Aluno: Guido Margonar Moreira RA: 2150948

**Parte - A**

**15.1**

No começo do capítulo é definido que o processamento de consulta se trata de um conjunto de atividades para extrair dados de um banco de dados. Composto pelas etapas de Análise e tradução, otimização e avaliação. Antes de qualquer processamento de consulta, a consulta precisa ser traduzida para uma forma interna baseada na álgebra relacional estendida. Em seguida, a consulta é otimizada para gerar um plano de execução eficiente. A avaliação da consulta ocorre quando o plano de execução é escolhido e o resultado da consulta é gerado. A escolha do plano de execução é responsabilidade do sistema para minimizar o custo da avaliação.

**15.2**

Em seguida é explicada a importância de comparar os custos de diferentes planos de avaliação de consulta em bancos de dados, e para isso é necessário estimar os custos das operações individuais e combiná-las para chegar ao custo total de um plano de avaliação. O custo pode ser medido em termos de vários recursos, como acessos ao disco, tempo de CPU e custo de comunicação em sistemas paralelos e distribuídos.

O modelo de custo do PostgreSQL inclui custos de CPU por tupla, entrada de índice e operador ou função. Para estimar o custo de um plano de execução de consulta, os dois fatores mais importantes são o número de transferências de bloco do local de armazenamento e o número de acessos de E/S aleatória, que exigem uma busca de disco em meio de armazenamento magnético. As SSDs não realizam uma operação de busca física, mas têm um trabalho extra para iniciar uma operação de E/S. As SSDs podem suportar cerca de 10.000 leituras aleatórias de blocos de quatro kilobytes por segundo e suportar 400 megabytes/segundo de throughput em leituras sequenciais usando a interface SATA padrão.

**15.3**

O operador mais básico para acessar dados é a varredura de arquivo (file scan), que é um algoritmo de busca para localizar e recuperar registros que atendem a uma condição de seleção. A busca linear é o algoritmo de varredura mais simples e pode ser aplicado a qualquer arquivo, independentemente da classificação ou da disponibilidade de índices, mas pode ser mais lento que outros algoritmos. As estimativas de custo para diferentes algoritmos de seleção são mostradas na Figura 15.3.

Os algoritmos de busca que usam um índice são chamados de varreduras de índice. Existem três algoritmos de seleção baseados em um índice agrupado: A2 (igualdade sobre chave), A3 (igualdade sobre não chave) e um algoritmo de seleção baseado em um índice secundário: A4 (igualdade). A escolha do índice é baseada na condição de seleção e em suas estimativas de custo.

Uma seleção conjuntiva envolve a operação lógica "e" e é representada pela expressão σθ1∧θ2∧...∧θn(r). Por outro lado, uma seleção disjuntiva envolve a operação lógica "ou" e é representada pela expressão σθ1∨θ2∨...∨θn(r). Já a negação é representada por σ¬θ(r), onde o resultado é o conjunto de tuplas de r para as quais a condição θ é avaliada como falsa.

Para implementar uma operação de seleção conjuntiva ou disjuntiva, existem diferentes algoritmos, que podem usar índices simples ou compostos, bem como identificadores de registro. O algoritmo escolhido é aquele que resulta no menor custo de processamento.

Por fim, a implementação de seleções com condições de negação fica a cargo do sistema gerenciador de banco de dados (SGBD).

**15.4**

O processo de classificação de dados é fundamental em sistemas de banco de dados, permitindo a especificação de ordenação em consultas SQL e tornando possíveis operações relacionais eficientes, como junções. Uma forma de classificação é criar um índice sobre a chave de classificação e usá-lo para ler a relação em ordem classificada. No entanto, a leitura de tuplas na ordem classificada pode levar a um acesso de disco para cada registro, o que pode ser dispendioso. Para lidar com isso, pode ser preferível ordenar os registros fisicamente.

A classificação externa é usada para classificar relações que não cabem na memória. A técnica mais utilizada para classificação externa é o algoritmo sort-merge externo, que envolve duas etapas: criação de rodadas classificadas e mesclagem de rodadas para criar a relação classificada. O número de rodadas criadas pode ser maior do que a quantidade de blocos disponíveis na memória, então a mesclagem prossegue em vários passos, com cada passada reduzindo o número de rodadas por um fator de M - 1.

**15.5**

Nesta seção, são apresentados vários algoritmos para calcular a junção de relações e seus custos. A equijunção é definida como uma junção da forma r ⋈ r.A=s.B s, em que A e B são atributos ou conjuntos de atributos das relações r e s, respectivamente. Um algoritmo simples para calcular a junção é o algoritmo de junção por loop aninhado, que consiste em um par de loops for aninhados. O algoritmo é dispendioso, pois examina cada par de tuplas nas duas relações. O número de pares de tuplas a serem consideradas é nr × ns, em que nr indica o número de tuplas em r e ns denota o número de tuplas em s.

O algoritmo merge associa um ponteiro a cada relação e lê grupos de tuplas com mesmo valor nos atributos de junção para Ss. As tuplas correspondentes da outra relação são lidas e processadas à medida que são lidas. A junção merge requer que cada conjunto Ss caiba na memória principal e se não couber, uma junção por loop aninhado em bloco pode ser realizada.

O particionamento recursivo é usado para lidar com grandes volumes de dados que não cabem na memória principal. Ele é realizado em passadas repetidas, gerando balde para cada passada, que é lido e particionado novamente na próxima passada para criar partições menores. O tratamento de estouro é necessário quando o índice de hash é maior que a memória principal, que pode ser resolvido por meio de solução de estouro ou impedimento de estouro. A junção hash exige a leitura completa das duas relações, com escrita subsequente, resultando em 2(br + bs) transferências.

O algoritmo de junção hash híbrida é usado quando a memória é grande, mas não cabe toda a relação de montagem. Ele usa mais blocos como buffer para economizar acesso de escrita/leitura. É útil quando a memória é significativamente maior que a relação de montagem dividido pelo número de blocos. Junções com condições complexas podem ser implementadas usando técnicas de seleções complexas. Cada par de tuplas no resultado intermediário consiste em uma tupla de r e uma de s, e o resultado final consiste nas tuplas que satisfazem as condições restantes.

**15.6**

Esta seção trata das outras operações relacionais e operações relacionais estendidas, como eliminação de duplicatas, projeção, operações de conjunto, junção externa e agregação.

* Eliminação de duplicatas pode ser feita por meio da classificação ou do hashing.
* Projeção pode ser realizada por meio da projeção em cada tupla e, em seguida, removendo os registros duplicados.
* Pesquisa por palavras-chave em documentos é muito utilizada no contexto das buscas na Web.
* Interseção pode ser implementada com eficiência, mesclando as listas a partir de varreduras simultâneas de todas as listas.

Para as operações de conjunto (união, interseção e diferença), é possível classificar as relações de entrada e varrê-las apenas uma vez para produzir o resultado. Se as relações não estiverem classificadas, é necessário incluir o custo da classificação. Outra opção é usar a técnica de hashing, que consiste em particionar as relações e criar partições, onde é possível aplicar as operações de conjunto. Para a junção externa, pode-se calcular a junção correspondente e acrescentar outras tuplas ao resultado da junção para chegar ao resultado da junção externa.

**15.7**

Nesta subseção são apresentadas duas técnicas e avaliação de expressões com operações relacionais: avaliação materializada e avaliação canalizada. A avaliação materializada avalia uma operação de cada vez e armazena os resultados em relações temporárias, enquanto a avaliação canalizada combina várias operações em uma canalização. O custo dessas técnicas pode variar e, em alguns casos, apenas a avaliação materializada é viável. O texto também sugere o uso de dois buffers para aumentar a eficiência da avaliação materializada.

As operações geram tuplas continuamente e as colocam em seus buffers de saída até que estejam cheios. Quando o buffer está cheio, a operação espera até que sua operação pai remova tuplas do buffer para que o buffer tenha espaço para mais tuplas. Essa abordagem requer mais memória, mas pode ter melhor desempenho.

Já nas canalizações para dados de stream contínuo, como entradas de sensores. Consultas contínuas podem ser escritas para responder aos dados à medida que eles chegam, e as canalizações controladas por produtor são as mais adequadas para avaliar essas consultas. Muitas dessas consultas executam agregação com janelas, em que as tuplas são agregadas separadamente em cada janela de tamanho fixo. O resultado da agregação em uma janela pode ser gerado assim que o sistema saiba que nenhuma nova tupla será recebida para essa janela.

**Parte - B**

Exercicios :

**15.2**

πT.branch\_name((πbranch\_name, assets(ρT (branch))) ⨝T.assets > S.assets

(πassets (ρ(branch\_city= ’Brooklyn’)(ρS(branch)))))

**15.5**

A maior relação pode ser guardada inteira na memória, enquanto a maior pode ser lida bloco por bloco executando um loop join onde a maior é a relação externa, o número de operações é br + bs, enquanto a memória requisitada é 2 páginas + o menor dentre br e bs.

**Referências:**

Sistema de Banco de Dados de Abraham Silberschatz - Minhas biblioteca

ChatGPT