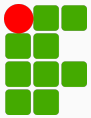
Banco de dados II

05.A - Visão Geral da Avaliação de Consultas

Marcos Roberto Ribeiro



Instituto Federal Minas Gerais - Campus Bambuí

2018

Introdução

- Para avaliar uma consulta, os SGBD traduzem o código SQL para planos de execução/avaliação
- Os planos são representados na forma de árvores onde os nós são operadores da álgebra relacional
- Além dos operadores os planos contém informações sobre qual algoritmo usar para avaliar cada operador
- O SGBD usa um processo de **otimização de consulta** para encontrar um **bom** plano de execução

Tabelas Consideradas nos Exemplos

Tabela `marinheiros`

```
marinheiros (id_marinheiro: integer, nome_marinheiro: string,
             avaliacao: integer, nascimento: date)
```

- Registros de 50 bytes
- 500 páginas com 80 registros cada

Tabela `reservas`

```
reservas(id_marinheiro: integer, id_barco: integer,
         dia: date, nome_responsavel: string)
```

- Registros de 40 bytes
- 1000 páginas com 100 registros cada

Catálogo do Sistema

- Tabelas especiais que armazenam **metadados** sobre os bancos de dados
- Também conhecido como **dicionário de dados**
- Dados sobre tabelas: nome da tabela, nome do arquivo, estrutura do arquivo, nomes e tipos dos atributos, índices da tabela, restrições de integridade
- Dados sobre índices: nome do índice, estrutura, atributos da chave de pesquisa
- Os catálogos são armazenados em forma de tabela. Por quê?

Estatísticas do Catálogo do Sistema

- Estatísticas (atualizadas periodicamente):
  Cardinalidade: Número de registros/tuplas (*NTuplas*) de cada tabela
  Tamanho: Número de páginas (*NPaginas*) de cada tabela
  Cardinalidade do Índice: Número de chaves distintas (*NChaves*) de cada índice
  Tamanho do Índice: Número de páginas (*INPaginas*) de cada índice
  Altura do Índice: Número de níveis (*IAltura*) não folha de cada índice do tipo árvore
  Faixa de Índice: Valores mínimo (*IBaixo*) e máximo (*IAlto*) da chave de pesquisa de cada índice

Introdução à Avaliação de Operadores

- Cada operador pode possuir diversos algoritmos para avaliação
- Nenhum algoritmo é universalmente superior
- Alguns fatores que influenciam tais algoritmos
  - Tamanho das tabelas
  - Índices e ordenações existentes
  - Quantidade de buffers disponíveis
  - Política de substituição de buffers
- Três técnicas comuns:

  Indexação: Uso de índice para obter apenas às tuplas que atendem à uma determinada condição

  Iteração: Varrer as tuplas de uma tabela ou as entradas de um índice

  Particionamento: Decomposição de uma operação em outras operações mais simples sobre partições de dados

Caminhos de Acesso

- Um *caminho de acesso* é uma forma de recuperar as tuplas de uma tabela
- Estes caminho podem afetar significativamente o custo do operador
- Exemplo: seleção usando índice
- A **seletividade** de um caminho de acesso é o número de páginas recuperadas usando tal caminho
- É melhor usar o caminho **mais seletivo** (que recupera o menor número de páginas)

Algoritmos para Operações Relacionais

Seleção

- Pesquisa por um registro que atenda a certas condições
- Se houver índice:
  - Verifique se é viável usar o índice
- Senão, varra a tabela

Projeção

- A maior dificuldade está na eliminação de duplicatas (`DISTINCT`)
- Algumas estratégias para esta situação:
  - Ordenar primeiro os dados
  - Avaliação somente de índice (se todos os campos estiverem na chave do índice)

Algoritmos para Operações Relacionais - Junção I

- Operações caras e comuns (existem diversos algoritmos)
- Exemplo: `reservas` $\bowtie_{\texttt{id\_marinheiro}=\texttt{id\_marinheiro}}$ `marinheiros`

Junção de Loops Aninhados Indexados

- Supondo que haja um índice hash sobre `marinheiros.id_marinheiro`
- Para cada tupla de `reservas`, use o índice para verificar a correspondência
- Custo:
  - Varredura de `reservas`: $100 \times 1.000 = 100.000$
  - Obtenção da correspondência em `marinheiros`: 1,2 E/S (média de um índice hash) + 1 página de `marinheiros`
  - Total: $1.000 + 10.000 \times (1 + 1,2) = 221.000$

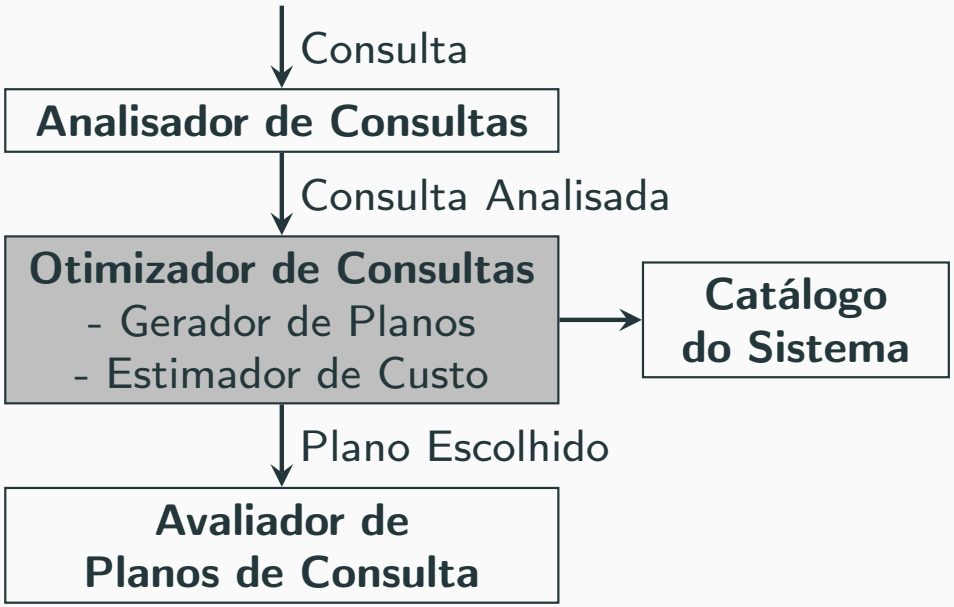
Algoritmos para Operações Relacionais - Junção II

Junção Sort-Merge

- Supondo que não hajam índices
- Ordenamos as tabelas sobre o atributo de junção e depois varremos para fazer a junção
- Custo:
  - Ordenação em duas passagens considerado apenas o custo de E/S e leitura/gravação em cada passagem:
    - `reservas`: $2 \times 2 \times 1.000 = 4.000$
    - `marinheiros`: $2 \times 2 \times 500 = 2.000$
  - Consideramos mais uma varredura nas tabelas ordenadas
  - Total: $4.000 + 2.000 + 1.000 + 500 = 7.500$

Introdução à Otimização de Consultas

- Uma das tarefas mais importantes do SGBD
- Uma consulta pode ser avaliada de várias formas e custo destas avaliações pode ser muito diferente
- É muito difícil encontrar o plano **ideal**, mas podemos encontrar um **bom** plano



Tarefas do Otimizador

- Gerar de planos alternativos (não considera todos, o número é muito grande)
- Avaliar o custo de cada plano alternativo
- Escolher o plano com menor custo

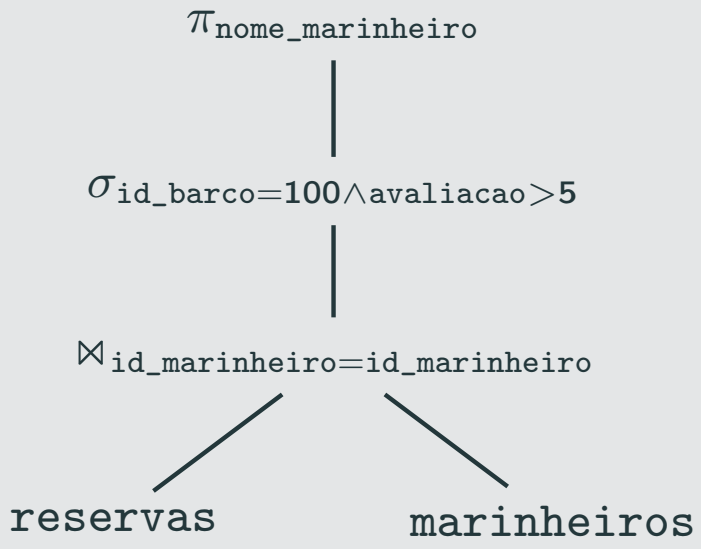
Planos de Avaliação de Consultas

- Um plano é uma árvore de operadores relacionais
- Contém informações adicionais sobre método de acesso e algoritmos

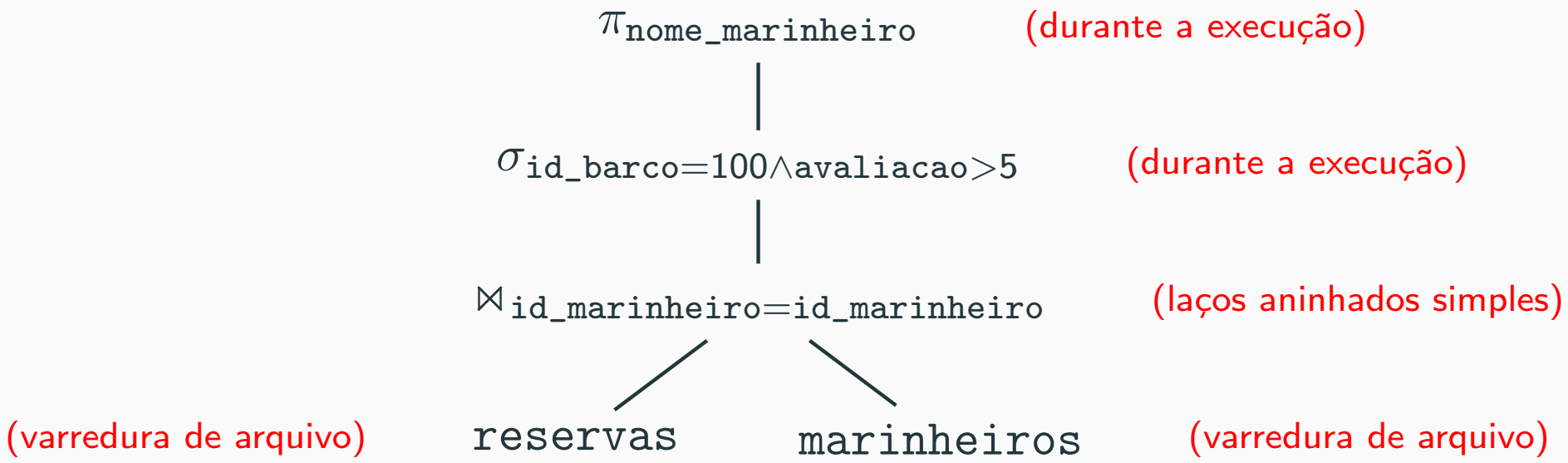
Consulta

```
SELECT m.nome_marinheiro
FROM   reservas r, marinheiros m
WHERE  r.id_marinheiro = m.id_marinheiro
  AND  r.id_barco = 100
  AND  m.avaliacao > 5;
```

Plano



Plano de Execução Completo



Avaliação Pipeline

- O resultado de um operador pode ser encaminhado para outro operador sem a utilização de tabelas temporárias
- Economia de gravar os dados e lê-los de volta
- Quando se usa tabelas temporárias, dizemos que as tuplas são **materializadas**
- A avaliação encadeada (*pipeline*) é escolhida sempre que possível
- Quando um operador usa a avaliação encadeada, dizemos que tal operador é aplicado **durante a execução**

A Interface Iteradora

- Normalmente a implementação dos operadores possui uma interface iteradora uniforme
- Esta interface esconde os detalhes de implementação e possui as seguintes funções

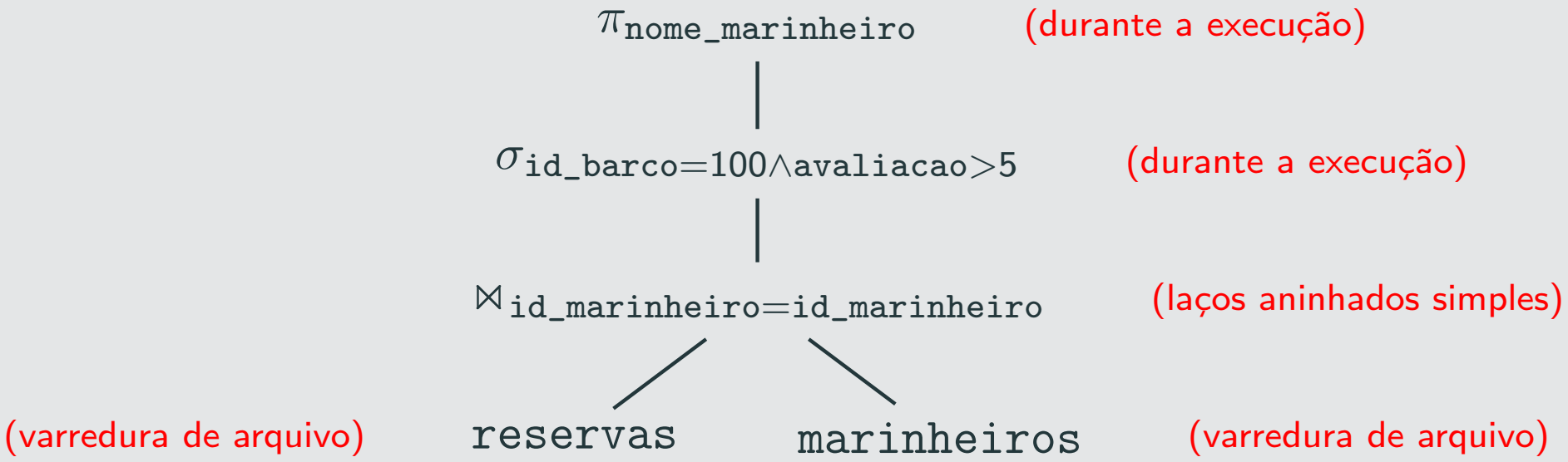
  open(): inicializa o operador

  get_next(): processa a(s) tupla(s) de entrada

  close(): finaliza o operador desalocando recursos usados
- A interface iteradora suporta *pipeline* naturalmente
- A decisão de materializar fica dentro do operador

Planos Alternativos - Exemplo de Motivação I

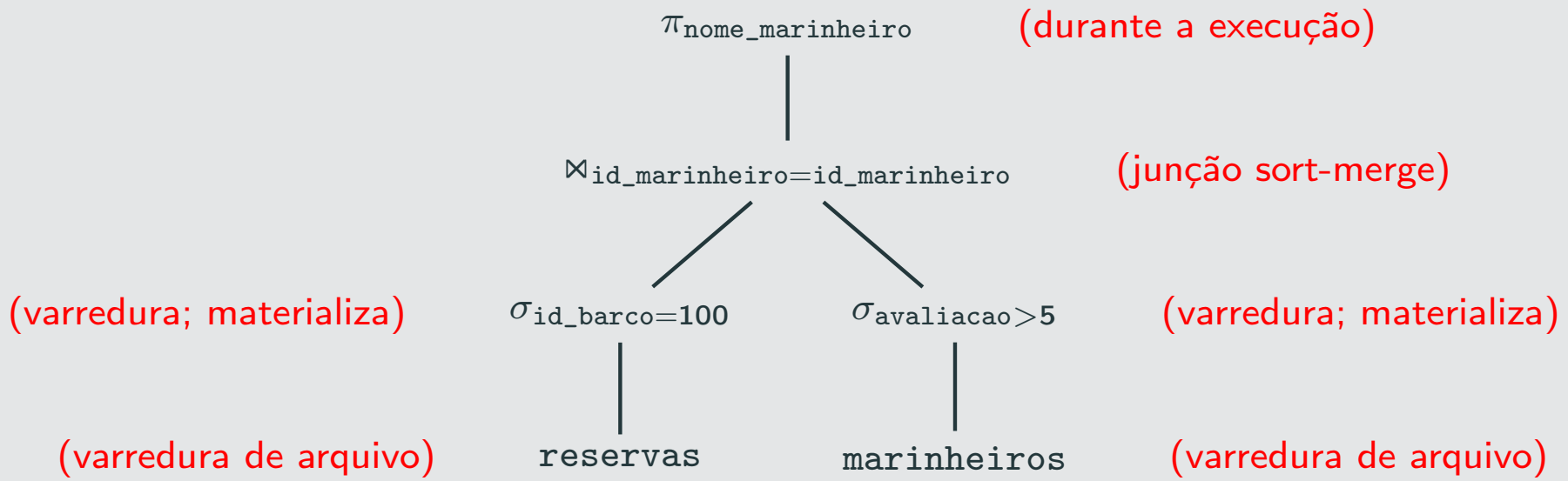
Varredura e Junção com Loops Aninhados



- Para cada página de `reservas`, leia todas as páginas de `marinheiros` fazendo a junção
- Custo: $1.000 \times 500 = 500.000$ E/S

Planos Alternativos - Exemplo de Motivação II

Empurrando Seleções



- Varredura/seleção de `reservas`: 1.000
- Gravação de T1 (supondo distribuição uniforme de reservas): 1.000 reservas / 100 barcos $= 10$
- Varredura/seleção de `marinheiros`: 500
- Gravação de T2 (supondo distribuição uniforme de avaliações entre 1 e 10): 50% de $500 = 250$
- Ordenação em duas passagens de T1 e T2: $2 \times 2 \times 10 = 40 + 2 \times 2 \times 250 = 1.000$
- Junção de T1 e T2: $10 \times 250 = 2.500$
- Custo final: $1.000 + 10 + 500 + 250 + 1.000 + 2.500 = 5.260$

Tarefas de Um Otimizador Típico

- Usa equivalências da álgebra relacional para identificar obter expressões algébricas alternativas
- Para cada expressão algébrica equivalente, considera as implementações disponíveis para os operadores para gerar planos
- Avalia os custos dos planos e seleciona aquele com menor custo
- Expressões algébricas são consideradas equivalentes se produzirem o mesmo resultado
- Exemplos:
  - Seleções e produtos cartesianos podem ser combinados em junções
  - Junções podem ser reordenadas extensivamente
  - Seleções e junções podem ser *empurradas* para frente das junções

Planos de Profundidade à Esquerda

Árvore 1

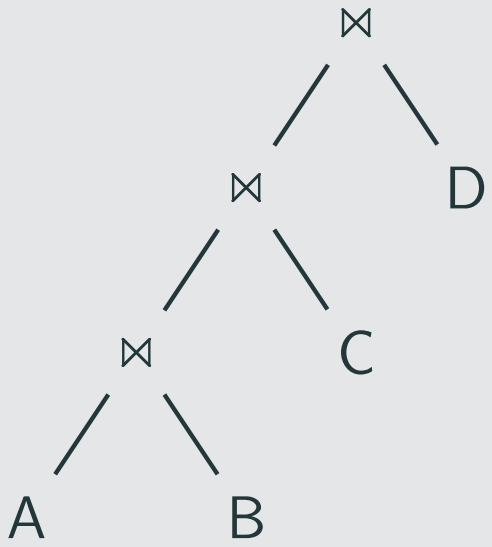
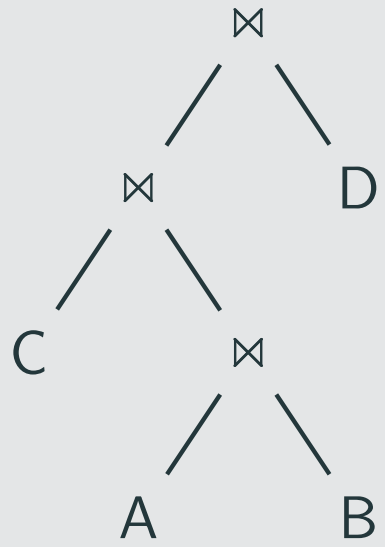


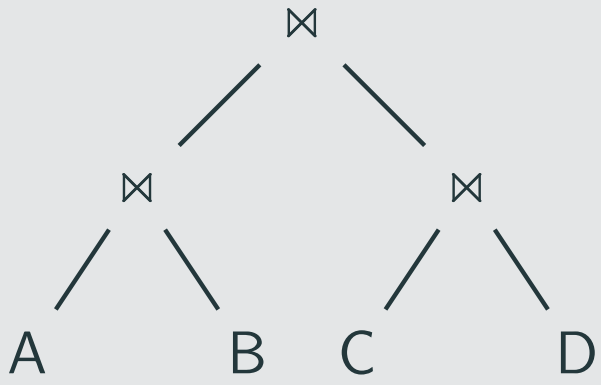
Tabela Externa

Tabela Interna

Árvore 2



Árvore 3



Árvores Lineares: Pelo menos um filho de junção é tabela

Árvore de Profundidade à Esquerda: O filho direito da junção sempre é uma tabela

Planos de Profundidade à Esquerda

- Os otimizadores usam *programação dinâmica* para pesquisar os planos de profundidade à esquerda
- A media que o número de junções aumenta, o número de planos alternativos pode crescer muito. É necessário podar o espaço de planos alternativos
- Árvores de profundidade à esquerda permite generalizar os planos *integralmente encadeados* (pipeline em todas as junções)

Referências I


Date, C. J. (2004).
***Introdução a sistemas de bancos de dados*.**
Elsevier, Rio de Janeiro.


Elmasri, R. and Navathe, S. B. (2011).
***Sistemas de banco de dados*.**
Pearson Addison Wesley, São Paulo, 6 edition.


Ramakrishnan, R. and Gehrke, J. (2008).
***Sistemas de gerenciamento de banco de dados*.**
McGrawHill, São Paulo, 3 edition.


Silberschatz, A., Korth, H. F., and Sudarshan, S. (2007).
Sistema de bancos de dados.