RECUPERAÇÃO DE FALHAS

Profa.Dra. Maria Madalena Dias

RECUPERAÇÃO DE FALHAS

- Introdução
- O Log do Sistema
- Ponto de Commit
- *Checkpoint*
- Efeito Cascata
- Técnicas de Recuperação
  - Atualização Adiada
  - Atualização Imediata

Introdução

- Mecanismos de recuperação de falhas:
  - garantem a consistência do BD após falhas do sistema
  - garantem as propriedades de atomicidade e durabilidade das transações
- Recuperação:
  - implica que o BD será restaurado para algum estado (correto) do passado

Introdução

- Algoritmos têm duas partes:
  - ações realizadas durante a execução normal da transação para garantir que exista informação suficiente na recuperação de uma falha
  - ações tomadas depois da ocorrência de uma falha para garantir a consistência do BD e a atomicidade da transação

O Log do Sistema

- Arquivo especial armazenado em memória secundária (armazenamento em disco)
- Também chamado de *journal*
- Mantém informações de todas as operações das transações que afetam os valores dos dados do BD
- Utilizado pelos mecanismos de recuperação de falhas para recuperar o BD para um estado consistente

O Log do Sistema

- Entradas:
  - [start_transaction, T]
  - [write_item, T, X, old_value, new_ value]
  - [read_item, T, X]
  - [commit, T]
  - [abort, T]
- Essas entradas podem não ser necessárias a alguns protocolos de recuperação

O Log do Sistema

- Falha do sistema
  - ***undo*** o efeito das operações *write* de T percorrendo o log do fim para o começo e alterando os valores do item de dado X para *old_value*
  - entradas necessárias: *old_value*
    - valor antigo do item de dado
    - também chamado de *before image* (BFIM)

7

O Log do Sistema

- Falha do sistema
  - ***redo*** o efeito das operações *write* de T percorrendo o log a partir do início de T até o seu fim, refazendo todos os valores de X para *new_value*
  - entradas necessárias: *new_value*
    - valor novo do item de dado após a alteração
    - também chamado de *after image* (AFIM)
- Possibilidade de efeito cascata
  - read_item (para operação de *redo*)

8

Transação 1	**Transação 2**	**Transação 3**
begin_transaction read_item (x) x: = x + 5 write_item (x) commit_transaction		
	begin_transaction read_item (z) z := z – 1 write_item (z)	
		begin_transaction read_item (a) a: = a + 5 write_item (a) commit_transaction
	read_item (b) b := b – 1 write_item (b) commit_transaction	

9

valores iniciais: x = 10, z = 10, a = 5, b = 5

Execução das Transações

- begin_transaction
- read_item (x)
- x := x + 5
- write_item (x)
- commit_transaction
- begin_transaction
- read_item (z)
- z := z - 1
- write_item (z)
- begin_transaction
- read_item (a)
- a := a + 5
- write_item (a)
- commit_transaction
- read_item (b)
- b := b - 1
- write_item (b)
- commit_transaction

T1 T2 T3 T2

Arquivo de Log

[start_transaction, T1]
[read_item, T1, x]
[write_item, T1, x, 10, 15]
[commit, T1]
[start_transaction, T2]
[read_item, T2, z]
[write_item, T2, z, 10, 9]
[start_transaction, T3]
[read_item, T3, a]
[write_item, T3, a, 5, 10]
[commit, T3]
[read_item, T2, b]
[write_item, T2, b, 5, 4]
[commit, T2]

10

- **Arquivo de Log**
- [start_transaction, T1]
- [read_item, T1, x]
- [write_item, T1, x, 10, 15]
- [commit, T1]
- [start_transaction, T2]
- [read_item, T2, z]
- [write_item, T2, z, 10, 9]
- [start_transaction, T3]
- [read_item, T3, a]
- [write_item, T3, a, 5, 10]
- [commit, T3]
- [read_item, T2, b]
- [write_item, T2, b, 5, 4]

Falha do sistema após a entrada [commit, T3]

Transação T1 : redo
Transação T2 : undo
Transação T3 : redo

11

Ponto de Commit

- Uma transação atinge o ponto de commit quando:
  - todas as operações da transação que acessam o BD foram executadas com sucesso; e
  - o efeito destas operações foi armazenado no log
- Se uma transação atingiu seu ponto de commit, então
  - escreve a entrada [commit, T] no log; e
  - finaliza com sucesso

12

Checkpoint

- Entrada adicional no arquivo de log
  - [checkpoint]
- Indica que todas as transações que escreveram a entrada [commit, T] no log antes da entrada [checkpoint] tiveram seus resultados refletidos no BD

13

Checkpoint

- Realizar um *checkpoint* envolve as seguintes ações:
  - suspender a execução das transações temporariamente
  - forçar a escrita de todas as operações das transações finalizadas com sucesso (*commit*) dos *buffers* de memória principal para o disco
  - escrever a entrada [checkpoint] no log
  - ativar a execução das transações suspensas temporariamente

14

Checkpoint

- Registro de *checkpoint* no log pode incluir informações adicionais
  - uma lista de identificadores de transações ativas
  - endereços do primeiro e último registros do log para cada transação ativa
- Vantagem
  - pode ajudar a desfazer as operações das transações quando estas forem *rolled back*

15

Efeito Cascata

- Também chamado de aborto cascata
- Transações *uncommitted*
  - têm que ser *rolled back*
  - porque leram um item de dado de uma transação que falhou
- Exemplo:
  - $r_1(x)$; $w_1(x)$; $r_2(x)$; $r_1(y)$; $w_2(x)$; $w_1(y)$; a_1
  - r_2 (x) lê de $w_1(x)$
  - transação 1 falha ⇨ transação 2 também falha

16

Efeito Cascata

- Se uma transação T_i é desfeita
- então qualquer transação T_j (i <> j) que
- tenha lido o valor de algum item de dado
- escrito por T_i deve ser desfeita

também.

- Escalonamentos que evitam efeito cascata:
  - cada transação T no escalonamento S somente deve ler o que foi escrito por transações que foram finalizadas com sucesso (*committed*)

17

Técnicas de Recuperação

- São sempre dependentes do mecanismo de controle de concorrência utilizado
- Falhas catastróficas
  - mecanismo de recuperação
    - restaura uma cópia do BD de uma fita *backup*
    - reconstrói o BD refazendo (*redoing*) as operações que foram finalizadas com sucesso (*commit*), a partir de informações armazenadas no log
  - técnica utilizada
    - *backup* do BD e do log periodicamente

18

Técnicas de Recuperação

- Falhas não catastróficas
  - mecanismo de recuperação baseado nas informações armazenadas no log
    - desfaz operações que causaram inconsistência (*undo*)
    - refaz operações que não tiveram suas alterações refletidas no BD (*redo*)

19

Técnicas de Recuperação - Falhas não Catastróficas -

- Atualização Adiada
  - adia quaisquer atualizações no BD para depois que a transação atingiu o seu ponto de commit
  - somente depois deste ponto é realizada a atualização do BD
- Atualização Imediata
  - permite que o BD seja atualizado por qualquer operação da transação antes desta atingir o seu ponto de commit

20

Técnica de Atualização Adiada

- Antes do *commit*
  - alterações armazenadas no *workspace* da transação
- Durante o *commit*
  - alterações primeiro armazenadas no log de maneira permanente e somente depois escritas no BD

21

Técnica de Atualização Adiada

- Se uma transação falha antes do ponto de *commit*
  - então não é necessário desfazê-la (não alterou
  - valores no BD (*no undo*)
- Se ocorre qualquer falha depois que uma transação atingiu seu ponto de *commit*
  - então é necessário refazê-la (*redo*)
  - algoritmo NO-UNDO/REDO

22

Protocolo de Atualização Adiada

- Uma transação não pode alterar o BD até que atinja seu ponto de *commit*
- Uma transação não pode atingir seu ponto de *commit* até que suas operações de atualização sejam armazenadas no *log* (posteriormente as alterações do *log* serão refletidas no BD)

23

Técnica de Atualização Adiada

- Algoritmo Simples
  - mantenha duas listas de transações
    - uma para transações que agintiram o seu ponto de *commit* desde o último *checkpoint* realizado (fila 1)
    - outra para transações ativas (fila 2)
  - aplique a operação *redo* a todas as operações de escrita das transações da fila 1 com base no *log* (na mesma ordem com a qual elas foram escritas no log)
  - reinicialize as transações da fila 2

24

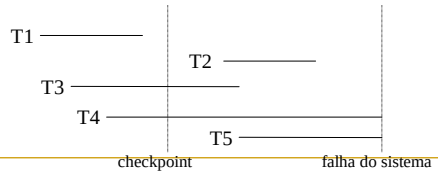
Técnica de Atualização Adiada

- Algoritmo para a operação *redo*
  - refaça as operações de escrita examinando as entradas [write_item, T, X, old_value, new_value] no *log*
  - para cada entrada, altere o valor do item de dado X para *new_value*

25

Exercício

- Considere o gráfico a seguir:
  - quais transações devem ser refeitas (*redo*)?
  - quais transações devem ser desfeitas (*undo*)?
- Considere a técnica de atualização adiada



26

Técnicas de Atualização Imediata

- Permitem que o BD seja atualizado por quaisquer operações da transação antes desta atingir seu ponto de *commit*
- Armazenam as alterações no *log* (em disco) antes de aplicá-las ao BD
- Se uma transação falha depois de realizar mudanças no BD, mas antes do ponto de *commit*
- então é necessário desfazê-la (*undo*)

27

Técnicas de Atualização Imediata

- Duas categorias de algoritmo
- **undo/no-redo**
  - todas as alterações de uma transação são armazenadas no BD antes que esta escreva a entrada [commit, T] no log: no redo
- **undo/redo**
  - uma transação pode escrever a entrada [commit, T] no log antes que suas alterações tenham sido refletidas no BD: redo

28

Técnicas de Atualização Imediata

- Algoritmo Simples (**undo-redo**)
  - mantenha duas listas de transações
    - uma para transações que atingiram o seu ponto de *commit* desde o último *checkpoint* realizado (fila 1)
    - outra para transações ativas (fila 2)
  - aplique a operação *undo* a todas as operações de escrita das transações da fila 2 com base no *log* (ordem inversa de escrita no *log*)
  - aplique a operação *redo* a todas as operações de escrita das transações da fila 1 com base no *log* (mesma ordem de escrita no *log*)

29

Técnicas de Atualização Imediata

- Algoritmo para a operação undo
  - desfaça as operações de escrita examinando as entradas [write_item, T X, old_value, new_value] no *log*
  - para cada entrada, altere o valor do item de dado X de *new_value* para *old_value*

30