Processamento e Otimização de Consultas

Técnicas de otimização Baseadas em estimativa de custo

- Implementação das operações
- Exemplos de funções de custo

Processamento e otimização de consultas

- O otimizador de consulta não deve depender somente de regras heurísticas (árvore algébrica)
  - Será necessário estimar e comparar os custos da execução da consulta utilizando diferentes estratégias de execução
  - Escolher estratégia com menor custo
  - O número de estratégias considerado deve ser limitado para não reduzir o desempenho
- Custos da execução de uma consulta
  - Considera: Acesso à memória secundária, armazenamento, processamento, uso de memória, comunicação
- Algoritmos para ordenação externa
  - Implementação do select e join
  - Implementação do project e operações de conjuntos

Abordagens para execução da consulta

- ***Materializada:*** o resultado da operação é armazenado como uma relação temporária.
- ***Pipeline*** : resultado de uma operação é transferido para a próxima operação sem a criação de uma tabela em disco.
  - Economia de custos de armazenamento e de leitura posterior
  - Sempre que o algoritmo do operador para o qual é transferido o resultado permitir, a técnica de pipeline é utilizada.

Estimativa do Tamanho do Resultado

SELECT <lista de atributos>
FROM < lista de relações R1,...,Rk >
WHERE <cond1 ^ cond2 ^....^condn>

- Nr máximo de tuplas no resultado = M1xM2x....Mk, onde Mi = tamanho de Ri
- Cláusula WHERE atua como um *redutor* desta estimativa
- Cada condição do WHERE tem o seu *fator de redução* próprio.Exemplos:
  - R.A = valor →*Fator de redução* = 1/Nr chaves(I), caso exista um indice I com chave A para a relação R
  - R.A IN (Subconsulta)
    - Fator de redução: M/N onde
      - M = tamanho do resultado da Subconsulta
      - N = número de valores do atributo R.A

Algoritmos para ordenação externa

Estratégia mergsort

- Estratégias de ordenação para arquivos grandes, armazenados em disco
- Exige espaço de buffer na memória principal

Ordenação

- **Corridas (Runs) que caibam nos buffers são lidas para a memória principal.**
- **As corridas são ordenadas (classif. Interna) e escritas de volta no disco**
- **O tamanho da corrida e o número inicial de corridas(n_R) são determinados a partir do número de blocos (b) e espaço de buffer (n_B)**

Fusão das corridas ordenadas

- **As corridas são fundidas em uma ou mais passagens**
- **Em cada passagem, precisamos 1 bloco de buffer para a corrida e 1 bloco de buffer para resultados**

Implementação do Select

Algoritmo de busca para seleções simples

S_1: Busca linear (força bruta)→ recupera cada registro do arquivo e verifica a condição de seleção

S_2: Busca binária→ se a condição de seleção envolve igualdade de atributo chave para o qual o arquivo está ordenado

S_3: Índice primário ou chave hash para recuperar um único registro

S_4: Índice primário para recuperar múltiplos registros: Se a condição for <, >, >=, <= em um campo chave com um índice primário

S_5: Índice clustering para recuperar múltiplos registros→ comparação de igualdade em atributo que não é chave

S_6: Índice secundário (arvore B^+) em uma comparação de igualdade → se o campo for chave recupera um único registro, e se o campo não for chave recupera múltiplos registros.

Implementação do Select

Algoritmo de busca para seleções complexas

São consideradas seleções complexas aquelas que possuem uma condição conjuntiva (várias condições simples combinadas usando o operador and)

S_7: Índice individual: se o atributo envolvido em qualquer uma das condição simples possuir um caminho de acesso que permita a utilização de um dos algoritmos de (2) a (6)

S_8: Índice composto: se dois ou mais atributos estiverem envolvidos numa igualdade e houver um índice composto ou estrutura hash envolvendo estes atributos.

S_9: Interseção de registros: se índices secundários estiverem estiverem disponíveis para mais de um dos atributos de uma das condições simples e se os índices envolverem ponteiros para registros em vez de blocos

- recuperar o conjunto de ponteiros de registros que satisfaça a condição
- Fazer a interseção destes ponteiros de registros
- Recuperar os registros apontados pelo resultado da interseção

Implementação do Select

Resumo

Se o select tem apenas uma condição individual:

- Se existe caminho de acesso envolvendo o atributo → o caminho de acesso é utilizado
- Caso contrário → utilizar força bruta

Se o select possui condições conjuntivas:

- A otimização é necessária sempre que mais de um atributo possuir caminho de acesso
- O otimizador deve escolher o caminho que recupera o menor número de registro, de maneira mais eficiente e de menor custo
- O otimizador considera a seletividade de cada condição

Seletividade (sl)

sl= nº de reg. que satisfazem a condição/ nº total de registros do arquivo
Sl=0 → nenhum registro satisfaz a condição
Sl=1 → todos os registros satisfazem a condição

Implementação do Join (⋈)

Vamos considerar apenas: Equijoin e natural join e Junção de 2-vias (envolve atributos dos 2 arquivos)

Algoritmos: considerando as relações (arquivos) R e S e os atributos A e B envolvidos na junção com A∈R e B∈ S

J_1: Laços aninhados (força bruta): para cada registro t de R (laço externo) recupera um registro s de S (laço interno) e testa se t[A]=s[B]

J_2: Laço aninhado indexado (laço único) → se existir um índice (ou chave hash) para um dos atributos, p.e. para B ∈ S. Recupera cada registro t de R e usa o caminho de acesso recuperar s de S que satisfaçam t[A]=s[B]

J_3: Mergesort: se os registros de R e S estiverem ordenados fisicamente pelos atributos A e B. Os arquivos são varridos simultaneamente comparando-se t[A] com s[B].

J_4: Hash: Os arquivos R e S são particionados em um mesmo arquivo hash utilizando a mesma função com os atributos A e B como chave hash

- Fase de separação: passa uma vez pelo arquivo com menor número de registros (p.e. R) e coloca os seus registros no cesto (buckets) do arquivo hash
- Fase de sondagem:Passa uma vez por S e (1) procura cesto apropriado, e (2) combina aquele registro com todos os registros de R que estão no cesto

Implementação do Project

- É direta se a lista de projeção inclui o atributo chave
- Se a lista de atributos não inclui a chave → registros duplicados devem ser eliminados
  - Ordena o resultado e elimina as duplicatas ou
  - Faz o hashing do arquivo incluindo-se o registro (tupla) no cesto apropriado, apenas se não for duplicata

Implementação de Operações de Conjunto

- Produto cartesiano
  - É uma operação muito cara pois, o resultado contém um registro para cada combinação de registros das relações envolvidas e todos os atributos delas
  - Deve ser evitada na execução da consulta.
- União, Interseção e diferença
  - Usando variações do mergesort
    - Ordena as duas relaçoes segundo o mesmo atributo
    - Uma única varredura em cada arquivo é suficiente para produzir o resulatado
  - Usando Hashing
    - Um dos arquivos é particionado
    - Aplica-se o hash ao outro arquivo para verificar o primeiro de acordo com a operação.
      Por exemplo: se a oiperação é de União ao fazer o hashing do 2º arquivo, inclua o registro no resultado apenas se ele não apareceu no hashing do primeiro.

Componentes de custo de uma consulta

1. Custo de acesso do armazenamento secundário
   - Custo da busca, leitura e escrita de blocos de dados em disco
   - O custo é afetado por: tipos de estrutura de acesso, organização dos blocos no disco(se são adjacentes ou não)
2. Custo de armazenamento
   - Custo de armazenamento de arquivos temporários gerados por uma estratégia de execução
3. Custo de computação
   - Custo das operações sobre os buffers na memória principal. Incluem: busca, ordenação, fusão durante a junção e cálculos sobre os atributos
4. Custo do uso da memória
   - Custo do número de buffers de memória utilizados durante a execução da consulta
5. Custo de comunicação
   - Custo do transporte da consulta e de seus resultados

Componentes de custo de uma consulta

Ênfase a diferentes Componentes de custo

- Para banco de dados grande
  - → Ênfase: custo de acesso ao armazenamento secundário
  - → Funções de custo ignoram outros fatores e comparam as estratégias de execução da consulta em termos de n.º de transferências de blocos entre disco e memória principal
- Banco de dados pequeno
  - → Ênfase: Custo de computação
  - → A maioria dos arquivos envolvidos na consulta está na memória principal
- Banco de Dados distribuído
  - → O custo de comunicação também deve ser levado em conta por ter muitos sites envolvidos

Informações do catálogo utilizadas nas funções de custo

- r – número de registros do arquivo
- b – número de blocos ou np – número de páginas
- brf – fator blocagem (nr de registros por bloco)
- x – número de níveis de índice (para cada índice multinível (primário, secundário ou clustering)
- b_{li} – número de blocos do primeiro nível de índice
- d – número de valores distintos de um atributo
- sl – seletividade (em função de tuplas que satisfazem uma condição de igualdade)
- s – cardinalidade da seleção: é calculada como s=sl* r

Para atributo chave→d=r; sl=1/r ; e s=1

Para atributos não chave → sl=1/d; s= sl*r=r/d

Exemplos de funções de Custo

Select

SE considerarmos o custo em termos de nº de transferências (acessos) de blocos disco ⇔ memória principal

- Custo da abordagem da busca linear
  - Todos os blocos do arquivo são transferidos para se verificar a condição de seleção, assim C_S=b
  - No caso de busca pela chave, em média, metade dos blocos são transferidos se o registro for encontrado → C_S=b/2
- Busca Binária
  - $C_S=\log_2{}^b$ se o atributo da condição for a chave
  - $C_S=\log_2{}^b + \lceil s/bfr \rceil -1$ se o atributo não for a chave
- Índice primário para recuperar múltiplos registros
  - Se a condição for <, >, <=, >= no campo chave com um índice ordenado pode-se estimar de forma grosseira que metade dos registros satisfazem a condição → b/2 acessos →C_S= x + (b/2)

Exemplos de funções de Custo

Join(⋈)

→ A função usa a seletividade de junção (js) → estimativa do tamanho do arquivo (% de reg) do resultado do join

→ Considerando as relações R com atributo A e S com B

Seja |R| e |S| o número de tuplas das relações R e S, e c, a condição A=B, temos:

- Js = $|R \bowtie_c S| / |R \times S| = |R \bowtie_c S| / |R| * |S|$
  - → $|R \bowtie_c S| = js * |R| * |S|$
- Se A for chave de R então $js \leq (1/|R|) \rightarrow |R \bowtie_c S| \leq |S|$
- Se B for chave de S então $js \leq (1/|S|) \rightarrow |R \bowtie_c S| \leq |R|$

→ Ter informação sobre a seletividade de consultas que ocorrem com freqüência possibilita ao otimizador estimar o tamanho do resultado a partir do tamanho dos arquivos envolvidos na junção.

Exemplos de funções de Custo

Join ($R \bowtie_{A=B} S$)

Mergesort

- Se os arquivos já estão ordenados pelos atributos da junção

$C_J = b_R + b_S + (js * |R| * |S|) / bf_{RS}$)

A última parte da fórmula é o custo de gravação do arquivo resultante

Pode ser considerados diferentes n.º de buffers

- Se os arquivos não estão ordenados, o custo de ordenação será acrescentado

Laços aninhados indexado (laço único)

Se existe um índice para B de S com x_B níveis de índice

O otimizador pode recuperar cada registro t de R e usar o caminho de acesso para recuperar s de S que satisfaça t[A]=s[B]

Índice secundário no qual s_B é a cardinalidade de seleção de B

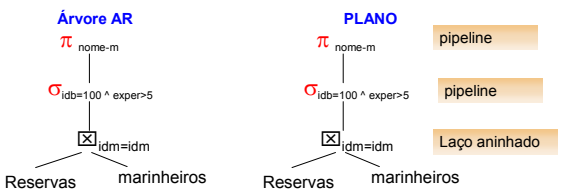
$C_J = b_R + (|R| * x_B * s_B) + (js * |R| * |S|) / bf_{RS}$)

Exemplo de Custo do plano de consulta

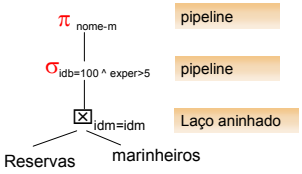
Marinheiros (*idm, nome-m, exper, idade)*

Reservas (*idm, id-barco, data, nome-r)*

- Reservas: 100 registros por página, 1000 págs.
- Marienheiros: 80 registros por página, 500 págs.



Exemplo de Custo do plano de consulta

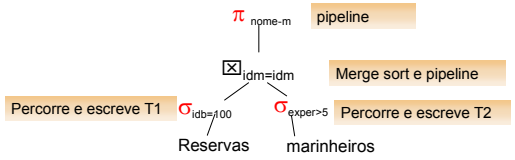


Custo: 500+500*1000 E/S

- ✓ Perde otimizações→ seleções podem ser feitas antes, não usa possíveis índices, etc.
- ✓ *Objetivo da otimização:→ Achar* planos mais eficientes que produzam a mesma resposta.

Exemplo de Custo do plano de consulta

Alternativa: Plano1- adiantar seleções



Com 5 buffers, o custo do plano será:

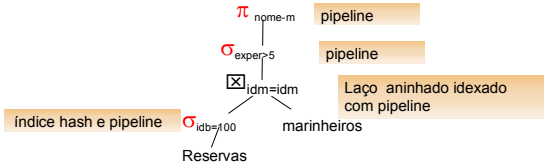
- Percorrer Reservas (1000) + escrever T1 (10 págs, se ∃ 100 barcos)= 1010 E/S
- Percorrer Marinheiros (500)+escrever T2 (250págs, exper>5→ 50%) = 750 E/S
- Ordenar T1 (2*2*10)=40 (leitura e gravação e 2 corridas)
- Ordenar T2 (2*4*250) (leitura e gravação e 4 corridas)
- Join (10+250) = 260

Custo Total: 1010 + 750 + 40 + 2000 + 260 = 4060 E/S.

Corridas com B buffers
n. corridas=2* N $\lceil \log^{N1}_{B-1} \rceil$+1
N1=np/B

Exemplo de Custo do plano de consulta

Alternativa 2- utilizar índices



- Solução ???