UFSM – Universidade Federal de Santa Maria DLSC

Prova 2–Implementação de BD **Prof:** Sérgio Mergen **Data:**07/07/2016

Nome do aluno:_____ Matrícula do aluno: _____

1)A Figura 1 apresenta um *schedule* que é **equivalente em conflito** a pelo menos **um** *schedule* serial. Com base nisso, responda as seguintes questões:

T1	T2	T3
Write(A = 10)		
	read(F)	
Write (B = 30)		
Write(D = 20)		
Commit		
	Read (D = 10)	
	Commit	
		Write (B)
		Write (F = 30)
		Commit

Figura 1 – *Schedule* equivalente em conflito a um *schedule* serial

a) **(1,5 ponto)** Mostre o log gerado supondo que seja usada **modificação imediata (sem *checkpoints*)**. Além disso, indique quais **ações de recuperação** seriam necessárias supondo que tenha ocorrido um erro **após o *read*(D) de T3**. Considere que todos itens de dados iniciem com valor **0** ;

b) **(1,5 ponto)**Mudando a **primeira** instrução de **T2**é possível gerar um schedule que **não é equivalente em conflito** a nenhum schedule serial mas **é equivalente em visão** a um *schedule* serial. Que mudança é essa? Com qual *schedule* serial ocorreria equivalência?

2) (1,0 ponto) Desenhe um *schedule* que apresente o problema de ***rollback* em cascata** em que um erro em uma transação provoca erros em outras três transações. Acrescente os comandos de ***abort*** para deixar claro em que momento cada uma das transações está sendo cancelada.

Observação: Para responder as questões três e quatro é necessário simular como o gerenciador de *locks* controla a aquisição e liberação de acesso aos recursos requisitados pelas transações da Figura 2. Na simulação, considere os seguintes aspectos:

- As transações são atendidas pelo gerenciador em ciclos de execuções, onde em cada ciclo todas transações tem o direito de tentar executar uma instrução.
- Dentro de cada ciclo a ordem de execução é determinada pelo número da transação.
- O gerenciador utiliza o protocolo *two fase locking.*

3)(2 pontos) Considere que seja usada a estratégia de **detecção de *deadlock*** baseada em **ciclos**. Caso ocorra um *deadlock*, deverá ser eliminada a **transação que originou o ciclo**. Mostre o *schedule* que seria gerado. As esperas **não** precisam ser indicadas. Os *aborts* e *commits* sim.

T1: read(A); Write(C); Read(D).	T2: write(A); Write(C).	T3: read(C); Write(B); Write(A).	T4: read(A); Write(D).

Figura 2 – Instruções de quatro transações (T1, T2, T3, T4)

4) Considere que seja usado o nível de serialização **read_uncommitted**, em que uma leitura não espera por uma escrita e uma escrita não espera por uma leitura. Além disso, **nenhuma** estratégia de detecção/prevenção de *deadlock* é usada.

a) **(1,5 ponto)**Mostre o ***schedule*** que seria gerado usando o **protocolo de *lock* relaxado** que aceite essa flexibilização.

b) **(1,5 ponto)**Com base no schedule gerado, é possível que uma **falha** em uma transação leve a um ***rollback* em cascata**. Indique onde deve ocorrer essa falha (qual **transação** e **instrução**) e explique qual a **outra transação** que teria que ser revertida.

5) (1 ponto)A estratégia de **prevenção de *deadlock*** baseada em ***timeout*** aborta as transações que estejam esperando por um período de tempo pré-determinado (*timeout*). O desafio é definir um timeout adequado. Qual o **problema** em definir um *timeout* **curto**? E qual o **problema** em definir um *timeout* **longo**?