Avaliação de Consultas – Capítulo 12
Resumo

João Fabio Pegorin Di Lello – RA 047481
Marcelo Coutinho – RA 002092

7 de Junho de 2005

1. Introdução

Assim como existem várias formas de se escrever uma consulta, o SGBD também possui várias maneiras de executá-la. Para isso, é necessário que a consulta seja avaliada e que se encontre uma maneira bastante eficiente de executá-la. Neste resumo veremos os artifícios utilizados pelo SGBD para avaliar e executar as consultas em um tempo adequado. A fonte de informação utilizada foi o capítulo 12 do livro Database Management Systems [Ramakrishnan and Gerhrke 2003].

2. Visão Geral Sobre Realização de Consultas

Uma consulta SQL é convertida para uma árvore de operadores da álgebra relacional, e a partir dessa árvore escolhe-se o algorítmo mais adequado para cada operador. Este procedimento é chamado de plano, onde cada operador é tipicamente implementado usando-se a interface *Iterator*. O otimizador de consultas gera várias alternativas de planos e estima qual é o plano mais eficiente para ser executado.

Durante o processo de otimização de consultas temos duas questões fundamentais que são como estimar o custo de um plano, e quais planos devem ser considerados para uma determinada consulta.

A Figura 1 mostra o processamento realizado pelo SGBD para executar uma consulta.

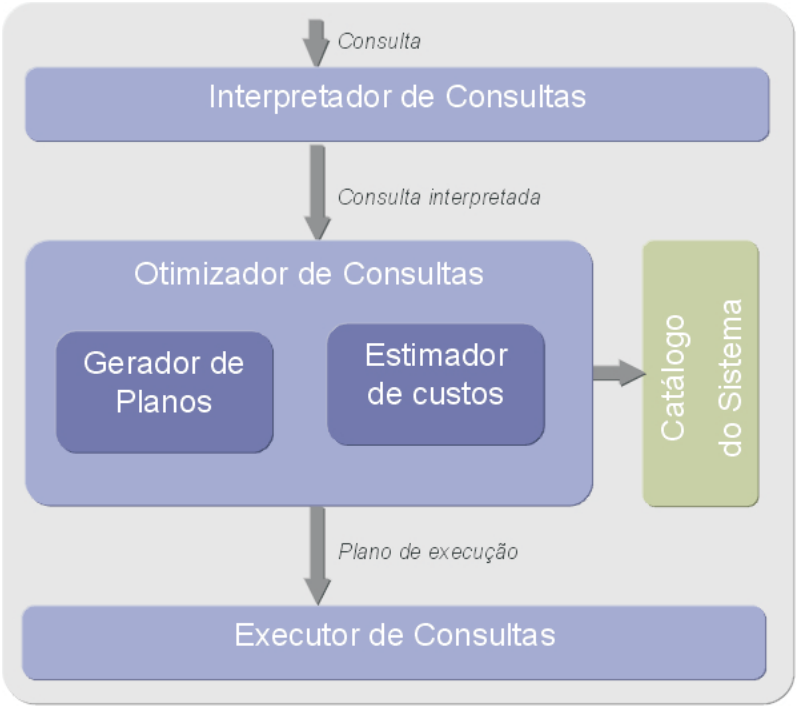


Figura 1. Fluxo de processamento de uma consulta.

3. Catálogos e Estatísticas do Sistema

O SGBD mantém várias informações sobre todos os objetos que integram um banco de dados. Essas informações, que também são armazenadas em um conjunto especial de tabelas, é chamada de catálogo do sistema ou dicionário de dados.

O Catálogo armazena informações gerais do sistema, como o tamanho do *buffer pool*, o tamanho de suas páginas, assim como informações específicas de tabelas (nome e tipo dos atributos, os índices, etc), índices e visões.

Além dessas informações, o catálogo armazena também alguns dados estatísticos sobre as tabelas e índices. Em geral, os dados estatísticos contêm pelo menos:

- **Cardinalidade:** O número de tuplas (*NTuples*) de cada tabela.
- **Tamanho:** O número de páginas (*NPages*) de cada tabela.
- **Cardinalidade do índice:** O número de chaves distintas (*NKeys*) de cada índice.
- **Tamanho do índice:** O número de páginas (*INPages*) de cada índice.
- **Altura do índice:** O número de níveis não-folha (*IHeight*) de índices *B+ tree*.
- **Faixa de domínio do índice:** O menor valor (*ILow*) e o maior valor (*IHigh*) presentes em cada índice.

A atualização dos dados estatísticos é um procedimento caro e por isso o SGBD atualiza estes dados do catálogo apenas periodicamente. Alguns SGBD armazenam em seus catálogos informações mais detalhadas, como por exemplo, histogramas dos valores de alguns campos.

4. Avaliação de operadores

4.1. Seleção

Os algorítmos para avaliação de operadores relacionais usam extensivamente idéias simples como indexação, iteração e particionamento. Existem várias formas de se recuperar uma tupla de uma tabela.

- Utilizando um índice hash quando temos na consulta um termo na forma $col = value$ e existe um índice em col.

  Se existir um índice hash em $< a, b, c >$ **a seleção casa somente com** $a = 5\ and\ b = 3\ and\ c = 6$**, não casa com** $b = 3$**, ou** $a = 5\ and\ b = 3$**.**
- Utilizando um índice em árvore, contanto que o(s) termo(s) na consulta esteja(m) na forma de um prefixo do índice em árvore.

  Ex: Índice em árvore em $< a, b, c >$ **casa com a seleção** $a = 5\ and\ b = 3$**, e** $a = 5\ and\ b > 6$**, mas não** $b = 3$**.**
- Realizando uma varredura de todos os dados da tabela ou do índice.

As formas de se recuperar uma tupla de uma tabela são chamadas de Caminhos de Acesso. Um método bastante utilizado consiste em achar o caminho de acesso mais seletivo, usando-o para recuperar tuplas, e aplicando os termos remanescentes que não casam com o índice. Outro método consiste em utilizar um índice para seleções, e neste caso o custo depende do nº de tuplas qualificadas, e do uso de clustering. Com isto o custo para achar dados de entrada qualificados é, em geral, pequeno, mas o custo de recuperar registros pode ser alto caso estes não estejam clusterizados. Veremos adiante mais detalhes sobre a seletividade de um termo.

4.2. Projeção

```
SELECT    DISTINCT R.sid , R.bid
FROM      Reserves R
```

Em uma operação de projeção, a parte cara é a remoção de duplicatas, por isso usa-se a técnica de Ordenação, que consiste em ordenar em $< sid, bid >$ e remover as duplicatas. (Pode-se otimizar descartando dados não necessários durante a ordenação.)

Outra opção é utilizar a técnica de Hashing , onde fazemos um hash em $< sid, bid >$ criando-se assim partições. Carregam-se as partições para a memória uma por vez, fazendo-se uma estrutura de hash na memória, e eliminando-se duplicatas.

Agora se há um índice com ambos R.sid e R.bid na chave de busca, pode ser mais barato ordenar os dados de entrada.

4.3. Junção

Em qualquer banco de dados relacional sempre há algum tipo de relacionamento entre informações armazenadas em diferentes tabelas. Dessa forma, muito frequentemente, há a necessidade de recuperar dados de diferentes tabelas segundo algum critério. Estas operações são chamadas operações de junção.

As operações de junção estão entre as operações mais custosas na execução de uma consulta. Contudo, estas operações têm sido amplamente estudadas e os sistemas gerenciadores de banco de dados geralmente implementam vários algorítmos que tratam as junções de forma bastante otimizada. Abaixo estão descritas algumas técnicas de junção.

Índices em laços aninhados (*Index Nested Loops)*

Esta técnica de junção pode ser utilizada sempre que houver um índice na coluna de união de uma das tabelas da relação. A técnica consiste em percorrer cada um dos registros da tabela que não possui o índice (T1) e para cada registro de T1 recuperar o registro correspondente na tabela que possui o índice (T2). Dessa forma, o SGBD consegue aproveitar o índice da tabela T2 agilizando a recuperação da informação da tabela T2.

Classificação&Intercalação (*Sort-Merge)*

Caso não haja índice nas colunas de união da relação não é possível utilizar a técnica de junção de índice em laços aninhados. Neste caso, é possível ordenar as duas tabelas da relação pelas colunas de união e depois percorrer as duas tabelas de forma intercalada, encontrando os registros de T1 que casam com T2 e vice-versa. Esta técnica de junção é chamada de Classificação&Intercalação.

Bloco em laços aninhados (*Block Nested Loop)*

Esta técnica consiste em ler uma certa quantidade de páginas de uma tabela T1 e percorrer todos os registros de uma segunda tabela T2 para encontrar os registros correspondentes. Isto é feito até que todas as páginas da tabela T1 tenham sido lidas.

5. Otimização de consultas & Estimativa de custos

As linguagens de consulta relacional nos permitem escrever uma mesma consulta de diversas formas sem que o seu resultado final seja alterado. Da mesma maneira, uma consulta pode ser executada de diversas formas e a diferença entre o tempo de execução da forma mais eficiente e da forma menos eficiente pode ser muito grande. De fato, não podemos esperar que o melhor plano seja sempre executado, mas esperamos que os piores planos sejam evitados.

O Otimizador de consultas é o responsável por identificar um plano de execução eficiente. Ele gera vários planos alternativos e elege o que possui o menor custo estimado.

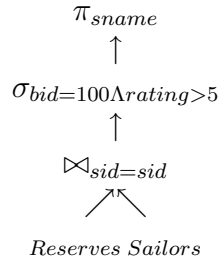
Dada uma consulta qualquer, sempre é possível representá-la na forma de uma expressão de álgebra relacional. A seguinte consulta:

```
SELECT S.sname
FROM   Reserves R, Sailors S
WHERE  R.sid = S.sid AND
       R.bid = 100 AND S.rating > 5
```

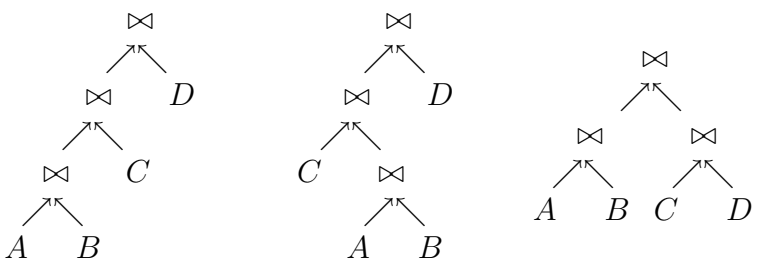
Pode ser expressa em álgebra relacional da seguinte forma:

$$\pi_{sname}(\sigma_{bid=100 \Lambda rating>5}(Reserves \bowtie_{sid=sid} Sailors))$$

A expressão algébrica sugere como executar a consulta – primeiro realiza a junção das tabelas *Reserves* e *Sailors*, depois faz as seleções e finalmente extrai a projeção do atributo *sname*. É possível também representar a expressão como uma árvore com os operadores da álgebra relacional.



O Otimizador de consultas sempre converte a consulta a ser otimizada em uma árvore para poder analisar a consulta montar as várias alternativas de planos de execução. Quanto maior o numero de relações envolvidas na consulta, maior o número de alternativas para o otimizador analisar. Como o universo de possíveis alternativas pode ser muito grande, o Otimizador deve considerar somente um conjunto de alternativas, do contrário, pode-se gastar mais tempo calculando qual é o plano mais eficiente do que executando o plano. Para isso, quando há mais de duas relações na consulta, o Otimizador considera apenas os planos com profundidade a esquerda (representado pelo primeiro item da figura abaixo).



Como a consulta é muitas vezes é composta de vários operadores, às vezes é interessantes realizar um *pipeline* do resultado de um operador para outro operador sem criar uma tabela temporária para armazenar os dados intermediários. O *pipeline* evita que os dados sejam salvos em tabelas temporárias e tenham que ser lidos novamente para continuar o processamento da consulta. Quando os dados são salvos em uma tabela temporária, dizemos que estes dados estão *materializados*. O *pipeline* causa uma sobrecarga menor do sistema se comparado com a *materialização* e sempre é escolhido caso seja suportado pelo algorítmo do operador.

Outra técnica muito utilizada é o adiantamento de seleções e projeções. Esta técnica consiste em aplicar uma ou mais seleções ou projeções antes de realizar a junção das tabelas. O adiantamento das seleções implica em menos tuplas a serem trabalhadas e isso melhora o tempo das junções (que pode ser bem alto, dependendo do tamanho da tabela). O adiantamento das projeções diminui a quantidade de colunas a serem lidas das tabelas e são especialmente úteis quando há necessidade de salvar dados em tabelas temporárias.

Para estimar o plano mais eficiente, o Otimizador utiliza os dados do catálogo do SGBD. Dessa forma, o Otimizador consegue estimar o tamanho do resultado de cada operador na árvore da consulta. Este resultado estimado é chamado de fator de redução.

Na consulta:

```
SELECT listaAtributos
FROM   listaRelação
WHERE  termo1 AND termo2 AND ... termok
```

O número máximo de tuplas resultantes corresponde ao produto da cardinalidade de todas as relações da cláusula FROM. Mas se houver algum termo na cláusula WHERE é bem possível que esse número seja reduzido. Assim, depois de calculado o fator de redução para cada termo, a cardinalidade do resultado da consulta será o produto da cardinalidade de todas as relações vezes o produto de todos os fatores de redução.

Para calcular os fatores de redução o Otimizador assume que todos os termos da cláusula WHERE são independentes.

- $col = valor$ – Fator de Redução $1/NKeys(I)$
- $col1 = col2$ – Fator de Redução $1/MAX(NKeys(I1), NKeys(I2))$
- $col > value$ – Fator de Redução $(High(I) - value)/(High(I) - Low(I))$

Os fatores de redução acima são válidos quando as colunas utilizadas nos termos possuem índices. Caso a coluna do termo não possua índice o Otimizador assume o valor $1/10$ como fator de redução.

6. Conclusão

Existem várias formas de executar uma consulta. O otimizador de consultas é o responsável por encontrar uma forma eficiente de executar essa consulta. Ele utiliza dados do catálogo do sistema para obter dados estatísticos e dessa forma estimar o custo de cada plano de execução. É muito importante avaliar e otimizar as consultas pois a performance do sistema pode ser drasticamente afetada dependendo do plano de execução escolhido.

7. Bibliografia

Referências

Ramakrishnan, R. and Gerhrke, J. (2003). *Database Management Systems*. McGraw-Hill, 3rd edition.