UFSM – Universidade Federal de Santa Maria DLSC

Prova 2 – Implementação de BD **Prof:** Sérgio Mergen

_____

Nome do aluno:_____ Matrícula do aluno: _____

1)(1,5 ponto) A Figura 1 apresenta um *schedule* que é **equivalente em conflito** a **um** schedule serial onde **T3 executa antes de T1**. Com base nisso, responda as seguintes questões:

a) Trocando a ordem de execução de duas instruções (de transações distintas, claro) é possível gerar um *schedule* equivalente em conflito a um *schedule* serial onde **T1 execute antes de T3**. Que troca é essa?

b) Quais *schedules* seriais seriam **equivalentes em conflito** ao *schedule* da figura após a mudança realizada pela letra a).

T1	T2	T3	T4	T5
		Read(A)		
	Write(F)			
				read(F)
Read(B)				
Read(C)				
			Read(C)	
		Read(C)		
			Read(F)	
	Write(B)			
				Write(C)
Write(A)				
Write(D)				

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

2) (1,5ponto) Desenhe um *schedule* que seja **serializável em visão** mas que **não** seja **serializável em conflito**.

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) Monte um *schedule* de execução usando a estratégia de detecção de *deadlock* baseada em **ciclos**. Caso ocorra um *deadlock*, deverá ser eliminada a transação que **originou** o ciclo. As esperas **não** precisam ser indicadas. Os ***aborts*** e ***commits*** sim.

4)(3 pontos) Para este exercício, considere que a última instrução de T3 (marcada em negrito) não exista. Ainda, considere que tenha ocorrido um *checkpoint* imediatamente antes do *commit* de T1. Todos os itens de dados tem 0 (zero) como valor inicial. Com base nisso, responda as seguintes questões:

a) Pela **ordem dos *commits***, qual *schedule* serial é equivalente ao *schedule* concorrente gerado a partir dessas transações.

b) Mostre o *log* de recuperação com ***checkpoints*** que seria gerado.

c) Caso ocorresse um erro **imediatamente antes** do *commit* de T2, quais ações de log teriam que ser realizadas?

T1:	T2:	T3:	T4:
read(A); Write(C = 3); Read(D).	write(A = 1); Write(C= 4).	read(C); Write(B = 5); **Write(A = 7).**	read(A); Write(C = 6).

Figura 2 – Instruções de quatro transações

5)(2 ponto) O nível de isolamento *ReadCommitted* permite a realização de escritas que seriam proibidas usando níveis de isolamentos mais restritivos. Considerando que o nível *ReadCommitted* esteja sendo usado, desenhe um *schedule* onde uma escrita desse tipo apareça. Mostre o problema que isso pode causar.