Algoritmos para Processamento e Otimização de Consultas

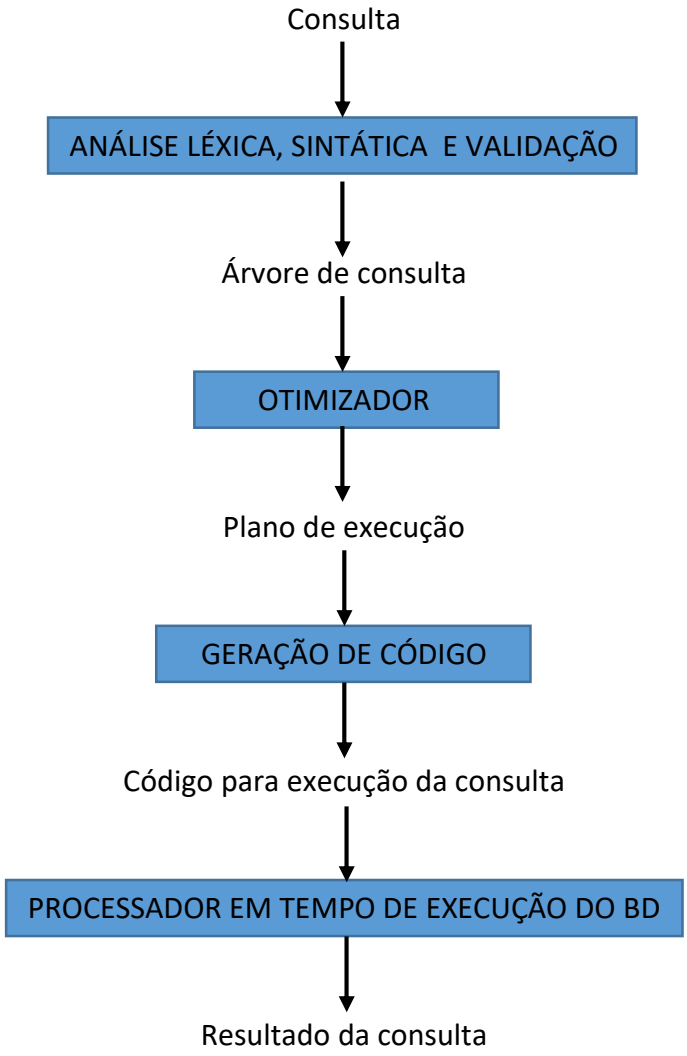
Ref: Sistemas de Banco de Dados, Elmasri e Navathe, 4ª ed. Addison Wesley

Trata-se de discutir as técnicas utilizadas por um SGBD para processar, otimizar e executar consultas expressas em uma linguagem de alto nível – SQL.

O otimizador é responsável pela análise léxica (identificar os itens léxicos da linguagem, como: palavras-chave da linguagem, nomes de tabelas, atributos e relacionamentos); pela análise sintática, que verifica se a consulta está formulada de acordo com as regras sintáticas da linguagem e pela validação da consulta (verificação se são válidos os atributos e nomes de relacionamentos e se são nomes com significados semânticos no esquema do BD)

A consulta é transformada em uma representação interna denominada árvore de consulta. É possível ainda representar a consulta por um grafo. Com base nessas informações o SGBD gera uma estratégia de execução para obter o resultado da consulta.

Passos do processamento de consultas de alto nível:



Conceitos sobre otimização de consultas:

1) Blocos de Consultas

Em geral, as consultas SQL são decompostas em **blocos de consultas** que podem ser escritos e otimizados em operadores da álgebra relacional.

Um **bloco de consulta** contém uma única expressão **SELECT-FROM-WHERE** e as cláusulas **GROUP BY** e **HAVING** (se elas fizerem parte do bloco).

Como SQL contém operadores de agregação, (MAX, MIN, SUM e COUNT) estes devem ser incluídos na álgebra relacional dita estendida.

Seja o esquema abaixo:

EMPREGADO (PNOME, MINICIAL, UNOME, SSN, ENDERECO, SEXO, SALARIO, SUPERSSN, DNO)
DEPARTAMENTO (DNOME, DNUMERO, GERSSN, GERDATAINICIO)
DEPTO_LOCALIZACOES (DNUMERO, DLOCALIZACAO)
PROJETO (PJNOME, PNUMERO, PLOCALIZACAO, DNUM)
TRABALHA_EM (ESSN, PNO, HORAS)
DEPENDENTE (ESSN, NOME_DEPENDENTE, SEXO, DATANSC, PARENTESCO)

Seja a seguinte consulta SQL:

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

Essa consulta contém uma subconsulta aninhada, assim, é decomposta em dois blocos.

O bloco interno:

```
(SELECT MAX (SALARIO)
FROM EMPREGADO
WHERE DNO=5)
```

E o bloco externo:

```
SELECT UNOME, PNOME
FROM EMPREGADO
WHERE SALARIO  >  c
```

Onde c representa o resultado do bloco interno.

O OTIMIZADOR DE CONSULTAS escolhe um plano de execução para cada bloco.

O bloco interno é avaliado apenas uma vez para produzir o salário máximo, que posteriormente é utilizado (constante c) pelo bloco externo.

Essa consulta é dita consulta aninhada não-correlacionada.

Cabe ressaltar que é muito mais complexo otimizar as consultas aninhadas correlacionadas, nas quais uma variável de tupla do bloco externo aparece na cláusula WHERE do bloco interno.

2) Algoritmos para ordenação externa

Ordenação externa refere-se a algoritmos de ordenação em arquivos que não cabem em memória principal, ou seja, são armazenados em disco.

A ordenação é um dos algoritmos primários usados no processamento de consultas. Sempre que uma consulta especifica a cláusula ORDER BY, o resultado da consulta deve ser ordenado. A ordenação é também usada nos algoritmos sort-merge (ordenação-fusão) usados no join e em outras operações (como: union e intersection) e também na eliminação de duplicatas em operação de projeção (DISTINCT em SELECT).

A ordenação pode ser evitada se houver um índice apropriado para os registros.

O típico algoritmo de ordenação externa usa uma estratégia sort-merge, que inicia ordenando pequenos subarquivos chamados **runs** (resultado parcial) do arquivo principal. Em seguida realiza o merge entre os runs ordenados, criando cada vez maiores subarquivos ordenados até chegar a um único arquivo ordenado. O algoritmo sort-merge (entre outros algoritmos) exige espaço em memória principal (buffer) onde a ordenação e a fusão são realizadas de fato.

O algoritmo realiza a leitura de blocos de dados para a memória onde são ordenados e gravados novamente em disco, em arquivos temporários. Em seguida os blocos ordenados são lidos para a realização da operação de fusão (merge) até que todo o arquivo esteja ordenado.

3) Algoritmos para operações SELECT e JOIN

Existem muitas opções para a operação SELECT. Vamos analisar estas opções com base no esquema indicado anteriormente (empregado, projeto, departamento e dependente).

Op1: $\sigma_{\text{SSN='123456789'}}$ **(EMPREGADO)**

Op2: $\sigma_{\text{DNUMERO>5}}$ **(DEPARTAMENTO)**

Op3: $\sigma_{\text{DNO=5}}$ **(EMPREGADO)**

Op4: $\sigma_{\text{DNO=5 AND SALARIO>30000 AND SEXO='F'}}$ **(EMPREGADO)**

Op5: $\sigma_{\text{ESSN='123456789' AND PNO=10}}$ **(TRABALHA_EM)**

<u>3a) Métodos de busca para seleções simples:</u>

S1. Busca linear (força bruta): Recupera cada registro do arquivo se seus valores satisfazem a condição de seleção.

S2. Busca binária: Se a condição de seleção envolver uma comparação de igualdade em atributo chave para o qual o arquivo está ordenado, a busca binária é mais eficiente do que a busca linear. Ex: Op1.

S3: Utilização de um índice primário (ou chave de hash): Se a condição de seleção envolver uma comparação de igualdade em um atributo-chave em um índice primário (ou chave de hash) é usado o índice ou chave de hash para recuperar o registro. Ex: Op1. Essa operação recupera apenas um registro.

S4: Uso de um índice primário para recuperar múltiplos registros: Se a condição de seleção for >, >=, < ou <= em um campo chave com um índice primário, por exemplo, em Op2 onde DNUMERO>5, é usado o índice para encontrar o registro que satisfaça a condição de igualdade correspondente (DNUMERO=5) e depois são recuperados os registros seguintes no arquivo, se **ORDENADO**. Se a condição for DNUMERO < 5, são recuperados os registros anteriores.

S5: Utilização de um índice cluster para recuperar múltiplos registros: Se a condição de seleção envolver uma comparação de igualdade **em um atributo que não seja chave** com um índice clustering, por exemplo, DNO=5 na Op3, use o índice para recuperar todos os registros que satisfazem a condição.

S6: Utilização de um índice secundário (árvore B+) em uma comparação de igualdade: Este método pode ser usado para **recuperar um único registro, se o campo de indexação for uma chave** (possui valores únicos) ou para **recuperar múltiplos registros se o campo de indexação não for chave**. Pode também ser usado para comparações <, <=, >, >=.

3b) Métodos de busca para Seleções complexas

3b1 - Operações com condições conjuntivas

Se uma condição da operação SELECT é formada por diversas condições simples conectadas pelo conectivo lógico AND, é denominada como **seleção conjuntiva** (por ex: Op4), o SGDB pode usar os seguintes métodos para implementar a operação:

S7: Seleção conjuntiva utilizando um índice individual: Se a seleção possuir um caminho de acesso que permita o uso de um dos métodos de S2 a S6, use a condição para recuperar os registros e depois verifique se cada registro recuperado satisfaz as condições restantes.

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 (ou estrutura hash) para os campos (ex: se houver um índice criado para a chave composta (ESSN, PNO) no arquivo TRABALHA_EM, na Op5, pode-se utilizar o índice diretamente.

S9: Seleção conjuntiva por meio de interseção de registros: Se existirem índices secundários (ou outros caminhos de acesso) para mais de um dos campos envolvidos nas condições simples da condição conjuntiva e se os índices incluírem ponteiros para registros (e não para blocos), cada índice pode ser usado para recuperar um conjunto de ponteiros de registros que satisfazem a condição individual. A interseção desses conjuntos de ponteiros resulta naqueles que satisfazem a condição conjuntiva e são usados para acessar o conjunto resposta da consulta. Se apenas algumas das condições possuírem índices secundários, após a interseção destes, cada registro resultante será testado para verificar se satisfaz às demais condições.

RESUMO: Sempre que uma condição individual especifica uma seleção, o SGBD verifica se existe um caminho de acesso associado ao atributo envolvido naquela condição. Se houver, o caminho será utilizado. Caso contrário, a abordagem de força bruta será utilizada. (ex: Op1, Op2 e Op3).

Em uma condição de seleção conjuntiva, quando mais de um atributo envolvido nas condições possuírem um caminho de acesso, o otimizador deve escolher o caminho que recupera o menor número de registros, de maneira eficiente, por meio de uma estimativa de custos e escolha do método de menor custo estimado.

Quando o otimizador está escolhendo entre múltiplas seleções simples em uma condição de seleção conjuntiva, ele, em geral, considera a seletividade de cada condição. Essa seletividade é definida pela razão entre o número de registros que satisfazem a condição e o número de registros do arquivo, ou seja, é um número entre 0 e 1. A seletividade = 0 significa que nenhum registro satisfaz a condição e a seletividade 1 significa que todos os registros a satisfazem.

As estimativas de seletividade dos atributos são mantidas frequentemente no catálogo e são informações utilizadas pelo otimizador.

3.b.2 - Operações com Condições Disjuntivas

São mais difíceis de processar do que as operações conjuntivas.

Op4A: $\sigma_{\text{DNO=5 OR SALARIO>30000 OR SEXO='F'}}$ **(EMPREGADO)**

Neste tipo de operação, pouca otimização pode ser feita porque os registros que satisfazem a condição disjuntiva são formados pela **união** dos registros que satisfazem as condições individuais. Assim, se qualquer uma das condições não possuir um caminho de acesso, será adotada a força bruta. A otimização pode ocorrer apenas se houver um caminho para toda condição. Assim, pode-se recuperar os registros que satisfazem cada condição e aplicar uma operação de união para evitar duplicidades.

3c) Implementação da operação JOIN

A junção é a operação que mais consome tempo no processamento de consultas. Como muitas das operações de junção encontradas nas consultas são variações de operações Equi Join e Natural Join, vamos considerar apenas estas.

Existem muitas maneiras de implementar uma junção em duas vias (junção em dois arquivos). As junções que envolvem mais de dois arquivos são chamadas de junções de múltiplas vias. O número de formas possíveis de implementar uma junção de múltiplas vias aumenta rapidamente. Aqui vamos estudas as técnicas para implementação das junções de duas vias. Iremos considerar apenas os algoritmos para implementar a junção na forma:

$$R \bowtie_{A=B} S$$

Em que A e B são atributos compatíveis com os domínios de R e S respectivamente.

Vamos avaliar quatro técnicas que são as mais comuns para executar este tipo de junção, utilizando as seguintes operações:

Op6: EMPREGADO $\bowtie_{\text{DNO=DNUMERO}}$ DEPARTAMENTO

Op7: DEPARTAMENTO $\bowtie_{\text{GERSSN=SSN}}$ EMPREGADO

J1: Junção de laços aninhados (nested-loop) (força bruta): O método funciona da seguinte forma: Para cada registro t em R (laço externo) recupere cada registro s de S (laço interno) e teste se os dois registros atendem a condição de junção $t[A] = s[B]$.

J2: Junção de laço único (single-loop) (utiliza uma estrutura de acesso para recuperar os registros correspondentes à junção): Se existir um índice (ou chave de hash) para um dos dois atributos da junção, por exemplo B de S, recupere cada registro t em R, um por vez (laço único), e, depois, use a estrutura de acesso para recuperar diretamente todos os registos s correspondentes em S, que satisfaçam $s[B] = t[A]$.

J3: Junção sort-merge (ordenação-fusão): Se os registros de S e R estiverem ordenados fisicamente pelos valores dos atributos de junção A e B, respectivamente, pode-se implementar a junção da maneira mais eficiente possível. Ambos os arquivos são varridos simultaneamente na ordem dos atributos de junção, fazendo a correspondência dos registros que possuem os mesmos valores para A e B.

Se os arquivos não estiverem ordenados eles devem ser classificados por uma ordenação externa. Neste caso, pares de blocos são copiados para buffers de memória e os registros de cada arquivo são varridos apenas uma vez, para fazer a correspondência com o outro arquivo (são atributos chave).

Se A e B não forem atributos chave, uma variação da junção sort-merge pode ser usada quando existirem índices secundários para ambos os atributos de junção. Os índices são usados para acessar os registros na ordem dos atributos de junção, mas se forem índices secundários, os registros estarão espalhados pelo disco ou seja, este método pode ser bastante ineficiente.

J4: Junção-hash (hash-join): Os registros dos arquivos R e S são particionados (hashed) em um mesmo arquivo hash, usando a mesma função hash, com os atributos da junção A de R e B de S. Primeiro, uma passagem pelo arquivo com menor número de registros (digamos, R) coloca seus registros nos buckets (buffer com registros com o mesmo resultado de hash) do arquivo hash. Em seguida, na segunda fase, uma única passagem pelo outro arquivo (S) usa a função para determinar o local adequado e aquele registro é combinado com todos os registros S naquele bucket. Caso o arquivo hash não caiba em memória, os buckets processados são armazenados em disco.

3d) Implementação de operações de agregação

Os operadores de agregação (MIN, MAX, COUNT, AVERAGE, SUM) quando aplicados a uma tabela inteira podem ser computados por meio da varredura da tabela ou pelo uso de um índice apropriado, se estiver disponível. Por exemplo, seja a consulta SQL:

```
SELECT MAX(SALARIO)
FROM EMPREGADO;
```

Se existir um índice ascendente para SALARIO na tabela EMPREGADO, o otimizador pode decidir pelo uso do índice de maior valor, seguindo pelo índice mais à direita em cada nó do índice, percorrendo desde a raiz até a folha mais à direita. O mesmo pode ser feito na operação MIN, usando o índice de menor valor (percorrendo o índice desde a raiz até a folha mais à esquerda). Isso é mais eficiente do que a varredura completa da tabela EMPREGADO.

O índice também pode ser utilizado para as agregações (COUNT, AVERAGE, SUM), mas apenas se o índice for denso (ou seja, se existir uma entrada de índice para cada registro do arquivo principal). Neste caso o cálculo é feito com os valores no índice. No caso de um índice esparso, o número real de entradas no índice é utilizado para corrigir o cálculo, exceto no caso de COUNT DISTINCT, onde o número de valores distintos pode ser contado a partir do próprio índice.

Se a cláusula GROUP BY for utilizada, o operador de agregação deve ser aplicado separadamente a cada grupo de tuplas. Assim, a tabela deve ser particionada em grupos de tuplas onde cada grupo possui o mesmo valor no atributo de agrupamento.

EX: Seja a seguinte consulta

```
SELECT DNO, AVG(SALARIO)
FROM    EMPREGADO
GROUP   BY DNO;
```

O método usado para estas consultas consiste em usar primeiro a ordenação ou o hashing para particionar o arquivo nos grupos. Em seguida, o algoritmo calcula a função de agregação para as tuplas, em cada grupo. Os grupos tem o mesmo valor para o atributo da agregação. No exemplo, o conjunto de tuplas para cada número do departamento é agrupado em uma partição e a média dos salários seria calculada para cada grupo ou partição. Se existir um índice clustering para o atributo de agrupamento então os registros já estarão particionados. Neste caso basta aplicar os cálculos a cada grupo.

3e) Implementação do outer join

A junção externa pode ser implementada via modificação dos algoritmos de junção, tais como a junção de loops aninhados ou a junção de loop único. Por exemplo, para obter a junção externa à esquerda usa-se a relação à esquerda como laço externo ou laço único porque toda tupla na relação à esquerda deve aparecer no resultado. Se houver tuplas correspondentes na outra relação, as tuplas que participam da junção serão produzidas e salvas no resultado. No entanto, se nenhuma tupla correspondente for encontrada, a tupla também é incluída no resultado, porém é completada com valores NULL. Os algoritmos de sort-merge e de junção hash também podem ser estendidos, para gerar as junções externas.
Ex. de junção externa:

```
SELECT UNOME, PNOME, DNOME
FROM (EMPREGADO LEFT OUTER JOIN DEPARTAMENTO ON DNO=NUMERO)
```

O resultado dessa consulta é uma tabela com os nomes dos empregados e os departamentos a eles relacionados.