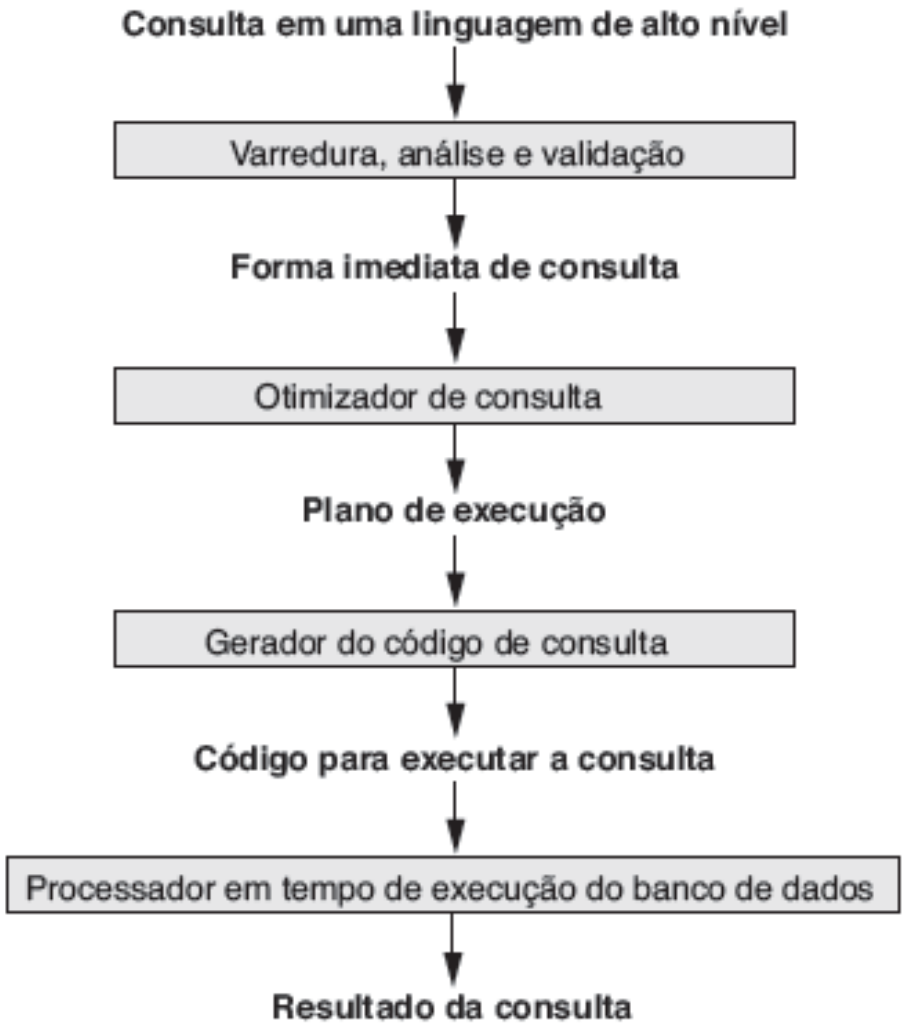
[MAC0426] Sistemas de Bancos de Dados
[IBI5013] Bancos de Dados para Bioinformática
Aula 26
Uma Brevíssima Introdução ao Processamento e Otimização de Consultas

Kelly Rosa Braghetto

DCC–IME–USP

17 de junho de 2016

Etapas típicas do processamento de uma consulta



Etapas típicas do processamento de uma consulta expressa em linguagem de alto nível (como a SQL)

1. **Varredura**: identifica os *tokens* – palavras reservadas, nomes de atributos e relações
2. **Análise Sintática**: verifica se a consulta está bem formulada segundo a sintaxe da linguagem
3. **Validação**: verifica se os nomes de atributos e relações são válidos e semanticamente significativos no esquema do BD sendo consultado
4. **Criação de uma representação interna da consulta**, na forma de uma *árvore de consulta* ou de um *grafo de consulta*

Etapas típicas do processamento de uma consulta expressa em linguagem de alto nível (como a SQL)

5. **Otimização de consulta**: escolhe uma estratégia de execução (= plano de consulta) para obter os resultados da consulta a partir dos arquivos do BD
   - "Otimização" não é um nome correto, porque a estratégia escolhida nem sempre é a ótima (basta ser razoavelmente eficiente)! Encontrar a melhor estratégia pode ser muito demorado ou exigir informações não disponíveis no catálogo do SGBD
6. **Geração de código**: gera o código para executar o plano escolhido
7. **Processamento em tempo de execução do BD:** executa o código da consulta para produzir seu resultado. O código pode ser:
   - Executado diretamente (modo interpretado)
   - Armazenado e executado posteriormente, sempre que desejado (modo compilado)

Traduzindo consultas SQL para a Álgebra Relacional

- As consultas SQL são decompostas em **blocos de consulta**
- Cada bloco de consulta contém uma única expressão SELECT-FROM-WHERE (que pode incluir GROUP BY e HAVING)
- Consultas aninhadas (subconsultas) são identificadas como blocos de consulta separados
- O otimizador de consultas escolhe um plano de execução para cada bloco de consulta

Exemplo de blocos de consultas

Consulta SQL

```
SELECT   Unome, Pnome
FROM     FUNCIONARIO
WHERE    Salario > ( SELECT   MAX (Salario)
                     FROM     FUNCIONARIO
                     WHERE    Dnr=5 );
```

Exemplo de blocos de consultas

Bloco mais interno da consulta SQL do slide anterior

```
( SELECT MAX (Salario)
  FROM   FUNCIONARIO
  WHERE  Dnr=5 )
```

Bloco mais externo da consulta SQL do slide anterior

```
SELECT  Unome, Pnome
FROM    FUNCIONARIO
WHERE   Salario > c
```

onde c é o resultado retornado do bloco interno.

Exemplo de blocos de consultas

Tradução dos blocos para a Álgebra Relacional

```
( SELECT MAX (Salario)
  FROM   FUNCIONARIO
  WHERE  Dnr=5 )
```

$$\Im_{\text{MAX Salario}}(\sigma_{\text{Dnr}=5}(\text{FUNCIONARIO}))$$

```
SELECT  Unome, Pnome
FROM    FUNCIONARIO
WHERE   Salario > c
```

$$\pi_{\text{Unome,Pnome}}(\sigma_{\text{Salario}>c}(\text{FUNCIONARIO}))$$

onde c é o resultado retornado do bloco interno.

Otimização de consultas em SGBDRs

Principais técnicas empregadas:

- Usar **regras heurísticas** para modificar a representação interna de uma consulta com o objetivo de melhorar o seu desempenho esperado
  - uma heurística é algo que funciona bem na maioria dos casos, mas não garante funcionar bem para todos eles
- **Estimar sistematicamente** o custo de diferentes estratégias e escolher o plano de execução com estimativa de custo mais baixa

Técnicas heurísticas para otimização de consultas

Resumo das regras heurísticas para otimização algébrica:

1. (Principal) Aplicar primeiro as operações que reduzem o tamanho dos resultados intermediários
   - Inclui a realização o mais cedo possível (mover para baixo na árvore) de operações de SELEÇÃO (que reduzem o número de tuplas) e de PROJEÇÃO (que reduzem o número de atributos)
2. Executar as operações SELEÇÃO e JUNÇÃO mais restritivas (ou seja, que resultam menos tuplas ou tuplas ou com o menor tamanho absoluto) antes de outras operações semelhantes
3. Reordenar os nós folhas da árvore entre eles mesmos, para evitar produtos cartesianos

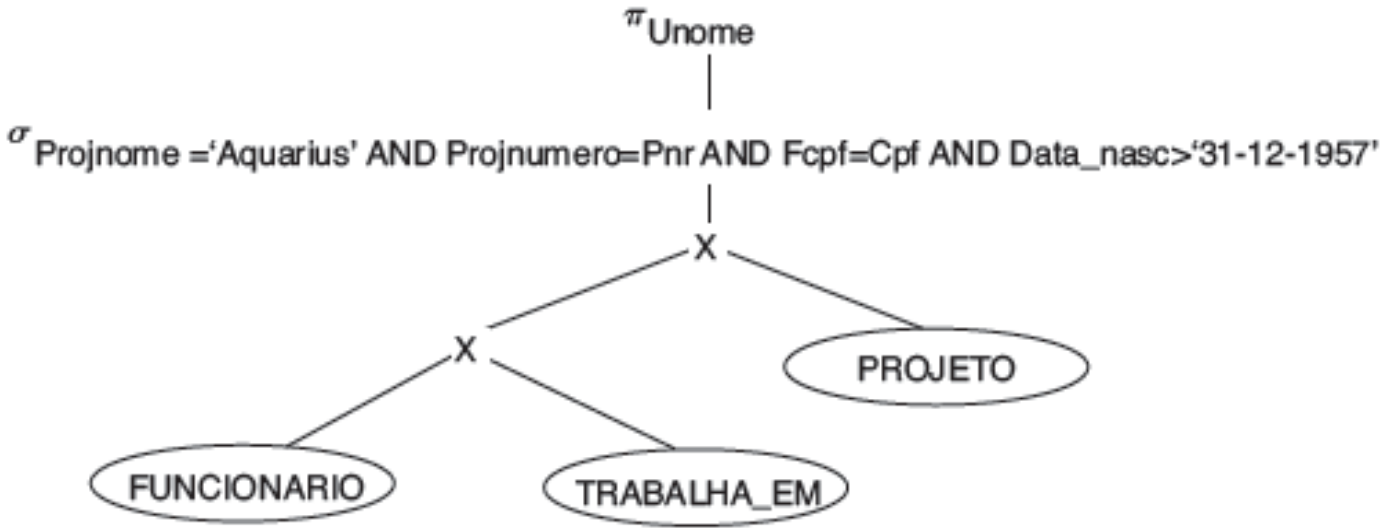
Técnicas heurísticas para otimização de consultas

Exemplo (parte 1)

- **Consulta SQL**

```
SELECT Unome
FROM   FUNCIONARIO, TRABALHA_EM, PROJETO
WHERE  Projnome = 'Aquarius' AND Projnumero = Pnr
       AND Fcpf = Cpf AND Data_nasc > '31-12-1957';
```

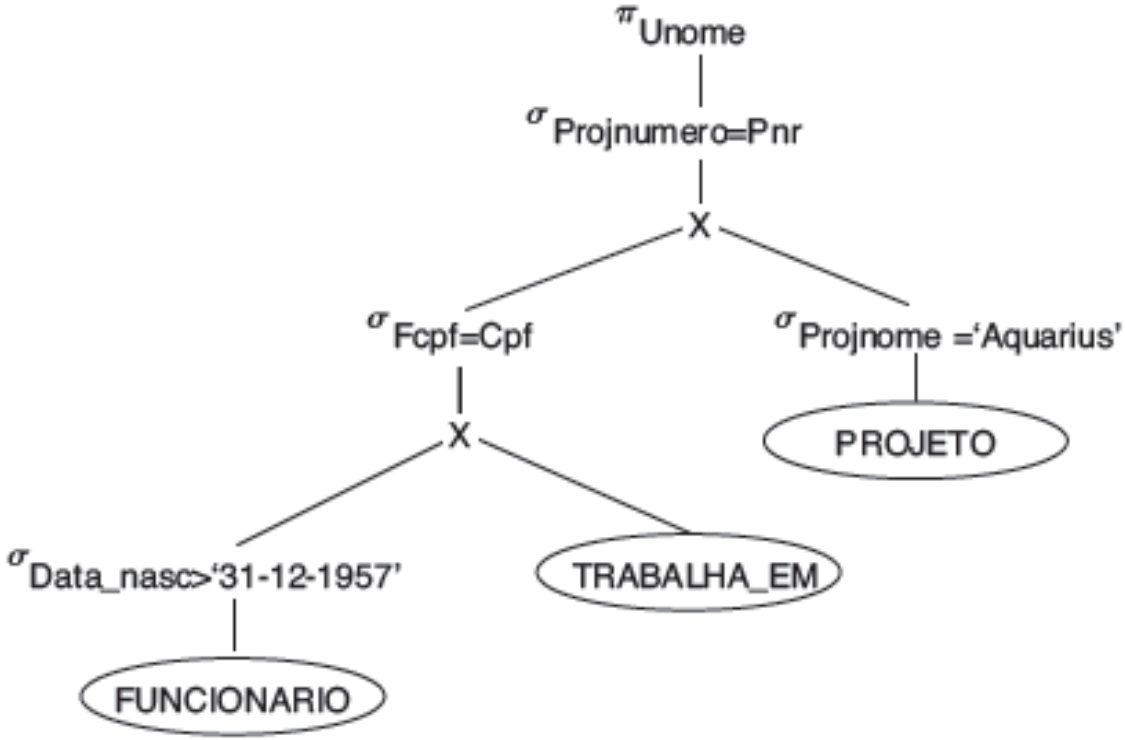
- **Árvore de consulta inicial (canônica)**



Técnicas heurísticas para otimização de consultas

Exemplo (parte 2)

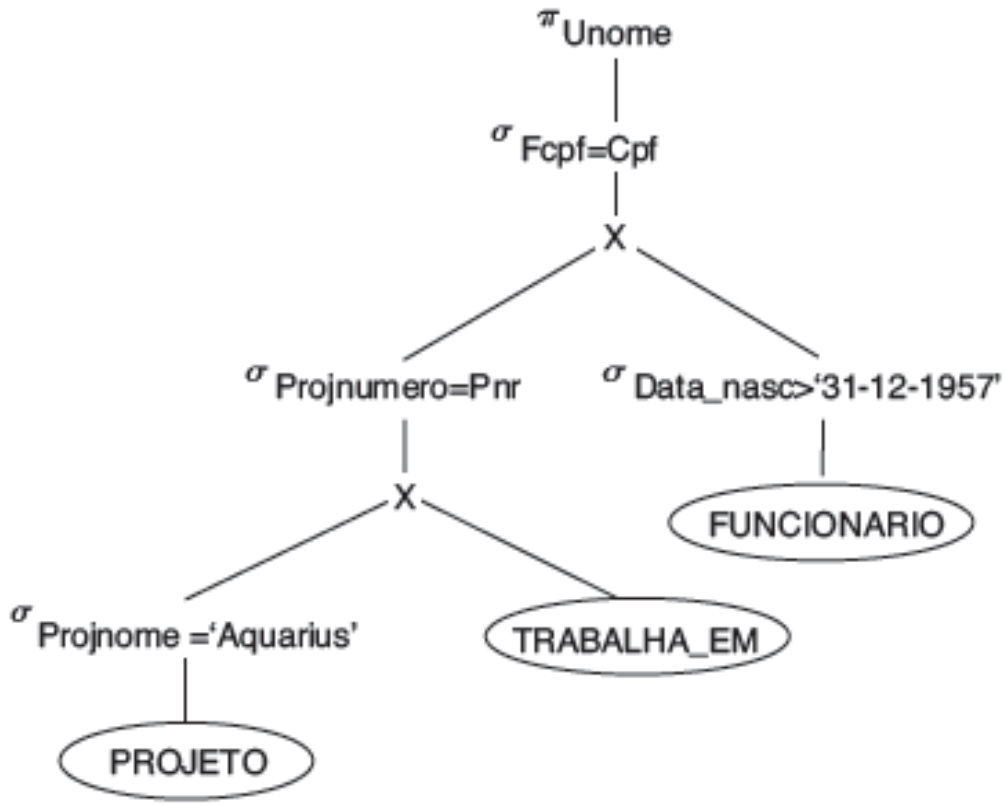
- Movendo as operações de SELEÇÃO mais para baixo



Técnicas heurísticas para otimização de consultas

Exemplo (parte 3)

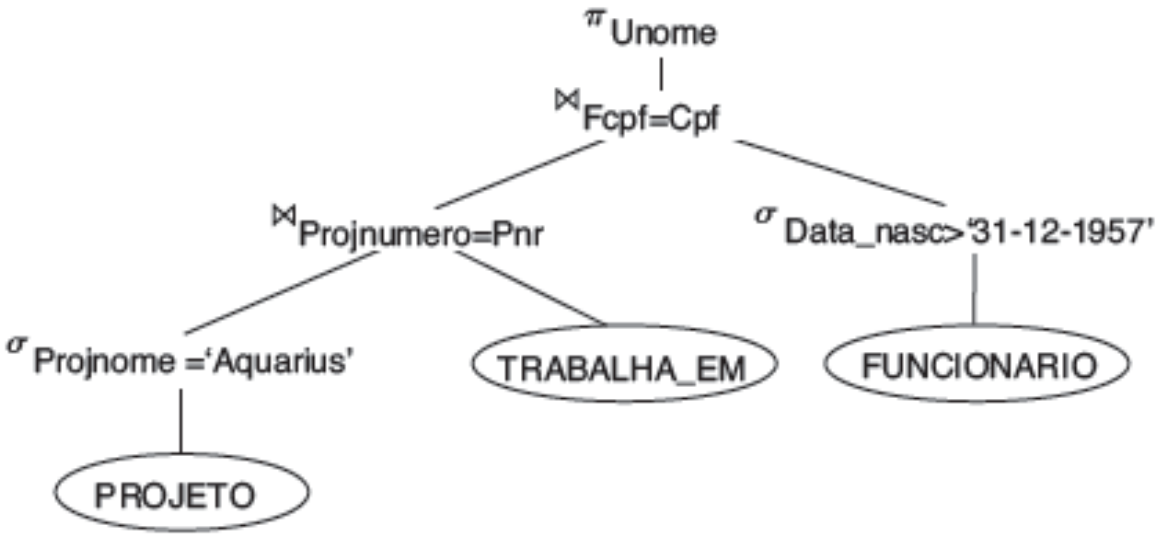
- Aplicando a SELEÇÃO mais restritiva primeiro



Técnicas heurísticas para otimização de consultas

Exemplo (parte 4)

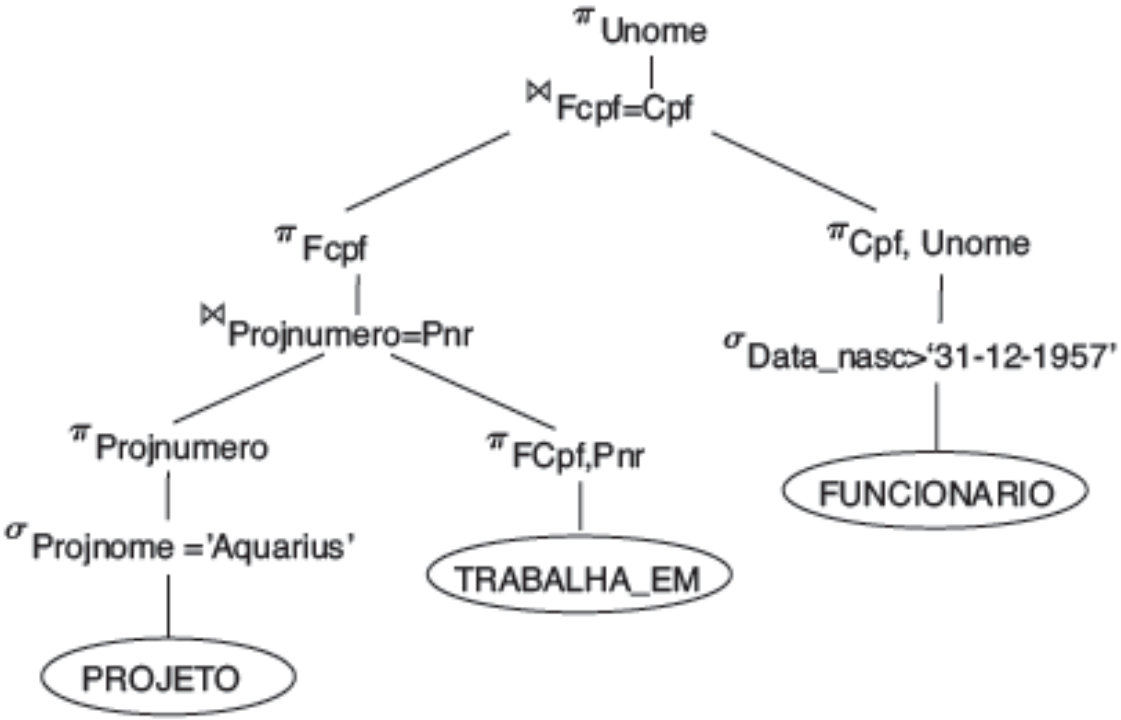
- Substituindo PRODUTO CARTESIANO e SELEÇÃO por JUNÇÃO



Técnicas heurísticas para otimização de consultas

Exemplo (parte 5) – árvore final otimizada

- Movendo PROJEÇÃO mais para baixo na árvore de consulta



Convertendo árvores de consulta em planos de execução de consultas

Um plano de execução inclui informações sobre:

- os métodos de acesso (ex.: índices) disponíveis para cada relação
- os algoritmos a serem usados para computar os operadores relacionais
- especificação de uso de avaliação *materializada* vs. *canalizada* (*pipeline*)
  - materializada – o resultado de uma operação é armazenado como uma relação temporária, que pode ser lida como entrada para uma próxima operação
  - *pipeline* – à medida que as tuplas resultantes de uma operação são produzidas, elas encaminhadas diretamente à próxima operação na sequência

Otimização de consultas baseada em custo

- Um otimizador de consultas não usa apenas regras heurísticas
- Ele também estima e compara o custo da execução de uma consulta utilizando diferentes estratégias de execução e algoritmos
  - seleciona o plano de execução com a menor estimativa de custo
- Para que a técnica funcione
  - são necessárias estimativas precisas
  - é preciso limitar o número de estratégias a serem avaliadas – senão *um tempo inviável será gasto nessa avaliação*
- Essa técnica é mais adequada para consultas compiladas – a otimização é feita em tempo de compilação e o plano resultante é armazenado e depois executado diretamente em tempo de execução
- Para consultas interpretadas, uma otimização em escala total pode atrasar o tempo da resposta

Otimização de consultas baseada em custo

Componentes considerados no custo de execução:

- custo de acesso ao armazenamento secundário
- custo de armazenamento em disco
- custo de computação (ou custo de CPU)
- custo de uso da memória principal
- custo de comunicação – custo do envio da consulta e de seus resultados entre o local do BD e o local onde a consulta foi originada. Em BDs distribuídos, também pode se referir à transferência de dados entre os vários computadores durante a avaliação da consulta.

Otimização de consultas baseada em custo

Exemplos de informações de catálogo do BD usadas para estimar o custo de uma consulta:

Para cada arquivo de dados:

- número de registros (tuplas)
- tamanho médio do registro
- número de blocos do arquivo
- número médio de registros por bloco
- número de níveis para cada índice multinível
- número de valores distintos de um atributo e sua seletividade
- ...

Otimização de consultas baseada em custo

Exemplo simples de função de custo para SELEÇÃO (1)

- S_1: **SELEÇÃO por pesquisa linear (força bruta)** – pesquisa-se todos os b blocos do arquivo para recuperar os registros que satisfazem a condição de seleção
  - $Custo(S_1) = b$
  - Para uma condição de *igualdade em um atributo chave*, somente metade dos blocos são pesquisados *em média* antes de que o registro procurando seja encontrado. Nesse caso, $Custo(S_1) = b/2$ se o registro for encontrado, e $Custo(S_1) = b$ no caso contrário.

Otimização de consultas baseada em custo

Exemplo simples de função de custo para SELEÇÃO (2)

- S_2: **SELEÇÃO usando um índice de árvore B+ sobre o atributo de chave primária** – para encontrar um registro pelo valor da sua chave, acessa-se $x + 1$ blocos do disco, onde x é o número de níveis da árvore
  - $Custo(S_2) = x + 1$

Ordenação externa

- Ordenação (externa) é uma operação muito usada no processamento de consultas SQL
  - ORDER BY, JOIN, UNION, INTERSECTS, SELECT DISTINCT
- A existência de um índices primário pode evitar a necessidade de ordenação para responder à consulta
- **Ordenação externa** se refere a algoritmos de ordenação adequados para grandes arquivos de registros que não cabem inteiros na memóra principal (como é o caso em BDs)

Exemplo de algoritmo de ordenação externa

External Merge-Sort – ideia geral:

1. Leia um *pedaço* dos dados do disco para a memória principal e ordene-o com algum método convencional, como o *quicksort*.
2. Escreva os dados ordenados de volta no disco.
3. Repita os passos 1 e 2 até que todos os pedaços estejam ordenados. Esses pedaços serão intercalados para formar o arquivo ordenado.
4. Leia para a memória principal um subpedaço de cada um dos pedaços ordenados no disco.
5. Faça uma intercalação dos subpedaços, armazenando o resultado em um buffer de saída na memória principal. Sempre que esse *buffer* encher, descarregue-o no disco, no arquivo ordenado final. E sempre que um dos subpedaços acabar, carregue nele o próximo subpedaço do seu pedaço ordenado de origem, até que nenhum dado mais tenha "restado" em nenhum dos pedaços ordenados.

Referências Bibliográficas

- *Sistemas de Bancos de Dados* (6ª edição), Elmasri e Navathe. Pearson, 2010. – Capítulo 19