Recuperação de falhas

- Garante a Atomicidade e Durabilidade
  - 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 estado inconsistente
  - Baseia-se em redundância de dados
  - Não é um mecanismo 100% seguro
  - Responsabilidade do subsistema de *recovery* do SGBD

Subsistema de *Recovery*

- Algoritmo para assegurar a consistência do BD
  - 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:
        - UNDO: desfazer uma atualização no BD
        - REDO: refazer uma atualização no BD

Subsistema de *Recovery*

Tipos de Falhas

- Previsíveis (Tratáveis e não tratáveis)
- Imprevisíveis
- Classificação das Falhas
  - **SEM** perda de informação – tratamento fácil
  - **COM** perda de informação - tratamento difícil

Sistema de Recuperação

- 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 secundária
    - Falha com maior probabilidade de ocorrência
    - Tempo de recuperação :pequeno

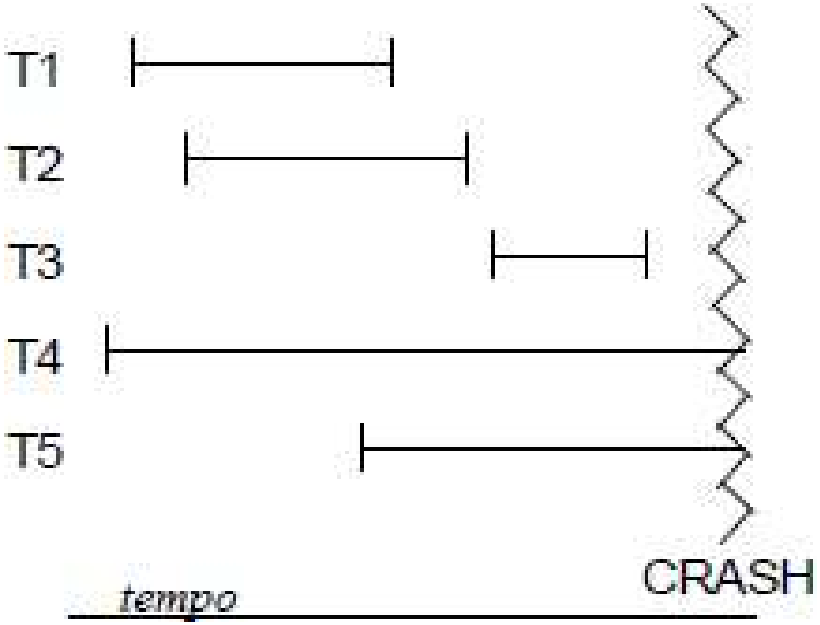
Sistema de Recuperação

- 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 o disco
    - Falha com probabilidade média de ocorrência
    - Tempo de recuperação: médio

Sistema de Recuperação

- 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
    - Tempo de recuperação: grande

Sistema de Recuperação



T1, T2, T3 terminaram com sucesso antes do CRASH: seus efeitos tem que permanecer no BD

T4, T5 têm que ser desfeitas

Sistema de Recuperação

- AÇÕES:
  - UNDO da transação (Transaction UNDO) :
    - Uma transação é abortada em função de falha
    - **Desfaz** os efeitos desta transação
  - UNDO Global (Global UNDO):
    - **Desfaz** os efeitos de todas as transações ativas antes da falha

Sistema de Recuperação

- AÇÕES:
  - REDO parcial (partial REDO) :
    - Transações terminaram com sucesso, mas suas ações podem não ter refletido no BD
    - **Refaz** os efeitos de um conjunto de transações
    -
  - REDO Global (Global REDO):
    - Todas as transações commited são perdidas
    - **Refaz** os efeitos de todas as transações commited

Técnicas de Recovery

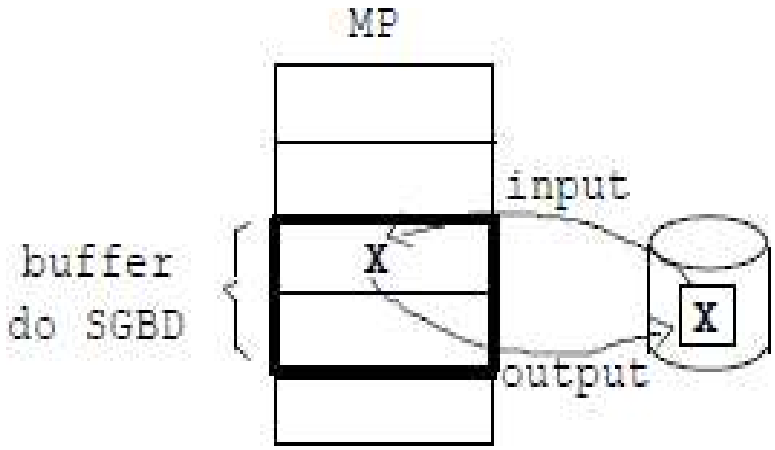
- Baseadas em Log
  - Modificação Adiada
  - Modificação Imediata
  - Recuperação de meio de armazenamento
- Baseadas em Shadow Pages

Armazenamento SGBD

- O BD é dividido em blocos (unidades de armazenamento)
- Blocos são feitas as transferências de dados usados pelas transações entre disco e mem.principal
- Blocos Físicos: residentes em disco
- Blocos Lógicos: residentes em mem. Principal
- Transações trabalham com dados na Mem. Principal (MP)

Armazenamento SGBD

- Movimentos de blocos:
  - INPUT (X) : transfere bloco onde está o dado x do disco para a MP
  - OUTPUX(X): transfere bloco onde está o dado x da MP para o disco



Armazenamento SGBD

- Operações:
  - READ(X, xi): variável xi recebe o conteúdo do dado x
    - Se bloco não esta na MP, então INPUT(X)
    - No buffer: xi=X
  - WRITE (X, xi): dado X recebe o conteúdo da variável xi
    - Se bloco não está na MP, então INPUT(x)
    - No buffer: X=xi
  - OUTPUT(x) : reflete no disco as alterações feitas em X pelas transações. Não precisa ser executado logo após o write (buffer).

Recuperação baseadas em log

Exemplo:

START

READ(X) → *INPUT(X), READ(X,xi)*

WRITE(X) → *WRITE(X,xi) (dado na MP)*

CRASH

OUTPUT(X)

Dado não foi para o BD

Recuperação baseadas em log

- **LOG** : estrutura (arquivo ou conjunto de arquivos) utilizada para registrar modificações no BD
- Técnica mais comum
- Grava-se as modificações primeiro no log sem modificar o BD
- Atualiza o BD

Recuperação baseadas em log

- Cada registro no log descreve uma gravação no BD e possui:
  1) Identificador da transação (executou o write)
  2) Nome do item de dado
  3) Valor antigo do dado: *Before Image (antes do write)*
  4) Valor novo do dado: *After Image (depois write)*
  5) Fim da transação

Recuperação baseadas em log

- Exemplo:

<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>
<write T3,A,10,19>
<commit T3>
<commit T2>
...

Recuperação baseadas em log Modificação Adiada

- Adia as operações de atualização até o estado parcialmente executado.
- Só armazena no BD depois do commit
- RECUPERAÇÃO:
  - Operação REDO (Ti) – ajusta os itens de dados com os novos valores.
  - REGISTRO DE LOG:
    - Início da transação;
    - Valor Novo;
    - Fim da Transação.

Recuperação baseadas em log Modificação Adiada

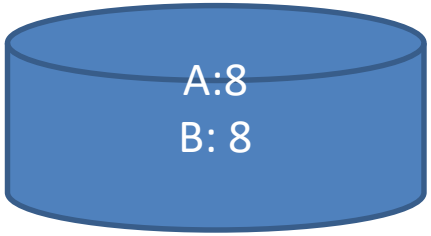
- RECUPERAÇÃO:
  - Antes do Commit:
    - A transação é ignorada;
    - Os valores no banco de dados são resgatados;
    - A transação é refeita.
  - Depois do Commit:
    - O banco de dados é atualizado com os valores do registro de log (**REDO**)

Modificação Adiada

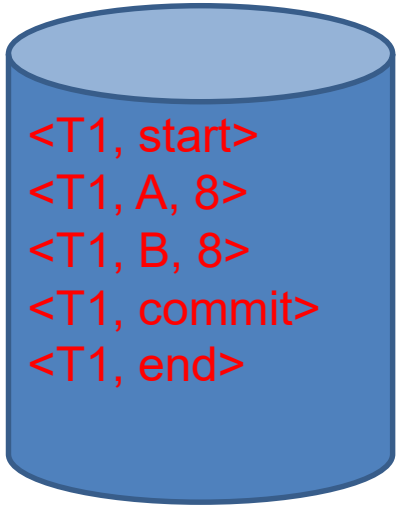
T1:
Read (A,t);
Write (A,t); t -> t* 2
Read (B,t);
Write (B,t); t -> t* 2

A=B

memória



disco



Recuperação baseadas em log Modificação Imediata

- O Banco de Dados é atualizado quanto a transação ainda está em um estado ativo: (a cada operação grava no BD)
- RECUPERAÇÃO:
  - Operação Undo (Ti) – ajusta os itens de dados com os valores antigos;
  - Operação Redo (Ti) – ajusta os itens de dados com os valores novos.

Recuperação baseadas em log Modificação Imediata

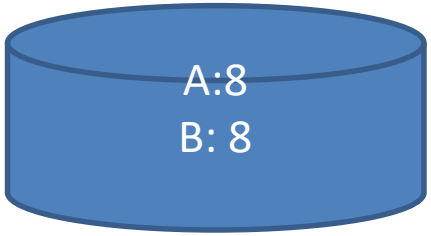
- RECUPERAÇÃO:
  - REGISTRO DE LOG:
    - Início da transação;
    - Valor antigo;
    - Valor Novo;
    - Fim da Transação.
  - Transações Desfeitas:
    - transações que possuem Start (Undo)
  - Transações Refeitas:
    - Transações que possuem Start e Commit. (Redo)

Modificação Imediata

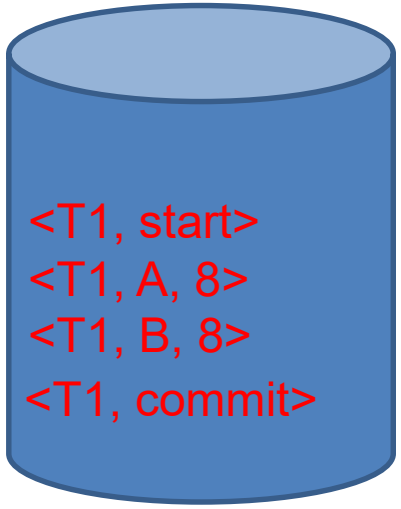
T1:
Read (A,t);
Write (A,t); t -> t* 2
Read (B,t);
Write (B,t); t -> t* 2

A=B

memória



disco



Log

Pontos de Verificação (Checkpoints)

- 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 T1, T2, ..., Tn>
      - Lista de transações ativas

Pontos de Verificação (Checkpoints)

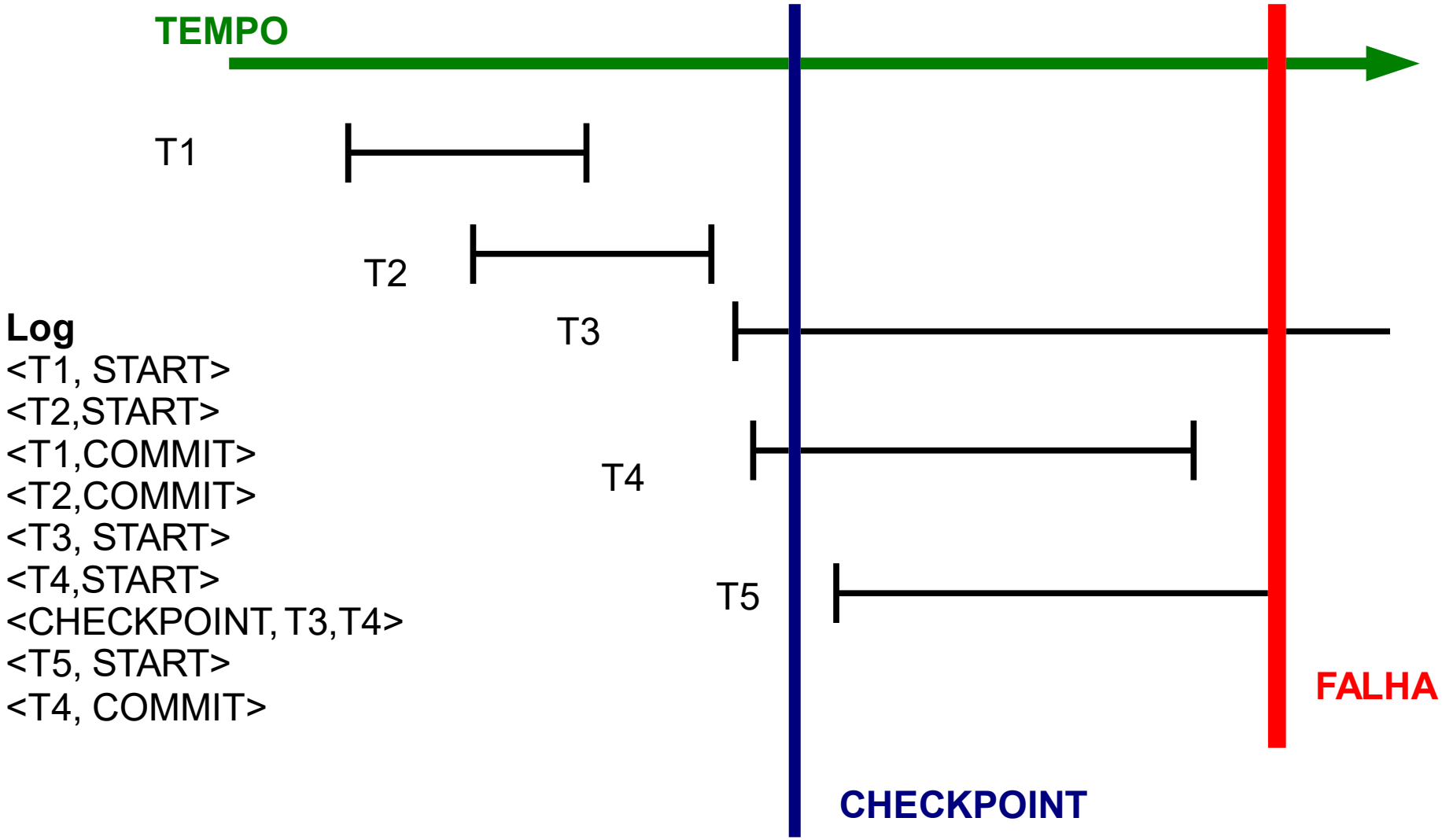
- Procedimento:
  - Suspensão de todas as transações
  - Descarga do buffer no disco
  - Inserção de um registro checkpoint no log e grava no disco
  - Retomada da execução das transações
- Vantagem:
  - Transações commited antes do checkpoint não precisam de REDO, já estão no disco

Pontos de Verificação (Checkpoints)

Percorrer o log (Modificação Adiada)

- **Funcionamento** (lista_refaz e lista_desfaz):
  - Lê log do FIM para INICIO (até encontrar checkpoint)
    - Para cada <Ti, COMMIT> - adiciona lista_refaz
    - Para cada <Ti, START> - se TI não está na lista_refaz, adiciona na lista_desfaz
    - Ao encontrar CHECKPOINT verifica a lista das transações ativas, se tá não estiver na lista_refaz, adiciona na lista_desfaz
    - Lê log do FIM → INICIO , executa UNDO na lista_desfaz
    - Lê log de INICIO->FIM e executa REDO para cada TI na lista_refaz

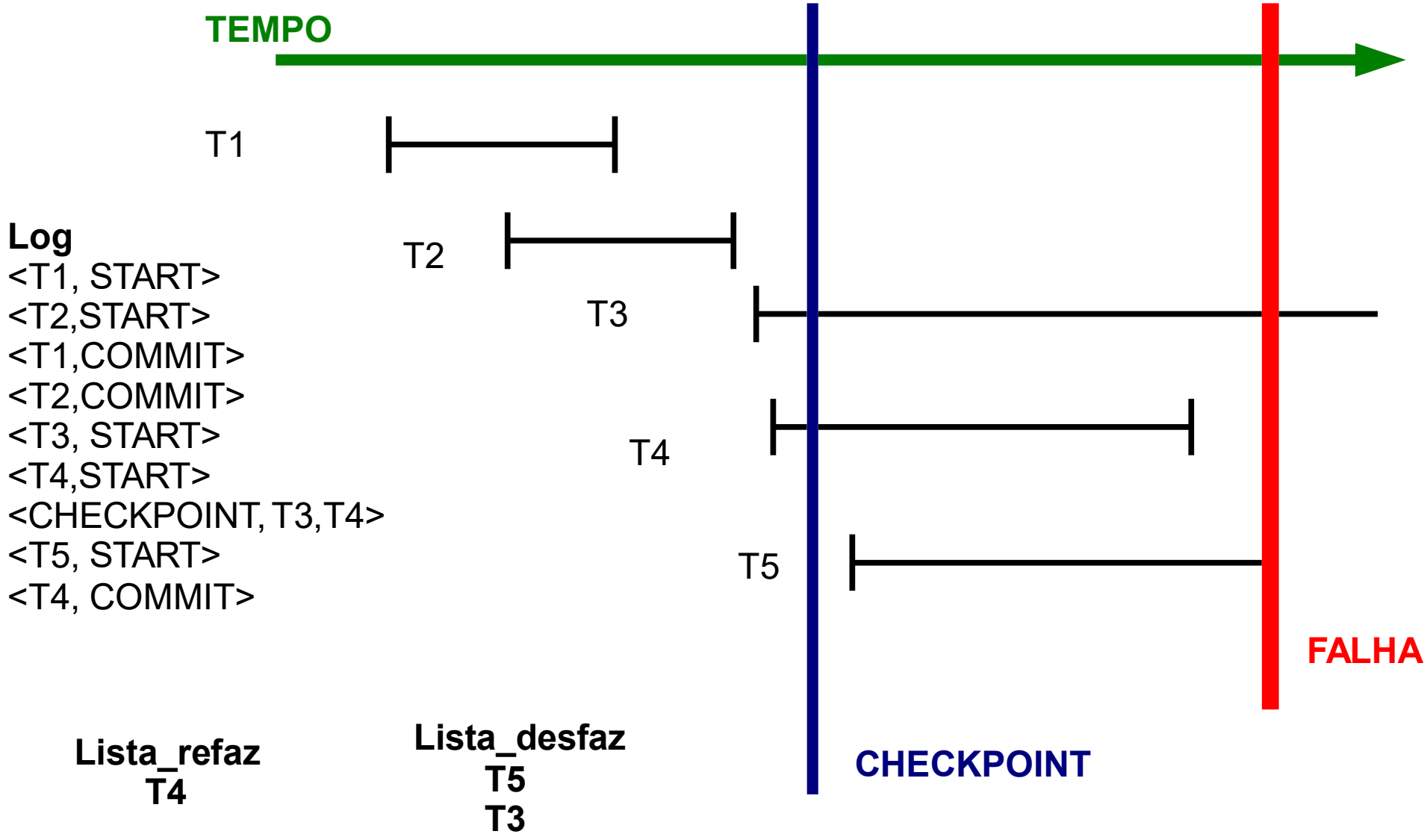
Pontos de Verificação (Checkpoints)



Log

<T1, START>
<T2,START>
<T1,COMMIT>
<T2,COMMIT>
<T3, START>
<T4,START>
<CHECKPOINT, T3,T4>
<T5, START>
<T4, COMMIT>

Pontos de Verificação (Checkpoints)



Log
<T1, START>
<T2,START>
<T1,COMMIT>
<T2,COMMIT>
<T3, START>
<T4,START>
<CHECKPOINT, T3,T4>
<T5, START>
<T4, COMMIT>

Lista_refaz
T4

Lista_desfaz
T5
T3

EXERCÍCIO

Log
<T4, START>
<T7,START>
<T1,START>
<T2,START>
<T7, COMMIT>
<T5,START>
<T4, COMMIT>
<T6, START>
<T5, COMMIT>
<T3, START>
<T6,COMMIT>

FALHA

Checkpoint após commit da T4

Quais transações serão REDO?
Quais transações serão UNDO?

EXERCÍCIO

Log
<T4, START>
<T7,START>
<T1,START>
<T2,START>
<T7, COMMIT>
<T5,START>
<T4, COMMIT>
<T6, START>
<T5, COMMIT>
<T3, START>
<T6,COMMIT>

FALHA

Checkpoint após commit da T4

Quais transações serão REDO?
Quais transações serão UNDO?

Lista_refaz	**Lista_desfaz**
T6	**T3**
T5	**T1**
	T2

EXERCÍCIO

Log
<T4, START>
<T7,START>
<T1,START>
<T2,START>
<T7, COMMIT>
<T5,START>
<T4, COMMIT>
<T6, START>
<T5, COMMIT>
<T3, START>
<T6,COMMIT>

FALHA

Checkpoint após commit da T4

Quais transações serão REDO?
Quais transações serão UNDO?

Lista_refaz	**Lista_desfaz**
T6	**T3**
T5	**T1**
	T2

Gerenciamento de Buffer

Cada registro de log -> disco

- Custo alto

Solução

Registros são mantidos no buffer de log (MP)

Vários registros são enviados ao disco ao mesmo tempo

Um registro de log pode residir apenas na MP antes de ser enviado ao disco

Registros perdidos em caso de falha

Regra: WAL (write-ahed-logging) Log de escrita antecipada

Gerenciamento de Buffer

Write-Ahead-Log (WAL)

Princípio básico

- A Transação Ti dá commit após o registro de log tiver sido gravado no disco
- Antes de gravar <Ti commit> no log, todos os registros de log desta transação devem ser enviados ao disco

O SGBD aplica técnicas de gerenciamento de buffer

Gerenciamento de Buffer

- **NOT-STEAL**
  - um bloco modificado por TX não pode ser gravado antes do commit
    - bloco possui um bit de status indicando se foi (1) ou não (0) modificado
    - vantagem: processo de recovery mais simples - evita blocos de dados inacabados gravados no BD

STEAL

- um bloco pode ser gravado antes do commit de Tx
  - 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
    - vantagem: não há necessidade de manter bloqueados por transações

Gerenciamento de Buffer

- **FORÇADA**
  - os blocos que mantêm dados atualizados por uma transação Tx são imediatamente gravados no BD quando Tx alcança o commit

    - deve-se saber quais os blocos que Tx atualizou dados
    - vantagem: garante a durabilidade de Tx o mais cedo possível

NÃO-FORÇADA

- os blocos que mantêm dados atualizados por Tx não são imediatamente gravados no BD quando Tx alcança o commit

  - Vantagem: blocos atualizados podem permanecer na cache e serem utilizados por outras transações, após o commit de Tx (reduz custo de acesso a disco)

Gerenciamento de Buffer

STEAL E NÃO FORÇADA
STEAL E FORÇADA
NÃO STEAL E NÃO FORÇADA
NÃO STEAL E FORÇADA

Gerenciamento de Buffer

STEAL E NÃO FORÇADA

UNDO – pode ter ido para o BD dados de tx's que não acabaram

REDO – dados de tx's com sucesso podem ter ficados no buffer

STEAL E FORÇADA

UNDO – pode ter ido para o BD dados de tx's que não acabaram

SEM REDO – dados de tX's com sucesso estão no banco

Gerenciamento de Buffer

NÃO STEAL E NÃO FORÇADA

SEM UNDO : BD só tem dados de tá's commited

REDO: dados podem não ter ido para o BD ainda

NÃO STEAL E FORÇADA

SEM UNDO

SEM REDO

– Só vai para o BD dados de tá's commited

Gerenciamento de Buffer

Qual das seguintes combinações de técnicas de gerenciamento de buffer requer um gerenciamento mais complexo por parte do SGBD?

- STEAL E NÃO FORÇADA
- NÃO STEAL E FORÇADA