Nome: Davi Cândido de Almeida_857859

Considere que o escalonamento S_a apresentado abaixo foi constituído a partir das transações T_1, T_2 e T_3 também apresentadas abaixo. Ressalta-se que, em um SGBDR diversas transações devem ser escalonadas para executarem simultaneamente, aumentando assim a concorrência e, consequentemente, diminuindo o tempo de processamento. No entanto, tal concorrência demanda a utilização de técnicas de controle de concorrência para garantir as propriedades de Atomicidade, Consistência, Isolamento e Durabilidade (ACID).

T_1 = r(x), r(y), w(x), r(z)

T_2 = r(z), r(x), r(y), w(z)

T_3 = r(y), r(z), w(y), r(x)

S_a = $r_3(y)$, $r_2(z)$, $r_1(x)$, $r_2(x)$, $r_3(z)$, $r_2(y)$, $w_3(y)$, $r_1(y)$, $w_2(z)$, $w_1(x)$, $r_3(x)$, $r_1(z)$

1. Considerando a técnica de controle de concorrência por bloqueio exclusivo (binário) com protocolo 2PL estrito e confirmação (*commit*) implícita (*commit* da transação ocorre logo após a última operação da transação no escalonamento), o escalonamento S_a possui *deadlock*? Entre quais transações? Qual o escalonamento que efetivamente será executado, considerando a técnica de resolução de *deadlock* que identifique o *deadlock* e mate a transação mais recente (aquela em que sua primeira operação se inicie depois da primeira operação das outras).

T2 tenta **w(z)**, mas T3 já fez **r(z)** → T2 espera T3 liberar z

T3 tenta **w(y)**, mas T2 já fez **r(y)** → T3 espera T2 liberar y

T1 tenta **w(x)**, mas T2 já fez **r(x)** → T1 espera T2 liberar x

T3 tenta **r(x)**, mas T1 já fez **r(x)** e está bloqueando escrita → T3 espera T1

Ciclo de espera (T1 → T2 → T3 → T1) ⇒ **deadlock entre T1, T2 e T3**

Timestamps (ordem de início no Sa):

r3(y) → T3 começou primeiro

r2(z) → T2

r1(x) → T1 → **mais recente**

Matar a transação recente, **logo T1 será abortada.**

Escalonamento efetivamente executado (**sem T1**):
Retirando as operações de T1 do Sa:

Sa' = r3(y), r2(z), r2(x), r3(z), r2(y), w3(y), w2(z), r3(x)

2. Considerando a técnica de controle de concorrência por bloqueio compartilhado (ternário) com protocolo 2PL estrito e confirmação (*commit*) implícita (*commit* da transação ocorre logo após a última operação da transação no escalonamento), o escalonamento S_a possui *deadlock*? Entre quais transações? Qual o escalonamento que efetivamente será executado, considerando a técnica de resolução de *deadlock* que identifique o *deadlock* e mate a transação mais recente (aquela em que sua primeira operação se inicie depois da primeira operação das outras)?

T2 faz rl(z), T3 faz rl(z), depois T2 tenta wl(z) e T3 tenta wl(y)

Deadlock ocorre também entre **T1, T2 e T3**

r3(y) → T3 começou primeiro

r2(z) → T2

r1(x) → T1 → **mais recente**

T1 será **abortada**

Escalonamento efetivamente executado:
Novamente, **se remove T1:**

Sa' = r3(y), r2(z), r2(x), r3(z), r2(y), w3(y), w2(z), r3(x)

3. Considerando a técnica de controle de concorrência por ordenação de registros de *timestamp*, qual o escalonamento que efetivamente será executado?

T3 será abortado logo o escalonamento efetivamente executado será:

Sa' = r2(z), r1(x), r2(x), r2(y), r1(y), w2(z), w1(x), r1(z)