UNIVALI – POLITÉCNICA – CIÊNCIA DA COMPUTAÇÃO

Disciplina: Banco de Dados I
Prof. Maurício Pasetto de Freitas, MSc.
Atividade Avaliativa

TRABALHO DESENVOLVIMENTO – M2

EM TRIOS – ENTREGA PARA O DIA 28/10 ATÉ ÀS 23:59 – **NÃO SERÃO ACEITAS ENTREGAS EM ATRASO**

Simulação de transações concorrentes com a detecção e resolução de Deadlocks.

O trabalho tem como objetivo implementar um programa que simula a execução de transações concorrentes utilizando threads, onde essas transações competem por recursos compartilhados e, em algum momento, provocam a ocorrência de um deadlock. O objetivo principal é compreender e implementar mecanismos de controle de concorrência em um cenário prático, aplicando técnicas de detecção e resolução de deadlock, como o algoritmo baseado em timestamp. Além disso, o programa exibirá, de forma clara, as interações das threads com os recursos compartilhados, como a obtenção e liberação de bloqueios, espera por recursos e a finalização em caso de deadlock, oferecendo uma visão prática do comportamento de sistemas concorrentes e dos desafios que envolvem a gestão de recursos compartilhados.

Você deverá criar um programa (com uso da linguagem de sua escolha), que atenda aos seguintes requisitos:

1. **Simulação de Transações Concorrentes**:
   - Utilize **threads** para simular transações que tentam acessar dois recursos compartilhados: os itens de dados **X** e **Y**.
2. **Controle de Acesso aos Recursos Compartilhados**:
   - Utilize uma estrutura semelhante a um **mutex** (bloqueio binário) para controlar o acesso exclusivo aos itens de dados **X** e **Y**. As threads devem tentar obter o bloqueio para acessar esses recursos de forma concorrente.
3. **Identificação de Deadlock**:
   - O programa deve incluir alguma técnica para identificar a ocorrência de **deadlock**, onde duas ou mais threads estão bloqueadas, cada uma esperando por um recurso que a outra detém, impossibilitando que qualquer uma avance.
4. **Execução com Múltiplas Threads**:
   - Dispare **5 threads** que concorrem pelos recursos **X** e **Y**. As threads devem ser capazes de tentar acessar os recursos em diferentes momentos.
5. **Técnica de Resolução de Deadlock**:
   - Ao detectar um deadlock, utilize a técnica de **timestamp** com o uso de **wait-die** ou **wound-wait** para resolver o problema de deadlock e permitir que as threads finalizem corretamente.
6. **Solicitações de Bloqueio Aleatórias**:

- As threads devem tentar acessar os recursos em tempos diferentes, utilizando tempos de espera randômicos para simular a solicitação de bloqueio em momentos aleatórios, aumentando as chances de ocorrer deadlock.

7. **Mensagens no Terminal**:
   - O programa deve exibir mensagens no terminal para acompanhar o estado das threads. Essas mensagens devem incluir:
     - Quando uma thread T(x) entra em execução.
     - Quando uma thread T(x) obtém o bloqueio de um recurso.
     - Quando uma thread T(x) está esperando por um recurso.
     - Quando uma thread T(x) libera (unlock) um recurso.
     - Quando uma thread T(x) finaliza sua execução.
     - Quando uma thread T(x) é finalizada em virtude de deadlock detectado.
8. **Ferramentas Sugeridas:**
   - **Python**: Fácil de implementar threads e locks, e há várias bibliotecas úteis para manipulação de grafos (ex.: networkx).
   - **Java**: Possui excelente suporte para programação concorrente, com bibliotecas como Thread e ReentrantLock.
9. **Tempo de Acesso Randômico**:
   - Para simular cenários concorrentes, as threads devem tentar adquirir recursos em tempos diferentes, gerados aleatoriamente. Isso ajudará a provocar situações de deadlock.

Critérios de Avaliação:

- Implementação correta do controle de acesso aos recursos utilizando bloqueio binário.
- Detecção e resolução eficiente do deadlock utilizando técnicas adequadas de timestamp.
- Gerenciamento adequado das threads e sincronização entre elas.
- Clareza e precisão das mensagens exibidas no terminal para acompanhar o estado das threads e recursos.
- Utilização adequada de tempos randômicos para simular a concorrência.
- Clareza na explicação relativa ao funcionamento do código.

Entrega:

- O código-fonte do programa deve ser entregue, juntamente com um arquivo **README** explicando como executar o programa.
- Gravar um vídeo, apresentando o código e como a implementação foi realizada. O link do vídeo em modo público deve ser compartilhado no momento da entrega do trabalho no Material Didático.

- Na aula do dia 14/10, quatro grupos para apresentarem o trabalho para a turma. Critério de escolha dos grupos será definido pelo professor.

ATENÇÃO: Deverá ser postado no Material Didático (Intranet → Portal do Aluno) o foi solicitado até às **23:59 do dia 28/10**.