INF / UFG

Disciplina

Banco de Dados

Conteúdo

Escalonamento Baseado em Recuperação

Recuperação É Necessária ?

Sempre que uma transação é submetida a um SGBD, o sistema é responsável por garantir que:

(i) tanto todas as operações na transação sejam concluídas com êxito, e seu efeito seja gravado permanentemente no banco de dados (***committed transaction***),

OU

(ii) a transação não tem qualquer efeito sobre o banco de dados ou quaisquer outras transações (***aborted transaction***).

Se uma transação falhar depois de executar algumas de suas operações, as operações já realizadas deve ser desfeitas.

Tipos de Falha

As falhas são geralmente classificados como **falhas de transação**, **falhas de sistema** e **falhas de mídia**.

Há várias razões possíveis para uma transação falhar:

1. Falha do computador(falha do sistema). Um erro de hardware, de software ou de rede ocorre no sistema durante a execução da transação. Falhas de hardware são geralmente falhas de media; porexemplo, falhas de memória principal.

2. Erro de transação ou sistema. Algumas operações na transação podem falhar, tal como *integer overflow*, divisão por zero ou erros lógicos de programação. Também, o usuário pode interromper a transação durante a sua execução.

Tipos de Falha

3. Condições de exceção detectadas pela transação. Por exemplo, uma condição de exceção pode ser detectada para uma regra de negócio, tal como saldo insuficiente em uma transação bancária. Esta exceção poderia ser programada na própria transação e, em tal caso, não seria considerada como uma falha de transação.

4. Aplicação de controle de concorrência. O mecanismo de controle de concorrência pode decidir:
(i) abortar uma transação, porque ela viola serialização, ou
(ii) abortar uma ou mais transações para resolver uma situação de impasse (*deadlock*) entre várias transações.
As transações abortadas nesses casos são normalmente reiniciadas automaticamente em um momento posterior.

Tipos de Falha

5. Falha de disco. Alguns blocos de disco podem perder seus dados (exemplo, falha no mecanismo físico de acesso). Isto pode acontecer durante operações de leitura ou de escrita da transação.

6. Problemas físicos e catástrofes. Refere-se a uma lista interminável de problemas que inclui falha de energia, falha de ar condicionado, incêndio, roubo, sabotagem, etc.

Falhas dos Tipos 1, 2, 3 e 4 são mais comuns que as de Tipos 5 e 6.

Sempre que uma falha dos Tipo 1 a 4 ocorre, o sistema tem de **manter informação suficiente** para recuperar rapidamente a falha.

Falhas do tipo 5 ou 6 requerem mecanismo de recuperação mais demorado.

Log do Banco de Dados

Para recuperar as falhas ocorridas, o sistema mantém um ***log*** sobre todas as operações da transação que afetam os valores dos itens de banco de dados, bem como outras informações necessárias ao mecanismo de recuperação.

O log é um arquivo sequencial, onde novos registros são gravados no seu final (do tipo *append-only*, que é afetado somente porfalhas do disco ou falhas catastróficas).

As transações no banco de dados serão primeiramente garantidas no arquivo de log e, em seguida, serão *committed* no banco de dados.

O arquivo de log deve periodicamente salvo (um novo arquivo de log vazio pode ser criado após a salva).

Tipos de Registros no Arquivo de *Log*

Considere que T refere-se a um único ID de transação que é gerada automaticamente pelo sistema.

[start_transaction, T]. A transação T começou sua execução.

[write_item, T, X, old_value, new_value]. A transação T alterou o valor do item de dados X de *old_value* para *new_value*.

[read_item, T, X]. A transação T leu o valor do item de dados X. Este tipo de registro pode não ser usado.

[commit, T]. A transação T foi concluída com êxito, e o seu efeito pode ser *committed* (gravado permanentemente) no banco de dados.

[abort, T]. A transação T foi abortada.

Exemplo de Arquivo de *Log*

Log

<start T_3>
<write T_3,B,15,12>
<start T_2>
<write T_2,B,12,18>
<start T_1>
<write T_1,D,20,25>
<commit T_1>
<write T_2,D,25,26>
<write T_3,A,10,19>
<commit T_3>
<commit T_2>
...

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

T_2

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

T_3

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

Transação X Recuperação X Concorrência

O conceito de transação é fundamental para muitas técnicas para controle de concorrência e de recuperação de falhas.

Escalonamento

Quando as transações estão sendo executadas ao mesmo tempo e de uma forma intercalada, em seguida, a ordem de execução das operações de todas as diversas transações é conhecido como **escalonamento** (*schedule*, ou histórico).

Um Escalonamento **S** de **n** transações **T1, T2, ..., Tn** é uma ordenação das operações das transações. As operações das diferentes transações podem ser intercaladas em **S**.

No entanto, para cada transação **Ti** que participa do escalonamento **S**, as operações de **Ti** em **S** devem aparecer na mesma ordem em que elas ocorrem em **Ti**.

Representação de Escalonamento

Para propósitos de recuperação e controle de concorrência, as operações pertinentes são: read_item, write_item, commit e abort.

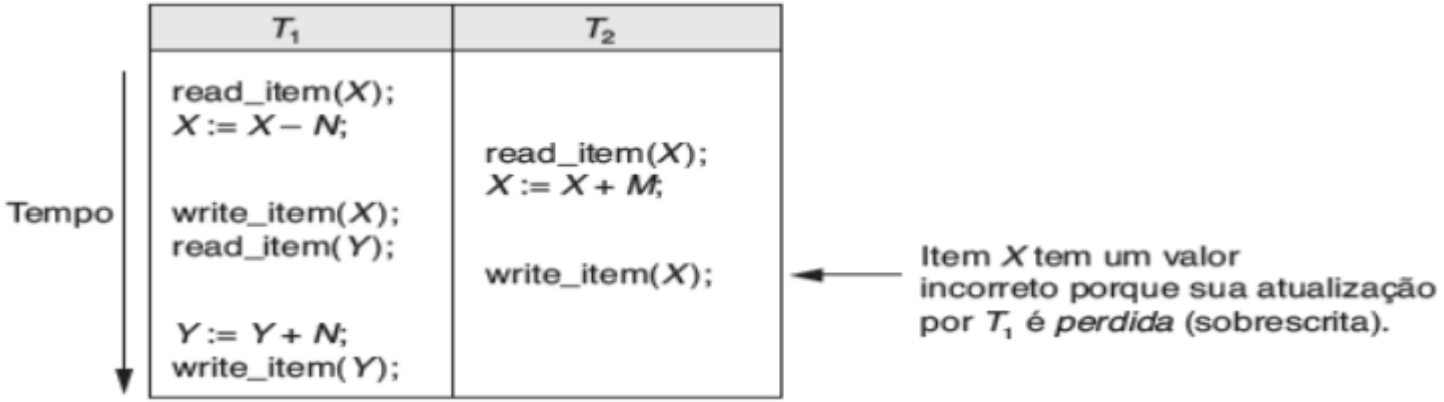
Uma notação abreviada para descrever um escalonamento:

a	begin_transaction
r	read_item
w	write_item
e	end_transaction
c	commit
a	abort

Representação de Escalonamento

Um escalonamento pode ser apresentado de forma simplificada somente com operações de leitura e escrita:

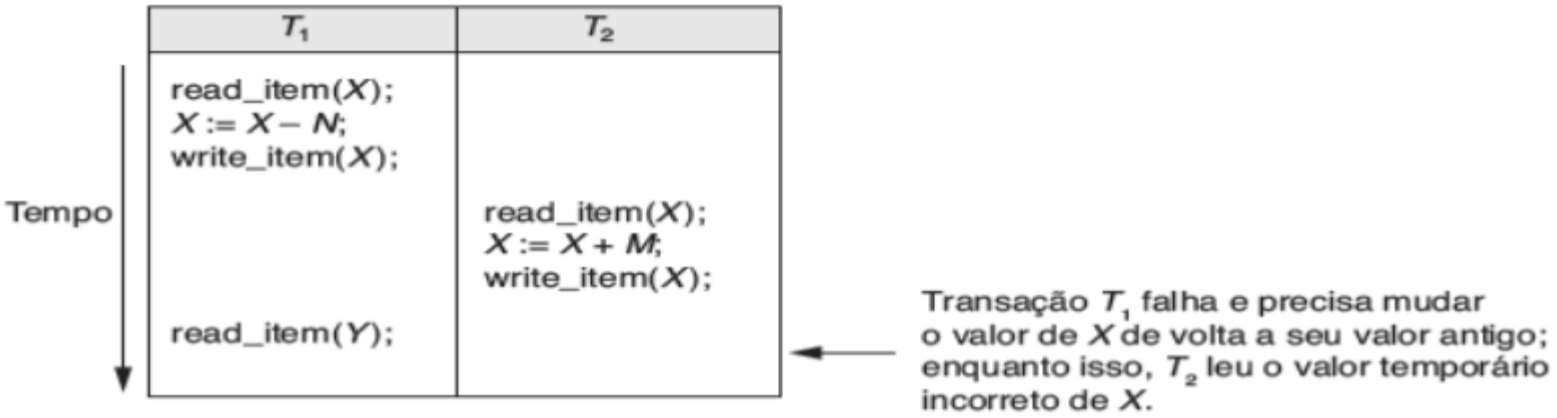
Sa: r1 (X), r2 (X); w1 (X); r1 (Y); w2 (X); w1 (Y);



Representação de Escalonamento

Na figura abaixo, se assumirmos que a transação T1 foi abortada após a operação read_item (Y):

Sb: r1 (X); w1 (X); r2 (X); w2 (X); r1 (Y); a1;



Transações em Conflito

Duas operações de um escalonamento estão em **conflito** quando:

(1) pertencem a diferentes transações; e
(2) acessam o mesmo item de dado **X**; e
(3) pelo menos uma das operações é um **write_item (X)**.

Considere o escalonamento

Sa: r1 (X), r2 (X); w1 (X); r1 (Y); w2 (X); w1 (Y);

Ha conflito em: r1 (X) e w2 (X),
r2 (X) e w1 (X),
w1 (X) e w2 (X).

Não há conflito em: r1(X) e r2(X),
w2(X) e w1(Y),
r1(X) e w1(X).

Transações em Conflito

Intuitivamente, duas operações são conflitantes se a alteração da sua ordem pode originar em um resultado diferente.

Exemplo de conflito *read-write*:

alterar r1(X),w2(X) para w2(X),r1(X)

>> o valor de X que é lido por T1 é modificado, pois será o valor de X que é alterado por w2(x).

Exemplo de conflito *write-write*:

alterar w1(X),w2(X) para w2(X),w1(X)

>> no primeiro o valor final de X é escrito por T2; no segundo, o valor final de X é escrito por T1.

Escalonamento Completo

Um escalonamento S de n transações T1, T2, ..., Tn é dito ser um **escalonamento completo** se:

(1) as operações em S são exatamente essas operações em T1, T2, ..., Tn, incluindo uma operação de *commit* ou *abort* como a última operação para cada transação no escalonamento;

(2) para qualquer par de operações da mesma transação Ti, a sua ordem relativa em S é a mesma em Ti;

(3) para quaisquer duas operações de conflito, uma deve estar antes da outra no escalonamento (é óbvio, mas ressalta que operações que não estão em conflito podem aparecerem qualquer ordem).

Uma vez que cada transação termina com *commit* ou *abort*, um escalonamento completo **não conterá quaisquer transações ativas em seu final.**

Escalonamento X Recuperabilidade

Para alguns escalonamentos, o processo de recuperação (falhas de sistema e de transação) é simples, enquanto que em outros escalonamentos o processo de recuperação pode ser muito caro.

Em alguns casos, não é mesmo possível recuperar corretamente o banco de dados após a ocorrência de uma falha.

Por isso, é importante caracterizar os tipos de escalonamentos :
(caracterizar teoricamente diferentes tipos de escalonamentos)

(i) a recuperação é possível;
(ii) a recuperação é relativamente simples.

Escalonamento X Recuperabilidade

Para atender a propriedade de durabilidade, uma vez que a Transação T for confirmada (*committed*), nunca deveria ser necessário revertê-la.

Os escalonamentos que, **teoricamente**, atender a tal critério são chamados **escalonamentos recuperáveis;** os demais são chamados **escalonamentos irrecuperáveis**.

Um Escalonamento S é recuperável se nenhuma Transação T em S for confirmada (*committed*), até todas as Transações T', que tenham escrito algum Item X que T lê, tenham sido confirmadas (*committed*).

Uma Transação T lê de uma Transação T' no Escalonamento S, se algum Item X for primeiro escrito por T' e depois lido por T:

>> além disso, T' não deveria ter sido abortada antes que T leia X, e não deveria haver qualquer transação que escreva X depois que T' tiver escrito X e antes que T leia X (a menos que essas transações, se alguma, tiver abortado antes que T leia X).

Escalonamento X Recuperabilidade

Alguns escalonamentos recuperáveis podem exigir um processo de recuperação complexo:

>> se a informação suficiente é mantida (no *log*), um algoritmo de recuperação pode ser criado para qualquer escalonamento recuperável.

Os escalonamentos (parcial) **Sa** e **Sb** abaixo

Sa : r1(X); r2(X); w1(X); r1(Y); w2(X); w1(Y);

Sb : r1(X); w1(X); r2(X); w2(X); r1(Y); a1;

são ambos recuperáveis, desde que satisfaçam a definição acima.

Considere o escalonamento **Sa*** :

Sa* : r1 (X), r2 (X); w1 (X); r1 (Y); w2 (X); c2; w1 (Y); c1;

Sa* é recuperável, mesmo que sofra do problema atualização perdida.

Escalonamento X Recuperabilidade

No entanto, considere os dois Escalonamentos (parciais) Sc e Sd:

Sc: r1(X); w1(X); r2(X); r1(Y); w2(X); c2; a1;
Sd: r1(X); w1(X); r2(X); r1(Y); w2(X); w1(Y); c1; c2;

Sc não é recuperável, porque T2 lê X de T1, mas T2 é confirmada (*committed*) antes de T1.
>> o problema ocorre se T1 abortar após a operação c2, então o valor de X que T2 leu não é mais válido e T2 deve ser abortada mesmo depois de ter sido confirmada (*committed*), **levando a um escalonamento não recuperável**.

Para o escalonamento ser recuperável, a Operação **c2** deve ser adiada até depois de T1 ser confirmada, conforme posto no Escalonamento **Sd**.

Se T1 for abortada (em vez de ser confirmada), então T2 também deveria ser abortada, conforme mostrado no Escalonamento **Se**:

Se: r1(X); w1(X); r2(X); r1(Y); w2(X); w1(Y); a1; a2;

Em **Se**, o aborto de T2 é aceitável, pois T2 ainda não havia sido confirmada, evitando que o escalonamento fosse não recuperável.

Escalonamento com Aborto em Cascata

Em um escalonamento recuperável, nenhuma transação confirmada (*committed*) precisa ser revertida: **transação confirmada é durável**.

No entanto, é possível ocorrerem em alguns escalonamentos recuperáveis um fenômeno conhecido como ***rollback* em cascata** (ou aborto em cascata):

>> **uma transação não confirmada precisa ser revertida, pois ela leu um item de dado de uma transação que falhou; ver Escalonamento Se:**

Se : r1(X); w1(X); r2(X); r1(Y); w2(X); w1(Y); a1; a2;

a Transação T2 precisa ser revertida, pois ela leu o Item X de T1, mas T1 abortou.

O ***rollback* em cascata** pode ser demorado, uma vez que muitas transações podem necessitar ser revertidas.

Escalonamento Sem Aborto em Cascata

Um escalonamento é dito **sem aborto em cascata** (*cascadeless*), se cada transação no escalonamento lê apenas os itens que foram escritos por transações confirmadas (*committed*).

A Operação **r2(X)** nos Escalonamentos Sd e Se deve ser adiada até que T1 tenha sido confirmada (ou abortada), garantindo que não haja aborto em cascata se T1 abortar.

Sd : r1(X); w1(X); r2(X); r1(Y); w2(X); w1(Y); c1; c2;
Se : r1(X); w1(X); r2(X); r1(Y); w2(X); w1(Y); a1; a2;

Escalonamento Estrito

No escalonamento estrito, transações não podem ler nem escrever um Item até a última transação que escreveu X tenha sido confirmada (ou abortada).

Escalonamentos estritos simplificam o processo de recuperação.

>> o processo de desfazer uma operação **write_item (X)** de uma transação abortada é simplesmente restaurar a sua imagem anterior (*old_value* ou *before image* – BFIM) do item de dados X.

Este procedimento simples funciona em escalonamentos estritos, **mas não funciona em escalonamentos recuperáveis ou sem aborto em cascata**.

Escalonamento Estrito

Considere o Escalonamento **Sf** :

Sf : w1(X, 5); w2(X, 8); a1;

Suponha que o valor de X foi originalmente 9 (imagem anterior de X), e no log é gravado w1 (X, 9, 5).

Se T1 abortar, o processo de recuperação irá restaurar o valor 9 para X, embora X tenha sido alterado para 8 por T2.

Embora o Escalonamento Sf seja sem aborto em cascata, ele **não é escalonamento estrito**, uma vez que permite T2 escrever o Item X mesmo que T1 (que escreveu antes em X) ainda não tenha sido confirmada (ou abortada). Um escalonamento estrito não tem esse problema.

Escalonamento Estrito

Qualquer escalonamento estrito é também sem aborto em cascata.
Qualquer escalonamento sem aborto em cascata é também recuperável.

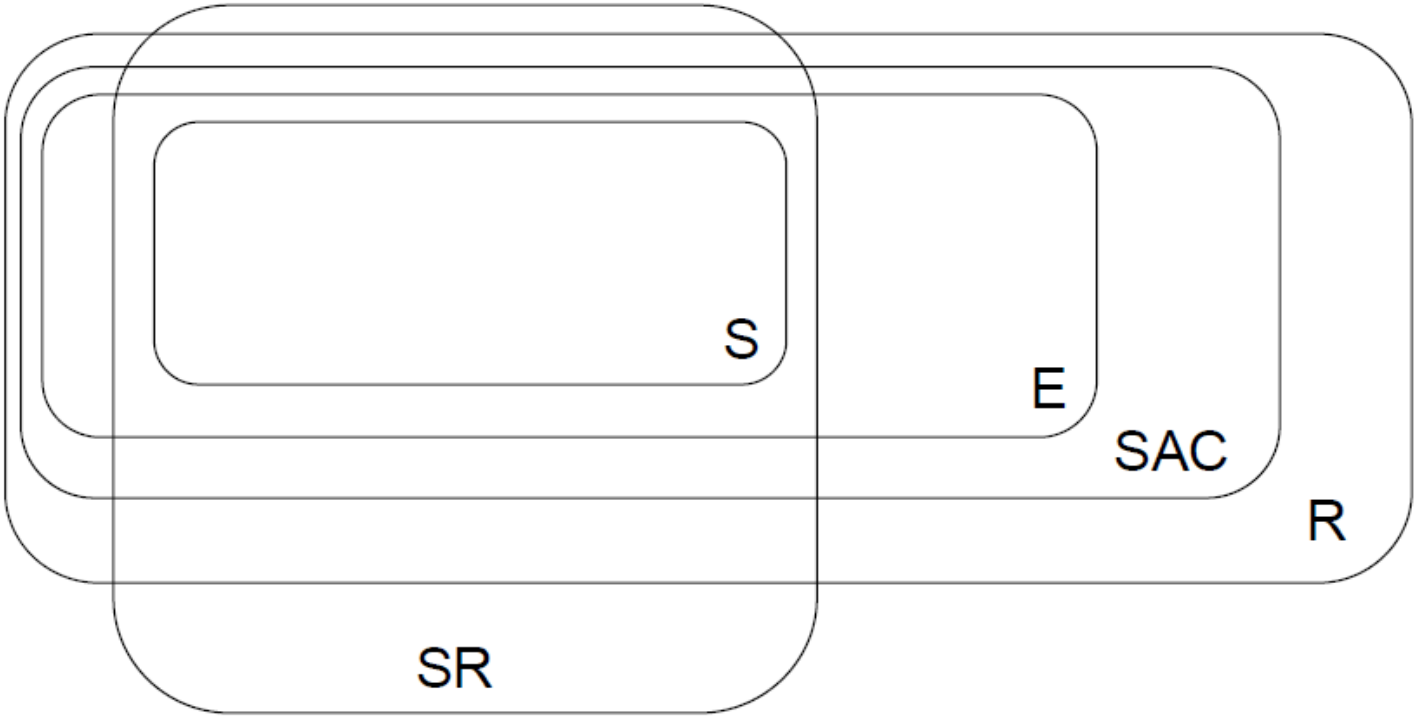
Para avaliar um conjunto de escalonamentos, é pertinente separar dois subconjuntos: recuperáveis e não recuperáveis.

Os escalonamentos sem aborto em cascata são um subconjunto dos escalonamentos recuperáveis.

Os escalonamentos estritos são um subconjunto dos escalonamentos sem aborto em cascata.

Relação entre Escalonamentos

- SR = escalonamento serializável
- R = escalonamento recuperável
- SAC = escalonamento sem aborto em cascata
- E = escalonamento estrito
- S = escalonamento serial



Exercício

Adicionar operação **commit** no final de cada uma das transações de T1 e T2. Liste todos os escalonamentos possíveis para as transações modificadas. Determine quais dos escalonamentos são recuperáveis, sem aborto em cascata e estritos.

(a)

T_1

read_item(X);
$X := X - N$;
write_item(X);
read_item(Y);
$Y := Y + N$;
write_item(Y);

(b)

T_2

read_item(X);
$X := X + M$;
write_item(X);

T1 : r1 (X); w1 (X); r1 (Y); w1 (Y); C1 ;
T2 : r2 (X); w2 (X); C2 ;

Exercício

T1 : r1 (X); w1 (X); r1 (Y); w1 (Y); C1 ;
T2 : r2 (X); w2 (X); C2 ;

(a)

T_1

read_item(X);
$X := X - N$;
write_item(X);
read_item(Y);
$Y := Y + N$;
write_item(Y);

(b)

T_2

read_item(X);
$X := X + M$;
write_item(X);

S1 : r1(X); w1(X); r1(Y); w1(Y); C1 ; r2(X); w2(X); C2 ; estrito
S2 : r1(X); w1(X); r1(Y); w1(Y); r2(X); C1 ; w2(X); C2 ; recuperável
S3 : r1(X); w1(X); r1(Y); w1(Y); r2(X); w2(X); C1 ; C2 ; recuperável
S4 : r1(X); w1(X); r1(Y); w1(Y); r2(X); w2(X); C2 ; C1 ; não recuperável
S5 : r1(X); w1(X); r1(Y); r2(X); w1(Y); C1 ; w2(X); C2 ; recuperável
S6 : r1(X); w1(X); r1(Y); r2(X); w1(Y); w2(X); C1 ; C2 ; recuperável
S7 : r1(X); w1(X); r1(Y); r2(X); w1(Y); w2(X); C2 ; C1 ; não recuperável
S8 : r1(X); w1(X); r1(Y); r2(X); w2(X); w1(Y); C1 ; C2 ; recuperável
S9 : r1(X); w1(X); r1(Y); r2(X); w2(X); w1(Y); C2 ; C1 ; não recuperável
S10 : r1(X); w1(X); r1(Y); r2(X); w2(X); C2 ; w1(Y); C1 ; não recuperável

Exercício

T1 : r1 (X); w1 (X); r1 (Y); w1 (Y); C1 ;
T2 : r2 (X); w2 (X); C2 ;

(a)

T_1

read_item(X);
$X := X - N$;
write_item(X);
read_item(Y);
$Y := Y + N$;
write_item(Y);

(b)

T_2

read_item(X);
$X := X + M$;
write_item(X);

S11 : r1(X); w1(X); r2(X); r1(Y); w1(Y); C1 ; w2(X); C2 ; recuperável
S12 : r1(X); w1(X); r2(X); r1(Y); w1(Y); w2(X); C1 ; C2 ; recuperável
S13 : r1(X); w1(X); r2(X); r1(Y); w1(Y); w2(X); C2 ; C1 ; não recuperável
S14 : r1(X); w1(X); r2(X); r1(Y); w2(X); w1(Y); C1 ; C2 ; recuperável
S15 : r1(X); w1(X); r2(X); r1(Y); w2(X); w1(Y); C2 ; C1 ; não recuperável
S16 : r1(X); w1(X); r2(X); r1(Y); w2(X); C2 ; w1(Y); C1 ; não recuperável
S17 : r1(X); w1(X); r2(X); w2(X); r1(Y); w1(Y); C1 ; C2 ; recuperável
S18 : r1(X); w1(X); r2(X); w2(X); r1(Y); w1(Y); C2 ; C1 ; não recuperável
S19 : r1(X); w1(X); r2(X); w2(X); r1(Y); C2 ; w1(Y); C1 ; não recuperável
S20 : r1(X); w1(X); r2(X); w2(X); C2 ; r1(Y); w1(Y); C1 ; não recuperável

Exercício

T1 : r1 (X); w1 (X); r1 (Y); w1 (Y); C1 ;
T2 : r2 (X); w2 (X); C2 ;

(a)

T_1

read_item(X);
$X := X - N$;
write_item(X);
read_item(Y);
$Y := Y + N$;
write_item(Y);

(b)

T_2

read_item(X);
$X := X + M$;
write_item(X);

S21 : r1(X); r2(X); w1(X); r1(Y); w1(Y); C1 ; w2(X); C2 ; estrito
S22 : r1(X); r2(X); w1(X); r1(Y); w1(Y); w2(X); C1 ; C2 ; sem aborto em cascata
S23 : r1(X); r2(X); w1(X); r1(Y); w1(Y); w2(X); C2 ; C1 ; sem aborto em cascata
S24 : r1(X); r2(X); w1(X); r1(Y); w2(X); w1(Y); C1 ; C2 ; sem aborto em cascata
S25 : r1(X); r2(X); w1(X); r1(Y); w2(X); w1(Y); C2 ; C1 ; sem aborto em cascata
S26 : r1(X); r2(X); w1(X); r1(Y); w2(X); C2 ; w1(Y); C1 ; sem aborto em cascata
S27 : r1(X); r2(X); w1(X); w2(X); r1(Y); w1(Y); C1 ; C2 ; sem aborto em cascata
S28 : r1(X); r2(X); w1(X); w2(X); r1(Y); w1(Y); C2 ; C1 ; sem aborto em cascata
S29 : r1(X); r2(X); w1(X); w2(X); r1(Y); C2 ; w1(Y); C1 ; sem aborto em cascata
S30 : r1(X); r2(X); w1(X); w2(X); C2 ; r1(Y); w1(Y); C1 ; sem aborto em cascata

Exercício

T1 : r1 (X); w1 (X); r1 (Y); w1 (Y); C1 ;
T2 : r2 (X); w2 (X); C2 ;

(a)

T_1

read_item(X);
$X := X - N$;
write_item(X);
read_item(Y);
$Y := Y + N$;
write_item(Y);

(b)

T_2

read_item(X);
$X := X + M$;
write_item(X);

S31 : r1(X); r2(X); w2(X); w1(X); r1(Y); w1(Y); C1 ; C2 ; sem aborto em cascata
S32 : r1(X); r2(X); w2(X); w1(X); r1(Y); w1(Y); C2 ; C1 ; sem aborto em cascata
S33 : r1(X); r2(X); w2(X); w1(X); r1(Y); C2 ; w1(Y); C1 ; sem aborto em cascata
S34 : r1(X); r2(X); w2(X); w1(X); C2 ; r1(Y); w1(Y); C1 ; sem aborto em cascata
S35 : r1(X); r2(X); w2(X); C2 ; w1(X); r1(Y); w1(Y); C1 ; estrito
S36 : r2(X); r1(X); w1(X); r1(Y); w1(Y); C1 ; w2(X); C2 ; estrito
S37 : r2(X); r1(X); w1(X); r1(Y); w1(Y); w2(X); C1 ; C2 ; sem aborto em cascata
S38 : r2(X); r1(X); w1(X); r1(Y); w1(Y); w2(X); C2 ; C1 ; sem aborto em cascata
S39 : r2(X); r1(X); w1(X); r1(Y); w2(X); w1(Y); C1 ; C2 ; sem aborto em cascata
S40 : r2(X); r1(X); w1(X); r1(Y); w2(X); w1(Y); C2 ; C1 ; sem aborto em cascata

Exercício

T1 : r1 (X); w1 (X); r1 (Y); w1 (Y); C1 ;
T2 : r2 (X); w2 (X); C2 ;

(a)

T_1

read_item(X);
$X := X - N$;
write_item(X);
read_item(Y);
$Y := Y + N$;
write_item(Y);

(b)

T_2

read_item(X);
$X := X + M$;
write_item(X);

S41 : r2(X); r1(X); w1(X); r1(Y); w2(X); C2 ; w1(Y); C1 ; sem aborto em cascata
S42 : r2(X); r1(X); w1(X); w2(X); r1(Y); w1(Y); C1 ; C2 ; sem aborto em cascata
S43 : r2(X); r1(X); w1(X); w2(X); r1(Y); w1(Y); C2 ; C1 ; sem aborto em cascata
S44 : r2(X); r1(X); w1(X); w2(X); r1(Y); C2 ; w1(Y); C1 ; sem aborto em cascata
S45 : r2(X); r1(X); w1(X); w2(X); C2 ; r1(Y); w1(Y); C1 ; sem aborto em cascata
S46 : r2(X); r1(X); w2(X); w1(X); r1(Y); w1(Y); C1 ; C2 ; sem aborto em cascata
S47 : r2(X); r1(X); w2(X); w1(X); r1(Y); w1(Y); C2 ; C1 ; sem aborto em cascata
S48 : r2(X); r1(X); w2(X); w1(X); r1(Y); C2 ; w1(Y); C1 ; sem aborto em cascata
S49 : r2(X); r1(X); w2(X); w1(X); C2 ; r1(Y); w1(Y); C1 ; sem aborto em cascata
S50 : r2(X); r1(X); w2(X); C2 ; w1(X); r1(Y); w1(Y); C1 ; ESTRITO ?

Exercício

T1 : r1 (X); w1 (X); r1 (Y); w1 (Y); C1 ;
T2 : r2 (X); w2 (X); C2 ;

(a)

T_1

read_item(X);
$X := X - N$;
write_item(X);
read_item(Y);
$Y := Y + N$;
write_item(Y);

(b)

T_2

read_item(X);
$X := X + M$;
write_item(X);

S51 : r2(X); w2(X); r1(X); w1(X); r1(Y); w1(Y); C1 ; C2 ; não recuperável
S52 : r2(X); w2(X); r1(X); w1(X); r1(Y); w1(Y); C2 ; C1 ; recuperável
S53 : r2(X); w2(X); r1(X); w1(X); r1(Y); C2 ; w1(Y); C1 ; recuperável
S54 : r2(X); w2(X); r1(X); w1(X); C2 ; r1(Y); w1(Y); C1 ; recuperável
S55 : r2(X); w2(X); r1(X); C2 ; w1(X); r1(Y); w1(Y); C1 ; recuperável
S56 : r2(X); w2(X); C2 ; r1(X); w1(X); r1(Y); w1(Y); C1 ; estrito