Universidade Federal do Ceará
Centro de Ciências
Departamento de Computação
Sistemas Gerenciadores de Banco de Dados – CK0117 (T02)
Lista para composição da nota da 2ª. AP
Prof. Dr.-Ing. Angelo Brayner

Equipe:

José Douglas Gondim Soares - Matrícula: 485347
Fernanda Costa de Sousa – Matrícula: 485404

1)

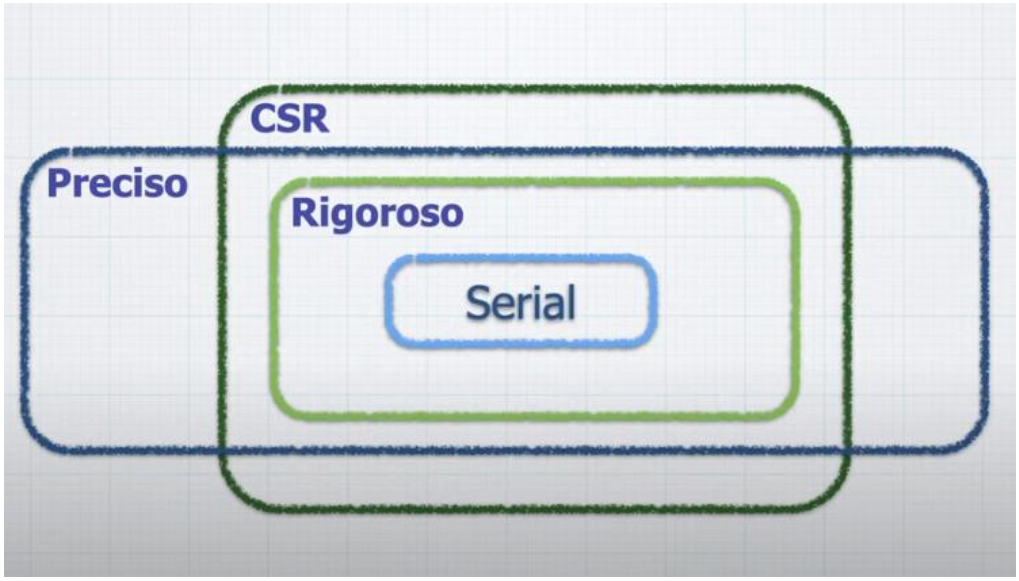

Fig. 1

a. CSR ∩ RG ≠ Ø

Isto é verdade, pois o conjunto RG está contido no conjunto CSR (Fig. 1), logo, a intersecção só poderia ser vazia se o conjunto RG ou (inclusivo) CSR fossem vazios. Também podemos argumentar que os escalonamentos pertencentes a RG são todos precisos e serializáveis por conflito. Assim, a intersecção de RG com CSR não pode ser vazia, pois existem escalonamentos comuns entre eles (todos em RG).

b. CSR - RG ≠ Ø

Isto é verdade, pois o conjunto RG está contido no conjunto CSR (Fig. 1), logo, ao retirarmos todos os escalonamentos RG (precisos e serializáveis por conflito) do conjunto CSR, ainda restarão aqueles escalonamentos que são serializáveis por conflito mas não são precisos.

2)

Após a falha, podemos identificar que T2 foi a transação ganhadora e T1 a perdedora. Vamos nos preocupar apenas com a transação perdedora T1. A última operação de escrita de T1 no arquivo de log foi a de LSN=30, que escreveu na página PM, mas a última operação escrita no último descarregamento da página PM foi a de LSN=5, o que implica que não precisamos realizar a operação UNDO na operação de LSN=30. Agora precisamos visitar a página PM pelo seu id e comparar o LSN da última operação escrita nela com LSN=5 e veremos que são iguais. Por causa disso precisaremos realizar um UNDO na operação de T1 com LSN=5.

Nesse ponto vamos olhar para a operação de LSN=25 e veremos que ela coincide com LSN do último descarregamento da Página PK. Isso indica que precisaremos fazer um UNDO nela também. Após essas operações, as mudanças que a transação T2 fez foram desfeitas.

3)

Primeiro o T1 irá bloquear a tupla de id_produto=1 para leitura. Em seguida, T2 irá bloquear a tupla de id_produto=2 para leitura. Nesse momento, T2 tentará bloquear a tabela Produtos para escrita, mas T1 já está bloqueando uma tupla de Produtos, o que faz T2 esperar pelo desbloqueio desse recurso. Após aproximadamente 10 segundos T1 tenta fazer um bloqueio de página (onde as tuplas de id_produto=1 e id_produto=2 estão), mas essa operação gera um Deadlock com T2, o que é resolvido abortando a transação T2(a mais recente). Após isso, o T1 termina sua execução e todas as operações de T2 são desfeitas.

4)

a) A operação *w3(u)* está solicitando um bloqueio de escrita sobre *u*, mas *u* já foi bloqueado por *T2* na operação *r2(u)*. Como a timestamp de *T3* é maior que a timestamp de *T2, T3* seria abortada, pois ela é a operação mais recente. Logo, o escalonamento S não poderia ser produzido

b) A operação *w3(u)* está solicitando um bloqueio de escrita sobre *u*, mas *u* já foi bloqueado por *T2* na operação *r2(u).* Como a timestamp de *T3* é maior que a timestamp de *T2, T3* não será abortada, será colocada para esperar. Logo, o escalonamento *S* pode ser produzido.

5)

O escalonamento S não é livre de Deadlock, pois considerando que os bloqueios só são liberados no momento do commit de uma transação, ainda haveriam bloqueios de leitura sobre as transações T3 e T1. O Deadlock é inevitável, pois o protocolo 2V2PL ocorre em duas fases, a de bloqueio e a de liberação e sabemos que durante a fase de bloqueio não podem haver liberações e na fase de liberações nenhuma operação pode ser bloqueada.