Transações Concorrentes

Um SGBD permite que várias sessões acessem um mesmo banco de dados ao mesmo tempo. O acesso simultâneo leva a execução de transações concorrentes, que leem e/ou gravam os mesmos dados ao mesmo tempo.

O acesso concorrente pode levar a fenômenos indesejáveis:

dirty read (leitura suja) - A transação lê dados escritos por uma transação simultânea não efetivada (uncommitted).

nonrepeatable read (leitura que não pode ser repetida) - A transação lê novamente dados lidos anteriormente, e descobre que os dados foram alterados por outra transação (que os efetivou após ter sido feita a leitura anterior).

phantom read (leitura fantasma) - A transação executa uma segunda vez uma consulta que retorna um conjunto de linhas que satisfazem uma determinada condição de procura, e descobre que o conjunto de linhas que satisfazem a condição é diferente por causa de uma outra transação efetivada recentemente.

Transações Concorrentes

Para evitar esses fenômenos, é definido um nível de isolamento entre as transações e o SGBD deve impedir esses fenômenos de acordo com o nível de isolamento selecionado:

Nível de Isolamento	Dirty read	Nonrepeatable read	Phanton read
READ UNCOMMITED	Possível	Possível	Possível
READ COMMITED	Impossível	Possível	Possível
REPEATABLE READ	Impossível	Impossível	Possível
SERIALIZABLE	Impossível	Impossível	Impossível

Plano de Execução

Quando transações são executadas concorrentemente, de maneira intercalada, a ordem de execução das operações das várias transações é conhecida como plano de execução (ou histórico) das transações.

Duas operações em um plano são ditas em conflito se satisfizerem todas as condições a seguir:

- **Pertencem a diferentes transações**
- **Acessam a um mesmo item de dado**
- **Ao menos uma das operações é de gravação do item**

Um SGBD deve procurar que as transações sejam ordenadas em um plano de execução restaurável e serializável.

Plano de Execução

Um plano de execução restaurável é um plano no qual uma vez efetivada um transação T, nunca seja necessário cancelar essa transação.

Um plano de execução é serializável se ele for equivalente a execução serial (sem operações intercaladas) de cada transação do plano.

Controle de Concorrência

Existem vários mecanismos utilizados para implementar o isolamento de transações. Algumas das principais técnicas são:

- **Controle de Concorrência por Bloqueio**
- **Controle de Concorrência Baseado em Ordenação por Timestamp**
- **Controle de Concorrência de Multiversão**
- **Controle de Concorrência de Validação (Otimista)**

Controle de Concorrência por Bloqueio

Para controlar a concorrência entre as transações, o SGBD pode utilizar o mecanismo de bloqueio.

Um bloqueio é uma variável associada a um item de dados que descreve a condição do item em relação as possíveis operações que podem ser aplicadas a ele.

Uma transação T2 não pode acessar um item de dado que esteja sendo utilizado pela transação T1. Uma transação deve obter o bloqueio de um item antes de acessá-lo. Os bloqueios obtidos por uma transação devem ser liberados quando a transação é concluída.

Controle de Concorrência por Bloqueio

O tipo de item de dados sobre o qual é feito o bloqueio define a granularidade do bloqueio.

Bloqueio do banco de dados – o banco inteiro é bloqueado, evitando assim que outra transação utilize qualquer outro item de dados do banco.

Bloqueio de tabela – a tabela inteira é bloqueada, evitando que qualquer outra transação acesse qualquer registro da tabela.

Bloqueio de página – uma página de disco inteira é bloqueada, a leitura e gravação dos discos é sempre feita por página, uma tabela pode ter várias páginas de disco e uma página de disco pode ter vários registros da tabela, o bloqueio de página permite o bloqueio de vários registros como um único item.

Bloqueio de registro – um registro inteiro é bloqueado, o bloqueio por registro e mais flexível do que o bloqueio de página, mas exige mais processamento do SGDB.

Bloqueio de campo – permite que diferentes transações acessem o mesmo registro desde que atualizem diferentes atributos do registro, usualmente não é implementado.

Controle de Concorrência por Bloqueio

Existem diferentes tipos de bloqueio que podem ser utilizados no controle de concorrência.

Bloqueios binários – um bloqueio binário pode ter dois estados, bloqueado ou desbloqueado.

Bloqueios compartilhados/exclusivos – uma transação pode solicitar dois tipos de bloqueio sobre um item:

- **bloqueio compartilhado - permite que outras transações também obtenham bloqueio compartilhado sobre o mesmo item, é usado quando uma transação pretende ler um item sem permitir que outra transação altere esse item.**
- **bloqueio exclusivo - não permite que nenhuma outra transação obtenha qualquer tipo de bloqueio sobre o item, é utilizado quando uma transação pretende alterar um item.**

Alguns algoritmos de controle de concorrência permitem a conversão de bloqueios, com a transformação de bloqueio compartilhado em exclusivo (promoção) ou a transformação de bloqueio exclusivo em compartilhado (rebaixamento).

Bloqueio de Duas Fases

Uma forma de garantir a serialização das transações é o bloqueio de duas fases.

A execução de uma transação pode ser dividida em duas fases:

Fase de crescimento (ou expansão), durante a qual o bloqueio de novos itens pode ser adquiridos, mas não podem ser liberados.

Fase de encolhimento, durante a qual bloqueios existentes podem ser liberados, mas novos bloqueios não podem ser adquiridos.

Desta forma, nenhum bloqueio pode ser liberado antes que todos os bloqueios sejam obtidos e nenhum bloqueio pode ser feito após a liberação de algum bloqueio.

Se for permitida a conversão de bloqueios, a promoção de bloqueios deve ser feita durante a fase de expansão e o rebaixamento deve ser feito durante a fase de encolhimento.

Bloqueio de Duas Fases

Existem algumas variações do bloqueio de duas fases sendo o mais comum o 2PL estrito, onde a transação não libera nenhum bloqueio exclusivo até que ela efetive ou aborte.

Deadlock

Um dos problemas do controle por bloqueio e a possibilidade da ocorrência de deadlock.

O deadlock ocorre quando cada transação em um conjunto de duas ou mais transações espera por algum item que esteja bloqueado por alguma outra transação T' no conjunto. Portanto, cada transação do conjunto esta em uma fila de espera, aguardando por uma das outras transações do conjunto liberar o bloqueio em um item.

O SGDB pode utilizar um protocolo de prevenção de deadlocks ou mecanismos para resolver situações de deadlock.

Deadlock

Os protocolos de prevenção de deadlocks podem ser baseados em timestamp, onde cada T transação recebe uma marca de tempo (timestamp) TS(T). Uma transação T_1 que inicie antes de uma transaçao T_2 recebe um timestamp menor $TS(T_1) < TS(T_2)$.

Caso uma transção T_i tente bloquear um objeto bloqueado por uma transação T_j, o deadlock pode ser evitado com base no esquema esperar-morrer ou ferir-esperar.

Esperar-Morrer – Se $TS(T_i) < TS(T_j)$, T_i pode esperar T_j liberar o bloqueio. Se $TS(T_i) > TS(T_j)$, T_i é abortada e reinicia posteriormente com o mesmo timestamp.

Ferir-Esperar – Se $TS(T_i) < TS(T_j)$, T_i aborta T_j. Se $TS(T_i) > TS(T_j)$, T_i pode esperar T_j liberar o bloqueio.

Deadlock

Também existem protocolos de prevenção de deadlock sem o uso de timestamp.

No algoritmo no waiting, se uma transação não estiver apta para a obter um bloqueio, ela será imediatamente abortada e, então, reiniciada com algum tempo de atraso.

No algoritmo cautios waiting, se T_i tenta bloquear um item bloqueado por uma transação T_j, se T_j não esta bloquada, T_i espera, se T_j esta bloqueada, T_i aborta.

Deadlock

O SGDB pode utilizar mecanismos para resolver situações de deadlock.

Uma técnica para detectar uma situação de deadlock é a manutenção de um grafo wait-for, onde cada transação T_i aguardando por um bloqueio é um nó do grafo e uma ligação direcional é criada para cada transação T_j que T_i esteja aguardando, se o grafo apresentar um ciclo, o sistema esta em deadlock e será necessário selecionar transações a abortar.

Outro esquema simples para resolver situações de deadlock é o uso de timeouts, transações que estejam aguardando por um tempo maior que o timeout definido são abortadas, é um esquema simples e com baixa sobrecarga para o SGBD.

Starvation

Outro problema do controle de concorrência por bloqueio é a starvation, quando uma transação fica aguardando indefinidamente. Esse problema pode ocorrer quando o esquema de espera por bloqueio de itens for parcial, dando prioridade a algumas transações sobre outras.

Para evitar esse problema, o esquema de espera deve ser imparcial, atendendo preferencialmente transações que aguardam a mais tempo.

Controle de Concorrência por Timestamp

Uma forma de garantir a serialização das transações sem utilizar bloqueio é atribuindo um timestamp e garantir que se a ação a_i da transação T_i entrar em conflito com a ação a_j da transação T_j, a_i ocorrerá antes de a_j, se $TS(T_i) < TS(T_j)$. Se alguma ação violar essa ordem, a transação será cancelada e reiniciada.

Para isso, cada objeto tem um timestamp de leitura e de gravação.

A permissão para que uma transação leia ou grave um objeto do banco de dados é feita por regras que garantam que o timestamp da transação não entre em conflito o timestamp de leitura ou de gravação do objeto.

Controle de Concorrência de Multiversão

O controle de multiversão tem por objetivo que uma transação nunca tenha que esperar para ler um objeto.

Cada gravação de um objeto gera uma nova versão do valor desse objeto e nas operações de leitura, uma transação T_i lera a versão mais recente cuja marca de tempo preceda $TS(T_i)$.

Se a transação T_i, tentar gravar um item, devemos garantir que o item não tenha sido lido por uma transação T_j , tal que $TS(T_i) < TS(T_j)$.

Esse mecanismo utiliza mais memória para armazenar diferentes versões do item, mas operações de leitura que seriam rejeitadas em outras técnicas podem ser realizadas com versões anteriores do objeto.

Controle de Concorrência Otimista

Considerando que na maior parte das transações não há conflito com as demais transações, as técnicas de controle de concorrência otimista não realizam a verificação de conflitos durante a execução da transação, mas apenas no final da transação para verificar se a mesma pode ser efetivada.

As transações tem três fases:

Leitura – onde a transação é executada e as alterações são gravadas em um workplace privado

Validação – verifica se a transação entra em conflito com as demais transações, cancelando a transação se necessário

Gravação – se a transação foi devidamente validada, as alterações feitas pela transação são copiadas para o banco de dados

As técnicas de controle de concorrência otimista apresentam melhor desempenho em ambientes com poucos conflitos pois diminuem as verificações realizadas. Mas em ambientes com muito conflitos, o cancelamento de muitas transações prejudica o desempenho do sistema.