Bases de Dados
Métodos de Acesso Físicos

Prof. Dr. Ives Renê V. Pola
ivesr@utfpr.edu.br

Departamento Acadêmico de Informática – DAINF
UTFPR – Pato Branco DAINF
UTFPR
Pato Branco - PR

Esta apresentação mostra os métodos de acesso físico usados para a execução de consultas em SGBD, como interpretar os resultados do comando *EXPLAIN QUERY* e como os métodos de de acesso exploram a estrutura de memória em disco.

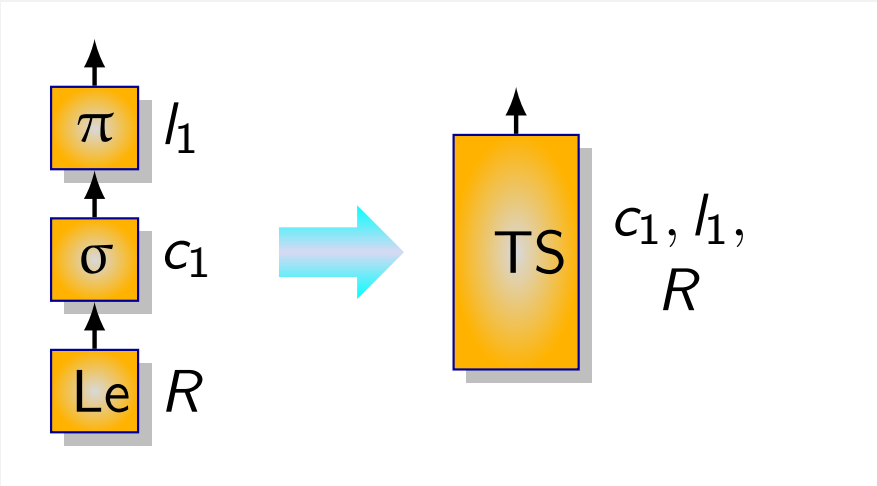
Roteiro

- Table Scan
- Index Scan
- Clustered Matching Index Scan
- Index-only Scan
- Multi-Index Scan

1. Métodos de Acesso Físico – Operadores de apoio

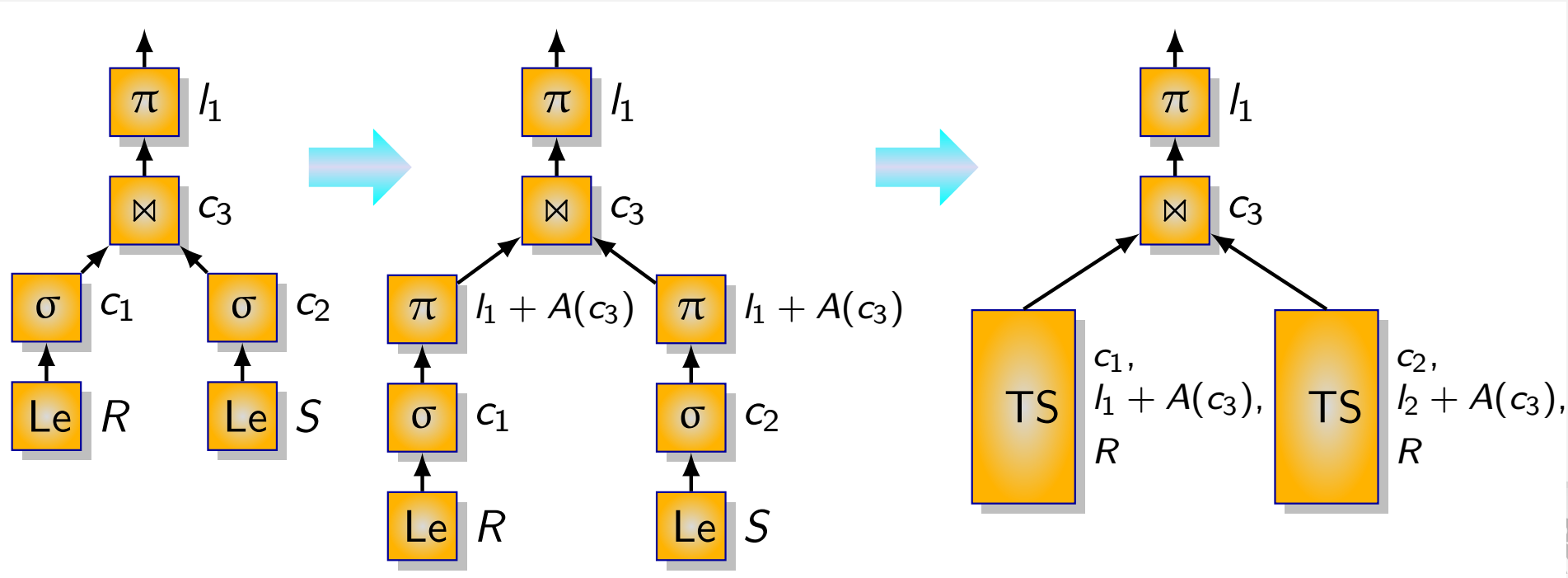
Métodos de Acesso para operações unárias

- Um Método de acesso para operações unárias empacota uma sequência de operadores Le R , σ c_i e π l_i . Por exemplo, seja o método de acesso físico *Table Scan* (a ser estudado a seguir):



Métodos de Acesso para operações unárias

- A escolha por um plano lógico 'ótimo' pode levar em conta o uso que será feito das operações de acesso físico.
- Por exemplo, antecipar a projeção apenas dos atributos que efetivamente serão usados é sempre uma operação útil, dado que todos os operadores de acesso físico já provêm naturalmente a capacidade de projeção.



Métodos de Acesso para operações unárias

Table Scan

- O método de acesso **Table Scan** é o método básico para acesso a qualquer relação.
  - Quer dizer, mesmo que uma consulta não consiga usar nenhum outro método, o *table scan* sempre pode ser usado.
  - Ele não depende do uso de nenhum índice.
- O *table scan*:

  ☞ **lê** sucessivamente todos os *extents* que formam uma relação,

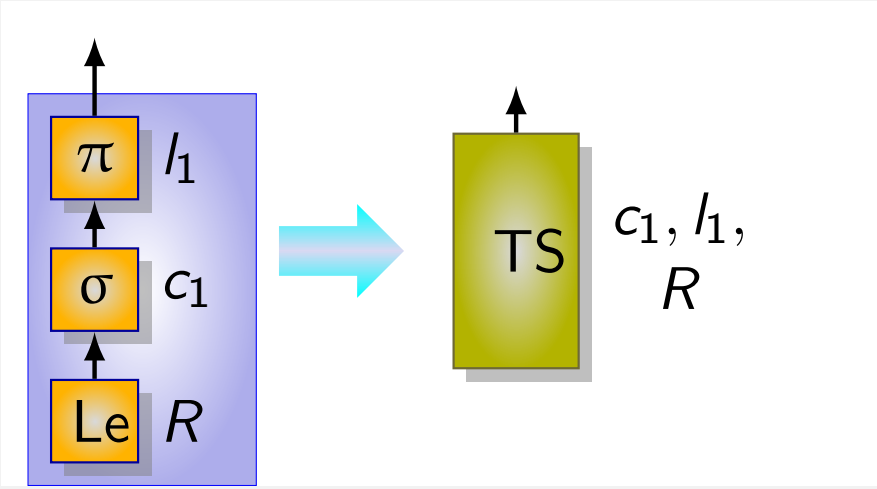
  ☞ aplica o critério combinado dos **operadores de seleção** "empacotados" para todas as tuplas de cada *extent*,

  ☞ e escreve no *buffer* de saída as tuplas projetadas segundo a lista de tuplas do **operador de projeção** "empacotado".
- **O custo** do *Table Scan* é a leitura sequencial das *NPags* que armazenam a relação: R(*NPags*)S .

Métodos de Acesso para operações unárias

Table Scan

- O **Table Scan** – TS empacota uma sequência de operadores Le R, σ c_i e π l_i:



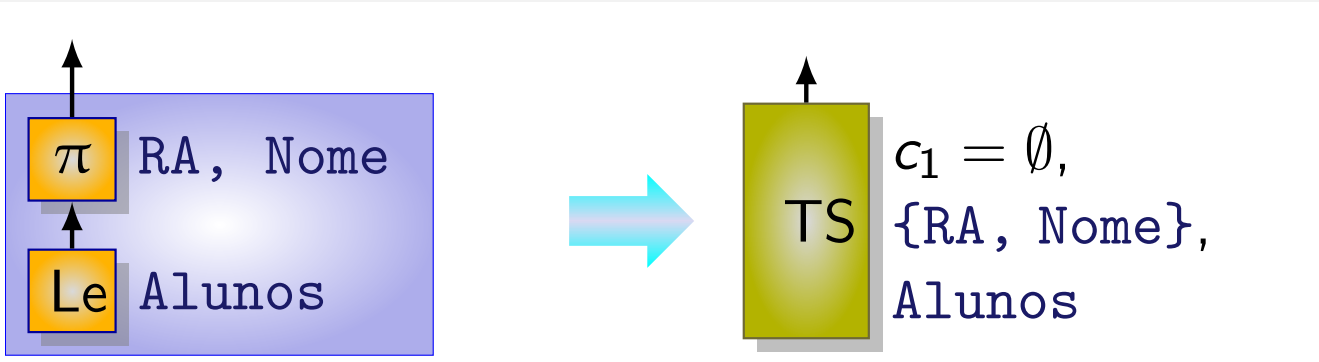
Métodos de Acesso para operações unárias

Table Scan – Exemplo Consulta 1

- Seja a consulta sobre a relação de alunos

```
SELECT RA, Nome FROM Alunos
```

- que gera a seguinte árvore de comandos. Como não pode ser usado nenhum índice, o plano de acesso físico gerado é:



- Considerando um total 3810 páginas, esse comando irá demorar $R3.810S = \frac{3.810}{1.000} = 3,81$ segundos.

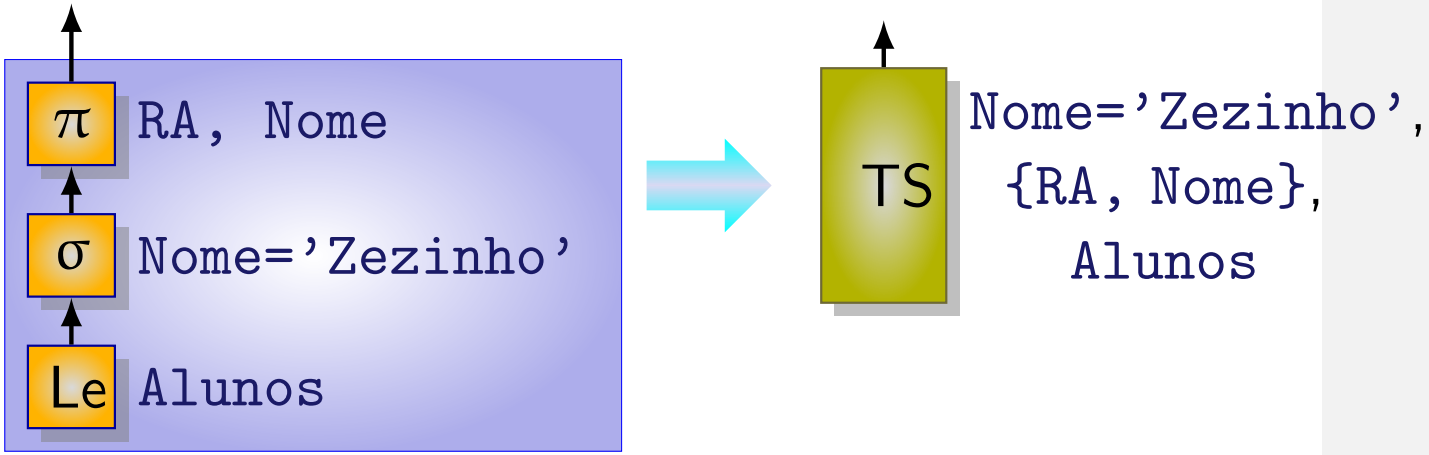
Métodos de Acesso para operações unárias

Table Scan – Exemplo Consulta 4

- Seja agora a seguinte consulta sobre a mesma relação de alunos

```
SELECT RA, Nome FROM Alunos
    WHERE Nome='Zezinho'
```

- Essa consulta gera a árvore de comandos:
- Como não existe nenhum índice sobre o atributo `Nome`, o plano de acesso físico gerado é:



- E da mesma maneira, esse comando irá demorar $R3.810S = \frac{3.810}{1.000} = 3,81$ segundos.

Métodos de Acesso para operações unárias

Index Scan

- O método de acesso **Index Scan** é o método básico para acessar uma relação usando um índice associado.
- Ele também empacota uma sequência de operadores Le R , σ c_i e π l_i , porém a fase de seleção faz uso de um índice.

Métodos de Acesso para operações unárias

Index Scan

- Vamos assumir que esteja sendo usado um índice B-tree:
- O *table scan*:

  ☞ Acessa o índice até a folha que contem o RowId da primeira tupla com a chave de busca,

  ☞ lê em sequência todas as folhas a partir da primeira, até achar uma chave diferente da chave de busca,

  ☞ **lê** todas as páginas apontadas pelas entradas das folhas lidas,

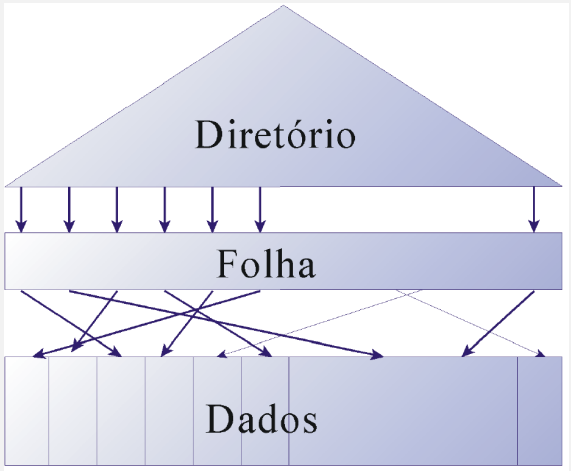
  ☞ caso haja mais critérios de busca não cobertos pelo índice, aplica o critério combinado dos demais **operadores de seleção** empacotados para todas as tuplas lidas,

  ☞ e escreve no *buffer* de saída as tuplas projetadas segundo a lista de tuplas do **operador de projeção** "empacotado".

Métodos de Acesso para operações unárias

Index Scan

- **O custo** do *Index Scan* é a leitura aleatória dos diretórios do índice (um para cada nível $H-1$ da árvore), mais a leitura sequencial das *NFolhas* que contém a chave de busca, mais a leitura das n tuplas recuperadas pelo índice:
  $\mathrm{R}H\mathrm{R} + \mathrm{R}(NFolhas)\mathrm{S} + \mathrm{R}n\mathrm{T}$.



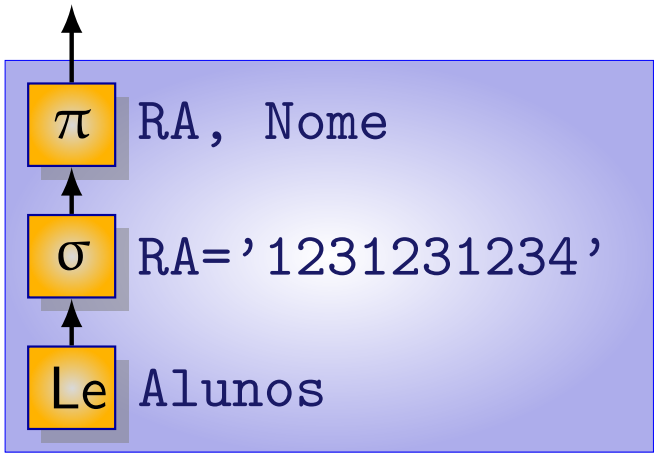
Métodos de Acesso para operações unárias

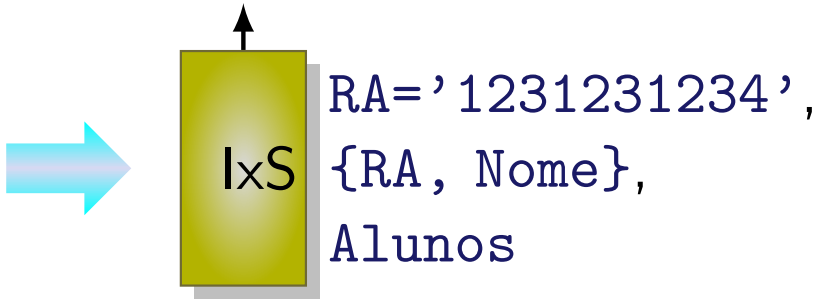
index Scan – Exemplo Consulta 2

- Seja a consulta do exemplo 2 sobre a relação de alunos:

```
SELECT RA, Nome FROM Alunos
    WHERE RA='1231231234'
```

- Agora o índice da chave primária pode ser usado. Portanto, o plano de acesso físico gerado é:





- Como calculado antes, esse comando irá demorar $R4R = \frac{4}{100} = 0,04$ segundos.

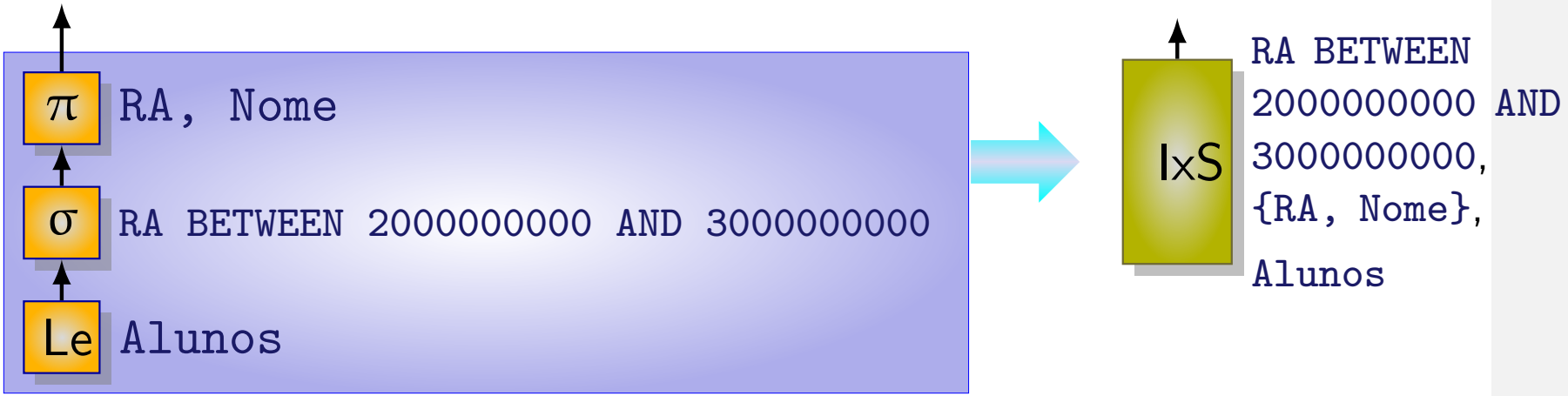
Métodos de Acesso para operações unárias

index Scan – Exemplo Consulta 3

- Nem sempre usar um índice é a melhor opção.
- Por exemplo, seja a consulta do exemplo 3 sobre a relação de alunos:

```
SELECT RA, Nome FROM Alunos
    WHERE RA BETWEEN 2000000000 AND 3000000000
```

- Vamos supor que é usado o índice da chave primária para responder a essa consulta. Então, o plano de acesso físico gerado é:



- Como calculado antes, esse comando irá demorar $\boxed{\text{R3R}} + \boxed{\text{R103S}} + \boxed{\text{R8.000T}} = \frac{3}{100} + \frac{103}{1.000} + \frac{8.000}{300} = 0,03 + 0,103 + 26,67 = 26,80$ segundos.

Métodos de Acesso para operações unárias

Index Scan

- Se a consulta for realizada usando um *table scan*, sem uso de índice, o tempo para acessar toda a relação é de $R3.810S = \frac{3.810}{1.000} = 3,81$ segundos.
- Se tentar usar o índice sobre a chave primária e um *index scan*, mesmo que ele permita acessar apenas $\frac{1}{10}$ da relação, o tempo de acesso aumenta para 26,80 segundos.
- Isso se deve a dois motivos:
  1. O acesso pelo índice gera uma "chuva de ponteiros" sobre as páginas de dados (uma mesma página pode ser lida mais de uma vez),
  2. O acesso é executado por acessos previsíveis, que usam o cache do disco mas não consegue aproveitar a proximidade das páginas em um *extent*.

Métodos de Acesso para operações unárias

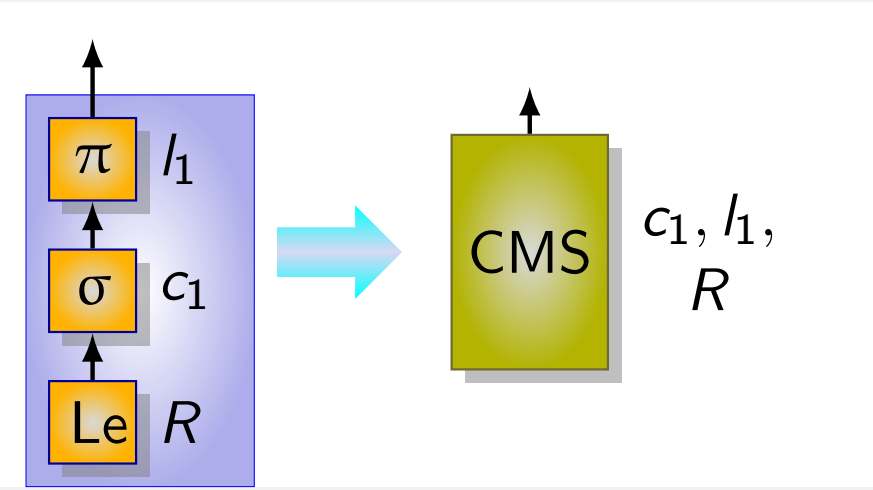
Clustered Matching Index Scan

- O método de acesso físico **Index Scan** pode ser implementado em um SGBD por mais de um algoritmo, para acessar relações que não estão *clusterizadas*:
  - Ele pode ser chamado *Equal Unique Index Lookup* quando a busca é por comparação por igualdade uma chave da relação;
  - Ele pode ser chamado *Unclustered Matching Index Scan* quando a busca é por comparação usando uma chave de busca que não é única;
  - Em geral, o acesso por *Index Scan* pode ser usado para comparação usando operadores de abrangência (*range predicates*) ou quando uma chave de busca não é única.
- Caso a relação esteja *clusterizada*, pode ser usado o método de acesso **Clustered Matching Index Scan**.

Métodos de Acesso para operações unárias

Clustered Matching Index Scan

- O método de acesso **Clustered Matching Index Scan** também empacota uma sequência de operadores Le R , σ c_i e π l_i usando um índice na execução do operador de seleção.



- O *Clustered Matching Index Scan* é semelhante ao *index scan*, a menos que ele é utilizado quando os dados são mantidos sincronizados com a lista de RID da folha do índice.

Métodos de Acesso para operações unárias

Clustered Matching Index Scan

- Vamos assumir que esteja sendo usado um índice B-tree. Então o *Clustered Matching Index Scan*:

  ☞ Acessa o índice até a folha que contem o RowId da primeira tupla com a chave de busca,

  ☞ lê em sequência todas as folhas a partir da primeira, até achar uma chave diferente da chave de busca,

  ☞ **lê** sequencialmente todas as paginas distintas apontadas pelas entradas das folhas lidas,

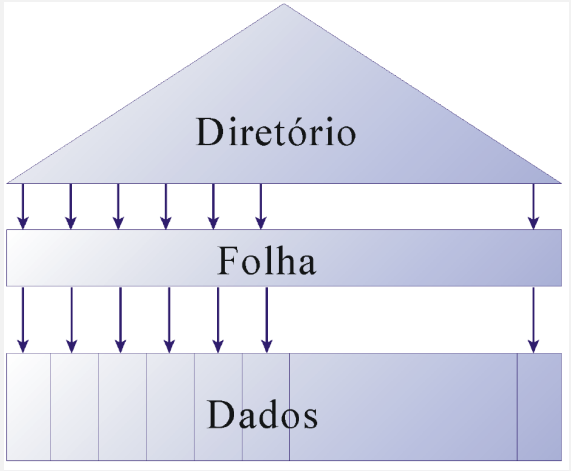
  ☞ caso haja mais critérios de busca não cobertos pelo índice, aplica o critério combinado dos demais **operadores de seleção** empacotados para todas as tuplas lidas,

  ☞ e escreve no *buffer* de saída as tuplas projetadas segundo a lista de tuplas do **operador de projeção** "empacotado".

Métodos de Acesso para operações unárias

Clustered Matching Index Scan

- **O custo** do *Clustered Matching Index Scan* é a leitura aleatória dos diretórios do índice (um para cada nível H da árvore), mais a leitura sequencial das *NFolhas* que contém a chave de busca, mais a leitura sequencial das $Sel(predicado)$ páginas recuperadas pelo índice:
$RHR + R(NFolhas)S + R(Sel(predicado) * NPags)S$.



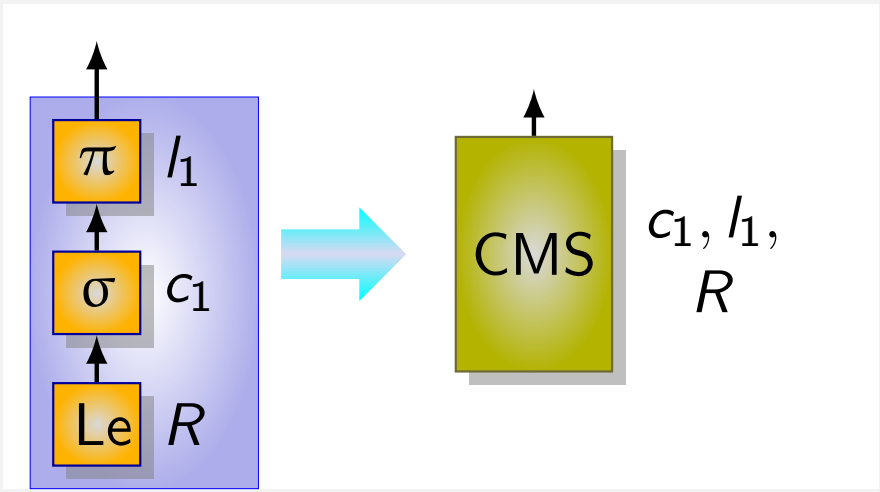
Métodos de Acesso para operações unárias

Clustered Matching Index Scan – Exemplo Consulta 2

- Vamos assumir que a relação alunos é *clusterizada* pela chave primária.
- Nesse caso, a consulta do exemplo 2 sobre a relação de alunos:

```
SELECT RA, Nome FROM Alunos
    WHERE RA='1231231234'
```

- que gera a árvore de comandos:
- Agora gera o plano de acesso físico:



Mas o método *Clustered Matching*

Index Scan não altera o tempo de busca, pois como antes, esse comando irá demorar

R4R $= \frac{4}{100} = 0,04$ segundos.

Métodos de Acesso para operações unárias

Clustered Matching Index Scan – Exemplo Consulta 3

- No entanto, a consulta do exemplo 3 sobre a relação de alunos:

```
SELECT RA, Nome FROM Alunos
    WHERE RA BETWEEN 2000000000 AND 3000000000
```

- embora gere a mesma árvore de comandos:
  - agora tem as tuplas sincronizadas com os nós-folha do índice, e portanto irá demorar $\boxed{\text{R3R}} + \boxed{\text{R103S}} + \boxed{\text{R}((1 - Sel(c_1)) * NTuplas)\text{S}}$, onde $c_1 = (\texttt{RA BETWEEN 2000000000 AND 3000000000})$;
  - $1 - Sel(\texttt{RA BETWEEN 2000000000 AND 3000000000}) = \frac{3000000000 - 2000000000}{9988776655 - 0011223344} \approx 0,1$
  - Portanto $\frac{3}{100} + \frac{103}{1.000} + \frac{0,1 * 8.000}{1.000} = 0,03 + 0,103 + 0,80 = 0,933$ segundos.

c_1, I_1, R

Métodos de Acesso para operações unárias

index Scan – Exemplo Consulta 3

- Comparando com o uso do *index scan* não *clusterizado*, vê-se que há um ganho de 26,80 para 0,93 segundos.
- Isso se deve a dois motivos:
  1. A "chuva de ponteiros" sobre as páginas de dados desaparece, pois os dados ficam na mesma sequência dos nós-folha;
  2. Cada página é lida apenas uma vez, pois múltiplos ponteiros que caem na mesma folha são eliminados por serem solicitados em sequência, e portanto garante-se que as páginas necessárias continuam no *buffer*.

Métodos de Acesso para operações unárias

Index-only Scan

- Uma relação somente pode ser *clusterizada* pela sua chave primária, portanto não é possível sincronizar as tuplas com outro índice.
- No entanto, é possível um efeito parecido com a seguinte observação:

É possível construir um índice que contém, sozinho, todos os dados necessários para responder a uma dada consulta.

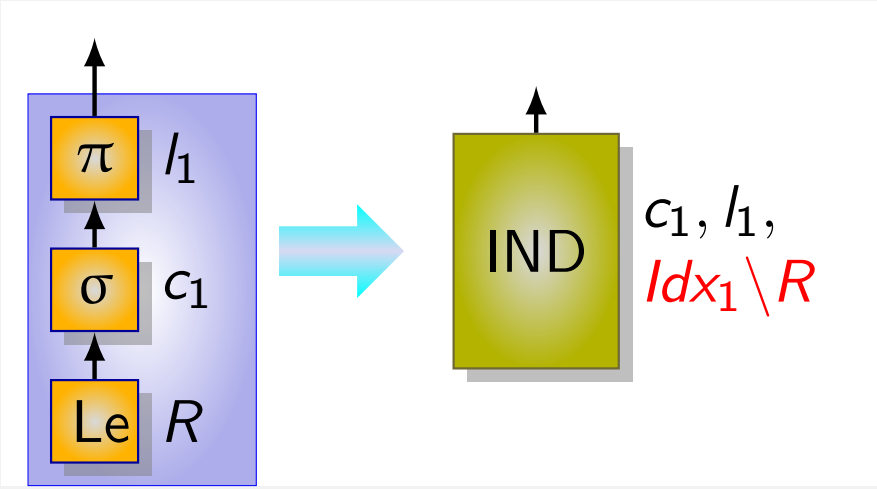
- Nesse caso, não é necessário o acesso aos dados: basta obter no índice os dados solicitados.
- O método de acesso que não faz a leitura dos dados, apenas do índice é chamado **Index-only Scan**.

☞ Veja que esse índice é equivalente ao *index scan* ou ao *clustered matching index scan*, pois como não é feito o acesso aos dados, é indiferente se os dados estão sincronizados ou não.

Métodos de Acesso para operações unárias

Index-only Scan

- O método de acesso *Index-only Scan* também empacota uma sequência de operadores Le R , σ c_i e π l_i :



- No entanto, o *Index-only Scan* muda bastante a estrutura da árvore de comandos da consulta, pois ele altera a leitura básica, que deixa de ser feita sobre os dados e passa a ser feita apenas no índice.

Métodos de Acesso para operações unárias

Index-only Scan

- Vamos assumir que esteja sendo usado um índice B-tree. Então o *Index-Only Scan*:

  ☞ Acessa o índice até a folha que contem o RowId da primeira tupla com a chave de busca,

  ☞ lê em sequência todas as folhas a partir da primeira, até achar uma chave diferente da chave de busca,

  ☞ monta cada chave lida como se fosse uma tupla,

  ☞ caso haja mais predicados cobertos pelo índice, aplica o critério combinado dos demais **operadores de seleção** empacotados para filtrar todas as tuplas montadas,

  ☞ e escreve no *buffer* de saída as tuplas montadas já projetadas segundo a lista de tuplas do **operador de projeção** "empacotado".

- A aplicação de múltiplos critérios de busca cobertos pelo índice é tratada de maneira especial:

Métodos de Acesso para operações unárias

Index-only Scan — Aplicação de múltiplos critérios de busca cobertos pelo índice

- Apenas **predicados indexáveis** podem ser utilizados como chave de busca em um método de acesso *Index-Only Scan*.
- Outros predicados que podem ser comparados no índice somente podem ser usados como um predicado para *screening*,
- e são usados como **predicados de restrição** para eliminar tuplas que passam pelos predicados indexáveis.

Métodos de Acesso para operações unárias

Index-only Scan — Aplicação de múltiplos critérios de busca cobertos pelo índice

- Por exemplo, suponha que a relação de alunos tem os seguintes atributos:
  `Alunos={RA, Nome, NomeMae, DataNasc, Altura, Cidade}`
- é criado o índice:
  `CREATE INDEX IdxNomeMae ON Alunos(Nome, NomeMae, DataNasc)`
- e é feita a seguinte consulta:

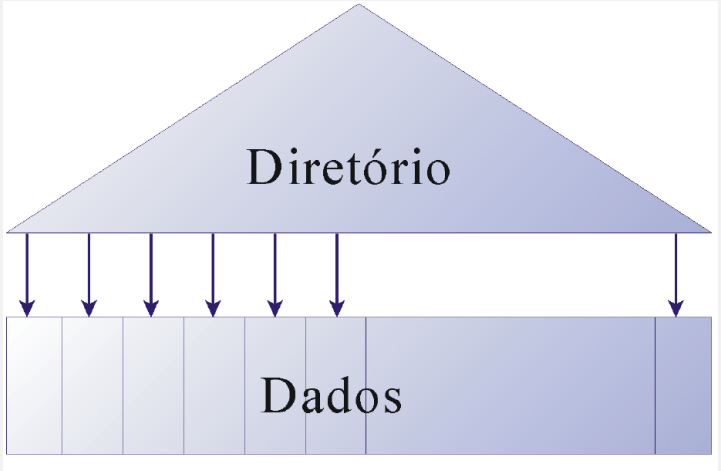
```
SELECT Nome, NomeMae FROM Alunos
    WHERE Nome=:nome AND DataNasc=:data;
```

- Como `NomeMae` não é atributo indexável (não existe predicado que o use), `IdxNomeMae` não pode ser usado para busca,
- mas o predicado `DataNasc=:data` pode ser usado como predicado de restrição no próprio método de acesso *Index-Only Scan*.

Métodos de Acesso para operações unárias

Index-only Scan

- O efeito geral do *index-only scan* é do acesso a uma estrutura de dados primária.
- **O custo** do *Index-only scan* é a leitura aleatória dos diretórios do índice (um para cada nível $H - 1$ da árvore), mais a leitura sequencial das *NFolhas* que contém a chave de busca:
  $R(H-1)R + R(NFolhas)S$.



Métodos de Acesso para operações unárias

Exemplo – Consulta 4 – Acesso indexado apenas por índice

- Suponha que é frequente solicitar a `Idade` de um aluno dado seu `Nome`, e que é frequente solicitar os `Nomes` de todos os alunos de uma dada `Cidade`,
- então os seguintes índices podem ser criados:

```
CREATE INDEX NomeIdade ON Alunos(Nome, Idade);
CREATE INDEX CidadeNome ON Alunos(Cidade, Nome);
```

- Como previsto, agora se quer saber a idade de um aluno:

```
SELECT Nome, Idade
    FROM Alunos
    WHERE Nome=:nome;
```

- Basta acessar o índice `NomeIdade`, obter todos os pares <Nome, Idade> cujas chaves têm valor do `Nome` dado, e retornar o resultado (sem acessar o segmento de dados).
- Assumindo que essa árvore ainda terá altura $H = 3$ e que o número médio de tuplas repetidas por chave é menor do que dois, o custo dessa consulta é $R(2)R + R(2)S \approx \frac{4}{100} = 0,04$s (acessar duas páginas em sequência é equivalente ao acesso aleatório).

Métodos de Acesso para operações unárias

Exemplo – Consulta 4 – Acesso indexado apenas por índice

- Vamos agora considerar a segunda consulta: Obter os `Nomes` de todos os alunos de uma dada `Cidade`, dada por:

```
SELECT Nome
    FROM Alunos
    WHERE Cidade=:cidade;
```

- Novamente, basta acessar o índice `CidadeNome`, obter todos os pares `<Cidade, Nome>` cujas chaves têm valor da `Cidade` dado, e retornar o resultado (novamente sem acessar o segmento de dados).
- Assumimos novamente que essa árvore ainda terá altura $H = 3$ mas o número médio de tuplas repetidas por chave é dado por: $\frac{Sel(pred) * Ntuplas}{TuplasPorPagina}$. Para isso, é necessário estimar o número de cidades distintas que têm alunos na relação. Usando as estimativas feitas no inicio da apresentação, temos que $Sel(\text{Cidade} = cte) * 80.000 = \frac{80.000}{700} = 114,29 \approx 114$.
- O tamanho e cada chave é $dir + 15 + 15 = 32$, então o número de tuplas por página é $\left\lfloor \frac{2.000 - 2*4}{32} \right\rfloor = 62.25 = 62$.
- Portanto, o custo dessa consulta é $R(2)R + R(114/62)S = \frac{2}{100} + \frac{1,84}{1000} \approx 0,04$s novamente.

Métodos de Acesso para operações unárias

Exemplo – Consulta 4 – Acesso indexado apenas por índice

- Veja que se não existisse o índice `CidadeNome`, seria necessário realizar um *table scan* sobre toda a relação, ao custo (já calculado no exemplo 1) de 3,81 segundos.
- Caso houvesse um índice sobre `Cidade` mas que não incluisse o `Nome`, seria necessário utilizar um *index scan* nesse índice, ao custo de $\mathrm{R}\mathit{H}\mathrm{R} + \mathrm{R}(\mathit{NFolhas})\mathrm{S} + \mathrm{R}\mathit{NTuplas}\mathrm{T} = \frac{3}{100} + \frac{80.000}{62*700*1.000} + \frac{114}{300} = 0,03 + 0,0018 + 0,38 = 0,41$ segundos,
- ou seja, aproximadamente 10 vezes mais demorado do que utilizar o índice `CidadeNome`.

Métodos de Acesso para operações unárias

Multi-Index Scan

- Se na consulta houver algum outro operador de seleção que envolva um atributo que não está no índice, o método *index-only scan* não pode ser usado.
- No entanto, uma outra observação pode ajudar novamente:

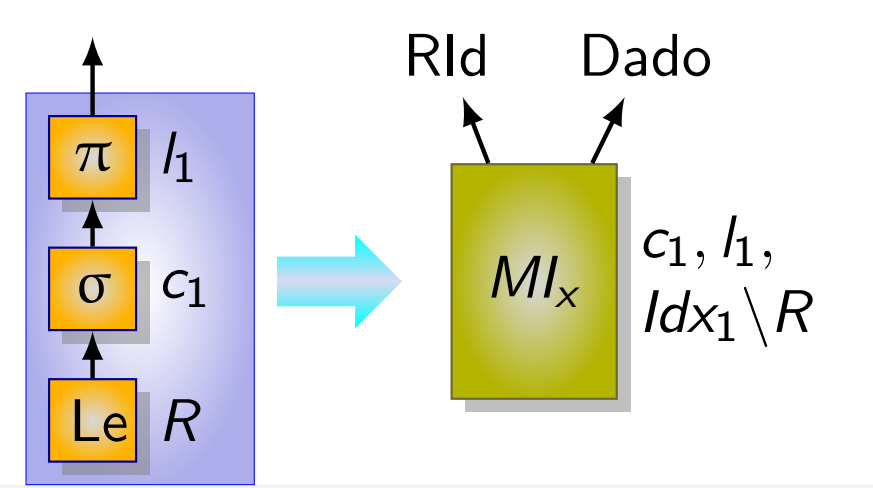
Todo índice, mesmo que seja usado apenas para *index-only scan*, tem o *RowId* para as tuplas de dados correspondentes.

- Isso significa que se houver dois ou mais índices que, juntos, contenham todos os dados necessários a uma consulta, é possível integrar os dados de todos eles usando o *RowId* como chave de junção, e não é necessário acessar os segmentos de dados.
- Mas para isso, o método que faz a leitura de um índice tem que recuperar os dados e os *RowId* como duas informações separadas.
- O método de acesso do índice recuperando as chaves e os *RowId*, sem fazer a leitura dos dados, é chamado **Multi-Index Scan** – MI_x.

Métodos de Acesso para operações unárias

Multi-Index Scan

- O método de acesso *Multi-Index Scan* – MI_x também empacota uma sequência de operadores [Le R], [σ c_i] e [π l_i]:



- No entanto, o *Multi-Index Scan* gera dois conjuntos de saída: a relação de tuplas (como todo método de acesso) e uma Lista de *RowId*, a *RId-List*.

Métodos de Acesso para operações unárias

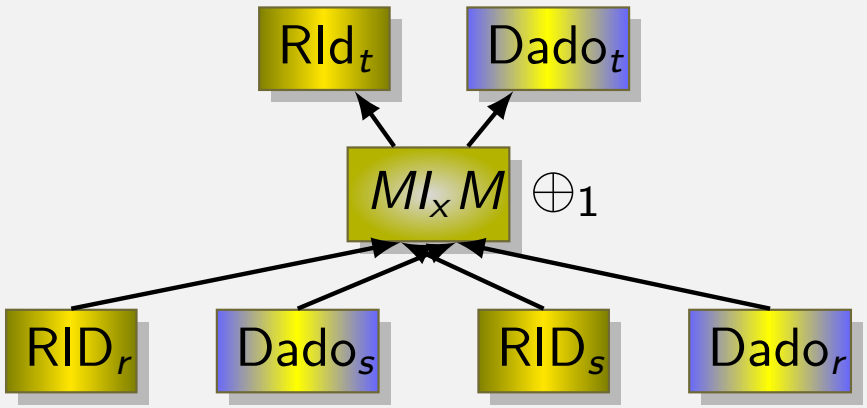
Multi-Index Scan

- Vamos assumir que esteja sendo usado um índice B-tree. Então o *Multi-Index Scan*:
  - ☞ Acessa o índice até a folha que contem o *RowId* da primeira tupla com a chave de busca,
  - ☞ lê em sequência todas as folhas a partir da primeira, até achar uma chave diferente da chave de busca,
  - ☞ monta cada chave lida como se fosse uma tupla,
  - ☞ e cria uma lista de *RowId*, a *Rid-list*,
  - ☞ caso haja mais critérios de busca cobertos pelo índice, aplica o critério combinado dos demais **operadores de seleção** empacotados para filtrar todas as tuplas montadas,
  - ☞ escreve no *buffer* de saída as tuplas montadas já projetadas segundo a lista de tuplas do **operador de projeção** "empacotado",
  - ☞ ordena a *Rid-list* e a escreve em uma área específica para *Rid-list* na estrutura de *buffers*.

Métodos de Acesso para operações unárias

Multi-Index Scan

- O método *Multi-Index Scan* somente é usado quando existe mais de um índice que pode ser usado para acesso *index-only*.
- Somente podem existir duas áreas para *Rid-list* simultaneamente.
- Isso significa que sempre deve ser aplicado um operador de integração de duas *Rid-list*, chamado **Multi-Index Merger** – MI_xM, antes que uma terceira possa ser aplicada:



Métodos de Acesso para operações unárias

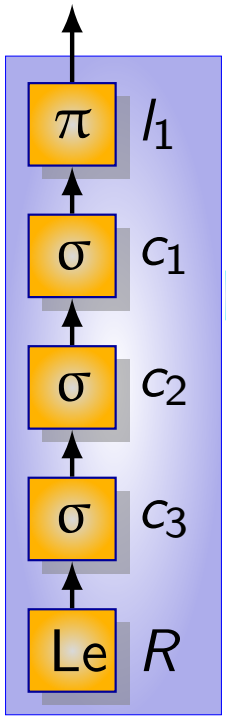
Multi-Index Scan

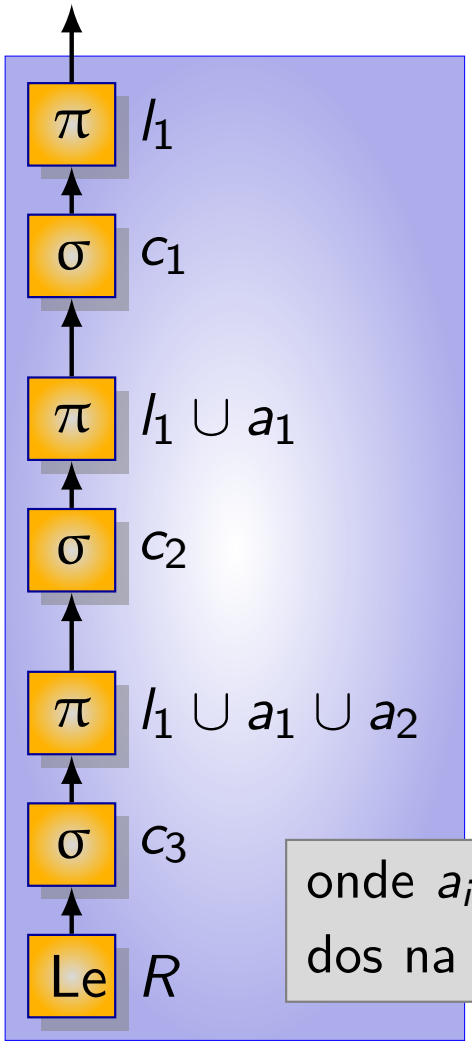
- Para exemplificar, vamos assumir que existam 3 índices Idx_1, Idx_2 e Idx_3, respectivamente sobre os atributos c_1, c_2 e c_3,
- e seja colocada a seguinte consulta:

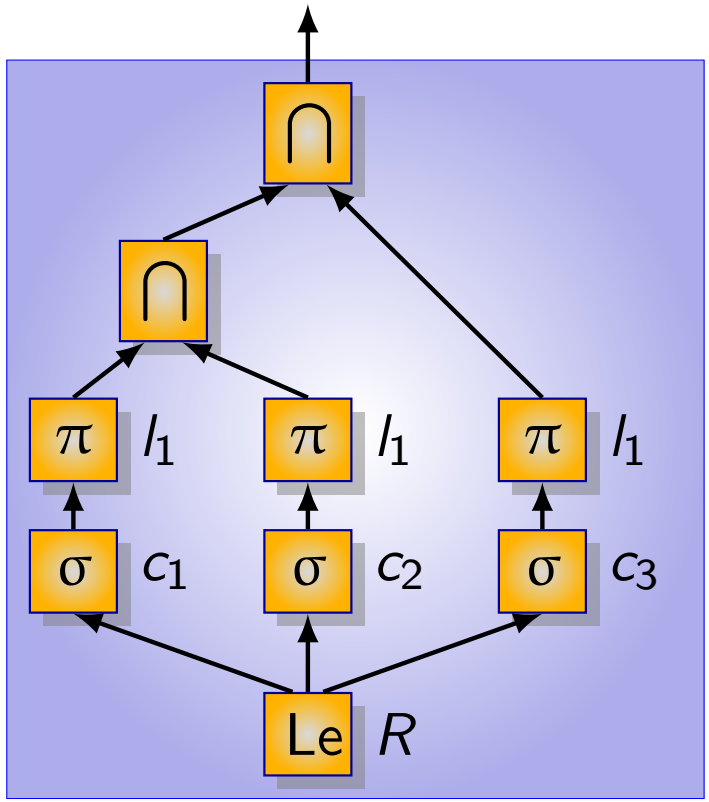
```
SELECT *
      FROM R
      WHERE c1 AND c2 AND c3;
```

Métodos de Acesso para operações unárias

Multi-Index Scan



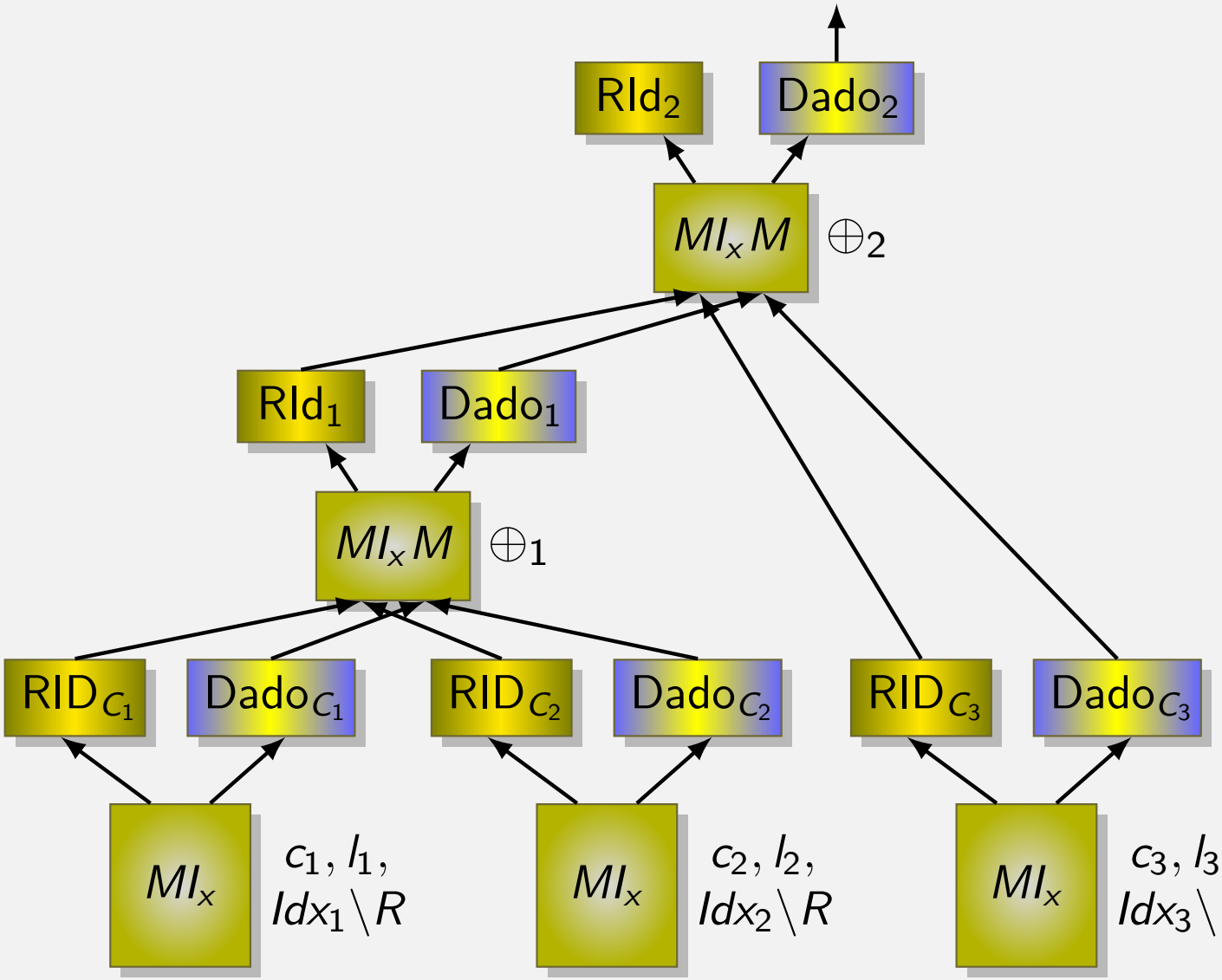




onde a_i são os atributos envolvidos na condição c_i

Métodos de Acesso para operações unárias

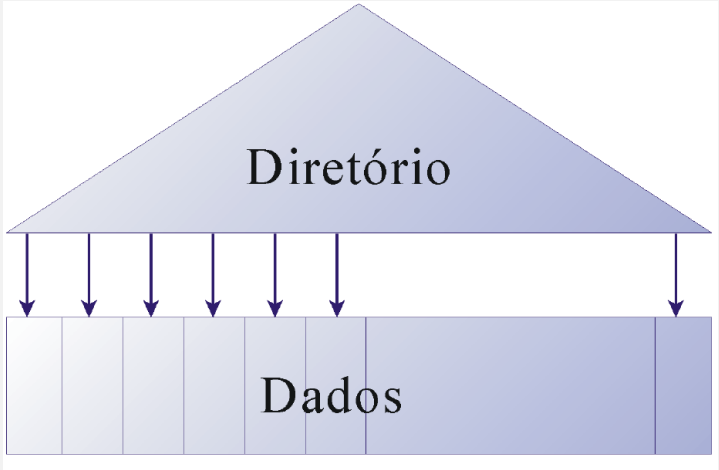
Multi-Index Scan



Métodos de Acesso para operações unárias

Multi-Index Scan

- O operador *Multi-Index Scan* é aplicado para cada índice usado. Portanto, ele é usado sempre mais de uma vez.
- **O custo** de cada execução do *Multi-Index Scan* é a leitura aleatória dos diretórios do índice (um para cada nível $H-1$ da árvore), mais a leitura sequencial das *NFolhas* que contém a chave de busca: $\mathrm{R}(H-1)\mathrm{R} + \mathrm{R}(NFolhas)\mathrm{S}$.



Métodos de Acesso para operações unárias

Multi-Index Scan

- Para poder utilizar o método de acesso físico *multi-index scan*, algumas condições devem ser seguidas:
- A previsão de espaço necessário para a ordenação dos métodos *multi-index scan* e do método *multi-index merge* intermediário deve indicar que ela pode ser completamente realizada em memória, caso contrário o *multi-index scan* não é escolhido.
- A seguir deve-se calcular a seletividade composta de todos os predicados indexáveis para cada índice.
- Sempre executa-se primeiro os índices que têm a maior seletividade composta de todos os predicados indexáveis desse índice.

Métodos de Acesso para operações unárias

Exemplo – Consulta 5 – Acesso indexado por múltiplos índices

- Suponha que os seguintes índices foram criados sobre a relação `Alunos`:

```
CREATE INDEX IdadeNome ON Alunos(Idade, Nome);
CREATE INDEX CidadeNome ON Alunos(Cidade, Nome);
```

- É solicitada a consulta sobre quais são os nomes dos alunos com menos de 20 anos que vêm de Coronel Vidida:

```
SELECT Nome, Idade
    FROM Alunos
    WHERE Cidade='Coronel Vivida' AND Idade<20;
```

- Primeiro calcula-se a seletividade de todos os predicados indexáveis. Para isso, usam-se as estatísticas disponíveis (exemplo):
  - $Sel(\text{Cidade} = \text{'Coronel Vivida'}) = \frac{1}{700}$
  - $Sel(\text{Idade} = 20) = \frac{1}{50}$
- Executa-se primeiro os índices que têm a maior seletividade composta de todos os predicados indexáveis de um índice.
- Neste caso processa-se primeiro o índice `CidadeNome` e depois o índice `IdadeNome`.

Métodos de Acesso para operações unárias

Exemplo – Consulta 5 – Acesso indexado por múltiplos índices

- O custo do índice `CidadeNome` é: $H = 3$, Número de chaves por folha$= \left\lfloor \frac{2.000-2*4}{2+15+15} \right\rfloor = 62,25 = 62$, número de tuplas repetidas por chave$NTuplas = \frac{80.000}{700} = 114,29 = 114$, número de folhas a serem lidas $= NFolhas = \left\lceil \frac{114}{62} \right\rceil = 1.84 = 2$.
- O custo para acessar o índice `CidadeNome` é: $\boxed{\text{R}(H-1)\text{R}} + \boxed{\text{R}(NFolhas)\text{S}} = \boxed{\text{R}(2)\text{R}} + \boxed{\text{R}(2)\text{S}} \approx \frac{4}{100} = 0,04$ segundos.
- o Custo para acessar os dados depois de filtrado esse índice é: $\boxed{\text{R}(\text{NTuplas})\text{T}} = \frac{114}{300} = 0,38\text{s}$.

Métodos de Acesso para operações unárias

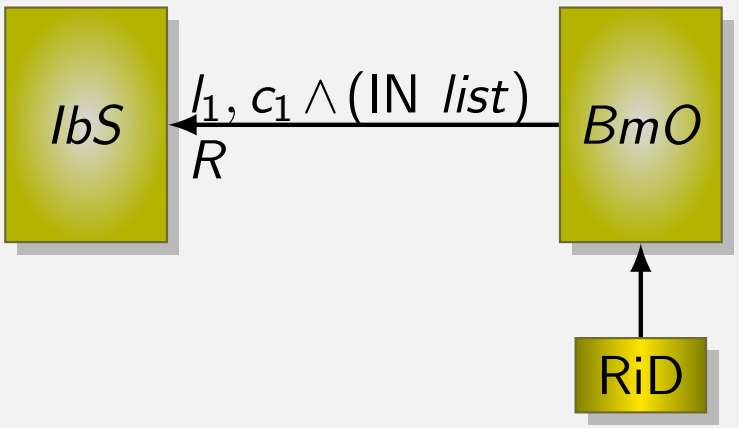
Exemplo – Consulta 5 – Acesso indexado por múltiplos índices

- O custo do índice `IdadeNome` é: $H = 3$, Número de chaves por folha$= \lfloor \frac{2.000-2*4}{2+15+2} \rfloor = 104,84 = 104$, número de tuplas repetidas por chave$NTuplas = \frac{80.000}{50} = 114,29 = 1.600$, número de folhas a serem lidas $= NFolhas = \lceil \frac{1.600}{104} \rceil = 14,38 = 15$.
- O custo para acessar o índice `IdadeNome` é: $R(H-1)R + R(NFolhas)S =$ $R(2)R + R(15)S = \frac{2}{100} + \frac{15}{1000} = 0,035$ segundos.
- A seletividade composta desses dois índices é $\frac{1}{700} \cdot \frac{1}{50} = \frac{1}{35.000}$, portanto aproximadamente $\frac{1}{35.000} \cdot 80.000 = 2.29$ tuplas serão recuperadas em média.
- o Custo para acessar os dados depois de filtrado pelos dois índices é: $R(NTuplas)T = \frac{3}{300} = 0,01$s.
- Mas como todos os dados necessários já estão obtidos ao custo de $0,04 + 0,035 = 0,075$s, não é necessário nenhum acesso aos dados.
- Finalmente, veja que as ordenações que devem ser executadas em memória corresponde a ordenar 114 *RowIds* para o índice `CidadeNome`, depois ordenar os 1.600 *RowIds* para o índice `IdadeNome` e fazer o *merging* de ambas *Rid-lists*.

Operadores de apoio aos Métodos de Acesso

Bitmap Sorter

- Outro operador de apoio interessante é o **Bitmap Sorter** – *BmO*.
- Ele transforma uma *Rid-list* em um predicado do tipo `IN (lista)` para ser usado em um outro método de acesso aos dados, que seja capaz de "filtrar" dados lidos de uma relação de entrada a partir de uma *Rid-list*.



- O Método de Acesso *IbS* corresponde a qualquer dos métodos já estudados que acessam uma relação-base, agora estendido para filtrar o acesso pelas tuplas indicadas pela `Rid-list`.

Operadores de apoio aos Métodos de Acesso

Exemplo – Consulta 6 – Acesso por múltiplos índices e aos dados

- Por exemplo, suponha que os mesmos índices criados na Consulta 5 do exemplo anterior existam sobre a relação `Alunos`:

```
CREATE INDEX IdadeNome ON Alunos(Idade, Nome);
CREATE INDEX CidadeNome ON Alunos(Cidade, Nome);
```

- Mas agora é solicitada a consulta sobre quais são os nomes, idades e RA dos alunos com menos de 20 anos que vêm de Coronel Vivida:

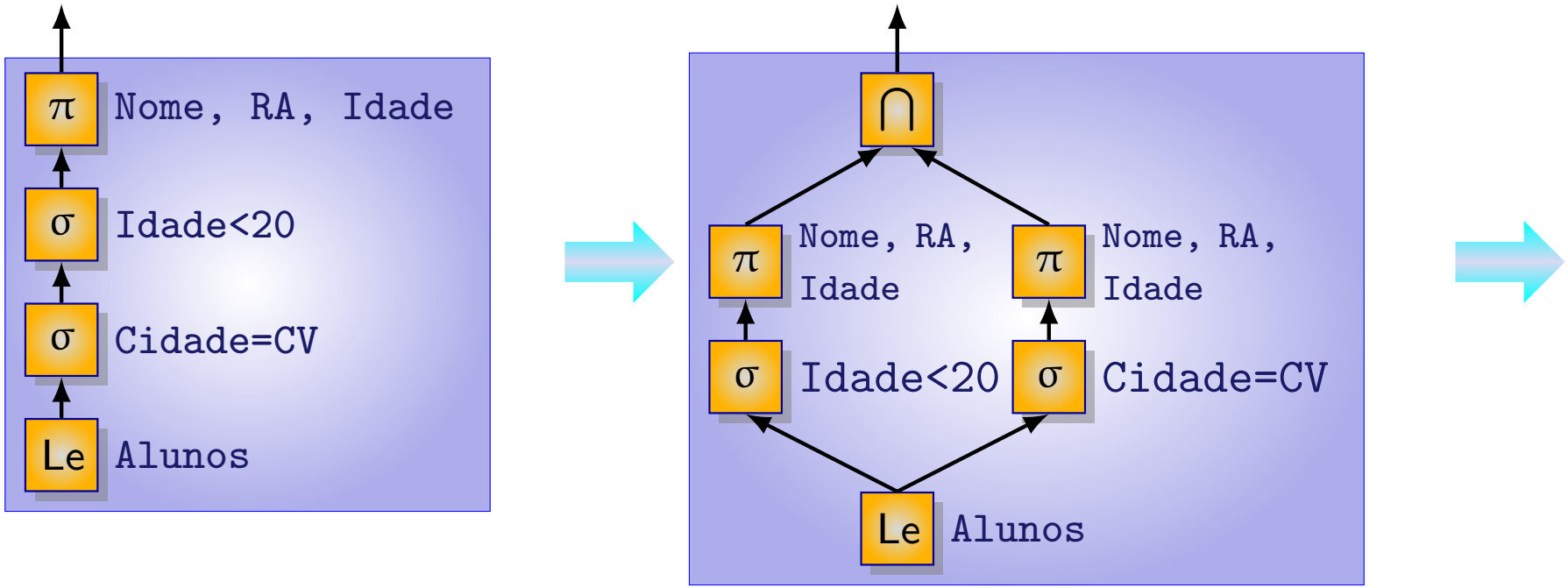
```
SELECT Nome, RA, idade
    FROM Alunos
    WHERE Cidade='Coronel Vivida' AND Idade<20;
```

- Nesse caso, nenhuma combinação de índice de busca pode suprir todos os dados necessários.
- Embora exista uma chave primária sobre o atributo `RA`, ela não pode ser usada como chave de busca porque não tem um predicado sobre ela.
- No entanto, o uso combinado dos dois índices `CidadeNome` e `IdadeNome`, como se viu no exemplo anterior, permite restringir o acesso necessário a uns poucos dados.
- Então pode-se usar os índices para obter a *Rid-list* das tuplas que devem ser lidas, e acessar só elas.

Operadores de apoio aos Métodos de Acesso

Bitmap Sorter

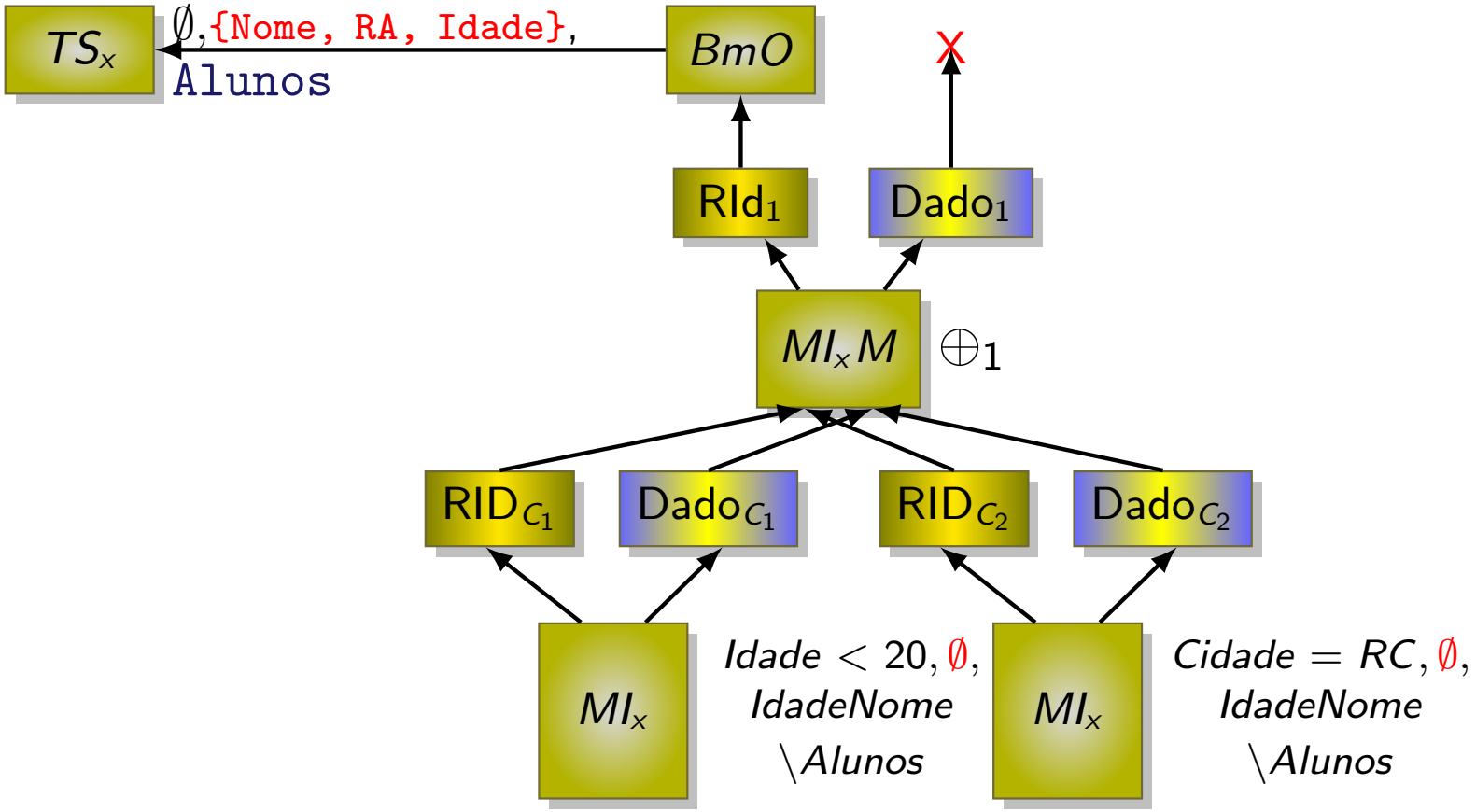
```
SELECT Nome, RA, Idade
    FROM Alunos
    WHERE Cidade='Coronel Vivida' AND Idade<20;
```



Operadores de apoio aos Métodos de Acesso

Bitmap Sorter

```
SELECT Nome, RA, Idade
    FROM Alunos
    WHERE Cidade='Coronel Vivida' AND Idade<20;
```



Operadores de apoio aos Métodos de Acesso

Bitmap Sorter

- De fato, o operador *Bitmap Sorter* gera:
  - a *Rid-list* das tuplas que precisam ser acessadas,
  - junto com um *bitmap* dos *extents* do segmento de dados que precisam ser acessados.
- Com o *bitmap* dos *extents*, o método de acesso usado pode ordenar os acessos, e acessar uma única vez cada *extent* que contém qualquer quantidade de tuplas.
- Portanto, o operador *Bitmap Sorter* permite eliminar a " chuva de ponteiros" sobre as tuplas, a partir de qualquer índice utilizado.

Operadores de apoio aos Métodos de Acesso

Exemplo – Consulta 6 – Acesso por múltiplos índices e aos dados

- Prosseguindo com o exemplo anterior, os mesmos acessos já calculados serão utilizados, o que corresponde ao custo de $0,04 + 0,035$ segundos para os índices `CidadeNome` e `IdadeNome` respectivamente, sem contar ainda o acesso aos dados.
- Como nenhum dos índices usáveis tem o atributo `RA`, ele é obtido direto nas tuplas.
- Como o operador *Multi-Index Merger* já gerou a *Rid-list* de todas as tuplas necessárias, o operador *Bitmap Sorter* extrai os *extents* necessários e passa essa lista para um método de acesso, que lê os dados, faz a projeção final e responde a consulta.
- O custo para o acesso aos dados será o de acessar aleatoriamente as 3 tuplas que foram identificados anteriormente, correspondente a $\text{R3R} = \frac{3}{100} = 0,03$.
- Portanto o custo total dessa consulta será $0,04 + 0,035 + 0,03 = 0,105$ segundos.

Operadores de apoio aos Métodos de Acesso

Exemplo – Consulta 6 – Acesso por múltiplos índices e aos dados

- Vamos considerar um plano de acesso alternativo que não use o método *Multi-Index Scan*.
- Nesse caso, somente um índice pode ser utilizado. Vamos escolher o mais seletivo: `CidadeNome`.
- Seria então utilizado o método *Index Scan*, a um custo de: $\mathrm{R}\mathit{H}\mathrm{R} + \mathrm{R}(\mathit{NFolhas})\mathrm{S} + \mathrm{R}\mathit{n}\mathrm{T} = \frac{3}{100} + \frac{2}{1.000} + \frac{114}{300} = 0,412$ segundos, quatro vezes mais demorado.
- O problema desse acesso é a chuva de ponteiros que o índice `CidadeNome` causa sobre as páginas de dados.
- Além disso, o predicado `(Idade<20)` somente pode ser usado para filtragem, pois não podem ser usados dois índices para acesso indexado.

Roteiro

1. Métodos de Acesso Físico – Operadores de apoio
   - Bitmap Sorter

Bases de Dados
Métodos de Acesso Físicos

Prof. Dr. Ives Renê V. Pola
ivesr@utfpr.edu.br

Departamento Acadêmico de Informática – DAINF
UTFPR – Pato Branco DAINF
UTFPR
Pato Branco - PR

FIM