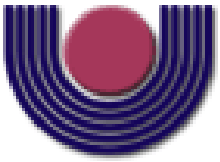


UNIOESTE – Universidade Estadual do Oeste do Paraná

CENTRO DE CIÊNCIAS EXATAS E TECNOLÓGICAS

Colegiado de Informática

Curso de Bacharelado em Informática

Algoritmos para Processamento e Otimização de Consultas

Adriano Douglas Girardello

Ana Paula Fredrich

Tiago Alexandre Schulz Sippert

CASCAVEL

2009

ADRIANO DOUGLAS GIRARDELLO

ANA PAULA FREDRICH

TIAGO ALEXANDRE SCHULZ SIPPERT

ALGORITMOS PARA PAROCESSAMENTO E OTIMIZAÇÃO DE CONSULTAS

.....

Este trabalho faz parte da avaliação da disciplina de Banco de Dados do curso de Bacharel em Informática, do Centro de Ciências Exatas e Tecnológicas da Universidade Estadual do Oeste do Paraná - Campus de Cascavel.

Orientador: Carlos José Maria Olguin.

CASCAVEL

2009

ADRIANO DOUGLAS GIRARDELLO

ANA PAULA FREDRICH

TIAGO ALEXANDRE SCHULZ SIPPERT

ALGORITMOS PARA PROCESSAMENTO E OTIMIZAÇÃO DE CONSULTAS

.....

Este trabalho faz parte da avaliação da disciplina de Banco de Dados do curso de Bacharel em Informática, do Centro de Ciências Exatas e Tecnológicas da Universidade Estadual do Oeste do Paraná – Campus de Cascavel, aprovado pelo professor:

_____

Prof. Carlos José Maria Olguin

Colegiado de Informática, UNIOESTE

Cascavel, 24 de Setembro de 2009.

“A mente que se abre a uma nova idéia jamais voltará ao seu tamanho original.” (Albert Einstein)

AGRADECIMENTOS

Agradecemos aos nossos colegas pela companhia, e apoio durante a realização dos trabalhos nos laboratórios, ao professor Olguin por ter nos fornecido o livro que serviu como base para a elaboração deste trabalho, e principalmente a Deus por ter nos dado capacidade e sabedoria de realizarmos trabalhos como este.

Lista de Figuras

Figura 1 - Passos do Processamento de uma Consulta de Alto Nível

Figura 2 - Esboço do algoritmo sort-merge para ordenação externa

Lista de Abreviaturas e Siglas

SGBD – Sistema Gerenciador de Banco de Dados

SQL – String Query Language

Sumário

1 - Introdução
2 - Passos do Processamento de uma Consulta
    2.1 - Moldar a Consulta em alguma forma interna
    2.2 - Converter para a forma canônica
    2.3 - Escolher procedimentos candidados de baixo nível
    2.4 - Gerar planos de consultas e escolher o mais econômico
3 - Traduzindo Consultas SQL para Álgebra Relacional
4 - Algoritmo para Ordenação Externa (External Sorting)
5 - Algoritmo para Operações Select e Join
    5.1 - Implementação da Operação SELECT
        5.1.1 - Métodos de Busca para Seleções Simples
        5.1.2 - Métodos de Busca para Seleções Complexas
    5.2 - Implementação da Operação JOIN
        5.2.1 - Métodos para a Implementação de Junções
        5.2.2 - Efeitos da Disponibilidade de Espaço de Buffer
6 - Combinação de Operações Usando Pipelines
7 - Utilização de Heurísticas na Otimização de Consultas
    7.1 - Notações de Árvores de Consulta e de Grafos de Consultas
    7.2 - Conversões de Árvores de Consulta em Planos de Execução de Consultas
8 - Utilização de Seletividade e Estimativa de Custo na Otimização de Consultas
    8.1 - Componentes do Custo para a Execução de uma Consulta
9 - Otimização Semântica de Consultas
10 - Otimização Física
11 - Estatísticas de Bancos de Dados
    11.1 - Estatísticas do BD2
    12.1 - Estatísticas do Ingres
13 - Conclusões

Resumo

Este trabalho aborda as técnicas utilizadas por um Sistema Gerenciador de Banco de Dados para processar, otimizar e executar consultas de alto nível. Daremos uma visão geral das técnicas utilizadas por SGBDs no processamento e na otimização de consulta de alto nível e de como as operações podem ser combinadas durante o processamento da consulta.

Palavras-chave: algoritmo de processamento, otimização de consulta.

Capítulo 1

Introdução

Abordaremos as técnicas utilizadas por um SGBD para processar, otimizar e executar consultas de alto nível. Uma consulta expressa em linguagem de alto nível, como SQL, deve primeiro passar por uma análise léxica, uma análise sintática e ser validada. Na análise léxica identificamos os itens léxicos da linguagem, tais como palavras-chaves da SQL, nomes de atributos e nomes de relacionamentos, enquanto que na análise sintática verificamos a sintaxe da consulta para determinar se ela está formulada de acordo com as regras sintáticas da linguagem de consulta. Por fim, a consulta também deve ser validada por meio da verificação de que todos os atributos e nomes de relacionamentos são válidos, e se são nomes com significados semânticos no esquema do banco de dados específico que está sendo consultado.

Então uma representação interna da consulta é criada, geralmente como uma estrutura de dados de árvore chamada árvore de consulta ou uma estrutura gráfica chamada grafo de consulta. Então o SGBD deve planejar uma estratégia de execução para a recuperação do resultado da consulta a partir dos arquivos no banco de dados. Para efetuar uma consulta existe muitas estratégias de execução possíveis, e o processo de escolha de uma estratégia adequada para o processamento de uma consulta é chamado *otimização de consulta*.

Capítulo 2

Passos do Processamento de uma Consulta

A partir de uma consulta de alto nível é realizado uma análise léxica, sintática e a validação da mesma para garantir que esta esteja de acordo com a linguagem de consulta. Após esta fase, a consulta passa por um *otimizador de consulta* que tem a função de produzir um plano de execução. Obtido o plano de execução, o mesmo passará pelo *gerador de código* que gera o código que executa o plano. Assim obtemos o código para executar a consulta.

O código então passa pelo *processador em tempo de execução em banco de dados* que executa o código da consulta, podendo este ser interpretado ou compilado, afim de obter o resultado da consulta. Se um erro ocorrer, uma mensagem é, ou deverá ser, gerada pelo processador em tempo de execução do banco de dados. Podemos observar visualmente estes passos na Figura 1.

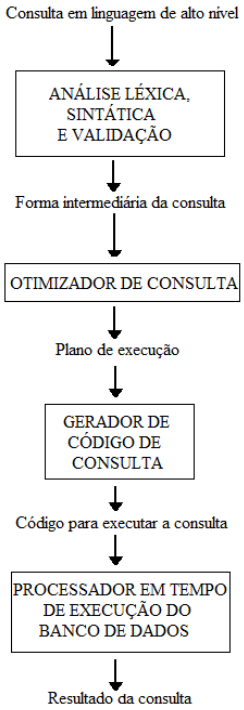


Figura 1: Passos do Processamento de uma Consulta de Alto Nível

Também em uma outra visão dos estágios de processamento de consultas, conforme (Date) identificamos quatro estágios principais: Moldar a consulta em alguma forma interna; Converter para forma canônica; Escolher procedimentos candidatos de baixo nível; Gerar planos de consultas e escolher o mais econômico. Em seguida entramos em detalhes de cada um dos estágios.

2.1 Moldar a Consulta em Alguma Forma Interna

Nesse estágio devemos converter a consulta original em uma representação mais adequada à manipulação pela máquina, eliminando peculiaridades da sintaxe concreta da linguagem de consulta e pavimentando o caminho para estágios posteriores ao de otimização.

Podemos escolher vários formalismos para a representação interna, desde que seja rico o suficiente para representar todas as consultas na linguagem de consulta externa, e neutro no sentido de não prejudicar escolhas subseqüentes. Em geral a forma de representação interna é a árvore de sintaxe abstrata ou árvore de consulta.

Para objetivos de otimização a representação mais conveniente é a álgebra relacional ou o cálculo relacional.

2.2 Converter para a Forma Canônica

Este é o estágio em que são executadas as otimizações que funcionam garantidamente, sem importar os valores dos dados ou os acessos físicos. As linguagens relacionais permitem que uma consulta seja expressa de várias formas diferentes, mas o desempenho da consulta não deve depender da forma que o usuário a escreveu. Assim o próximo passo é converter a representação eliminando essas diferenças superficiais, encontrando uma representação mais eficiente que a original.

Assim para a saída do primeiro estágio, o otimizador transforma em alguma forma mais eficiente utilizando-se de regras de transformação bem definidas.

2.3 Escolher Procedimentos Candidatos de Baixo Nível

Depois de converter a representação em uma forma mais desejável, o otimizador deve decidir como executar essa consulta. Agora são necessárias considerações como a existência

de índices ou outros caminhos de acesso, agrupamento físico, distribuição de valores de dados, e outros.

A estratégia é considerar que para cada operação de baixo nível, com certas interdependências, o otimizador terá à sua disposição um conjunto de procedimentos de implementação predefinidos. Cada procedimento terá a fórmula de custo associada, as quais são usadas no próximo estágio.

Após, utilizando informações do catálogo quanto ao estado do banco de dados e utilizando a informação de interdependência, o otimizador escolhe um ou mais procedimentos para implementar cada uma das operações de baixo nível, procedimento chamado de seleção de caminho de acesso.

2.4 Gerar Planos de Consultas e Escolher o mais Econômico

Neste estágio final de otimização, é elaborado um conjunto de planos de consulta, e é escolhido o melhor desses planos. Cada plano é construído com a combinação do conjunto de procedimentos de implementação. Normalmente haverão muitos planos candidatos, os quais geralmente não precisariam ser gerados, pois a tarefa de escolher entre um deles, o mais econômico, poderia demorar muito. Assim é preciso manter o conjunto de planos gerados dentro do limite, reduzindo o espaço de pesquisa.

A escolha do melhor plano exige um método para atribuir um peso de execução para cada um deles. Em geral, o custo do plano é a soma dos custos dos procedimentos individuais que formam esse plano. Como todas as consultas envolvem a geração de resultados intermediários durante a execução (exceto as mais simples), o otimizador deve avaliar o tamanho dos resultados intermediários para avaliar as fórmulas de custos.

Capítulo 3

Traduzindo Consultas SQL para Álgebra Relacional

A linguagem SQL é usada na maioria dos SGBDs comerciais. Primeiramente, uma consulta SQL é traduzida em uma expressão equivalente de álgebra relacional estendida - e é representada por uma estrutura de dados de árvore de consulta -, que então, é otimizada. Em geral, as consultas SQL são decompostas em blocos de consultas, que formam as unidades básicas que podem ser traduzidas em operadores algébricos e otimizados.

Exemplo:

SELECT UNOME, PNOME
FROM EMPREGADO
WHERE SALARIO > (SELECT MAX (SALARIO)
FROM EMPREGADO
WHERE DNO=5);

O bloco interno poderia ser traduzido para a seguinte expressão da álgebra relacional estendida: $_{F}$ MAX SALARIO(σ DNO=5(EMPREGADO)) e o bloco externo na seguinte expressão: π UNOME,PNOME (σ > C (EMPREGADO)), onde C é o resultado retornado pelo bloco interno. O otimizador de consultas, então, escolheria um plano de execução para cada bloco.

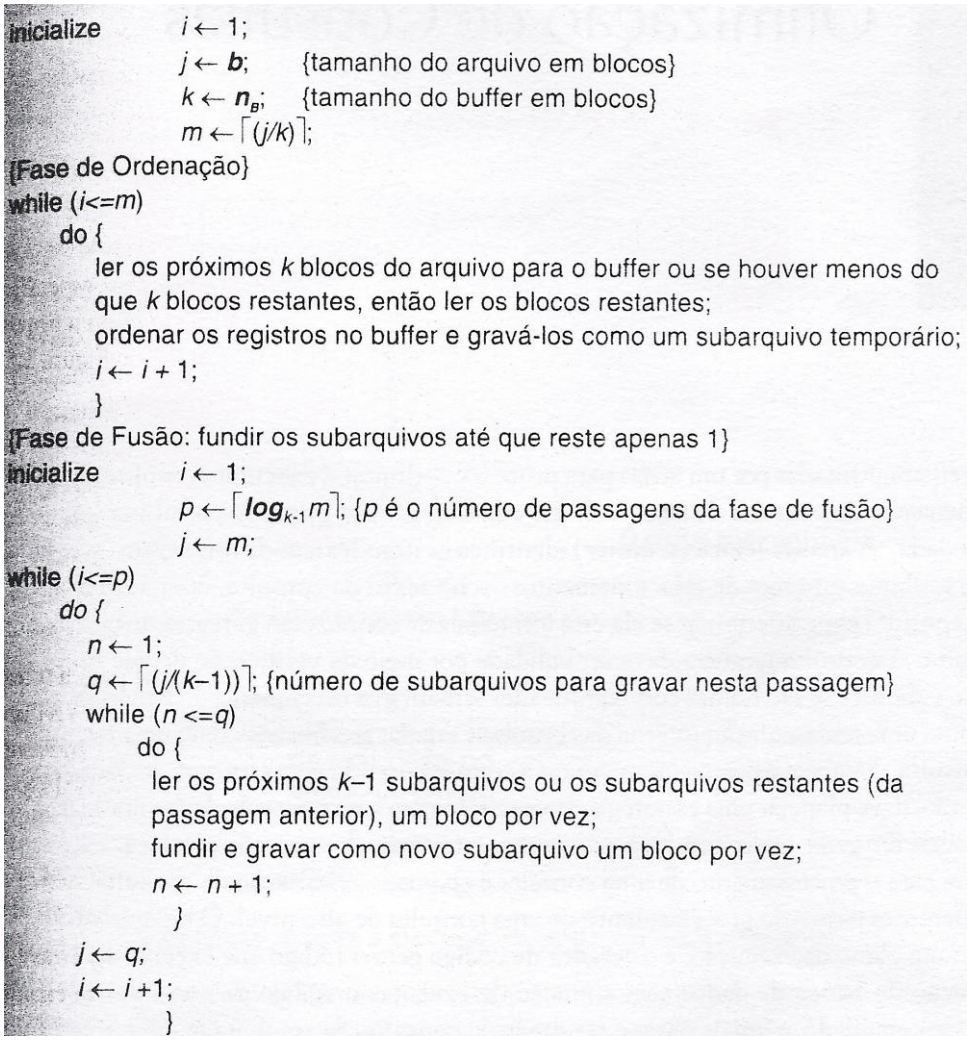
Capítulo 4

Algoritmo para Ordenação Externa (*External Sorting*)

A ordenação é um dos algoritmos primários utilizados no processamento de consultas. Como por exemplo, quando ocorre uma consulta SQL que contenha a cláusula ORDER BY o resultado da consulta deverá ser ordenado. A ordenação também é utilizada no algoritmo sort-merge sendo este utilizado no JOIN e em outras operações tais como UNION e INTERSECTION, e em algoritmos de eliminação de duplicatas na operação PROJECT.

A ordenação externa é adequada para arquivos de registros grandes, que são armazenados em disco e não cabem inteiramente na memória principal, como a maioria dos arquivos de bancos de dados.

O algoritmo de ordenação externa usa uma estratégia sort-merge que inicia ordenando pequenos sub-arquivos - chamados runs (resultado parcial) - do arquivo principal e, então, realiza a fusão dos runs ordenados, criando maiores sub-arquivos ordenados, que, por sua vez, são fundidos. Este algoritmo, assim como outros algoritmos de banco de dados, exige espaço de buffer na memória principal, onde a ordenação e a fusão são realizados de fato. O esboço do algoritmo básico pode ser visto na Figura 2, consistindo nas duas fases já explicadas, ordenação e fusão.



```
inicialize        i ← 1;
                  j ← b;      {tamanho do arquivo em blocos}
                  k ← n_B;    {tamanho do buffer em blocos}
                  m ← ⌈(j/k)⌉;
{Fase de Ordenação}
while (i<=m)
    do {
       ler os próximos k blocos do arquivo para o buffer ou se houver menos do
       que k blocos restantes, então ler os blocos restantes;
       ordenar os registros no buffer e gravá-los como um subarquivo temporário;
       i ← i + 1;
       }
{Fase de Fusão: fundir os subarquivos até que reste apenas 1}
inicialize        i ← 1;
                  p ← ⌈log_{k-1} m⌉; {p é o número de passagens da fase de fusão}
                  j ← m;
while (i<=p)
    do {
       n ← 1;
       q ← ⌈(j/(k–1))⌉; {número de subarquivos para gravar nesta passagem}
       while (n <=q)
            do {
              ler os próximos k–1 subarquivos ou os subarquivos restantes (da
              passagem anterior), um bloco por vez;
              fundir e gravar como novo subarquivo um bloco por vez;
              n ← n + 1;
                }
       j ← q;
       i ← i +1;
                }
```

Figura 2: Esboço do algoritmo sort-merge para ordenação externa

Na fase de ordenação, runs (partes ou pedaços) do arquivo que caibam no espaço de buffer disponível são lidos para a memória principal, ordenados usando o algoritmo de classificação interna e escritos de volta no disco como sub-arquivos ordenados temporários. O tamanho de um run e o número inicial de runs (n_R) são determinados pelo número de blocos de arquivo(b) e o espaço de buffer disponível (n_B).

Na fase de fusão, os runs ordenados são fundidos durante uma ou mais passagens. O grau de fusão (d_M) é o número de runs que podem ser fundidos em cada passagem. Em cada passagem, um bloco de buffer é necessário para manter um bloco de cada um dos runs que serão sendo fundidos, e um bloco é necessário para conter um bloco de resultado da fusão.

Capítulo 5

Algoritmo para Operações Select e Join

5.1 Implementação da Operação SELECT

Para a execução de um SELECT existem muitas opções, sendo que algumas dependem que o arquivo possua caminhos específicos de acesso. Podemos dividir os métodos de busca em dois grandes grupos: Métodos de Busca para Seleções Simples e Métodos de Busca para Seleções Complexas.

Um SGBD coloca à disposição muitos métodos para otimizar uma consulta de seleção. O otimizador deve escolher o método apropriado para a execução de cada operação SELECT em uma consulta, utilizando fórmulas que estimam os custos de cada método e escolhendo o método com o menor custo estimado.

5.1.1 Métodos de Busca para Seleções Simples

Uma variedade de algoritmos de busca é possível para a seleção de registros de um arquivo. Eles também são conhecidos como varreduras de arquivo porque varrem os registros de um arquivo para buscar e recuperar os registros que satisfazem a condição de seleção. Se o algoritmo de busca envolve o uso de um índice, o índice de busca é chamado de índice de varredura. Alguns métodos de busca:

- S1. *Busca Linear (força bruta)*: Recupera cada registro do arquivo e testa se seus valores de atributos satisfazem a condição de seleção.
- S2. *Busca Binária*: Se a condição de seleção envolver uma comparação de igualdade em um atributo-chave para o qual o arquivo está ordenado, pode-se usar a busca binária - que é mais eficiente que a busca linear.
- S3. *Utilização de um Índice Primário (ou Chave de Hash)*: Se a condição de seleção envolver uma comparação de igualdade em um atributo-chave com um índice

primário (ou chave de hash), use o índice primário (ou chave de hash) para recuperar o registro.

- S4. *Utilização de um Índice Primário para Recuperar Múltiplos Registros*: Se a condição de comparação for >, >=, < ou <= em um campo chave com um índice primário, use o índice para encontrar o registro que satisfaça a condição de igualdade correspondente e depois recupere todos os registros seguintes no arquivo (ordenado).
- S5. Utilização de um Índice Cluster para Recuperação de Múltiplos Registros: Se a condição de seleção envolver uma comparação de igualdade em um atributo que não seja chave com um índice clustering, use o índice para recuperar todos os registros que satisfaçam a condição.
- S6. Utilização de um Índice Secundário (arvore B^+) em uma Comparação de Igualdade: Esse método de busca pode ser usado para recuperar um único registro se o campo de indexação for uma chave ou para recuperar múltiplos registros se o campo de indexação não for chave. Ele também pode ser usado para comparações envolvendo >, >= < ou <=.

5.1.2 Métodos de Busca para Seleções Complexas

Se uma condição de uma operação SELECT é uma condição conjuntiva - ou seja é formada por diversas condições simples conectadas pelo conectivo lógico AND - o SGBD pode usar os seguintes métodos adicionais para implementar a operação.

- S7. *Seleção Conjuntiva Utilizando um Índice Individual*: Se um atributo envolvido em qualquer condição simples individual da condição conjuntiva possuir um caminho de acesso que permita o uso de um dos métodos de S2 a S6, use aquela condição para recuperar os registros e depois verifique se cada registro recuperado satisfaz as condições simples restantes da condição conjuntiva.
- S8. *Seleção Conjuntiva Utilizando um Índice Composto*: Se dois ou mais atributos estiverem envolvidos em condições de igualdade na condição conjuntiva e houver um índice composto para a combinação dos campos, podemos usar o índice diretamente.
- S9. *Seleção Conjuntiva por Meio da Integração de Registros*: Se índices secundários estiverem disponíveis para mais de um dos campos envolvidos nas condições simples de uma condição conjuntiva, e se os índices incluírem ponteiros de registros, cada

índice poderá ser usado para recuperar o conjunto de ponteiros de registro que satisfaça a condição individual.

Quando ocorre que uma condição individual especifica a seleção, verifica-se se existe um caminho de acesso no atributo envolvido naquela condição. Havendo um caminho de acesso, o método correspondente aquele caminho é utilizado, não havendo caminho, a abordagem da força bruta da busca linear (S1) é utilizada.

É necessário que se otimize consultas de seleção principalmente em condições de seleção conjuntivas, sempre que mais que um dos atributos possuam um caminho de acesso. Dessa forma o otimizador sempre escolhe o caminho de acesso que recupera o menor número de registros, por meio da escolha do método com menor custo estimado.

Uma condição disjuntiva, é muito mais difícil de processar e de otimizar. Com condições desse tipo podemos realizar poucas otimizações, pois os registros que satisfazem a condição disjuntiva são a união dos registros que satisfazem as condições individuais. Por isso se haver apenas um caminho de acesso para toda condição, poderemos otimizar a seleção por meio da recuperação dos registros que satisfazem cada condição, e então aplicar a operação de união para eliminar duplicidades.

5.2 Implementação da Operação JOIN

Uma das operações que mais consomem tempo de processamento de consultas é a operação JOIN. Há possibilidades de fazer uma junção de duas vias (junção em dois arquivos) ou junções de múltiplas vias (junção em mais de dois arquivos). O número de maneiras possíveis para executas as junções de múltiplas vias aumenta rapidamente. A seguir apresentamos as principais técnicas para a execução de junção.

5.2.1 Métodos para a Implementação de Junções

- J1. *Junção de Laços Aninhados (Nested-Loop) (Força Bruta)*: Para cada registro em um conjunto do laço externo, recupere cada registro do conjunto do laço interno, testando se os registros satisfazem a condição de junção.
- J2. *Junção de Laço Único (Single-Loop)*: Se existir um índice para um dos atributos da junção recupere todos os registros de um conjunto e depois recupere os registros do outro conjunto que correspondam a condição de junção.

- J3. *Junção Sort-Merge (Ordenação-Fusão)*: Se os registros dos dois conjuntos estiverem ordenados pelos valores de junção, poderemos implementar a junção de melhor maneira possível. Sendo ordenados, os índices proporcionam a capacidade de acessar os registros em ordem, porém os registros estão fisicamente espalhados pelos blocos do arquivo, tornando este método bastante ineficiente.
- J4. *Junção-Hash (Hash-Join)*: Os registros dos dois conjuntos são particionados em um mesmo arquivo hash. Uma primeira passagem pelo menor conjunto coloca seus registros nos buckets do arquivo, fase chamada de separação. Em uma segunda fase é feita a sondagem é feita uma passagem no segundo conjunto, combinando os registros.

5.2.2 Efeitos da Disponibilidade de Espaço de Buffer

Nos vários algoritmos de junção, um efeito importante sobre os mesmos é o tamanho disponível do buffer. Sempre é vantajoso ler para a memória a quantia máxima de blocos possíveis do arquivo, reduzindo assim o número de acessos total ao disco. Uma vez que o buffer está cheio ele é escrito no disco e reutilizado.

Capítulo 6

Combinação de Operações Usando *Pipelines*

Normalmente uma consulta SQL é traduzida em uma expressão de álgebra relacional, composta de várias operações relacionais. Ao executarmos cada operação por vez, geramos arquivos temporários gerando sobrecarga excessiva. Muito tempo é gasto na geração e ordenação de tais arquivos.

Para reduzir o número de arquivos temporários, uma estratégia é usar códigos de execução de consulta que correspondam a algoritmos de combinação das operações em uma só consulta. Essa estratégia é chamada de pipelining.

O que pode ser feito ainda é criar dinamicamente código de execução de consulta e implementar múltiplas operações, assim conforme o resultado de uma consulta é produzido, ele pode ser imediatamente ser utilizado como entrada para as próximas operações.

Capítulo 7

Utilização de Heurísticas na Otimização de Consultas

A representação interna de uma consulta geralmente está na forma de estrutura de dados de árvore de consulta ou de um grafo de consulta. A fim de melhorar seu desempenho, técnicas de otimização que utilizam regras heurísticas para modificar a representação interna são usadas.

O analisador sintático de uma consulta de alto nível primeiro gera uma representação interna inicial, que depois é otimizada de acordo com regras de heurística. Na seqüência, um plano de execução de consulta é gerado para executar grupos de operações com base nos caminhos de acesso disponíveis para os arquivos envolvidos na consulta.

Uma das principais regras heurísticas é aplicar as operações SELECT e PROJECT antes de aplicar o JOIN ou outras operações binárias. Isso deve ao tamanho do arquivo resultante de uma operação binária - tal como o JOIN -, que geralmente é uma função multiplicativa dos tamanhos dos arquivos de entrada. As operações SELECT e PROJECT reduzem o tamanho de um arquivo e, por isso, devem ser aplicadas antes de uma junção ou outra operação binária.

7.1 Notações de Árvores de Consulta e de Grafos de Consultas

Uma árvore de consulta é uma estrutura de dados de árvore que corresponde a uma expressão da álgebra relacional. Ela representa as relações de entrada de uma consulta como nós folhas da árvore e representa as operações da álgebra relacional como nós internos.

Uma execução de árvore de consulta consiste na execução de uma operação de nó interno sempre que seus operandos estiverem disponíveis - e depois da substituição do nó interno pela relação que resulta da execução da operação. A execução termina quando o nó raiz é executado e produz a relação de resultado da consulta.

7.2 Conversões de Árvores de Consulta em Planos de Execução de Consultas

Um plano de execução para uma expressão de álgebra relacional representada como uma árvore de consulta inclui informações sobre os métodos de acesso disponíveis para cada relação, bem como os algoritmos a serem utilizados na computação dos operadores relacionais representados em árvore.

Capítulo 8

Utilização de Seletividade e Estimativa de Custo na Otimização de Consultas

Um otimizador de consultas não deve depender somente de regras heurísticas; ele também deve estimar e comparar os custos da execução de uma consulta usando diferentes estratégias de execução - e deve escolher a estratégia com a menor estimativa de custo. Para essa abordagem funcionar, estimativas precisas de custo são necessárias, de forma que diferentes estratégias sejam comparadas de maneira correta e realista. Além disso, deve-se limitar o número de estratégias de execução possíveis.

8.1 Componentes do Custo para a Execução de uma Consulta

O custo da execução de uma consulta inclui os seguintes componentes:

1. *Custo de Acesso ao Armazenamento Secundário*: Esse é o custo da busca, da leitura e da escrita de blocos de dados que residem em armazenamento secundário, principalmente em discos. O custo da busca por registros em um arquivo depende dos tipos de estruturas de acesso daquele arquivo, tais como ordenação, hashing e índices primários e secundários. Além disso, fatores tais, como se os blocos de arquivo são alocados de maneira adjacente no mesmo cilindro do disco ou se são espalhados no disco, afetam o custo de acesso
2. *Custo de Armazenamento*: Esse é o custo de armazenamento de quaisquer arquivos temporários que sejam gerados por uma estratégia de execução para a consulta.
3. *Custo de Computação*: Esse é o custo da realização, na memória, das operações sobre os buffers de dados durante a execução da consulta. Tais operações incluem a busca e a ordenação de registros, a fusão de registros em uma junção e a realização de cálculos em valores de campos.

4. *Custo de Uso de Memória*: Esse é o custo referente ao número buffers de memória necessários durante a execução da consulta.
5. *Custo de Comunicação*: Esse é o custo do transporte da consulta e de seus resultados de um site de banco de dados para o site ou terminal onde a consulta se originou.

Para grandes bancos de dados, a ênfase principal é na minimização do custo de acesso ao armazenamento secundário.

Capítulo 9
Otimização Semântica de Consultas

A otimização semântica de consultas é uma abordagem diferente, que consiste em usar restrições especificadas no banco de dados a fim de transformar uma consulta em outra que seja mais eficiente para ser executada. Essa técnica pode ser utilizada em conjunto com as técnicas já aplicadas anteriormente.

O otimizador semântico verifica a existência de restrições no banco antes de executar uma consulta, dependendo das restrições existentes ele modifica a consulta ou nem a executa, dessa forma podendo economizar um tempo de processamento considerável.

Essa otimização, dependendo do número de restrições, pode implicar em um aumento no tempo de execução da consulta, pois pode pesquisar muitas restrições para ver quais ele se aplica realmente. Com a inclusão de regras ativas no banco de dados, as técnicas de otimização semântica podem ser completamente incorporadas aos SGBDs do futuro.

Capítulo 10
Otimização Física

A fim de reduzir a controvérsia para acessos a disco, o banco de dados pode ser dividido em dois discos, permitindo diversos acessos a disco servindo em paralelo. Porém, para explorar o potencial de acessos paralelos a disco, precisamos escolher uma boa distribuição de dados entre os discos.

Um algoritmo para tal distribuição é o algoritmo ligação paralela de dois modos, e requer diversos processadores para acessar relações em paralelo. Uma das técnicas, chamada de fatiamento de disco, divide as tuplas de relações individuais entre diversos discos.

A ligação física ótima difere para diferentes consultas. O administrador do banco de dados precisa escolher uma organização física que acredita ser boa para a mistura de consultas do banco de dados. De tal forma o otimizador de consultas do Sistema de Banco de Dados precisa escolher entre as várias técnicas paralelas e seqüenciais, estimando o custo de cada técnica em uma dada organização física.

Capítulo 11

Estatísticas de Bancos de Dados

Outro estágio do processo geral de otimização, os estágios de "seleção de caminho de acesso", fazem o uso das chamadas estatísticas de dados armazenadas no catálogo. Listamos a seguir as principais estatísticas mantidas por dois produtos comerciais, DB2 e Ingres:

11.1 Estatísticas do BD2

- Para cada tabela básica:
  - Cardinalidade
  - Número de páginas ocupadas por esta tabela
  - Fração de "espaço de tabela" ocupado por esta tabela.
- Para cada coluna de cada tabela básica:
  - Número de valores distintos nesta coluna
  - Segundo valor mais alto nesta coluna
  - Segundo valor mais baixo nesta coluna
  - Somente para colunas indexadas, os dez valores que ocorrem com maior freqüência nesta coluna e o número de vezes em que ocorrem.

* Para cada índice:

- Uma indicação quanto a ser este um "índice de agrupamento" (isto é, um índice usado para agrupar dados logicamente relacionados em posições fisicamente contíguas no disco).
- Se for o caso, a fração da tabela indexada ainda em sequência de agrupamento em cluster.
- Número de páginas por folhas neste índice.
- Número de níveis neste índice.

11.2 Estatísticas do Ingres

* Para cada tabela básica:
    - Cardinalidade
    - Número de páginas primárias para esta tabela
    - Número de páginas de overflow para esta tabela
* Para cada coluna de cada tabela básica:
    - Número de valores distintos nesta coluna
    - Valores máximo, mínimo e médio para esta coluna
    - Valores efetivos nesta coluna e o número de vezes em que eles ocorrem.

Capítulo 12
Conclusões

Como conclusão, devemos mencionar que muitos dos produtos de hoje infelizmente incluem certos inibidores de utilização, dos quais os usuários devem pelo menos estar cientes (embora pouco se possa fazer em relação a eles na maioria dos casos). O inibidor de otimização é uma característica do sistema em questão que impede o otimizador de fazer um trabalho tão bom quanto poderia. Os inibidores em questão incluem linhas duplicadas, lógica trivalorada e a implementação de sql da lógica trivalorada.

Glossário

Tupla	Uma seqüência ordenada de n elementos, que pode ser definida pela recursão do par ordenado.

Referências Bibliográficas

DATE, C. J. 1941 - Introdução a sistemas de bancos de dados; Tradução Vandemberg Dantas de Souza, Publicare Consultoria e Serviços. Rio de Janeiro: Campus, 2000.

ELMASRI, Ramez - Sistemas de bancos de dados / Ramez Elmasri e Shamkant B. Navathe; São Paulo: Pearson Addison Wesley, 2005.

KORTH, Henry F. - Sistemas de Bancos de Dados / Henry F. Korth, Abraham Silberschatz; Tradução Maurício Heihachiro Galvan Aba; Revisão Técnica Sílvio Carmo Palmieri, 2ª ed - São Paulo: Makron Books, 1993.