LISTA 005 – MATHEUS HENRIQUE BUTKOSKI SILVA

TADS – UNIVEL

1.

Transação pode ser definido como um conjunto de uma ou mais operações que formam uma unidade lógica de processamento a ser executada. Existem quatro tipos de operações: **Inserção, exclusão, modificação, recuperação.**

Estas devem sempre possuir a garantia de serem executadas da maneira correta pelo banco de dados. As operações podem ser embutidas juntamente com o programa em que o banco de dados funciona, ou serem especificadas através de uma linguagem de consulta como o SQL.

Ao se utilizar o SQL, as transações serão definidas por declarações no formato:

BEGIN TRANSACTION
.
.
.
END TRANSACTION

As operações que forem descritas entre os dois comandos formam uma transação, que será vista pelo usuário como uma única instrução. Quando se trata de transações, dois tipos podem ser definidos:

Transação de leitura: a transação não atualiza o banco de dados e retorna os dados solicitados.

Transação de leitura-gravação: a transação atualiza o banco de dados e retorna os dados solicitados.

2.

ACID define os quatro pilares que uma transação precisa possuir, para que a

Integridade do banco de dados seja garantida:

Atomicidade: É a garantia de que a transação será feita totalmente ou não será feita. Dessa forma a transação nunca será feita “pela metade”. Se por ventura uma operação da transação falhar, consequentemente, toda a transação falhará.

Consistência: Se a transação for executada completamente sem interferência de outras, deve levar o banco de dados de um estado consistente para outro.

Isolamento: É a capacidade de uma transação não interferir no andamento da outra, garantindo que sejam feitas individualmente, visto que diversas transações ocorrem simultaneamente

Durabilidade: É a preservação dos dados das operações, após elas terem sido realizadas. Dessa forma, mesmo que falhas graves aconteçam, essas mudanças não devem ser perdidas.

Se as quatro propriedades forem seguidas corretamente será garantido que as transações serão realizadas de forma confiável.

3.

No bloqueio multiversão existem três modos de bloqueio para um item: leitura, gravação e certificação. Dessa forma, o estado de LOCK(X) para um item X pode ser um dentre os seguintes: bloqueado para leitura, bloqueado para gravação, bloqueado para certificação ou desbloqueado. No esquema de bloqueio padrão, apenas com bloqueios de leitura e gravação um bloqueio de gravação é um bloqueio exclusivo.

A ideia do bloqueio multiversão é permitir que outras transações T leiam um item X enquanto uma única transação T mantém um bloqueio de gravação sobre X.

4.

Um deadlock ocorre quando uma transação, em um conjunto de duas ou mais transações, está esperando por algum item bloqueado por alguma outra transação no conjunto. Criando assim uma fila, que espera até que uma das transações do conjunto libere o bloqueio.

4.1

Para impedir um deadlock é necessário um protocolo de prevenção de deadlock, que requer que cada transação bloqueie todos os itens que precisar com antecedência.

Outra forma também envolve ordenar todos os itens do banco de dados e garantir que uma transação que precisa de um ou mais itens o bloqueará de acordo com a ordem estabelecida.

Existem dois esquemas que impedem o deadlock, eles são chamados de:

Wait-die- uma transação A que iniciou no instante 1 e uma transação B que iniciou no instante 2, logo, o *timestamp* de A B, então, a transação A será abortada e será reiniciada posteriormente com o mesmo rótulo de tempo.

Wound-wait - uma transação A que iniciou no instante 1 e uma transação B que iniciou no instante 2, logo, *o timestamp* de A B, então, a transação A tem permissão para esperar.

Em ambos os esquemas, a transação mais nova acaba sendo abortada pela transação que é mais velha, se elas estiverem envolvidas em um deadlock.

4.2

Técnica em que o sistema verifica se um deadlock realmente existe. Caso haja pouca interferência entre as transações essa técnica pode ser atraente. Isso pode acontecer se as transações forem curtas e cada uma bloquear apenas alguns itens, ou se a carga da transação for leve, mas se a transação for longa e de demandar muitos itens poderá ser vantajoso usar um esquema de prevenção de deadlock.

Uma forma simples para que um deadlock seja detectado é através da construção de um grafo de espera por parte do sistema, em que um nó será criado para cada transação que está sendo executada.

4.3

Timeout é um esquema para lidar com deadlocks, e suas vantagens são a praticidade e simplicidade. Em um Timeout, o sistema irá definir um tempo-limite, que quando uma

transação espera por um tempo superior ao definido, essa operação será abortada, pois o sistema entenderá que um deadlock pode ser criado.

5.

Um *timestamp* é um identificador criado pelo banco de dados, a fim de identificar uma transação. Estes são atribuídos conforme a ordem em que as transações foram submetidas, assim pode-se relacionar um *timestamp* ao horário de início de uma transação. Além disso as técnicas baseadas em *timestamps* não utilizam bloqueios, dessa forma não haverá deadlocks.

O algoritmo precisa garantir que a ordem em que o item está sendo acessado não está violando a ordem do rótulo de tempo, e para isso o algoritmo associa a cada item X do banco de dados dois valores de rótulo de tempo (TS), sendo:

read_TS(X) - rótulo de tempo de leitura;

write_TS(X) - rótulo de tempo de gravação;

Esses timestamps são atualizados sempre que uma nova instrução read_TS(X) ou write_TS(X) é executada. Sempre que uma transação read e write é desfeita pelo esquema de controle de concorrência, esta transação recebe um novo timestamp e é reiniciada.

O rótulo de tempo (TO) compara o rótulo de tempo de T com read_TS(X) e write_TS(X) para garantir que o rótulo da transação de tempo não seja violado, ou seja, o protocolo de ordenação de timestamp irá garantir que qualquer operação read ou write que esteja em conflito sejam executadas por ordem de timestamp.

6.

Pode-se referir ao tamanho dos itens de dados como **granularidade.** Quanto maior a granularidade de um item, maior seu tamanho, logo menor o grau de concorrência permitido.

Porém, quanto menor o tamanho do item, maior é o número de itens no banco. Como cada item está associado a um bloqueio, o sistema terá uma quantidade maior de bloqueios ativos para serem tratados pelo gerenciador de bloqueios. Mais operações de bloqueio e desbloqueio serão realizadas, causando uma sobrecarga maior.

6.1

O melhor tamanho de uma granularidade depende da transação, dessa forma o banco de dados admite múltiplos níveis de granularidade. Entretanto, para este protocolo ser executado com eficiência serão necessários tipos adicionais de bloqueios.

O bloqueio de intenção será necessário, e sua ideia é que uma transação indique que tipo de bloqueio (compartilhado ou exclusivo) ela exigirá, através do caminho da raiz até o nó. São três tipos de bloqueios de intenção:

Intention-Shared (IS): Bloqueio explícito em um nível inferior da árvore, mas apenas com bloqueios compartilhados.

Intention-Exclusive (IX): Bloqueio explícito em um nível inferior com bloqueios exclusivos ou compartilhados.

Compartilhado e exclusivo com intenção (SIX): A sub-árvore com raiz por esse nó é bloqueada explicitamente no modo compartilhado e o bloqueio explícito está sendo feito em um nível inferior com bloqueios de modo exclusivo.

Além disso, um protocolo de bloqueio apropriado precisa ser usado. Ele é o **MGL** (Multiple granularity locking) e consiste nas seguintes regras:

- A compatibilidade de bloqueio deve ser aderida;

- A raiz da árvore precisa ser bloqueada primeiro, em qualquer modo;

- Um nó N pode ser bloqueado por uma transação T no modo S ou IS somente se o nó pai N já estiver bloqueado pela transação T no modo IS ou IX;

- Um nó N só pode ser bloqueado por uma transação T no modo X, IX ou SIX se o pai do nó N já estiver bloqueado pela transação T no modo IX ou SIX;

- Uma transação T só pode bloquear um nó se ela não tiver desbloqueado qualquer nó;

- Uma transação T só pode desbloquear um nó, N, se nenhum dos filhos do nó N estiver atualmente bloqueado por T;

Assim, o bloqueio começa pela raiz e “desce” pela árvore até que seja encontrado o nó que precisa ser bloqueado, enquanto o desbloqueio realizará o caminho contrário até que seja desbloqueado.

Otimização de Consulta

1.

Otimização de consulta é um conceito que se baseia em chegar ao plano mais eficiente e econômico usando as informações disponíveis sobre o esquema e o conteúdo das relações envolvidas. Seu propósito é escolher a melhor estratégia possível para realizar uma consulta.

2.

Uma árvore de consulta é uma estrutura de dados que representa as relações de entrada da consulta como nós folha da árvore e as operações de álgebra relacional como nós internos.

Executar uma árvore de consulta consiste na execução de um nó interno em que a ordem começará nos nós folha e terminará no nó raiz, assim que este produzir o resultado da consulta.

Por muitas vezes, ao se utilizar a álgebra relacional, mutas árvores de consultas podem ser semanticamente equivalentes porem diferentes em estrutura. Em alguns casos, quando uma seleção é feita, ela pode gerar árvores de consulta que irão criar arquivos grandes que não serão executados, assim sendo, a otimização heurística irá simplificar e otimizar as árvores de consultas iniciais em árvores de consulta finais, que serão mais eficientes para execução.

3.

Um plano de execução para uma expressão da álgebra relacional representada como uma árvore de consulta inclui informações sobre os métodos de acesso disponíveis para cada

relação e também os algoritmos a serem usados na computação dos operadores relacionais representados na árvore.

4.

Ao se utilizar um banco de dados pequeno, em que a maioria dos dados dos arquivos está armazenado na memória, os custos de computação são visados para serem minimizados. Já quando se utiliza um banco de dados maior, a minimização dos custos deve estar presente para armazenamento secundário, utilizando-se funções simples.

Dessa forma, alguns custos estarão inclusos ao se fazer uma consulta entre eles:

- **Custo de Acesso ao Armazenamento secundário:** É o ato de transferir blocos de dados entre o armazenamento de disco secundário e os buffers.

- **Custo de armazenamento em disco:** É o ato de armazenar no disco quaisquer arquivos intermediários que sejam gerados por uma estratégia de execução para a consulta.

- **Custo de computação:** É o ato de realizar operações na memória em registros dentro dos buffers de dados durante a execução da consulta.

- **Custo de uso da memória:** É o ato que diz respeito ao número de buffers da memória principal necessários durante a execução da consulta.

- **Custo de comunicação:** É o ato de envio da consulta e seus resultados do local do banco de dados até o local ou terminal em que a consulta foi originada.

Referências:

Livro “Sistemas de Banco de Dados”

http://www.bosontreinamentos.com.br/bancos-de-dados/conceitos-de-bancos-de-dados-o-que-e-uma-transacao/

https://blog.betrybe.com/tecnologia/acid-porque-usar/

https://www.devmedia.com.br/protocolo-com-base-em-timestamp-controle-de-concorrencia-em-bancos-de-dados/27810