

Complementos de Bases de Dados – Processamento de *Queries* –

Engenharia Informática

2º Ano / 1º Semestre

Cláudio Miguel Sapateiro

claudio.sapateiro@estsetubal.ips.pt

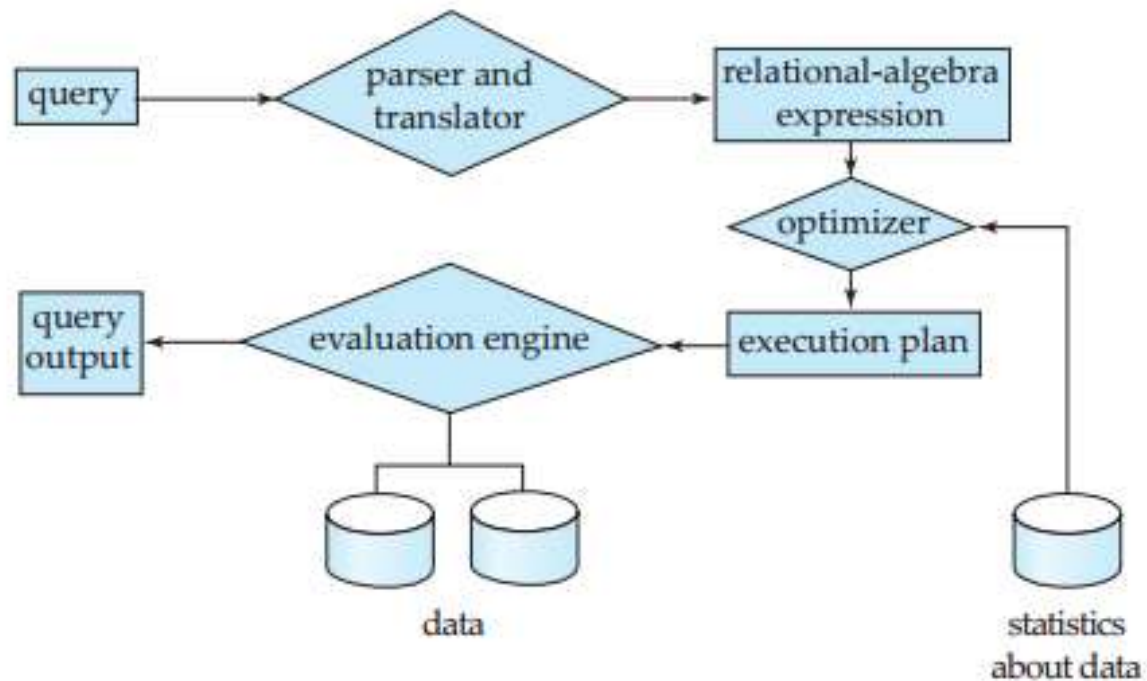
DSI :: Escola Superior de Tecnologia de Setúbal :: Instituto Politécnico de Setúbal

Sumário

- Fases do processamento de uma consulta
 - Análise Sintática
 - Planos lógico e físico
- Otimização da consulta

Processamento e Otimização

Fluxo

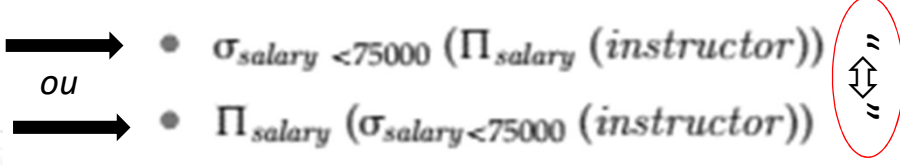


Processamento e Otimização

Parser

- Para o início do processamento “efetivo” o SGBD tem de traduzir/representar a *query* para uma forma “acionável” ao nível de “sistema”
 - O SQL é para humanos!
 - O formato suporta-se na *Álgebra Relacional* e suas equivalências
- O **parser** realiza duas tarefas principais
 - A validação sintática (garantindo que a *query* está “conforme” com as relações da BD a que se refere – utilizando o catálogo)
 - Traduz a *query* para uma árvore que representa as “sub-operações” que terão de ser realizadas » **Plano de Execução (ou Avaliação)**

Exemplo: `select salary`
`from instructor`
`where salary < 75000;` $\xrightarrow{\text{ou}}$ $\bullet \sigma_{\text{salary} < 75000} (\Pi_{\text{salary}} (\text{instructor}))$
 $\bullet \Pi_{\text{salary}} (\sigma_{\text{salary} < 75000} (\text{instructor}))$



Processamento e Otimização

Álgebra Relacional

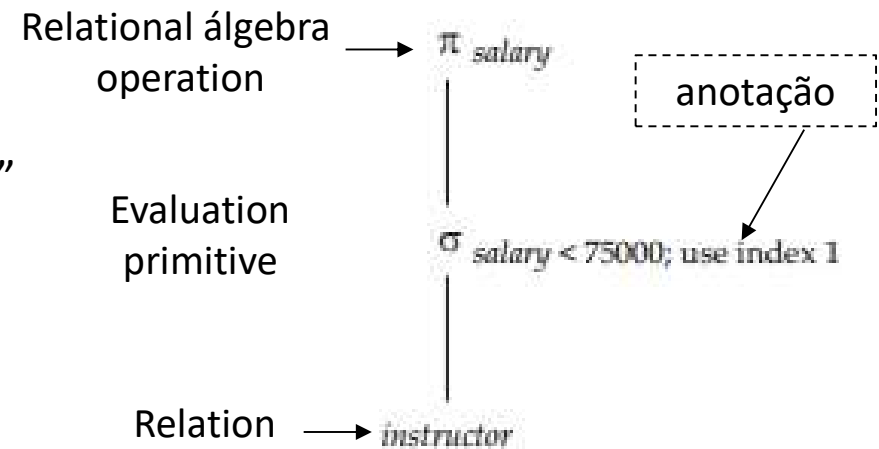
Algumas das operações previstas:

- δ (delta) – remoção de tuplos duplicados
- π (pi) - projeção
- ρ (ró) – renomear atributos ou relações
- σ (sigma) - seleção de tuplos
- χ (chi) - produto de relações
- γ (gama) – agrupar e agregar tuplos de relações
- τ (tau) – ordenação de tuplos
- \bowtie - junção natural de relações (por atributos comuns)
- \bowtie_{θ} - junção teta de relações (junção por predicado θ)
- $\bowtie^>$ – antijunção: tuplos da relação à esquerda que não satisfazem a junção
- \ltimes – semi-junção: projeta apenas as colunas da relação à esquerda
- \ltimes – junção externa direita: inclui os tuplos da relação à direita, sem correspondência com a relação à esquerda
- \ltimes – junção externa esquerda: inclui os tuplos da relação à esquerda, sem correspondência com a relação à direita
- $\ltimes\ltimes$ – junção externa

Processamento e Otimização

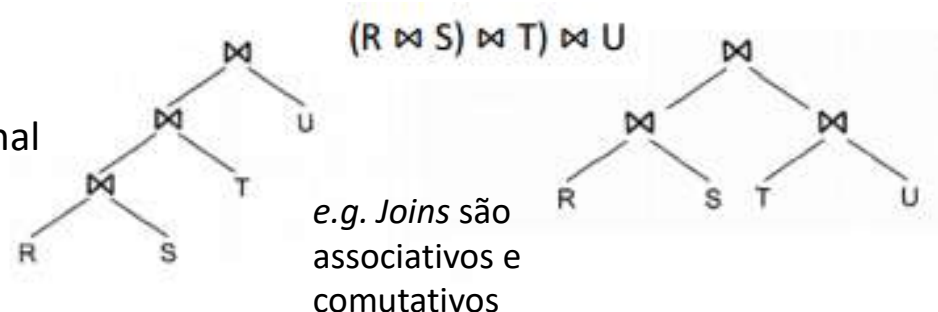
Planos de Execução (avaliação)

- Diferentes planos de execução têm associados diferentes “**custos de execução**”
- Não é da responsabilidade de quem elabora a *query*, estrutura-la da forma que reduza o seu “custo de execução” (bom pelo menos no sentido deste contexto ...)



- ❖ Será o *query optimizer* que gerará planos alternativos de execução e avaliará para cada o seu custo de execução afim de selecionar o que efetivamente será executado, considerando:

- as propriedades (e custos) dos operadores da álgebra relacional
- heurísticas



Processamento e Otimização

A Geração das Alternativas

- A transformação da consulta inicial em alternativas na *forma canónica SPJ* (seleção-projeção-junção) dos operadores da álgebra relacional, envolve:
 - Formar uma lista dos produtos das relações
 - Aplicar a essa lista o predicado da cláusula WHERE
 - Aplicar os agrupamentos
 - Aplicar à essa lista o predicado da cláusula HAVING
 - Aplicar as projeções
 - Aplicar as ordenações.
- ❖ As regras de equivalência a álgebra relacional podem permitir grandes reduções no número de tuplos a processar (redução dos tuplos/dados nos processamentos intermédios)
- ❖ Também as heurísticas permitem ganhos de performance na execução, e.g.
 - ✓ Reorganizar as seleções para que as mais restritivas sejam executadas primeiro (o que reduz o número de tuplos a passar aos operadores ascendentes)
 - ✓ Descer projeções o mais possível (reduz o tamanho dos tuplos)
 - ✓ ...

Processamento e Otimização

Exemplo

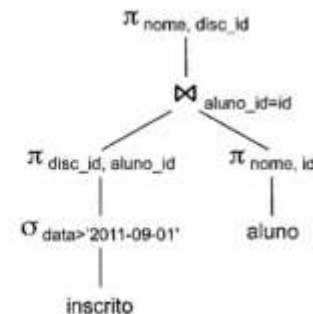
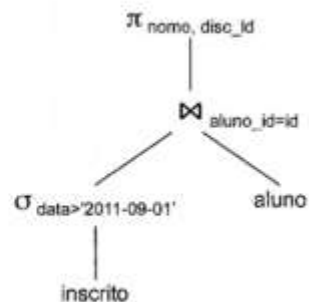
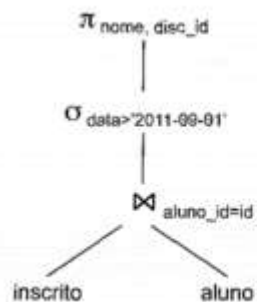
Query

```
select nome, disc_id
from inscrito i, aluno a
where i.aluno_id = a.id
and i.data > '2011-09-01'
```

Plano original

$$\pi_{a.nome, i.disc_id} (\sigma_{i.data > '2011-09-01' \wedge i.aluno_id = a.id} (inscrito \times aluno))$$

Alternativas no
Plano Lógico



Processamento e Otimização

Avaliação - Estimação do Custo

- A avaliação de alternativas lógicas de planos de execução poderá ponderar diversos fatores, incluindo:
 - Acessos ao disco
 - Tempo de CPU
 - e no caso de BDs distribuídas o custo da comunicação
- Deverão ser considerados os custos (*físicos*) associados a cada “sub-operação” e combinados num custo total da *query*
- Tipicamente a grande fatia do custo a ser ponderada (e mais “facilmente” estimada) prende-se com as necessidades das operações relativamente ao acesso ao disco

Processamento e Otimização

Exemplo

- No custo de acesso ao disco considera-se e.g.
 - Number of seeks * average-seek-cost
 - Number of blocks read * average-block-read-cost
 - Number of block written * average-block-write-cost
(custo de escrita > custo de leitura)
 - Simplificações:
 - Consideramos simplesmente *number of block transfers* & *Number of disk seeks*
 - e ignoramos CPU
 - Assim se
 - t_T = nº of seconds to transfer a data block
 - t_S = average block-access time (disk seek time + rotacional latency)
- ❖ Então uma operação envolvendo “*b*” blocos e que realize “*S*” *seeks* demorará :: $b * t_T + S * t_S$ [s]

Processamento e Otimização

Avaliação - Estimação do Custo II

- Contudo as estimativas são de fiabilidade questionável porque:
 - A resposta dependerá dos conteúdos e ocupação do *memory buffer*
 - Com vários discos dependerá de como os dados estão distribuídos
 - ❖ e.g. um plano A pode precisar de mais leituras que B e ainda assim ser mais rápido na execução dependendo da distribuição dos dados pelos discos!
- **Na prática, é impossível determinar um custo exato, da mesma forma que não é possível gerar em tempo útil todas as alternativas possíveis de processamento de uma dada consulta**
- O custo calculado pelo SGBD não é o custo real de execução, mas um valor que permite comparar planos alternativos entre si
 - Alguns SGBD comparam efetivamente o custo calculado com o obtido para atualizar algoritmos de estimativas

Processamento e Otimização

Exemplo

Operação: Selection

	Algorithm	Cost	Reason
A1	Linear Search	$t_s + b_r * t_T$	One initial seek plus b_r block transfers, where b_r denotes the number of blocks in the file.
A1	Linear Search, Equality on Key	Average case $t_s + (b_r/2) * t_T$	Since at most one record satisfies condition, scan can be terminated as soon as the required record is found. In the worst case, b_r blocks transfers are still required.
A2	Primary B ⁺ -tree Index, Equality on Key	$(h_i + 1) * (t_T + t_s)$	(Where h_i denotes the height of the index.) Index lookup traverses the height of the tree plus one I/O to fetch the record; each of these I/O operations requires a seek and a block transfer.
A3	Primary B ⁺ -tree Index, Equality on Nonkey	$h_i * (t_T + t_s) + b * t_T$	One seek for each level of the tree, one seek for the first block. Here b is the number of blocks containing records with the specified search key, all of which are read. These blocks are leaf blocks assumed to be stored sequentially (since it is a primary index) and don't require additional seeks.
A4	Secondary B ⁺ -tree Index, Equality on Key	$(h_i + 1) * (t_T + t_s)$	This case is similar to primary index.

Existirá diferenças na resposta, conforme campos ativados

- e.g. nos critérios do Where,
 - estejam sob indexação ou não, e
 - em caso afirmativo de que tipo

A5	Primary B ⁺ -tree Index, Comparison	$h_i * (t_T + t_s) + b * t_T$	Identical to the case of A3, equality on nonkey.
A6	Secondary B ⁺ -tree Index, Comparison	$(h_i + n) * (t_T + t_s)$	Identical to the case of A4, equality on nonkey.

Processamento e Otimização

Avaliação - Estimação do Custo III

- Outras operações que não a seleção têm cálculos mais complexos!
 - Joins , sorts,
- Ainda, terá de haver lugar a avaliação de expressões para lá de operações individuais (i.e. de toda árvore e seu contexto de execução), considerando a modalidade:
 - *Materialization*: persistência de resultados temporários (relações) em disco se excedem buffer
 - À que juntar ao somatório de custo das operações mais estas!
 - *Pipelining*: modalidade que vai disponibilizando às operações ascendentes resultados à medida que se disponha destes (por oposição a somar o processamento de cada operador em tempo integral)

Processamento e Otimização

Estatísticas sobre os custos

“The cost of an operation depends on the size and other statistics of its inputs:
Given an expression such as $a \star (b \star c)$ to estimate the cost of joining a with $(b \star c)$, we need to have estimates of statistics such as the size of $b \star c$ ”

- **O catálogo da BD guarda informação de cada relação:**
 - NR ou $|R|$: Número de tuplos da relação R;
 - BR: Número de blocos com tuplos de R;
 - SR: Tamanho de um tuplo de R em bytes;
 - FR: Número de tuplos de R que cabem num bloco.
 - VD(C): Número de valores distintos da coluna C. Se C tiver valores únicos, por exemplo se for uma chave candidata, $VD(C) = NR$. Alguns SGBD armazenam também a densidade da coluna C: $1/VD(C)$;
 - Percentagem de valores nulos;
 - min(C): Valor mínimo da coluna C;
 - max(C): Valor máximo da coluna C.
 - A informação guardada sobre os índices:
 - fi: Número de descendentes por cada nó interno, no caso de índices do tipo Btree;
 - Hi. Altura do índice i, igual a 1 no caso de ser um índice de hashing;
 - LBi: Número de blocos ao nível das folhas do índice i.

Processamento e Otimização

Estatísticas sobre os custos

- As estatísticas são muita vezes calculadas com base numa amostra dos dados
 - e.g. umas dezenas ou centenas de milhares de tuplos numa relação de milhões
 - deve ser uma amostra aleatória (evitar enviesamentos)
- As estatísticas não estão sempre a ser atualizadas!
- A informação estatística sobre as relações pode ser atualizada com periodicidade controlada pelo DBA
 - comandos para atualizar estatísticas da BD:
UPDATE STATISTICS; ANALYZE
- As actualizações podem ser despoletadas “manualmente”
ou periodicamente e/ou associadas a eventos
(e.g. threshold de NR, disparity between estimates actual query execution performances/costs)

Processamento e Otimização

Exemplo

- Pressupostos:
 - 1000 registos em *Staff*
 - 50 registos '*Manager*'
 - 50 registos em *Branch*
 - 5 registos '*London*'
 - Não existem índices associados
 - Hipótese (didática) os tuplos são acedidos singularmente (na realidade são em bloco) e com operadores *blocking* (o acesso e produção são feitos em sequência)
 - A memória é suficiente para processar as operações parciais

```
SELECT *
FROM Staff s, Branch b
WHERE s.branchNo = b.branchNo AND
      (s.position = 'Manager' AND b.city = 'London');
```

- Alternativas:
 - (1) $\sigma_{(position='Manager') \wedge (city='London') \wedge (Staff.branchNo=Branch.branchNo)} (Staff \times Branch)$
 - (2) $\sigma_{(position='Manager') \wedge (city='London')} (\sigma_{Staff.branchNo=Branch.branchNo} (Staff \bowtie Branch))$
 - (3) $(\sigma_{position='Manager'} (Staff)) \bowtie_{Staff.branchNo=Branch.branchNo} (\sigma_{city='London'} (Branch))$

- Acessos aos dados?

$$1. \quad (1000 + 50) + 2 * (1000 * 50) = 101\,050$$

$$2. \quad 2 * 1000 + (1000 + 50) = 3050$$

$$3. \quad 1000 + 50 + 50 + 5 + (50 + 5) = 1160$$

Processamento e Otimização

Plano Físico

- O plano físico (*Execution Plan*) contém o acoplamento entre os operadores relacionais e os operadores físicos escolhidos para os implementar.
- Um plano físico especifica como a consulta vai ser executada, sendo que a sua construção parte do plano lógico anterior, adicionando os operadores físicos mais adequados a cada operação lógica, juntamente com os respectivos custos associados
- Geralmente, um plano lógico pode originar vários planos físicos

Processamento e Otimização

Plano Físico II

- Os operadores físicos implementam as operações da álgebra relacional
- O mesmo operador relacional pode ser implementado por mais do que um operador físico e o mesmo operador físico pode implementar mais do que um operador relacional
 - Exemplos:
 - O varrimento de uma tabela pode projetar ao mesmo tempo algumas colunas e efetuar restrições nos tuplos
 - O operador físico "ordenar" pode ser usado por ORDER BY ou por uma junção por ordenação e fusão

Processamento e Otimização

Observações

- Os planos de execução gerados são guardados
 - Para poderem serem reutilizados
 - Uma vez que uma recompilação tem um custo associado
- Contudo,
 - A reutilização de um plano nem sempre é a melhor opção
 - Dependerá da actual distribuição de dados na relação
 - e eventuais alterações à *metadata*
(e.g. remoção de uma *Constraint* ou *Index*)

mini Sumário

1. Ciclo de vida da *query*
2. Plano de execução

05:00



Exercícios

1. Em que consiste a fase de “*Parsing & Translate*” do processamento da *query*?
2. O que distingue os planos de avaliação/execução de nível lógico e nível físico.

Recolha
das
Folhas de
Resposta

CORREÇÃO DOS EXERCÍCIOS