BD
 - requer um Log de UNDO/REDO
- Utiliza 2 listas de transações
 - lista-REDO: IDs de transações *committed*
 - possuem *commit* gravado no Log
 - lista-UNDO: IDs de transações ativas
- Procedimento
 1. faz uma varredura *backward* do Log, realizando UNDO das transações na lista-UNDO
 2. faz uma varredura *forward* do Log, realizando REDO das transações na lista-REDO

Técnica UNDO/REDO - Exemplo



Técnica UNDO/REDO

- A propriedade de **idempotência** de operações UNDO e REDO é válida
 - fazer UNDO ou REDO uma vez ou várias vezes produz o mesmo resultado
 - situações em que ocorrem falhas durante o processo de *recovery*
- Técnica mais trabalhosa de *recovery*
 - tanto UNDO quanto REDO devem ser realizados
 - porém, o gerenciamento de *buffer* é mais simples

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

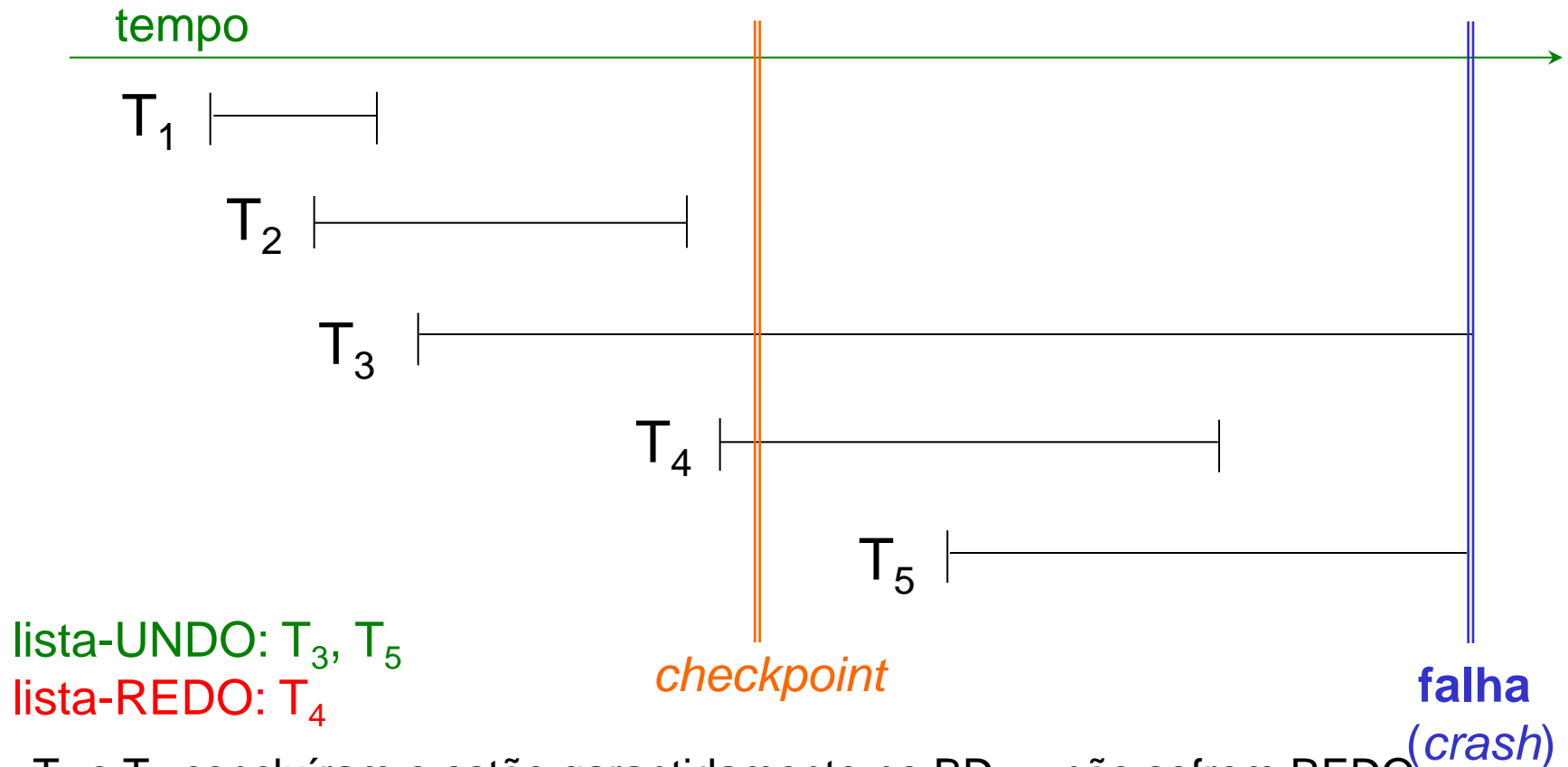
Checkpoint

- 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*
 - $\langle \text{checkpoint } T_1, T_2, \dots, T_n \rangle$
 - lista de transações ativas

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
 1. **FORCE** dos dados
 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



- 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*
 1. percorre-se o *Log backward* até alcançar um registro *Checkpoint*
 - se achou `<commit Tx>`, insere T_x na **lista-REDO**
 - se achou `<start Tx>` e T_x não está na lista-REDO, insere T_x na **lista-UNDO**
 2. analisa-se cada transação T_x no registro *checkpoint*
 - se T_x não estiver na lista-REDO, insere T_x na **lista-UNDO**

Técnica UNDO/REDO c/ Checkpoint

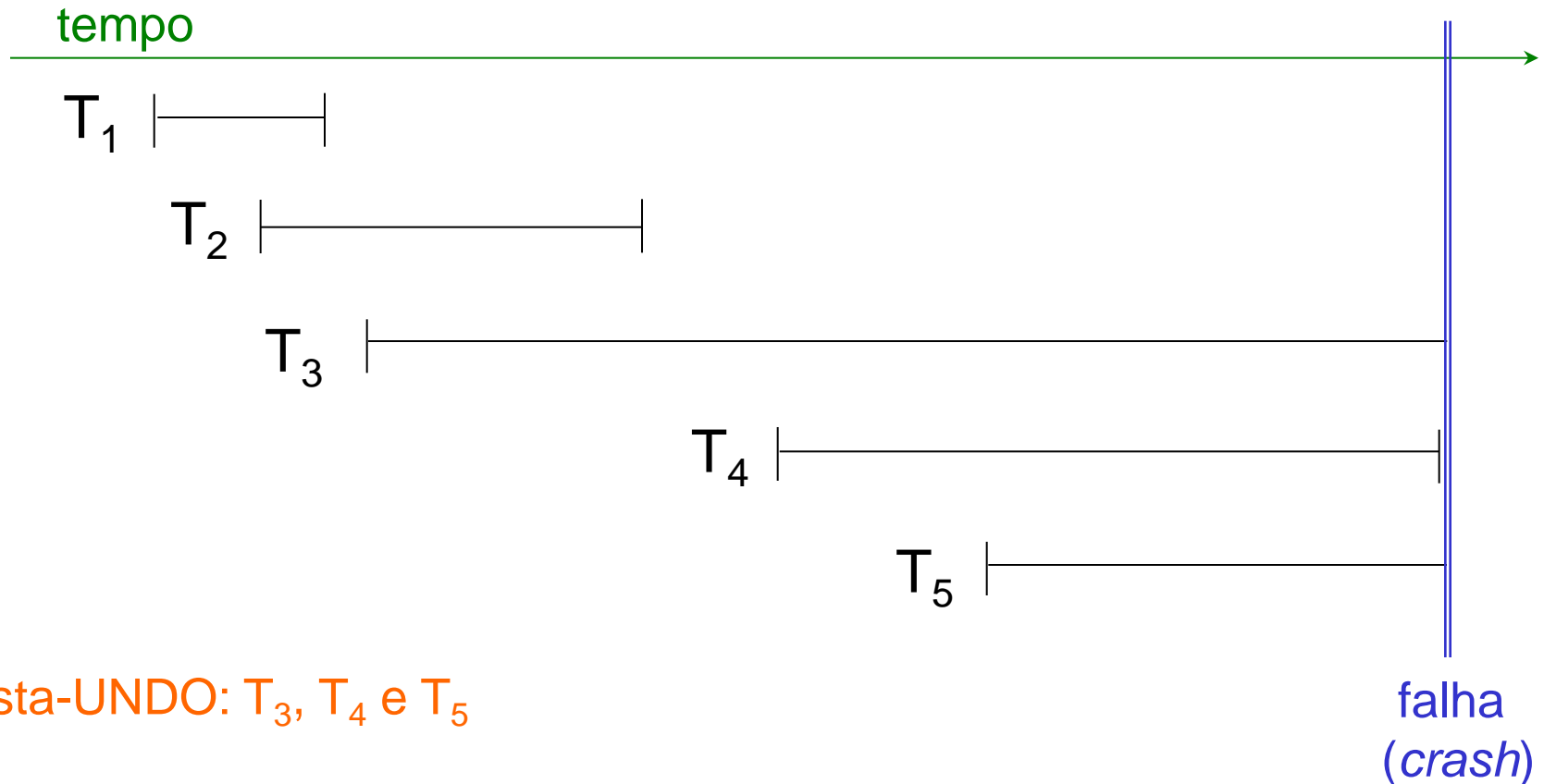
- Procedimento de *recovery* (cont.)
 3. percorre-se de novo o *Log backward*, até que todas as transações em lista-UNDO tenham sofrido UNDO
 - marca-se na lista-REDO as transações T_x cujos registros $\langle \text{start } T_x \rangle$ estão sendo encontrados nessa varredura
 4. se existem transações 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
 5. percorre-se o *Log forward* do ponto de parada, realizado REDO das transações na lista-REDO
- Vantagem
 - não é necessário varrer sempre todo o *Log*

Continua.....

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 $\langle \text{commit } T_x \rangle$ 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
 - 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 não precisam sofrer UNDO
- Técnica
 - **NO-UNDO/REDO**

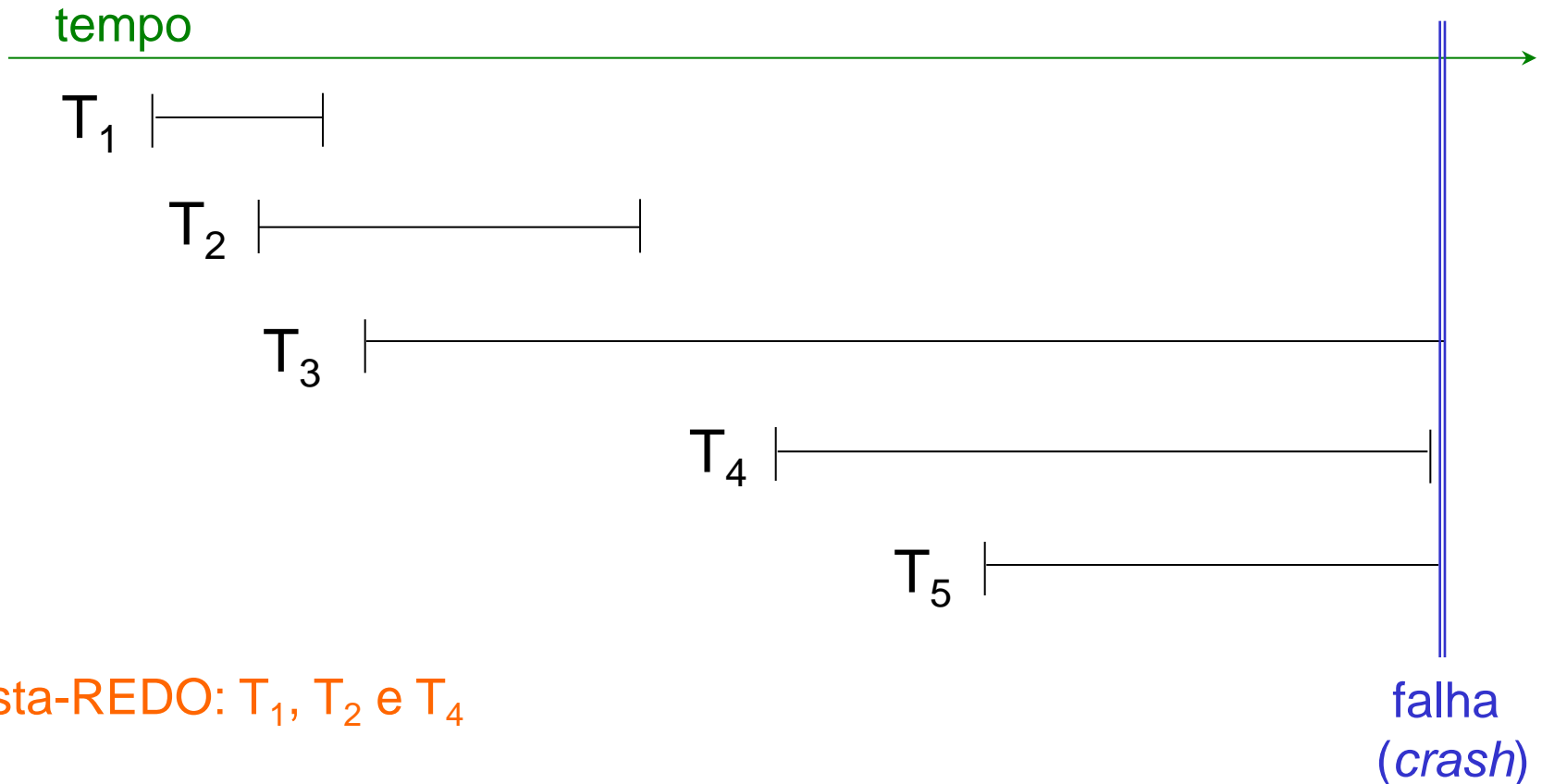
Técnica NO-UNDO/REDO

- 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 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



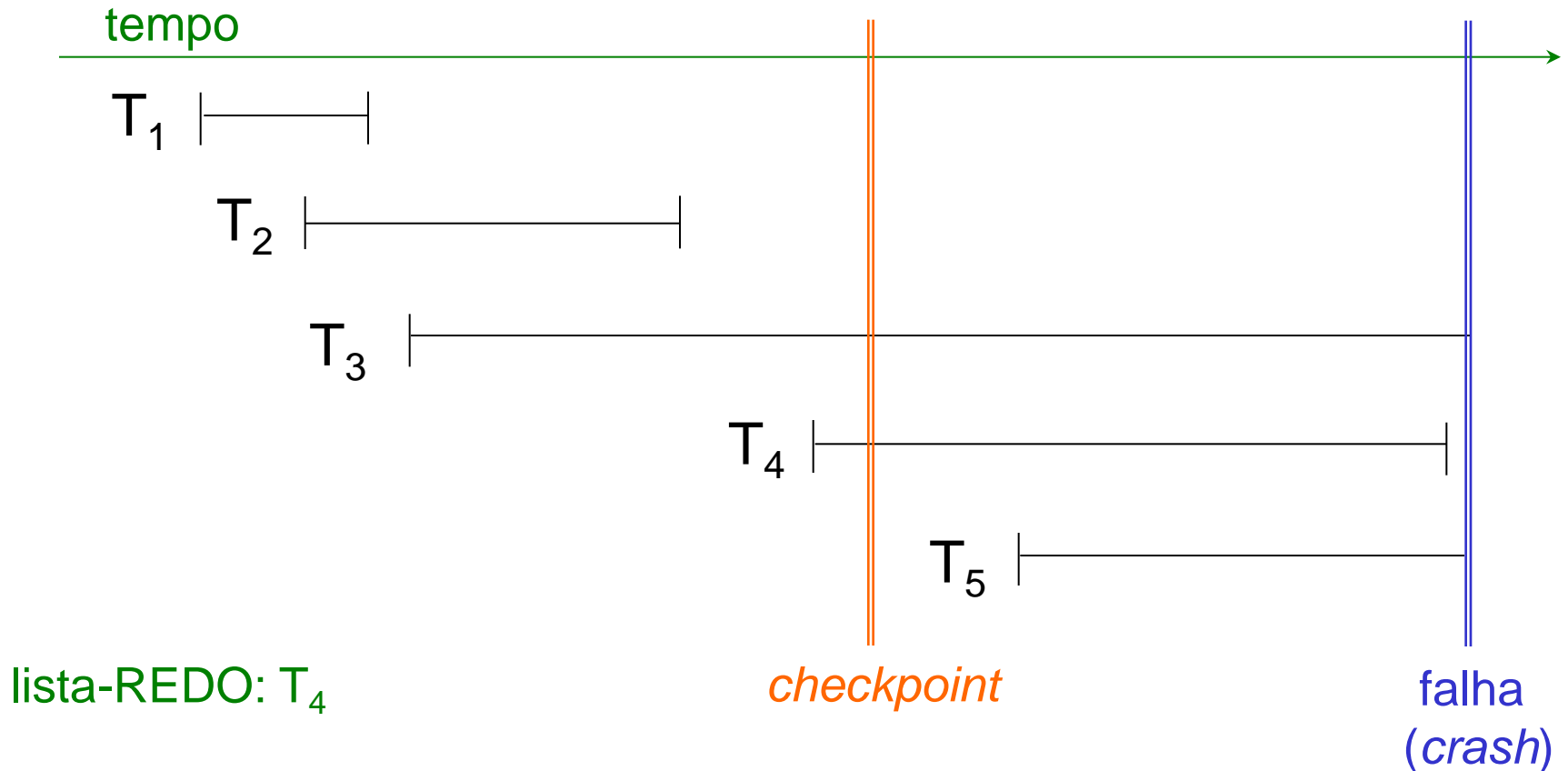
lista-REDO: T_1 , T_2 e T_4

- 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



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

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 frequência igual ou superior ao *archive* do BD

ARCHIVE/DUMP/REDO - Exemplo

