João dos Santos Neto

1. O que é o protocolo de bloqueio em duas fases? Como ele garante a serialização?

O protocolo de bloqueio de duas fases é um mecanismo usado em sistemas de gerenciamento de banco de dados para garantir a sequência e a consistência de transações simultâneas. Ele garante a serialização seguindo as restrições impostas pelas fases de crescimento e encerramento:

A fase de crescimento garante que uma transação não possa adquirir novos bloqueios após liberar algum. Isso ajuda a evitar situações em que duas transações adquirem bloqueios simultaneamente e, em seguida, entram em um impasse (deadlock), onde cada uma está esperando pelo bloqueio que a outra possui.

A fase de encerramento garante que uma transação não possa adquirir mais bloqueios após começar a liberá-los. Isso evita situações em que uma transação tenta liberar alguns bloqueios e, enquanto isso, adquire outros, o que poderia levar a problemas de serialização.

2. Quais são algumas variações do protocolo de bloqueio em duas fases? Qual dessas variações é o mais viável. (Informação não contida nos slides - leia o conteúdo sobre bloqueio em duas fases no capítulo “Técnicas de Controle de Concorrência” Elsmari e Navathe ou outras fontes).

As variações do protocolo de bloqueio em duas fases consistem em:

Strict Two-Phase Locking: Nessa variação, as transações mantêm todos os bloqueios adquiridos até o final da transação, sem liberar nenhum bloqueio antes.

Rigorous Two-Phase Locking: Essa é uma extensão do 2PL que impõe um conjunto mais rigoroso de restrições nas fases de crescimento e encerramento.

Conservative Two-Phase Locking: Essa variação permite que as transações liberem alguns bloqueios adquiridos antes de terminarem, mas impõe restrições rigorosas sobre a ordem em que os bloqueios são liberados para evitar anomalias de consistência.

Preclaiming: Essa é uma técnica que permite que as transações adquiram todos os bloqueios necessários no início, antes de começar sua execução.

Deadlock Avoidance: Não é uma variação direta do 2PL, mas uma técnica usada em conjunto para evitar impasses. Algoritmos de prevenção de impasses, como o algoritmo do banqueiro, podem ser aplicados para garantir que as transações não entrem em situações de impasse.

3. Discuta os problemas de deadlock e inanição e as diferentes técnicas para lidar com esses problemas.

Deadlock: O deadlock ocorre quando várias transações estão presas esperando recursos que nunca serão liberados, causando uma paralisação no sistema.

Inanição: A inanição acontece quando certas transações são consistentemente preteridas em relação a outras, impedindo seu progresso e afetando o desempenho do sistema.

Técnicas para Lidar com Deadlock e Inanição:
Detecção e Resolução de Deadlock: Identificação e eliminação de impasses; cancelamento seguro de transações em impasses.

Prevenção de Deadlock: Algoritmo do banqueiro; limitação de recursos para evitar impasses.

Evitação de Deadlock: Uso de informações sobre recursos para evitar impasses.
Prioridades e Inanição: Atribuição de prioridades para evitar inanição.

Temporização e Reatribuição de Recursos: Definição de limites de tempo para uso de recursos.

Mecanismos de Desbloqueio e Espera: Uso de timeouts para liberar recursos.

Controle de Concorrência Baseado em Versões: Uso de versões isoladas dos dados para reduzir deadlock e inanição.

4. Compare os bloqueios binários com os bloqueios exclusivo/compartilhado. Por que esse último tipo de bloqueio é preferível?

Os bloqueios binários (leitura/gravação) e os bloqueios exclusivo/compartilhado são usados para controlar o acesso concorrente a recursos em sistemas de gerenciamento de bancos de dados. Os bloqueios binários permitem leituras simultâneas, mas exigem exclusividade para gravações. Os bloqueios exclusivo/compartilhado permitem leituras compartilhadas e exclusividade para gravações.

O bloqueio exclusivo/compartilhado é geralmente preferível devido a:
- Melhor isolamento entre leituras e gravações, evitando anomalias de leitura sujas.
- Maior flexibilidade para adaptar-se a cenários equilibrados de leitura e escrita.
- Maior consistência e controle sobre a concorrência, o que é crucial para a integridade dos dados.

5. Descreva o princípio básico dos protocolos esperar-morrer e ferir-esperar para a prevenção de deadlock.

Os protocolos "esperar-morrer" e "ferir-esperar" são usados para prevenir impasses em sistemas de gerenciamento de bancos de dados.
No protocolo esperar-morrer, transações mais novas podem ser interrompidas se tentarem bloquear recursos já alocados para transações mais antigas.
No protocolo ferir-esperar, transações mais antigas são interrompidas se tentarem bloquear recursos já alocados para transações mais novas. Ambos os protocolos visam evitar impasses, permitindo interrupções controladas de transações para manter a continuidade das operações.

6. O que é um rótulo de tempo (timestamp)? Como ele pode ser gerado sistema?

Um rótulo de tempo (timestamp) é um valor numérico associado a transações em um sistema de gerenciamento de bancos de dados. Ele indica o momento de início ou a ordem relativa das transações.

Os rótulos de tempo são usados para evitar anomalias de consistência, tomar decisões de acesso a recursos e prevenir impasses. Eles podem ser gerados por relógios do sistema, contadores incrementais, carimbos de data e hora, relógios lógicos ou relógios vetoriais.

7. Resuma o método de ordenação por timestamp para o controle de concorrência. Descreva duas variações desse método (Informação não contida nos slides).

O método de ordenação por timestamp é uma técnica de controle de concorrência em bancos de dados. As transações recebem rótulos de tempo que indicam sua ordem relativa. As transações são executadas seguindo a ordem dos rótulos de tempo. Duas variações incluem o protocolo de timestamp de emissão, onde o timestamp do recurso é verificado antes da escrita, e o protocolo ferir-esperar, onde transações mais novas podem interromper as mais antigas para obter recursos. O método evita impasses e anomalias de consistência, mas pode resultar em inanição. O uso de timeouts e reatribuição de recursos pode mitigar a inanição.