CK0117 - Sistemas de Bancos de Dados - 2019

Javam C. Machado, Eder C. M. Gomes, Davi B. Gomes

TRABALHO III - Transações e Controle de Concorrência

O trabalho consiste em desenvolver em C++ ou Java um gerenciador de transações de banco de dados com suporte para o controle de concorrência baseado em bloqueio em duas fases.

O programa receberá como entrada um arquivo com **n** histórias, uma história por linha, por exemplo:

BT(1),BT(2),R_2(x),R_1(y),W_1(y),R_2(y),W_1(z),CM(1),W_2(y),R_2(z),W_2(z),CM(2)

onde:

- BT(A): inicia uma transação com identificador de valor A
- R_1(x): transação 1 deseja ler o item x
- W_1(x): transação 1 deseja escrever o item x
- CM(A): transação A confirma suas operações

O programa terá duas classes principais a Tr_Manager e a Lock_Manager. A primeira gerencia uma matriz de transações onde cada linha representa uma transação contendo o identificador da transação, seu *timestamp* de início e seu estado, cujo valor pode ser *Active*, *Committed*, *Aborted*. A segunda classe mantém duas estruturas de dados (1) uma tabela de bloqueios sobre itens de dados, chamada Lock_Table e (2) uma lista de espera chamada Wait_Q, que mantém, para cada item de dado, uma lista FIFO de identificadores de transações que esperam pelo item. O identificador da transação deve estar associado ao modo de espera. Portanto supondo It_{250} um item de dado, T_{20} uma transação e LS um bloqueio no modo compartilhado, se T_{20} espera por It_{250} para fazer uma leitura, a Wait_Q de It_{250} terá um elemento $[T_{20}, LS]$. A Lock_Table deverá ser implementada como uma estrutura matricial onde cada linha da matriz mantém o identificador do item bloqueado, o Id da transação que obteve o bloqueio e seu respectivo modo de bloqueio, S para bloqueio compartilhado e X para bloqueio exclusivo. A interface do Lock_Manager deve implementar pelo menos as funções:

LS(Tr, D) : Insere um bloqueio no modo compartilhado na Lock_Table sobre o item D para a transação Tr se puder, caso contrário cria/atualiza a Wait_Q de D com a transação Tr.

LX(Tr, D) : Insere um bloqueio no modo exclusivo na Lock_Table sobre o item D para a transação Tr se puder, caso contrário cria/atualiza a Wait_Q de D com a transação Tr.

U(Tr, D) : Apaga o bloqueio da transação Tr sobre o item D na Lock_Table.

A classe Lock_Manager deve implementar o controle de *deadlock* baseado na estratégia de prevenção utilizando a técnica *Wait_Die*, fazendo uso do *timestamp* de cada transação encontrado na matriz gerenciada pelo Tr_Manager. Em caso de *deadlock* iminente quando

um bloqueio sobre um item de dado é requisitado ao Lock_Manager e a fim de evitar *deadlocks*, o Lock_Manager deve (1) colocar a transação solicitante na Wait_Q do objeto se for o caso ou (2) sinalizar um *Rollback* da transação solicitante para o Tr_Manager, e este atualiza assim o estado da transação para *Aborted*.

Data da entrega: sexta-feira - 31 de maio de 2019 no LEC.