**2. Concurrency Control**

2.1. a) O escalonamento apresentado não seria permitido em Strict 2-Phase Locking, uma vez que a transação T1 efetua uma escrita de A antes da transação T2 efetuar também ela uma escrita sobre A. Assim, para que não houvesse conflitos, T1 teria que fazer lock(A) antes de fazer write(A) e fazer unlock(A) depois disso. Como T1 faz uma escrita de B depois de T2 fazer uma escrita de A, isto significa que T1 teria que fazer lock(B) antes de fazer write(B) e fazer unlock(B) depois disso. Concluindo, como T1 fazia unlock(A) depois de fazer write(A) e antes de fazer lock(B), os unlocks dessa transação não seriam todos efetuados no final da mesma e, como tal, não cumpririam os requisitos de Strict 2-Phase Locking.

b) Admitindo que T1 entrou no sistema antes de T2, que entrou no sistema antes de T3, então sabe-se que o valor de TS(T1)-1 < TS(T2)-2 < TS(T3)-3. Quando T3 efetua write(B) com sucesso, então W-timestamp(B) passa a ser 3. De seguida, quando T1 deseja efetuar write(B), como TS(T1), que é 1, é menor do que W-timestamp(B), que é 3, a operação é rejeitada e é efetuado um rollback sobre T1. Como tal, o escalonamento não é permitido no protocolo timestamp-based.

2.2 a) Este escalonamento também não seria permitido em Strict 2-Phase Locking, uma vez que um dos requisitos é que todos os unlocks sejam feitos no fim da transação. Como a transação T1 realiza unlock(A) antes de fazer lock-S(B), este requisito não é cumprido.

b) Admitindo que T1 entrou no sistema antes de T2, então sabe-se que o valor de TS(T1)-1 < TS(T2)-2. Quando T2 efetua write(B) com sucesso, então W-timestamp(B) passa a ser 2. De seguida, quando T1 deseja efetuar read(B), como TS(T1), que é 1, é menor ou igual do que W-timestamp(B), que é 2, a operação é rejeitada e é efetuado um rollback sobre T1. Como tal, o escalonamento não é permitido no protocolo timestamp-based.