Instituto de Informática
Banco de Dados
Prof. Plínio de Sá Leitão Júnior
Exercício 10

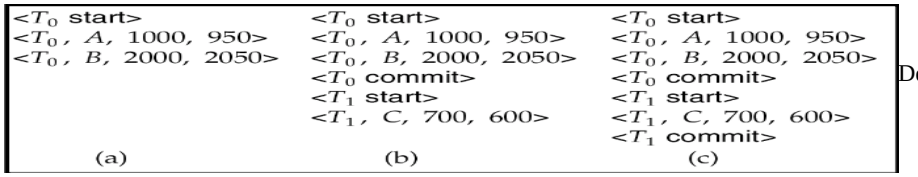
Considere o *log* abaixo (Transações T1, T2, T3 e T4).

<< start_transaction, T1 >>
<< read_item, T1, A >>
<< read_item, T1, D >>
<< write_item, T1, D, 20, 25 >>
<< commit, T1 >>
<< start_transaction, T2 >>
<< read_item, T2, B >>
<< write_item, T2, B, 12, 18 >>
<< start_transaction, T4 >>
<< read_item, T4, D >>
<< write_item, T4, D, 25, 15 >>
<< start_transaction, T3 >>
<< write_item, T3, C, 30, 40 >>
<< read_item, T4, A >>
<< write_item, T4, A, 30, 20 >>
<< commit, T4 >>
<< read_item, T2, D >>
<< write_item, T2, D, 15, 25 >>

Preencha a tabela abaixo, com respeito às transações que devem sofrer UNDO e REDO.

Falha <u>antes</u> de gravar no *log* a operação	**Algoritmo para recuperação / gerenciamento de cache do BD**	**Transações que sofrem UNDO**	**Transações que sofrem REDO**
<< write_item, T2, D, 15, 25 >>	Atualização adiada		
<< write_item, T2, D, 15, 25 >>	Atualização imediata, todas as atualizações são gravadas no banco de dados antes do *commit* da transação.		
<< write_item, T2, D, 15, 25 >>	Atualização imediata		
<< write_item, T2, D, 15, 25 >>	*No-steal*		
<< write_item, T2, D, 15, 25 >>	*Steal* , *no-force*		
<< write_item, T4, D, 25, 15 >>	Atualização adiada		
<< write_item, T4, D, 25, 15 >>	Atualização imediata, todas as atualizações são gravadas no banco de dados antes do *commit* da transação.		
<< write_item, T4, D, 25, 15 >>	Atualização imediata		
<< write_item, T4, D, 25, 15 >>	*No-steal*		
<< write_item, T4, D, 25, 15 >>	*Steal* , *no-force*		

Sejam: o algoritmo de Atualização Imediata para a recuperação de dados; e o conteúdo de 3 arquivos de *log*, conforme abaixo. Considere que houve uma falha no sistema após a gravação do último registro em cada arquivo de *log*.



Determine o valor dos itens de

dados A, B e C após o processo de recuperação. Se não for possível saber o valor de algum item de dado, responda com **???**.
(a) A = _____ , B = _____ , C = _____
(b) A = _____ , B = _____ , C = _____
(c) A = _____ , B = _____ , C = _____

Sobre a recuperação de transações concorrentes, some as falsas. RESPOSTA: _____
(01) todas as transações compartilham uma única cache do SGBD e um único *log*;
(02) um bloco na cache do SGBD pode ter itens de dados atualizados por uma ou mais transações;
(04) os registros de *log* de diferentes transações podem ser intercalados no *log*.

No grafo *wait-for*, um arco (Ti → Tj) significa que **Ti** precisa bloquear um dado, o qual está bloqueado por **Tj**. Na figura abaixo: S=bloqueio compartilhado, X=bloqueio exclusivo, R=leitura e W=escrita.

T1:	S(A), R(A),		S(B)	
T2:		X(B),W(B)		X(C)
T3:			S(C), R(C)	
T4:				X(B)

(a) construa o grafo *wait-for* para o escalonamento acima;

(b) o escalonamento resulta em *deadlock* ? Justifique a sua resposta.

Sobre as variações para o protocolo de bloqueio de duas fases (2PL), considere:
(01) 2PL básico; (02) 2PL rigoroso;
(04) 2PL estrito; (08) 2PL conservador.
Observe o número de cada variação 2PL e some os números daquelas que atendem às sentenças:
( ) obtém todos os bloqueios dentro do estado de transação;
( ) obtém todos os bloqueios fora do estado de transação;
( ) possui fase de crescimento em estado de transação;
( ) possui fase de encolhimento em estado de transação;
( ) é livre de *deadlock*;
( ) não é livre de *deadlock*;
( ) libera os bloqueios exclusivos somente no final da transação;
( ) libera os bloqueios compartilhados somente no final da transação.

Sobre as técnicas para a tratamento (prevenção/detecção) de *deadlock*, considere:
(01) esperar-ou-morrer (*wait-die*); (02) ferir-ou-esperar (*wound-wait*).
(04) sem-espera (*no-waiting*); (08) espera-cautelosa (*cautious-waiting*).
Observe o número associado a cada técnica e some os números daquelas que atendem às sentenças:
( ) sempre aborta a transação mais nova;
( ) uma transação pode ser abortada e reiniciada várias vezes;
( ) pode ocorrer que alguma transação entre em estado de espera;
( ) sempre aborta a transação mais velha;
( ) testa se ocorreu *deadlock*, antes de abortar alguma transação;
( ) baseada em *timestamp*;
( ) uma transação pode ser abortada desnecessariamente, pois não há *deadlock*;
( ) para haver o aborto de uma transação, primeiro detecta-se a ocorrência de *deadlock*;

Sobre recuperação após falhas, some as verdadeiras:
(01) checkpoint é um recurso que visa: aumentar a performance do sistema de banco de dados e reduzir o tempo do processo de recuperação;
(02) *fuzzy checkpoint* é um recurso para amenizar a degradação de performance pelo emprego de *checkpoint*;
(04) UNDO/REDO e NO-UNDO/REDO são algoritmos para atualização imediata;
(08) UNDO/-NO-REDO e NO-UNDO/REDO são algoritmos para atualização adiada;
(16) *force* e *steal* são políticas para atualização imediata;
(32) recuperação após falha é um processo 100% seguro.

Compare atualização imediata e atualização postergada, na perspectiva do processo de recuperação que usa o mecanismo de *checkpoint*.