Isolamento: cada transação deve ser executada sem interferência de outras transações, mesmo durante execuções concorrentes.

durabilidade: mudanças provocadas por um commit devem ter sua persistência garantida, mesmo na ocorrência de falhas.

"manter a consistência é geralmente responsabilidade do programador"

níveis de isolamento:

0 : não permite leituras sujas
1 : " " atualizações perdidas
2 : " " leituras sujas nem atualizações perdidas
3 : " " " " " " " e provê leituras repetitivas

paramos aqui em 5/8 por causa fala Lucas. Sugestão índice do livro exercícios 6ª edição seal gratuito

21.4 Históricos de transações (schedules)

ordenação das operações das transações (S)

$T_1, T_2, \ldots, T_m$ = transações

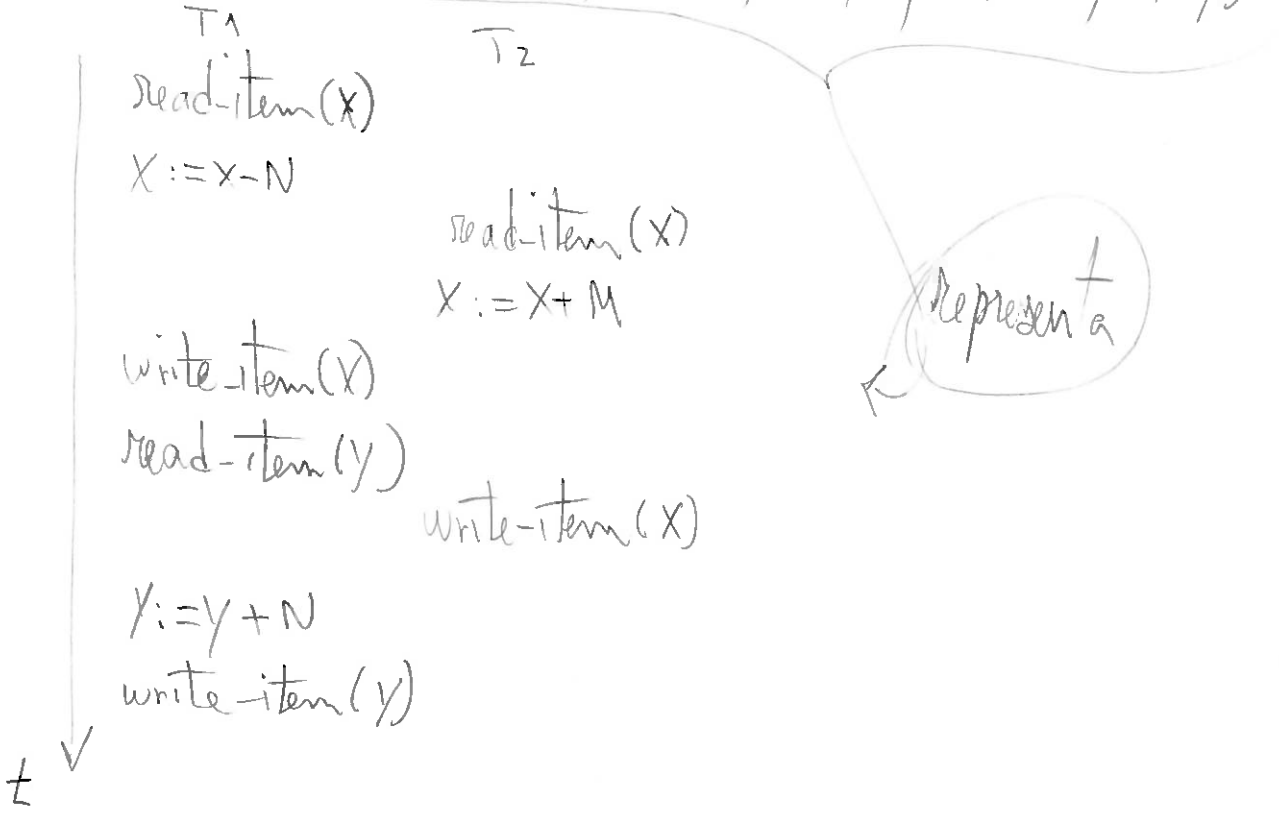
Operações de T_i em S precisam aparecer na mesma ordem em que ocorrem em T_i

notação abreviada: read_item = r, write_item = w, begin_transaction = b, end_transaction = e, commit = c, abort = a

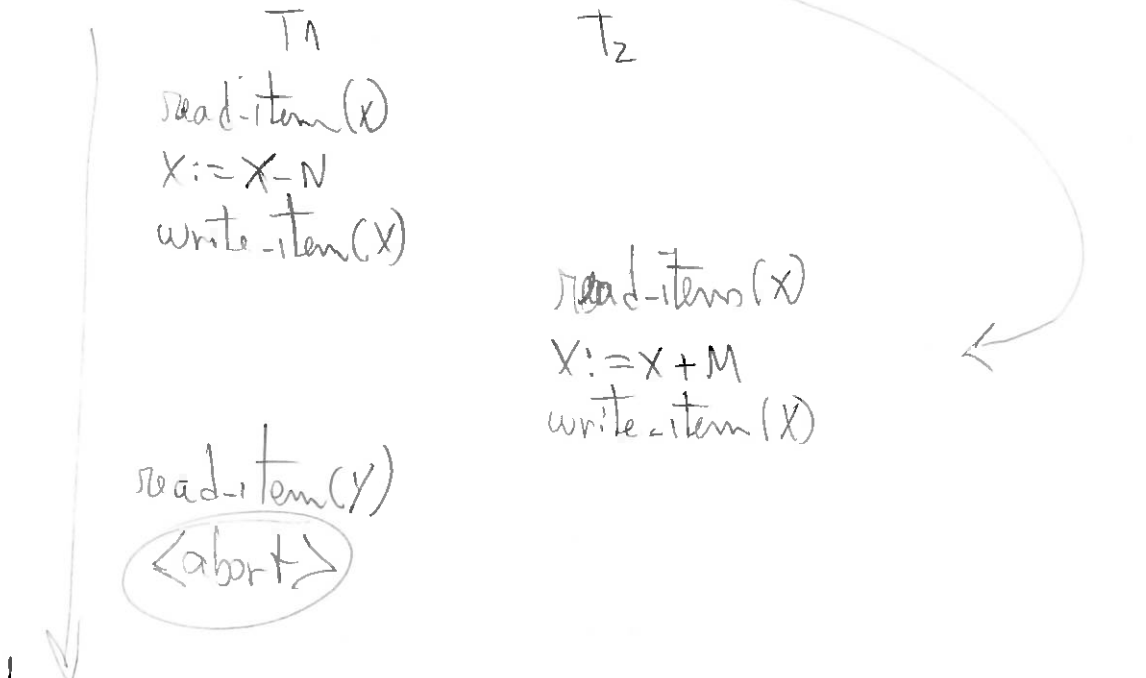
. id da transação aparece subscrito

exemplos:

Sa : $r_1(X); r_2(X); w_1(X); r_1(Y); w_2(X); w_1(Y)$



Sb : $r_1(X); w_1(X); r_2(X); w_2(X); r_1(Y); a_1;$



duas operações em um schedule estão em conflito: se...

1) elas pertencem a diferentes transações; e
2) elas acessam o mesmo item X ; e
3) pelo menos uma das operações é write-item(X)

ex: Sa e Sb

$r_1(X)$ e $w_2(X)$ estão em conflito
$r_2(X)$ e $w_1(X)$ "
$w_1(X)$ e $w_2(X)$ "
$r_1(X)$ e $r_2(X)$ não estão em conflito (leituras)
$w_2(X)$ e $w_1(Y)$ " (operam em dados distintos)
$r_1(X)$ e $w_1(X)$ " (mesma transação)

intuitivamente: duas operações estão em conflito se a mudança de ordem entre elas puder resultar em algo diferente

21.5 Serialização

schedules seriais: T_1 e T_2, ou T_2 e T_1, sem intercalação → inaceitáveis, na prática

com intercalação: é preciso verificar a corretude ...
(não-seriais)
↓
quando não é executado de forma serial

(travam outras transações enquanto operam I/O, podem ser muito longas, etc)

exemplo de schedule não-serial equivalente a um schedule serial:

T_1	T_2
r(X)	
X := X - N	
w(X)	
	r(X)
	X := X + M
	w(X)
r(Y)	
Y := Y + N	
w(Y)	

t ↓

Como determinar se um schedule não-serial é correto?

Um schedule S de n transações é serializável se for equivalente a algum schedule serial das mesmas n transações.

Como definir equivalência entre dois schedules?

equivalentes no resultado: pode ser acidental. Ex:

S_1	S_2
r(X)	r(X)
X := X + 10	X := X * 1.1
w(X)	w(X)

não podemos observar apenas o resultado final

São equivalentes apenas para X = 100

equivalência de conflito

def Dois schedules são equivalentes em conflito se a ordem de duas operações em conflito quaisquer for a mesma nos dois schedules.

ex:

S_1 ocorre $r_1(x)$ e $w_2(x)$
S_2 " $w_2(x)$ e $r_1(x)$
> não são equivalentes em conflito

def Um schedule S é serializável de conflito se ele for equivalente (em conflito) a algum schedule serial S'.

ex: schedule D (não-serial) é equivalente (em conflito) ao schedule A (serial)

⇒ schedule D é serializável de conflito

A

T_1	T_2
r(X)	
X := X − N	
w(X)	
r(Y)	
Y := Y + N	
w(Y)	
	r(X)
	X := X + M
	w(X)
r(X)	
X := X + M	
w(X)	
	r(Y)
	Y := Y + N
	w(Y)

D

T_1	T_2
r(X)	
X := X − N	
w(X)	
	r(X)
	X := X + M
	w(X)
r(X)	
X := X + M	
w(X)	
	r(Y)
	Y := Y + N
	w(Y)

[Quando são equivalentes, podemos "reordenar" as operações]

poderia ser movido para cá, pois $r_1(Y)$ e $w_1(Y)$ não estão em conflito com $r_2(X)$ e $w_2(X)$ (acessam dados diferentes)

21.5.2 Teste de serialização por conflito de um schedule

Construir um grafo de precedência (ou grafo de serialização)
- observar apenas operações read_item e write_item
- grafo direcionado

Algoritmo (21.1)

1. Criar um nó T_i para cada transação
2. Criar aresta $T_i \rightarrow T_j$ quando T_i executar um write_item(X) e depois T_j executar um read_item(X)
3. Criar aresta $T_i \rightarrow T_j$ quando T_i executar um read_item(X) e depois T_j executar um write_item(X)
4. Criar aresta $T_i \rightarrow T_j$ quando T_i executar um write_item(X) e depois T_j executar um write_item(X)
5. S é serializável se, e somente se, o grafo não tiver ciclos.

ex:

fig 21.5 e fig 21.7 (slides).

mais exemplos / exercícios de aplicação do algoritmo → desenhar ou impresso grande

fig 21.5 c)

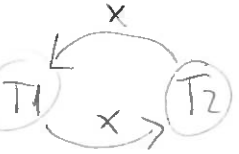


fig. 21.8 e fig 21.8 continuação

slides — desenhar ou impresso grande