Banco de Dados

Gerenciamento de Transações - Continuação

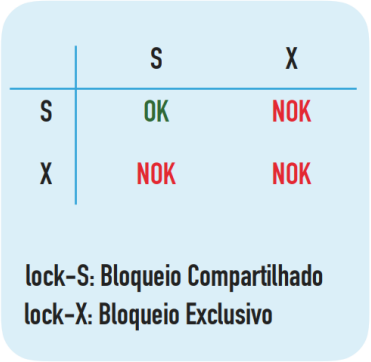
- Escalonamento Serializável por Visão
  - Dois escalonamentos S e S' são equivalentes por visão se:
    - Se a transação T_i lê o valor inicial do item de dado X em S, ele também lê o valor inicial de X em S'
    - Se T_i lê o valor de X escrito por T_j em S, o mesmo acontece em S'
    - Se T_i é o último a gravar X em S, ele também é o último a gravar X em S'
  - Um escalonamento é serializável por visão se ele é equivalente por visão a um escalonamento serial.
  - Equivalente ao escalonamento serial < T1,T2,T3 >
  - Um escalonamento serializável por conflito sempre é serializável por visão
  - Um escalonamento serializável por visão nem sempre é serializável por conflito
  - Definição alternativa
    - Considere que existe uma transação T_0 que escreve todos os itens de dados
    - Considere que existe uma transação T_f que lê todos os itens de dados
    - Condição para dois escalonamentos S e S' serem equivalentes por visão: Se T_i lê o valor de X escrito por T_j em S, o mesmo acontece em S'
  - Algoritmo

    1. Crie um vértice para T_0, T_f e para cada transação no escalonamento
    2. Se a transação T_j lê um item de dado X escrito pela transação T_i e $ineqj$, crie uma aresta de T_i para T_j
    3. Se uma transação T_k ($k \neq i$ e $k \neq j$) também escreve o mesmo item de dado X, crie um **par** de arestas de T_k para T_i e de T_k para T_i
       - Se T_i for T_0 não crie a aresta de T_k para T_0
       - Se T_j for T_f não crie a aresta de T_f para T_k
    4. Considere os grafos gerados considerando **apenas um elemento de cada par de arestas** do passo anterior. Se em todos os grafos houver um ciclo, então o escalonamento não é serializável por visão; caso contrário, o escalonamento é equivalente a um escalonamento serial

- Escalonamento Serial Equivalente
  - Ordenação Topológica do grafo acíclico direcionado (DAG): ordenação linear dos vértices, na qual cada vértice aparece antes de seus descendentes
  - Algoritmo de Ordenação Topológica = Algoritmo de Kahn

```
L:= vazio;
S:= vértices sem arestas incidentes;
enquanto S não estiver vazio faça
      remova um vértice n de S
      insira n em L
      para cada vértice m com uma aresta e = (n, m) faça
            remova a aresta e do grafo
            se m não tiver mais arestas incidentes então
                  insira m em S
se o grafo não estiver vazio então
      o grafo contém pelo menos um ciclo
senão
      returne L
```

- Controle de Concorrência
  - Assegura o isolamento de execução concorrente
  - Garante efeito da serialidade utilizando bloqueios (evita a necessidade de conhecer previamente o agendamento)
  - Métodos de bloqueios: Compartilhado, 2PL, 2PL Estrito
  - Bloqueio Compartilhado
    - Compartilhado: Se T_i obteve bloqueio compartilhado S sobre Q, T_i pode ler mas não escrever: lock-s(A)
    - Exclusivo: Se T_i obteve bloqueio exclusivo X sobre Q, T_i pode ler e escrever: lock-x(A)
    - T_i desbloqueia Q com unlock(A)
    - Matriz de bloqueios



    - Ainda podem ocorrer problemas no banco
  - Bloqueio de 2 fases
    - Fase 1 (expansão): pode obter bloqueios, mas não pode liberar qualquer bloqueio
    - Fase 2 (encolhimento): pode liberar bloqueios, mas não pode obter novos bloqueios
    - As 2 fases mantém a serialidade do agendamento negando pedidos de recursos com bloqueios exclusivos
    - Ainda podem ocorrer impasses
  - Prevenção de impasse
    - Algoritmos de prevenção de impasse usam selo temporal (timestamp) para testar a prioridade das transações.
    - Esperar-Morrer: Quanto mais antiga a transação, maior probabilidade de espera
    - Ferir-Esperar: Transação antiga nunca espera