UNIVERSIDADE ESTADUAL DE CAMPINAS
PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO
INSTITUTO DE COMPUTAÇÃO

OTIMIZAÇÃO DE CONSULTAS RELACIONAIS

TRABALHO DE PÓS-GRADUAÇÃO

Alunos: João Paulo Silva Cintra
Renato Capello
Professor: Geovane Cayres Magalhães

CAMPINAS,
2005

1. INTRODUÇÃO

O seguinte trabalho abordará em detalhes uma típica otimização de consulta relacional, onde será mostrado como é realizado o processo de uma consulta SQL. Assim, quando uma consulta é enviada ao sistema, é necessário encontrar o melhor método para se descobrir à resposta usando a estrutura de banco de dados existentes.

A otimização de consulta é o processo de selecionar o plano de avaliação de consulta mais eficiente para uma consulta.

Então, a finalidade do otimizador de um banco de dados é livrar os usuários de suas complexidades e das exigências necessárias para se conseguir consultas de forma eficiente. Usar técnicas para reformular consultas em outra(s) que desempenham a mesma funcionalidade, mas com tempo de resposta menor que a consulta original é uma das principais características de um otimizador de consulta.

O otimizador faz com que reduza o esforço manual e repetitivo de identificar e corrigir comandos de SQL realizados de maneira incorreta, causando em baixos desempenhos. Isso é uma vantagem do otimizador evitando-se assim aquisição de novos hardwares mais poderosos para se alcançar ainda mais quanto ao desempenho.

Por isto, um SGBD deve oferecer soluções eficientes para obter dados de um banco de dados através de seu processador.

Serão apresentados os conceitos básicos necessários para a compreensão de como uma otimização de consulta relacional realiza suas tarefas em um sistema de banco de dados por meio de instruções SQL.

2. TRANSFORMAÇÃO DE CONSULTAS SQL PARA ÁLGEBRA

Em se tratando de otimização de consulta só faz sentido se falarmos em uma linguagem de alto nível como é o caso do SQL que é uma linguagem de manipulação e modificação de estruturas do banco de dados.

Quando uma consulta SQL é realizada pelo usuário, estas são decompostas em pequenas unidades chamadas de blocos de consulta, e estas são otimizadas um bloco por vez. Já em relação aos blocos aninhados esses são tratados como uma chamada de uma sub-rotina, feita uma vez por tupla mais externa. São esses blocos que serão convertidos pela álgebra relacional.

O modelo relacional é sem dúvida uma das tecnologias de banco de dados mais utilizada no mundo. Esse modelo é representado por operações da teoria dos conjuntos como união, produto cartesiano, intersecção e diferença e por operações específicas para banco de dados relacionais com a seleção, junção, agregação e projeção.

A primeira ação que o sistema tem de executar em uma consulta é traduzir a expressão em uma expressão equivalente da álgebra relacional e representá-la em uma estrutura de dados conhecida como árvore de consulta. Tendo convertido a representação em uma forma interna melhor, o otimizador deve então decidir como avaliar a consulta transformada representada pela forma convertida.

Um simples bloco de consulta em SQL não aninhado possui a cláusula SELECT que relaciona os atributos que representam os dados desejados da consulta, a cláusula FROM relaciona as tabelas envolvidas na consulta, a cláusula WHERE especifica as condições de escolha das tuplas das tabelas, a GROUP BY agrupa tuplas que possuem o mesmo valor para a lista de atributos de agrupamento em grupos e, sobre estes, são aplicados funções de agregação. Por último a cláusula HAVING que específica as condições de escolha dos grupos de tuplas produzidos pela cláusula GROUP BY.

Uma característica importante no modelo relacional que favorece seu desempenho são os catálogos que oferece informações sobre a estrutura de dados, os tipos e formatos de armazenamento e as restrições sobre os dados. Estas informações são armazenadas na

forma de tabelas e são requisitadas várias vezes e se uma implementação não planejada do catálogo poderá ocasionar um baixo desempenho no acesso aos dados.

3. ESTIMANDO O CUSTO DE UM PLANO

Para estimar o custo de várias estratégias de execução, devemos manter um rastro das informações no sistema que são os dicionários de dados, mantendo essas informações no catálogo do SGBD que serão acessadas pelo otimizador de consulta.

A Otimização de consulta consiste em traduzir uma consulta de usuário de alto nível em um plano eficiente para ter acesso aos dados do banco de dados. Então o otimizador se baseia em custos para refazer as consultas com baixo desempenho.

O otimizador verifica todos os possíveis planos para encontrar o plano com melhor desempenho de tempo, escolhendo uma estratégia para avaliar uma dada expressão relacional sem nenhuma garantia de que a estratégia escolhida para a implementação da consulta seja realmente a melhor. As estimativas de custos para a execução da consulta utilizam, assim, a consulta de menor custo. Essas estimativas são importantes, porém devemos ter cuidado com as estimativas, pois as opções de execução são muitas e não desejamos gastar o tempo da consulta fazendo excessivas estatísticas.

Diversos fatores estão relacionados com relação a estimativa de custo tais como o fato dos dados estarem de forma contínuas em disco ou não, se os dados estão ordenados ou não, procurar condições nos dados, levar os dados em uma memória temporária, entre outros.

A escolha do plano mais barato naturalmente necessita de um método para atribuição de custo ao plano em questão. A maioria dos sistemas de I/O em discos envolvidos, sendo que alguns também consideram a utilização da CPU. O plano de execução de uma consulta influencia diretamente no desempenho da mesma.

Algumas estatísticas que podem ser feitas incluem:

- Número de tuplas para cada relação.
- Número de valores distintos para cada índice secundário.

- A altura da B-tree para cada índece.
- Valor máximo e valor mínimo das chaves existentes.

Essas estatísticas nos permitem estimar os tamanhos dos resultados de várias operações, como também o custo para executar as operações.

Assim o otimizador de consultas deve calcular e comparar o custo de execução de cada plano e escolher o de menor custo.

4. ESTIMANDO TAMANHOS DOS RESULTADOS

O tamanho de uma estimativa é uma importante regra de estimativa de custo porque a saída de um operador pode ser à entrada de um outro operador, e o custo desse operador depende do tamanho dessa entrada.

Para conseguir uma estimativa do resultado de uma consulta contamos com o uso de um fator de redução. Estes fatores de redução são calculados assumindo que a distribuição dos valores do banco de dados é uniforme.

Quando tratamos de uma condição é importante notar que a seletividade de uma condição pode ser alta ou baixa, ou seja, uma condição pode eliminar poucos ou muitos registros do número total de registros armazenados. O otimizador pode armazenar estimativas de seletividade ou calculá-las. Por exemplo, a seletividade de um atributo chave em uma dada consulta é exatamente 1/n onde n é o número de elementos desta tabela.

O máximo de tuplas no resultado é produto das cardinalidades das relações da cláusula FROM. O Fator de Redução (FR) associado a cada termo, reflete o impacto do termo no tamanho do resultado.

Abaixo é mostrado algumas técnicas que o fator de redução utiliza na cláusula WHERE usando as estatísticas avaliada pelo catálogo do sistema:

- Cardinalidade = (NTuplas) * Fator de redução(FR)
- Termo coluna = valor (dado um índice I na coluna)
  - ✓ FR = 1/Nchaves(I)

- Termo coluna> valor (dado um índice I na coluna)
  - ✓ FR = Max(I) –(valor/Max(I))– Min(I)
- Termo coluna1 = coluna2
  - ✓ FR = 1/Max(NChaves(I1) ,NChaves(I2))
- Ausência de Índice: assumir FR = 1/10.

Estas estimativas do fator de redução são calculados assumindo que a distribuição dos valores do banco de dados é uniforme.

5. ÁLGEBRA RELACIONAL EQUIVALENTES

Uma regra de equivalência diz que uma expressão de duas formas são equivalentes: podemos transformar uma na outra preservando a equivalência. Manter a equivalência significa que as relações geradas pelas duas expressões têm o mesmo conjunto de atributos e contêm o mesmo conjunto de tuplas, embora seus atributos possam estar ordenados de forma diferente. As regras de equivalência são usadas pelo otimizador para transformar expressões em outras logicamente equivalentes. O processo é caro, tanto em termos de espaço quanto de tempo. Além disso, nem sempre é necessário gerar expressões usando regras de equivalências. Se as estimativas de custo da avaliação são levadas em consideração, um otimizador pode ser capaz de evitar o exame de algumas das expressões.

Nesse estágio, o otimizador pode escolher uma expressão equivalente avaliada e ainda obter o mesmo resultado.

Abaixo, listamos algumas equivalências da álgebra relacional.

5.1 SELEÇÕES

A operação de seleção é uma das mais caras que um sistema de banco de dados pode fazer. Essa operação carrega para a memória as tuplas que satisfazem a condição de seleção. Esta operação seleciona, a partir de uma relação de entrada, tuplas que satisfazem a um determinado predicado.

Duas importantes equivalências envolvendo seleção de operações.

Cascata:

$$\sigma_{c1 \wedge \ldots \wedge cn}(R) \equiv \sigma_{c1}(\ldots \sigma_{cn}(R))$$

Essa equivalência combina várias seleções dentro de uma seleção e troca por várias pequenas operações de seleções.

Comutativa:

$$\sigma_{c1}(\sigma_{c2}(R)) \equiv \sigma_{c2}(\sigma_{c1}(R))$$

Essa equivalência pode testar as condições c1 and c2 em qualquer ordem.

5.2 PROJEÇÕES

A operação de projeção, como a operação de seleção, reduz o tamanho das relações. Assim, toda vez que precisamos gerar uma relação temporária, é vantajoso aplicar imediatamente qualquer projeção possível.

Esse tipo de operação faz uma varredura seqüencial em todo o arquivo de dados.

Cascata:

$$\pi_{a1}(R) \equiv \pi_{a1}(\ldots(\pi_{an}(R)))$$

Essa equivalência é útil em conjunto com outras equivalências com a projeção comutativa com junção.

5.3 PRODUTO CARTESIANOS E JUNÇÕES

Duas importantes equivalências envolvem produto cartesiano e seleções. O produto cartesiano deve ser transformado em combinações de outras operações que produzam um resultado equivalente. Enquanto que a junção retorna a combinação de tuplas de duas relações que satisfazem um predicado.

Nessa fase operações como comutativa, associativa são importantes.

Um exemplo utilizando a operação comutativa, onde uma projeção comuta com uma seleção que usa somente atributos retidos pela junção. A seleção de um produto cartesiano pode ser transformada numa seleção.

Comutativa:

$$R \times S \equiv R \times S$$

$$S \infty R \equiv R \infty S$$

Esta propriedade é importante, pois nos permite escolher qual relação é a mais interna e uma outra para junção de duas relações.

Associativa:

Situação equivalente com junção e produto cartesiano são associativos.

$$R \times (S \times T) \equiv (R \times S) \times T$$

$$R \infty (S \infty T) \equiv (R \infty S) \infty T$$

Podemos realizar uma junção em R e S primeiro e então uma junção em T para o resultado, ou uma junção em S e T primeiro e então uma junção em R para o resultado. Por intuição por meio da associativa e do produto cartesiano, o resultado final contém as mesmas colunas.

6. ENUMERAÇÃO DE PLANOS ALTERNATIVOS DE CONSULTAS

Nesta seção iremos abordar formas de se estimar o custo de uma consulta, utilizando os métodos descritos anteriormente. Mesmo com esses métodos aproximados

ainda resta um problema: o número de planos alternativos é tipicamente muito grande. Veremos a seguir uma abordagem para esse problema.

6.1. CONSULTAS SOBRE UMA ÚNICA RELAÇÃO

Comecemos pelo caso mais simples: há apenas uma relação envolvida na consulta. Dessa forma, não iremos nos preocupar com junções, apenas com as operações de seleção e projeção, além de ordenação. Um plano para realizar uma consulta pode ter uma grande variação dado a existência ou não de índices:

- **Planos sem índices:** deve-se realizar "file-scan" para recuperar as tuplas e aplicar seleções em conjunto de projeções sobre as tuplas da relação. Em seguida, deve-se escrever as tuplas selecionadas (já com as projeções) em uma relação temporária. Em seguida, deve-se ordenar as tuplas sob algum campo (implementando, assim, a cláusula GROUP BY).
- **Planos utilizando índices:** podem ocorrer quatro casos:
    - **Single-index-access-path:** considera-se apenas um índice, as demais operações são realizadas posteriormente sem o uso de índices;
    - **Multiple-index-access-path:** utiliza-se mais de um índice e faz-se a intersecção dos resultados parciais obtidos;
    - **Sort-index-access-path:** caso o índice seja agrupado na ordem desejado pela cláusula GROUP BY, os atributos agrupantes são utilizados como prefixos da árvore;
    - **Index-only-access-path:** quando o índice contém todos os campos necessários, não é preciso acessar a relação correspondente.

6.1.1. ESTIMATIVA DE CUSTO PARA PLANOS DE UMA ÚNICA RELAÇÃO

Vamos considerar quatro possíveis casos:

- **Índice I na chave primária casa com a seleção:** caso a busca seja feita pela chave primária, o custo é *Altura(I) + 1* para um índice em árvore B+, ou aproximadamente 1,2 para índice em hash (seria 1 para um hash perfeito);
- **Índice agrupado I casa com uma ou mais seleções:** nesse caso, o custo é dado por *(NPáginas(I) + NPáginas(R)) * produto de FR das seleções casadas*;
- **Índice não-agrupado I casa com uma ou mais seleções:** de forma semelhante ao caso anterior, o custo é dado por *(NPáginas(I) + NTuplas(R)) * produto de FR das seleções casadas*. Note que como o índice não é agrupado, não se pode acessar diretamente as páginas corretas como no caso anterior, daí o aumento de custo implicado por *NTuplas(R)* no lugar de *NPáginas(R)*;
- **Varredura seqüencial do arquivo:** caso mais simples, o custo é dado por *NPáginas(R).*

Uma observação importante que se faz é que as consultas geralmente são feitas se eliminação de duplicatas nas projeções, mas quando na consulta é especificado DISTINCT, essa eliminação torna-se obrigatória.

6.2. CONSULTAS SOBRE MÚLTIPLAS RELAÇÕES

Quando mais de uma relação está envolvida dois novos problemas devem ser levados em conta: o algoritmo utilizado para a junção e o grande número de planos devido às propriedades de associatividade e comutatividade vistas anteriormente. Nesta seção iremos ver a abordagem utilizada pelo Sistema R: dentre todas as possíveis árvores de junção, serão consideradas apenas as árvores de profundidade à esquerda. Essa decisão é importante, pois além de realizar-se uma poda no número de planos a se analisar, considera-se somente as árvores que permitem o uso de pipelining – ou seja, evita a escrita de resultados intermediários

6.2.1. ENUMERAÇÃO DE PLANOS DE PROFUNDIDADE À ESQUERDA

Mesmo levando-se em conta apenas os planos com árvore de junção à esquerda, ainda restam vários graus de liberdade: a ordem em que se considera as relações, o método de acesso para cada relação e o método de junção utilizado para cada junção. Dessa forma, uma outra poda foi considerada no Sistema R: ao invés de se levar em consideração todas essas possibilidades, reliza-se um processo iterativo de N passos (N é o número de relações envolvidas), e para cada passo considera-se apenas o melhor plano envolvendo K relações ($1 \leq K \leq N$), da seguinte forma:

- **Passo 1:** encontra-se o melhor plano para cada relação;
- **Passo 2:** encontra-se a melhor forma de juntar a relação do(s) melhor(es) plano(s) obtido(s) no passo anterior (utilizada como relação mais externa) com uma outra relação.
- **Passo N:** encontra-se a melhor forma de juntar um (ou mais) plano de N-1 relações com a N-ésima relação que falta.

Em cada passo são mantidos apenas os planos mais baratos e eventualmente o planos mais barato para cada ordenação interessante das tuplas (ordenação que pode facilitar uma futura junção e/ou eliminar a necessidade de uma ordenação).

As cláusulas ORDER BY, GROUP BY, de agregação etc, são manipulados ocmo um passo final, usando ou um plano ordenado de forma interessante quanto ou um operador adicional de ordenação.

Um plano da etapa N-1 não é combinado com uma relação adicional a não ser que exista uma condição de junção entre eles, a não ser que todos os predicados no WHERE já tenham sido usados. Ou seja, se possível, evitar produtos cartesianos.

É importante ressaltar que apesar dessas podas no espaço de planos alternativos, ainda assim essa abordagem é exponencial no número de relações envolvidas.

6.2.2. ESTIMATIVA DE CUSTO PARA PLANOS DE MÚLTIPLAS RELAÇÕES

O número máximo de tuplas no resultado de uma consulta é dão pelo produto das cardinalidades das relações citadas na cláusula from (ou seja, o número de tuplas obtido pelo produto cartesiano entre todas elas). Dado que o fator de redução associado a cada

termo da cláusula WHERE reflete o impacto do termo na redução do tamanho do resultado (note: termos da forma conjuntiva normal) e que planos de múltiplas relações são construídos juntando-se uma nova relação por vez, tem-se que o custo do método de junção mais a estimativa da cardinalidade da junção resultam na estimativa de custo e do tamanho do resultado.

Resultados práticos indicam que bons planos permutam algoritmos de junção, que os piores planos (os que devem ser evitados!) possuem custo maior que 10000 e que o lucro de otimização é de cerca de 1000 vezes em relação a planos não otimizados.

6.3. CONSULTAS ANINHADAS

A abordagem do Sistema R para consultas aninhadas é: o bloco aninhado é otimizado independentemente, com cada tupla mais externa considerada como provedora de uma condição de seleção. Em contrapartida, o bloco mais externo é otimizado levando em consideração o custo de "chamar" e calcular o bloco aninhado.

A ordenação implícita desses blocos (é fixa a definição de qual bloco é o externo e qual é o aninhado) significa que algumas boas estratégias não são consideradas e, portanto, a versão não aninhada da consulta (se houver) costuma ser mais bem otimizada.

6.3.1. MELHORANDO O DESEMPENHO DE UMA CONSULTA ANINHADA

Quando se tem uma consulta aninhada, vimos que para cada tupla externa executa-se a consulta para o bloco aninhado. Entretanto, casos os blocos não sejam correlacionados (isto é, não se faz casamento de valores do bloco aninhado com o externo), tem-se uma situação melhor, na qual os blocos aninhados são executados somente uma vez, e o casamento é feito pelo resultado obtido do bloco aninhado. Portanto, sempre que possível, deve-se reescrever a consulta de forma que os blocos fiquem independentes entre si.

Outra otimização que se pode fazer é "achatar" a consulta, ou seja, reescrever a consulta (se possível) de forma a não se ter blocos aninhados. Como vimos anteriormente, consultas não aninhadas costumam ser mais bem otimizadas.

Bibliografia

Silberschatz, H. F. Korth, S. Sudarshan, “Database Systems Concepts – Third Edition” 1997.

2º R. Ramakrishnan, J. Gehrke, “Database Management Systems – Third Edition” 2003.

3º - Artigos diversos e WWW.