

PROCESSAMENTO DE TRANSAÇÕES

Profa. Dra. Maria Madalena Dias

PROCESSAMENTO DE TRANSAÇÕES

- Introdução
- Execução Intercalada
- Execução Paralela
- Conceito de Transação
- Operação de Leitura
- Operação de Escrita
- Propriedades das Transações
- Necessidade de Controle de Concorrência
- Necessidade de Recuperação de Falhas
- Estados da Transação e Operações Adicionais
- Log do Sistema
- Escalonamento de Transação

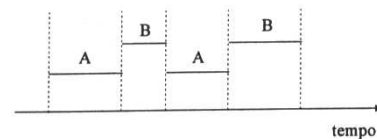
2

Introdução

- Ambiente multiusuário:
 - vários usuários utilizam o mesmo sistema ao mesmo tempo;
 - múltiplos programas (transações) compartilham a mesma CPU.
- Forma de execução dos programas:
 - intercalada (*interleaved*);
 - alguns comandos de um programa são executados e o programa é suspenso; alguns comandos de outro programa são executados, o programa é suspenso, e assim por diante.

3

Execução Intercalada



- SBD multiusuário:
 - recursos principais: dados armazenados no BD.

4

Discussão

- Se:
 - todos os programas forem pequenos;
 - todos os dados estiverem centralizados na memória principal;
 - todos os dados forem acessados por um único processador;
- Para quê concorrência

5

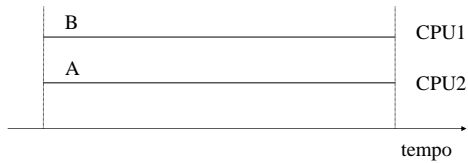
Discussão

- Caso geral
 - os programas apresentam tempo de resposta variante;
 - os programas são distribuídos entre muitos processadores, com muitas memórias disjuntas;
 - os dados podem estar distribuídos entre memórias rápidas e lentas;
- Assim, **deve existir concorrência.**

6

Execução Paralela

- 2 processos, 2 CPUs



- Pode ocorrer intercalamento de programas neste caso (duas ou mais CPUs)?

7

Transação

- Sequência de ações de leitura/escrita em objetos
- Objetos:
 - window
 - menu
 - keyboard
 - terminal
 - fila
 - registro de BD
 - outros processos

8

Transação

- Leitura
 - lê um objeto "olhando" algum aspecto de seu estado
- Escrita
 - escreve o objeto alterando o seu estado
- Criação e destruição de objetos
 - exemplo particular de escrita em seus estados

9

Transação

- Execução de um programa que acessa ou altera o conteúdo do BD
- Sequência de operações de escrita/leitura no BD
- Observações:
 - os dados do BD estão armazenados em memória secundária;
 - as operações que não sejam leitura/escrita não apresentam efeito para o BD. Exemplo: soma, subtração.

10

Operação de Leitura

- Representada por **r(x)** ou **read_item(s)**
- Lê item de dado **x** na variável de programa **x**
- Passos:
 - encontrar o endereço do bloco que contém **x**;
 - copiar o bloco para o *buffer* da memória principal (se necessário);
 - copiar o valor de **x** do *buffer* para a variável **x**.

11

Operação de Escrita

- Representada por **w(x)** ou **write_item(x)**
- Escreve o valor da variável de programa **x** no item de dado **x**
- Passos:
 - encontrar o endereço do bloco que contém **x**
 - copiar o bloco para o *buffer* da memória principal (se necessário)
 - copiar o valor da variável **x** para o *buffer*
 - escrever o novo valor do item de dado **x** no disco (atualização de BD)

12

Observações

- Transações submetidas pelos usuários podem
 - executar concorrentemente
 - acessar e alterar os mesmos itens de dados
- Execução não controlada pode
 - originar problemas como **inconsistência do banco de dados**

13

Exemplos

Transação 1

```
r(x)
x := x - n
w(x)
r(y)
y := y + n
w(y)

t1:r1(x)w1(x)r1(y)w1(y)
```

Transação 2

```
r(x)
x := x + m
w(x)

t2:r2(x) w2(x)
```

14

Propriedades das Transações

- Atomicidade
- Consistência
- Isolamento
- Durabilidade

15

Propriedades ACID

- Atomicidade
 - uma transação é uma unidade atômica (indivisível)
 - todas as operações das transações são finalizadas e refletidas no BD ou nenhuma delas é finalizada e refletida
- Durabilidade
 - os valores dos dados alterados durante a execução de uma transação devem persistir após a sua finalização

16

Propriedades ACID

- Consistência
 - transformações preservam a consistência do BD
 - a execução correta de uma transação leva o BD de um estado consistente a outro estado consistente
 - tarefa do programador que escreve os módulos do BD ou do módulo do SGBD que garante as restrições de integridade impostas pelo sistema

17

Propriedades ACID

- Isolamento
 - transações são isoladas umas das outras
 - cada transação assume que está sendo executada sozinha no sistema, o SGBD garante que os resultados intermediários da transação permaneçam escondidos de outras transações executando concorrentemente

18

Propriedades ACID

- Protocolos de controle de concorrência
 - ↓
 - garantem a consistência dos dados através de acessos concorrentes
 - ↓
 - isolamento
- Protocolo de recuperação de falhas
 - ↓
 - garantem a consistência dos dados após falhas do sistema
 - ↓
 - atomicidade e durabilidade

19

Consistência

- Estado de um BD
 - coleção de todos os valores de itens de dados armazenados no BD em um determinado momento
- Estado consistente de um BD
 - um estado consistente satisfaz às restrições especificadas no esquema, assim como quaisquer outras restrições que existam no BD

20

Restrições x Consistência

- Exemplos:
 - para cada elemento x, em uma fila duplamente encadeada, $prev(next(x)) = x$
 - EMP1 é uma réplica da tabela EMP
 - todos os gerentes de departamento são empregados e devem, portanto, aparecer na tabela EMP
 - se qualquer instância de EMP for alterada, a instância equivalente em EMP1 também deve ser alterada

21

Necessidade de Controle de Concorrência

- Problema da atualização perdida
- Problema da leitura incorreta
- Problema das somas incorretas
- Problema da leitura não repetida

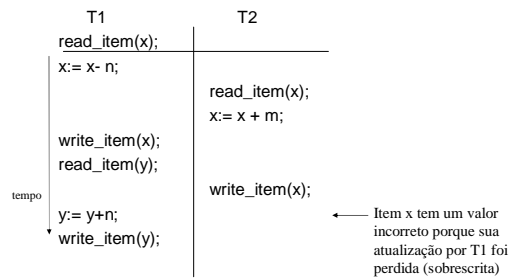
22

Problema da Atualização Perdida

- Ocorre quando duas transações que acessam os mesmos dados do BD têm suas operações intercaladas de modo a gerar um valor incorreto de algum item de dado
- Exemplo
 - dois programadores escrevem o mesmo programa ao mesmo tempo
 - cada cópia é alterada independentemente
 - a cópia 1 substitui a versão original
 - a cópia 2 substitui a versão 1
 - cópia 1: alterações perdidas

23

Problema da Leitura Incorreta



24

Problema das Somas Incorretas

T1	T2
<pre>read_item(x); x:= x - n; write_item(x);</pre>	<pre>sum:= 0; read_item(A); sum:= sum+A;</pre>
<pre>read_item(y); y:= y+n; write_item(y);</pre>	<pre>read_item(x); sum:= sum+x; read_item(y); sum:=sum+y;</pre>

← T3 lê x após n ser subtraído e lê y antes de n ser adicionado, então o resultado é um erro de soma

25

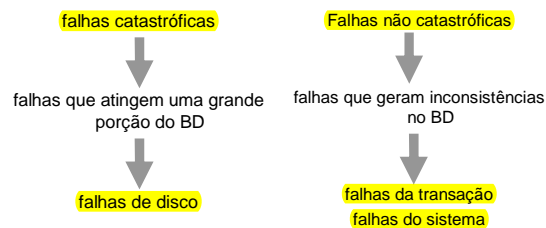
Problema da Leitura Não Repetida

T1	T2
<pre>read_item(x); x:= x + n;</pre>	<pre>read_item(x); x:= x + 10;</pre>
<pre>read_item(x); x:= x - 4;</pre>	<pre>write_item(x);</pre>

tempo

26

Necessidade de Recuperação de Falhas



27

Falhas da Transação

- Ocorrem quando uma transação não é finalizada com sucesso
- Exemplos
 - divisão por zero
 - leitura de um dado inexistente
 - transação é escolhida como vítima em algum protocolo de *deadlock*
 - valores de parâmetros incorretos
 - usuário interrompe uma transação através do CTRL-C

28

Falhas do Sistema

- Relacionadas à destruição dos dados da memória principal
- Exemplos
 - *bugs* no software do sistema gerenciador de BD
 - *bugs* no software do sistema operacional
 - falhas de hardware na CPU

29

Falhas de Disco

- Relacionadas à destruição de dados da memória secundária
- Também chamadas de problemas físicos
- Exemplo
 - quebra física do disco

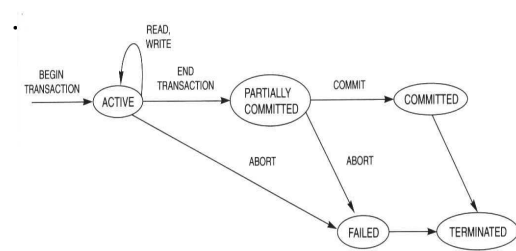
30

Estados da Transação

- **BEGIN_TRANSACTION**: Marca o início da execução da transação;
- **READ** ou **WRITE**: Operações de leitura e escrita sobre itens do BD;
- **END_TRANSACTION**: Especifica que as operações **READ** e **WRITE** terminaram e marca o limite final de execução da transação;
- **COMMIT_TRANSACTION**: sinaliza o fim com sucesso da transação;
- **ROLLBACK** (ou **ABORT**): sinaliza que a transação finalizou com erro, tal que quaisquer alterações realizadas pela transação devem ser desfeitas.

31

Estados da Transação



32

Operações Adicionais

- **UNDO**: Similar ao *rollback* exceto que é aplicada a uma única operação;
- **REDO**: Certas operações devem ser refeitas para garantir que todas as operações de uma transação *committed* (finalizada com sucesso) tenham sido aplicadas corretamente para o banco de dados.

33

O Log do Sistema

- O *log* (ou *jornal*) mantém uma trilha de todas as operações da transação que afetam os valores dos itens do BD.
- Informação necessária para permitir a recuperação de falhas de transação.
- Ponto de *commit*: uma transação *T* alcança seu ponto de *commit* quando todas as suas operações que acessam o BD foram executadas com sucesso e seus efeitos foram registrados no *log*; é dita ser uma transação "*committed*".
- *Checkpoints*: registra no *log*, periodicamente, os efeitos das operações de *write* de transações "*committed*".

34

Exercício

Transação 1
read_item(x)
x := x - 5

write_item(x)
read_item(y)
y := y + 50
write_item(y)

Transação 2
read_item(x)
temp := x * 0,1
x := x - temp
write_item(x)
read_item(y)

y := y + temp
write_item(y)

Existe problema?

35

Exercício

- Correlacione os itens da coluna ao lado com os três tipos de falhas a seguir:

1. Falha de disco	() escrita de dados em posições incorretos do disco
2. Falha da transação	() divisão por zero
3. Falha do sistema	() bugs no SGBD
	() leitura de um dado inexistente
	() bugs no SO
	() quebra física do disco

36

Escalonamento de Transação

- Ordem de execução de operações de várias transações executando concorrentemente de forma intercalada
- Um escalonamento S de n transações T_1, T_2, \dots, T_n é uma ordenação das operações das transações sujeitas à restrição que, para cada transação T_i que participa em S , as operações de T_i em S devem aparecer na mesma ordem em que elas ocorrem em T_i
- Exemplo (página 23):
 $S: r_1(x); r_2(x); w_1(x); r_1(y); w_2(x); c_2; w_1(y); c_1;$

37

Escalonamento de Transação

- Duas operações em um escalonamento estão em **conflito** se elas pertencem a diferentes transações, se elas acessam o mesmo item x e se uma das duas operações é um `write_item(x)`
- Um escalonamento S de n transações T_1, T_2, \dots, T_n é um **escalonamento completo** se as seguintes condições são mantidas:
 - As operações em S são exatamente aquelas operações em T_1, T_2, \dots, T_n , incluindo uma operação *commit* ou *abort* como última operação para cada transação no escalonamento;
 - Para qualquer par de operações da mesma transação T_i , sua ordem em S é a mesma em T_i ;
 - Para quaisquer duas operações em conflito, uma delas deve ocorrer antes da outra no escalonamento.

38

Escalonamento de Transação

- Um escalonamento S é dito ser **recuperável** se nenhuma transação T em S tenha atingido o *ponto de commit* até que todas as transações T' que tenha escrito um item que T lê tenha atingido o *ponto de commit*
- Exemplos:
 não recuperável:
 $S: r_1(x); w_1(x); r_2(x); r_1(y); w_2(x); c_2; a_1;$
 recuperável:
 $S: r_1(x); w_1(x); r_2(x); r_1(y); w_2(x); w_1(y); c_1; c_2;$

39

Serializabilidade de Escalonamentos

- Um escalonamento é dito ser **serial** quando as operações de cada transação são executadas consecutivamente, sem quaisquer operações intercaladas de outra transação; caso contrário, ele é dito ser **não serial**
- Limita concorrência ou intercalação de operações
- Um escalonamento S de n transações é **serializável** se ele é equivalente a algum escalonamento serial das mesmas n transações

40

Equivalência de Escalonamento

- Para dois escalonamentos serem equivalentes, as operações aplicadas a cada item de dado afetado pelos escalonamentos deveriam ser aplicadas àquele item em ambos os escalonamentos e na mesma ordem
- Equivalência de Conflito:
 - dois escalonamentos são ditos serem equivalentes de conflito se a ordem de quaisquer duas operações conflitantes é a mesma em ambos os escalonamentos
 - um escalonamento S é serializável de conflito se ele é equivalente (de conflito) para o mesmo escalonamento S'

41

Equivalência de Escalonamento

- Exemplo de equivalência de conflito

T_1	T_2	T_1	T_2
$r(x);$		$r(x);$	
$x := x - n;$		$x := x - n;$	
$w(x);$		$w(x);$	
$r(y);$			$r(x);$
$y := y + n;$			$x := x + m;$
$w(y);$			$w(x);$
	$r(x);$	$r(y);$	
	$x := x + m;$	$y := y + n;$	
	$w(x);$	$w(y);$	

42

Equivalência de Escalonamento

■ Equivalência de Visão

- o mesmo conjunto de transações participa em S e S' , sendo que S e S' incluem as mesmas operações daquelas transações
- para qualquer operação $r_i(x)$ de T_i em S , se o valor de x lido pela operação foi escrito por uma operação $w_j(x)$ de T_j (ou se ele é o valor original de x antes do escalonamento ter iniciado), a mesma condição deve permanecer para o valor de x lido pela operação $r_i(x)$ de T_i em S' .
- Se a operação $w_k(y)$ de T_k é a última operação a escrever o item y em S , então $w_k(y)$ de T_k deve também ser a última operação a escrever o item y em S' .

Ex: $S : r_1(x); w_1(s); w_2(x); w_3(x); c_1; c_2; c_3$ (Escalonamento Serial)

$S' : r_1(x); w_2(x); w_1(s); w_3(x); c_1; c_2; c_3$ (Equivalência de Visão)