INSTITUTO FEDERAL
FARROUPILHA

INSTITUTO FEDERAL DE
EDUCAÇÃO, CIÊNCIA E TECNOLOGIA

Admin. de Banco de Dados

Estruturas de Indexação

Prof.: Ícaro Lins Iglesias

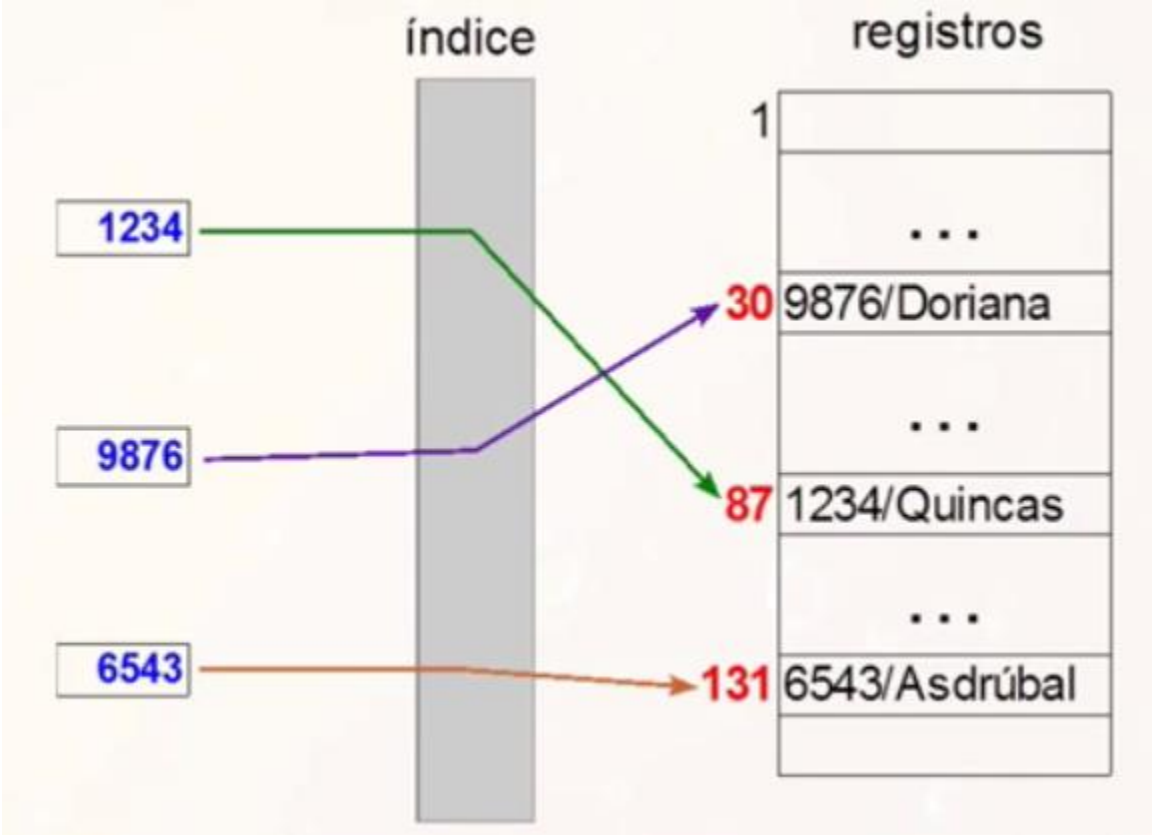
icaro.iglesias@iffarroupilha.edu.br

Motivação

- Até o momento estudamos técnicas de acesso realizadas diretamente nos arquivos (ordenados, desordenados, hashed)
- No mundo real, isto não ocorre, necessitando haver **estruturas de acesso auxiliares**, denominadas índices
- Objetivo: agilizar a recuperação de registros em resposta a certas condições de pesquisa
- São arquivos adicionais ao disco que oferecem **caminhos de acesso secundários** (localizam os registros sem afetar seu posicionamento físico no arquivo)
- Analogia ao mundo real: sumário de um livro

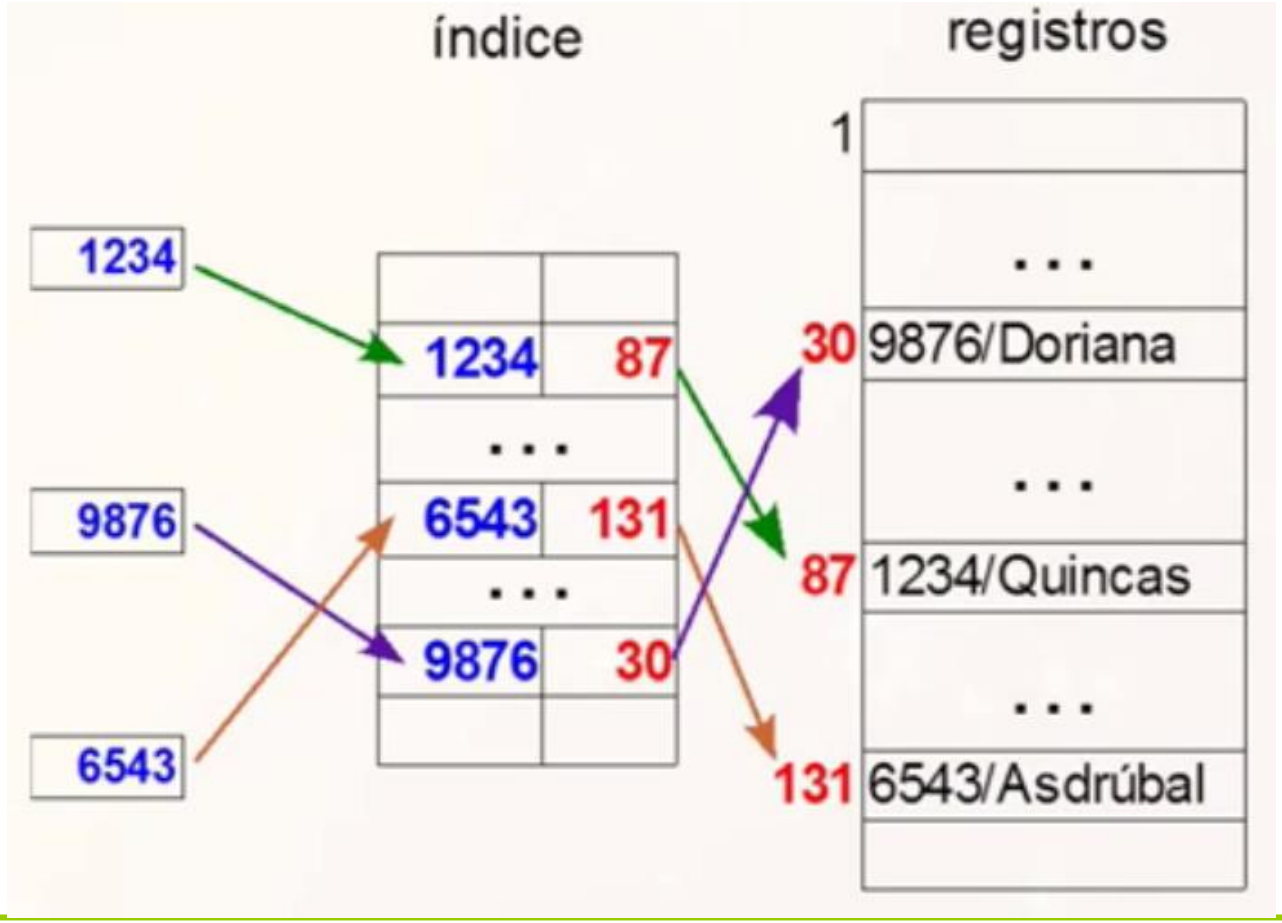
Índices

- Estrutura de dados
- Organiza registros
- Otimiza certas operações de recuperação



Exemplo de índice: lista ordenada de chaves

- Utiliza a busca binária para recuperar registros
- Por que a ordenação ocorre no índice e não no registro?



Vantagens ao usar Índices

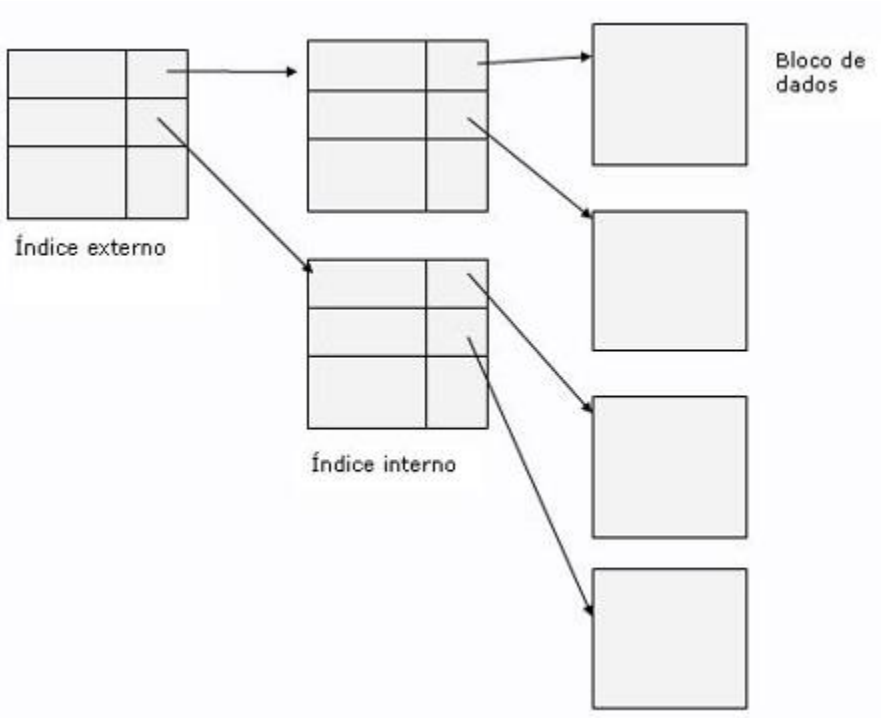
- Melhora a performance das consultas na maioria dos casos. Exemplo: Se tiver 50 linhas você consegue chegar onde quer em até 7 passos, sem o índice demoraria 50. Com mais de um bilhão de linhas, você alcança o que deseja usando índice em pouco mais de 30 passos.
- Estrutura auxiliar independente que pode ser carregada inteira para a RAM
- Por ser independente ao arquivo físico, permite a realização de várias ordenações sem alterar os registros fisicos

Desvantagens ao usar Índices

- Piora a performance em escritas de dados no banco de dados. A alteração do índice implica em acesso de leitura e escrita nele, apesar de ser uma operação eficiente se comparada com o acesso direto à tabela, não deixa de ter um custo adicional.
- Aumenta o consumo de espaço para armazenamento do banco de dados (memória e disco)
- Aumenta a necessidade de manutenção no banco de dados
- Pode diminuir a performance de consultas: operação adicional, soma do tempo gasto com os arquivos envolvidos, complexidade das consultas e volume dos dados.

Tipos de índices

- **Nível único**
  - ✓ Primário
  - ✓ Clustering
  - ✓ Secundário
- **Multinível**
  - ✓ Árvore B
  - ✓ Árvore B+



Terminologia

- Índice Único: não há duplicidade de valores na chave definida (Exemplo: Primary Key)
- Índice Primário: especificado no campo chave de ordenação de um arquivo ordenado (valor único de entrada).
- Índice de Agrupamento (clustering): utilizado quando diversos registros puderem ter o mesmo valor para o campo de ordenação (arquivos agrupados).
- Índice Secundário: utilizado em qualquer campo não ordenado de um arquivo. Um arquivo de dados pode ter vários índices secundários.

Índices Primários

- Índice cuja chave especifica a ordem sequencial do arquivo

Características

- Sua estrutura possui dois campos: valor da PK de âncora e endereço do ponteiro do bloco
- Esparso (não denso): possui entradas de índice para somente alguns dos valores de pesquisa
- Possui menos estradas do que o número de registros do arquivo
- Apresenta ancoragem de blocos: número de entradas de índices = número de blocos no arquivos de dados

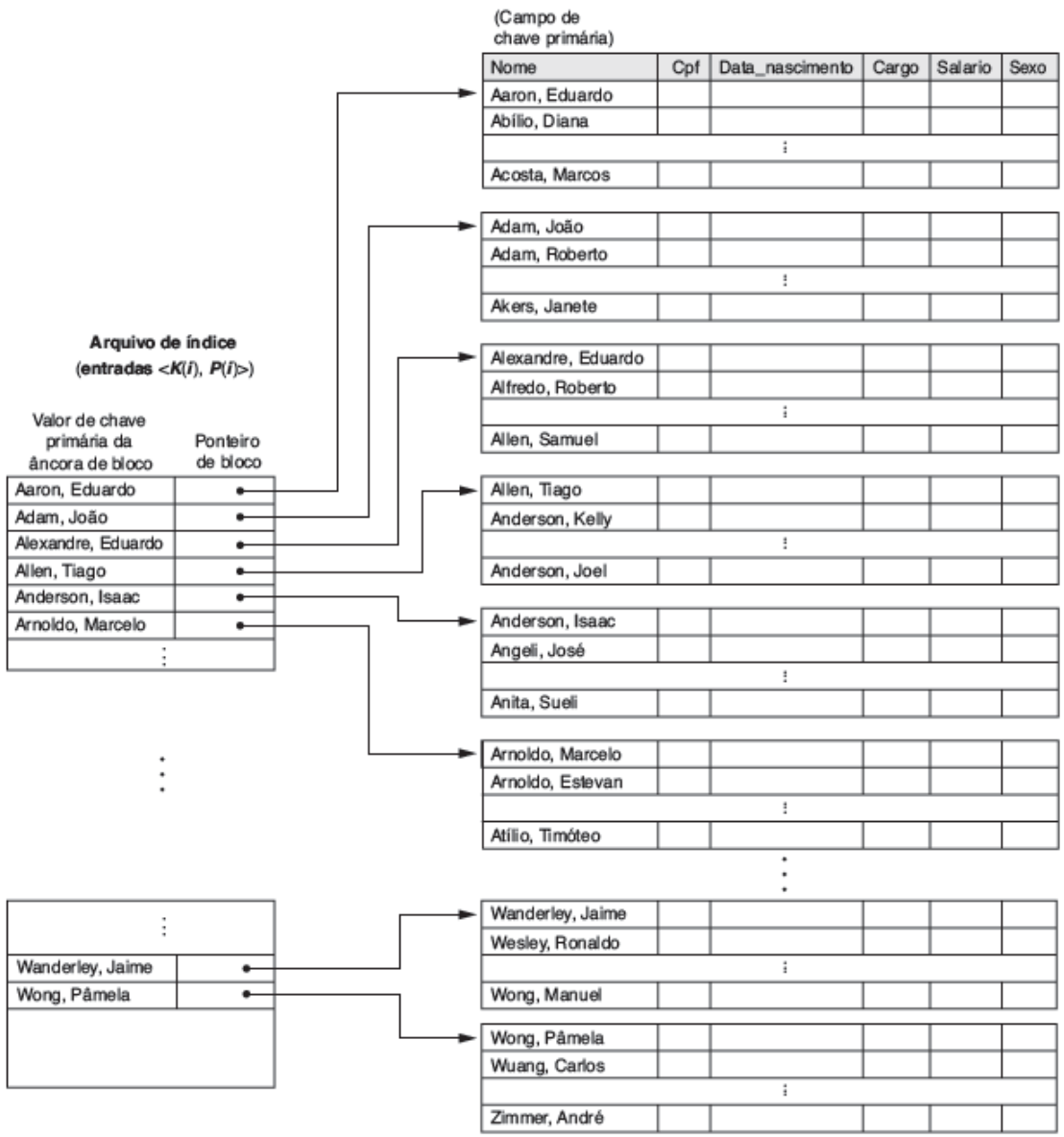


Figura 18.1
Índice primário no campo de chave de ordenação do arquivo mostrado na Figura 17.7.

Índices de agrupamento

- Se os registros de arquivo forem fisicamente ordenados em um campo não chave — que não tem um valor distinto para cada registro —, esse campo é chamado de campo de agrupamento, e o arquivo de dados é chamado de arquivo agrupado.
- Pode-se criar um tipo de índice diferente, chamado **índice de agrupamento**, para agilizar a recuperação de todos os registros que têm o mesmo valor para o campo de agrupamento.

Índices de agrupamento

- Índices clusterings, baseados no campo de ordenação não-chave de um arquivo.

Características

- Também é um arquivo ordenado com dois campos: o primeiro campo é do mesmo tipo do campo de agrupamento do arquivo de dados, e o segundo campo é um ponteiro de bloco de disco.
- Há uma entrada no índice de agrupamento para cada valor distinto do campo de agrupamento, e ele contém o valor e um ponteiro para o primeiro bloco no arquivo de dados que tem um registro com esse valor para seu campo de agrupamento.

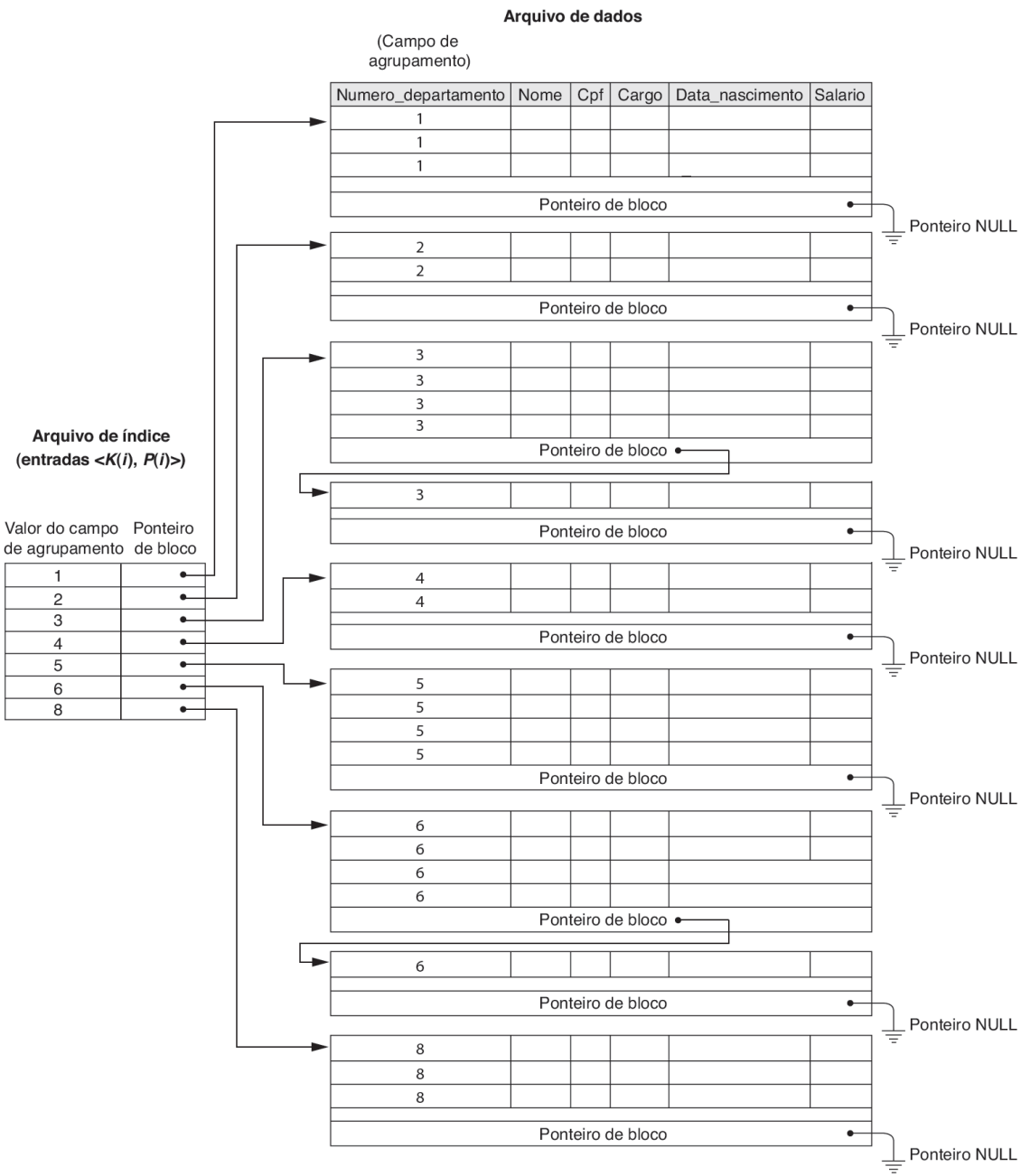


Figura 18.3
O índice de agrupamento com um cluster de bloco separado para cada grupo de registros que compartilham o mesmo valor para o campo de agrupamento.

Índices secundários

- Índice que oferece um meio secundário para acessar um arquivo de dados para o qual um acesso prímário já existe.

Características

- Pode ser criado em um campo que é uma chave candidata e tem um valor único em cada registro, ou em um campo não chave com valores duplicados.
- O índice é novamente um arquivo ordenado com dois campos: campo de índice e ponteiro de *bloco* ou um ponteiro de *registro*
- Modelo Relacional equivale a atributos PK ou UNIQUE
- Índice denso (não esparso): uma entrada de índice para cada registro no arquivo de dados.

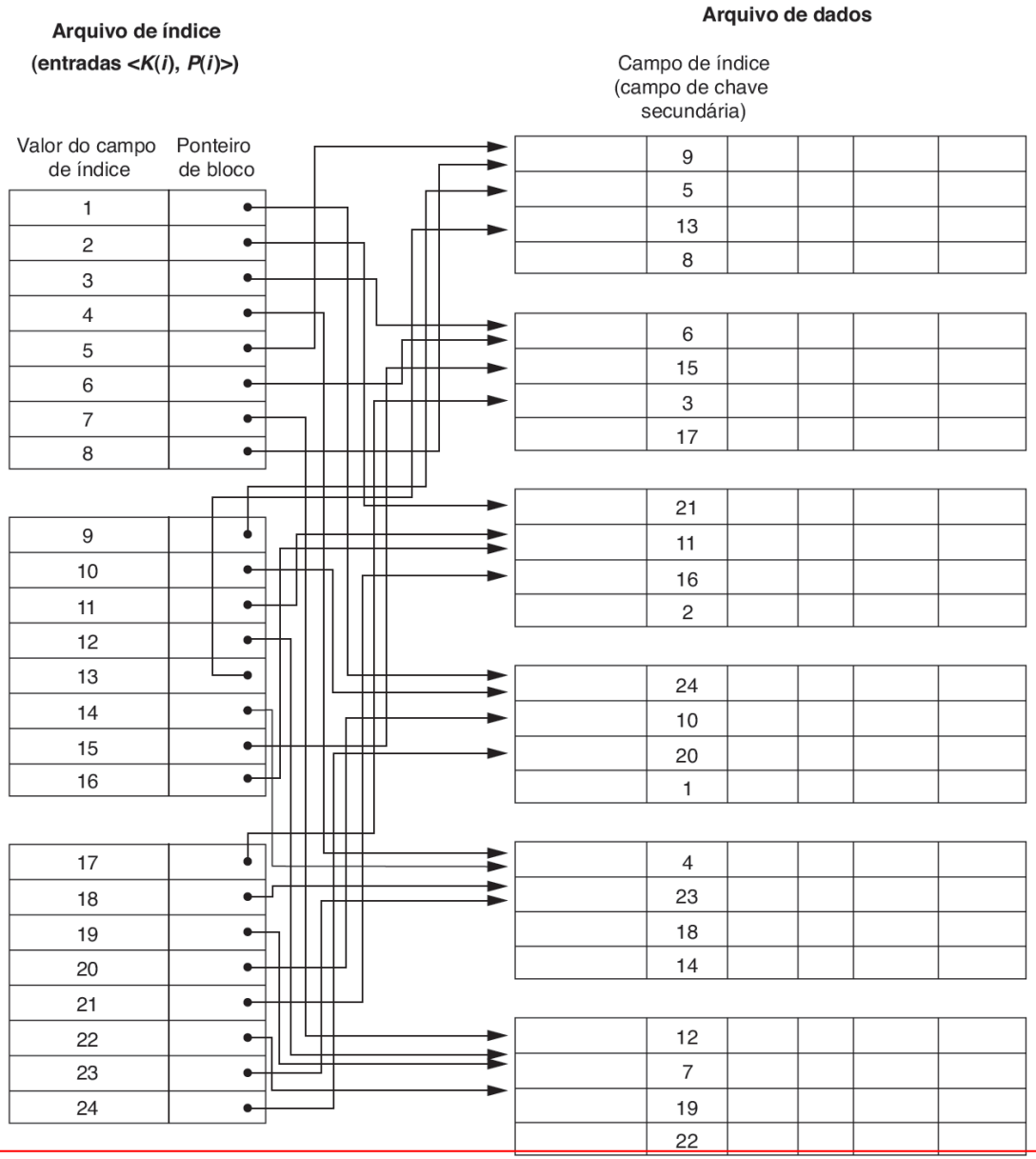


Figura 18.4
Um índice secundário denso (com ponteiros de bloco) em um campo de chave não ordenado de um arquivo.

Resumo tipos de índices

Tabela 18.2

Propriedades dos tipos de índice.

Tipo de índice	Número de entradas de índice (primeiro nível)	Denso ou não denso (esparso)	Ancoragem de bloco no arquivo de dados
Primário	Número de blocos no arquivo de dados	Não denso	Sim
Agrupamento	Número de valores de campo de índice distintos	Não denso	Sim/não[a]
Secundário (chave)	Número de registros no arquivo de dados	Denso	Não
Secundário (não chave)	Número de registros[b] ou número de valores de campo de índice distintos[c]	Denso ou não denso	Não

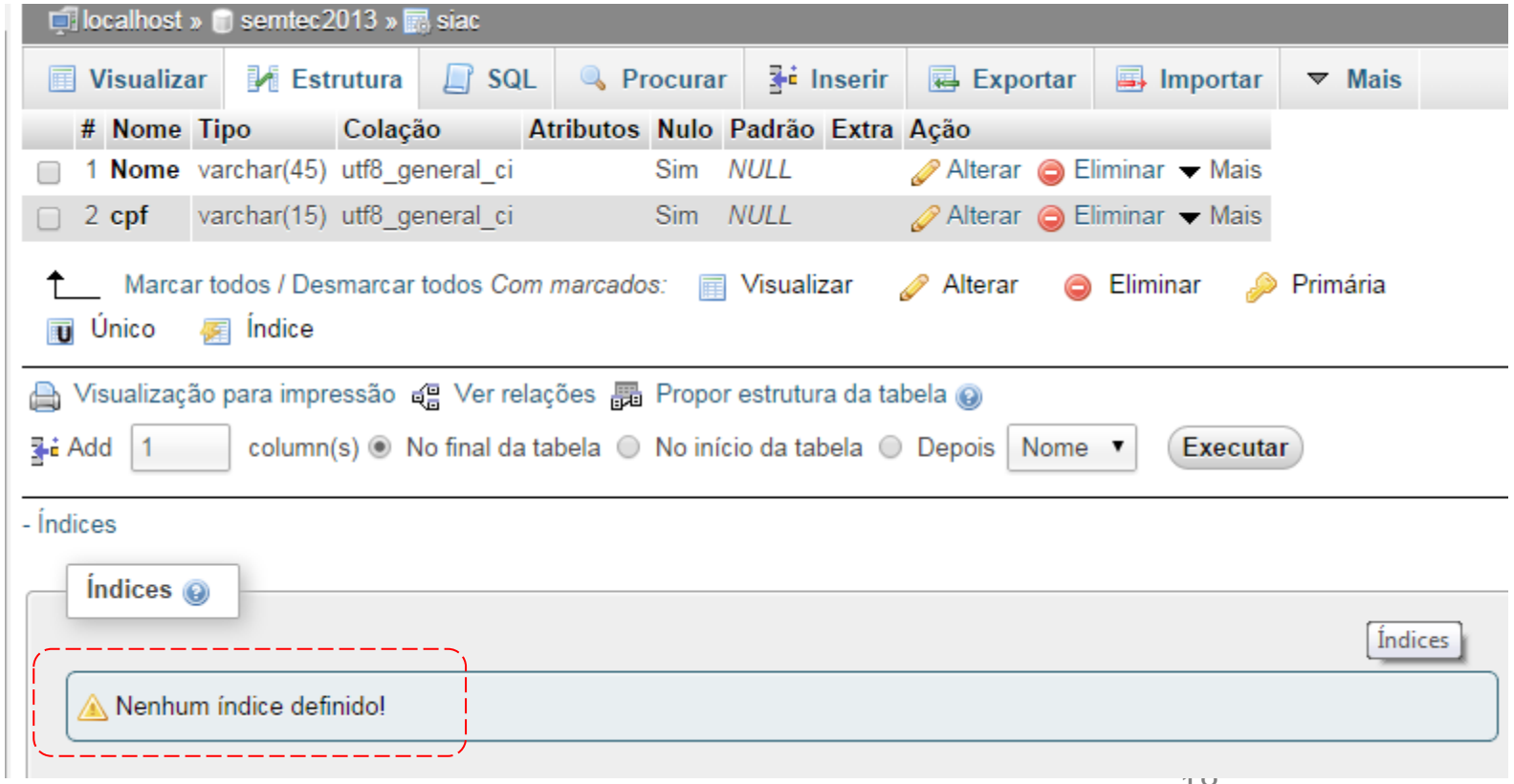
[a] Sim, se cada valor distinto do campo de ordenação iniciar um novo bloco; caso contrário, não.

[b] Para a opção 1.

[c] Para as opções 2 e 3.

DEMO 1

Exemplo de criação de índice utilizando o SGBD MySQL. Fonte: Bóson Treinamentos



Árvore e Indexação

- A indexação em BD é inspirada em árvores binárias, sendo implementadas como árvores balanceadas (BTREE)
- Os dados existem mesmo sem qualquer índice ou árvore
- Cada índice irá gerar uma árvore própria (colunas da tabela. Exemplo: cpf, renda)
- As colunas que compõem um índice estarão presentes tanto nos dados quanto na árvore deste (redundância)

Estrutura de blocos – SQL SERVER

- Blocos possuem tamanho de 8 KB
- Dados são guardados nestes blocos
- Cada bloco pode suportar uma quantidade R de registros, onde R = 8KB/tam do registro
- Quanto maior a largura dos registros, menos registros ele irá comportar, necessitando mais blocos para armazenar os dados
- Largura dos registros é proporcional ao tamanho de suas colunas

DEMO 2

- Define-se um índice primário para a coluna ID, com registros devidamente ordenados!
- Como as BTREE contém apenas as colunas participantes do índice, normalmente conseguirão ter mais entradas em uma página (bloco) de 8KB
- Exemplo:
- ID* = 4 bytes + Ponteiro* = 4bytes, Total 8 bytes
- Bloco = 8KB/8bytes = 1000 registros de índices por bloco

Coluna	Tamanho
ID [INT]	4 bytes
Nome [VARCHAR(100)]	50 bytes
CPF [CHAR(11)]	11 bytes
Renda [SMALLMONEY]	4 bytes
Sexo [CHAR(1)]	1 bytes
DataCadastro [DATE]	3 bytes
Municipio [CHAR(7)]	7 bytes
Outros	240 bytes

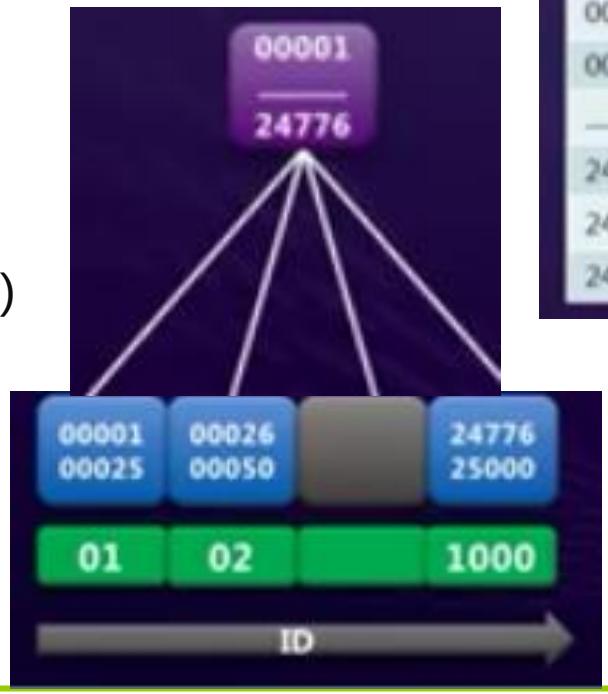
Montagem da BTREE

- 1 registro = 320 bytes, onde 8KB/320 = 25 registros
- 1000 blocos = 25000 registros (clientes, no exemplo)
- Leitura realizada em 2 blocos localiza qualquer ID entre 1 e 25.000

Bloco	ID Inicial	ID Final
0001	00001	00025
0002	00026	00050
0003	00051	00075
0004	00076	00100
0998	24726	24750
0999	24751	24775
1000	24776	25000

Nivel Raiz (bloco de índices+ponteiros)

Nivel folha (blocos de registros)



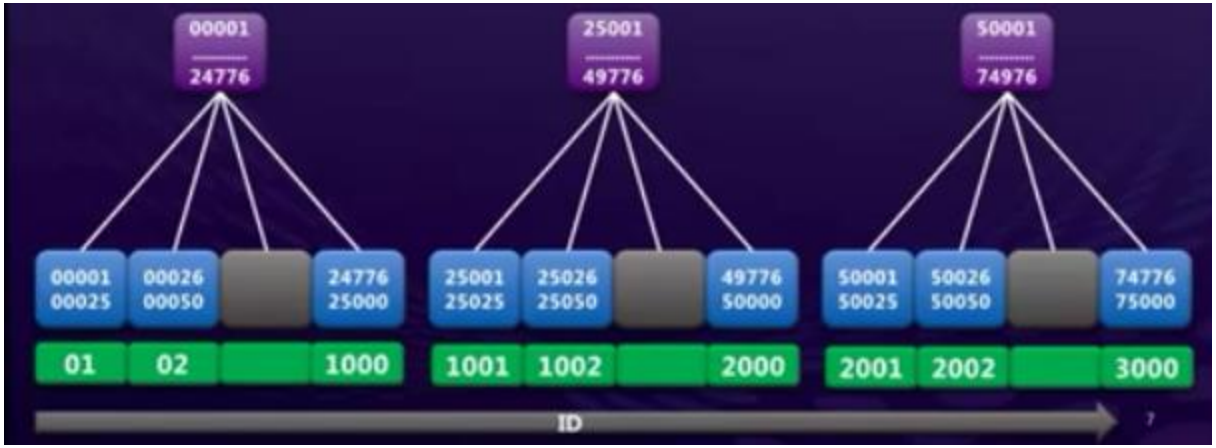
ID Inicial	Bloco
00001	0001
00026	0002
00051	0003
———	———
24726	0998
24751	0999
24776	1000

BTREE expandida - 2 níveis

Desta forma será necessária a leitura **em todos os blocos da raiz** para localizar um determinado registro.

Nivel Raiz (bloco de índices+ponteiros)

Nivel folha (blocos de registros)



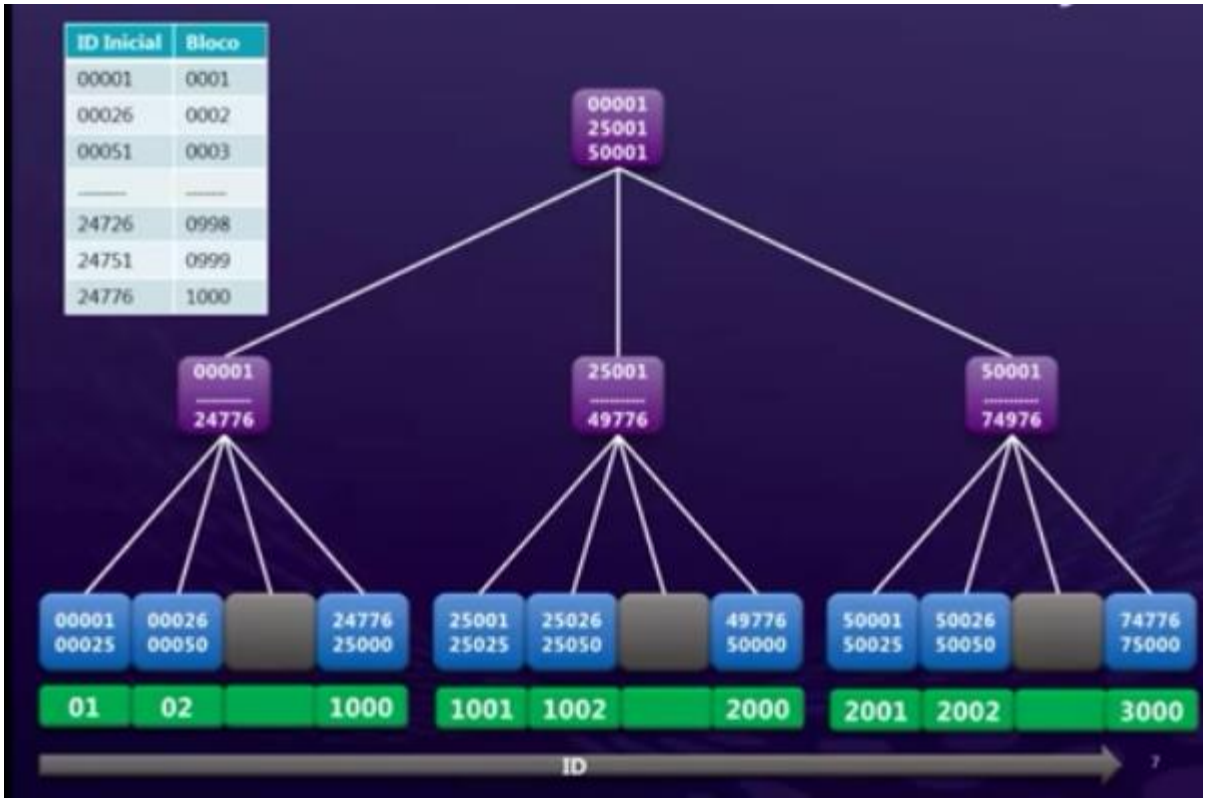
Árvore BTREE completa

Desta forma será possível encontrar qualquer registro entre 1 e 75000 com a leitura de apenas 3 blocos.

Nó Raiz

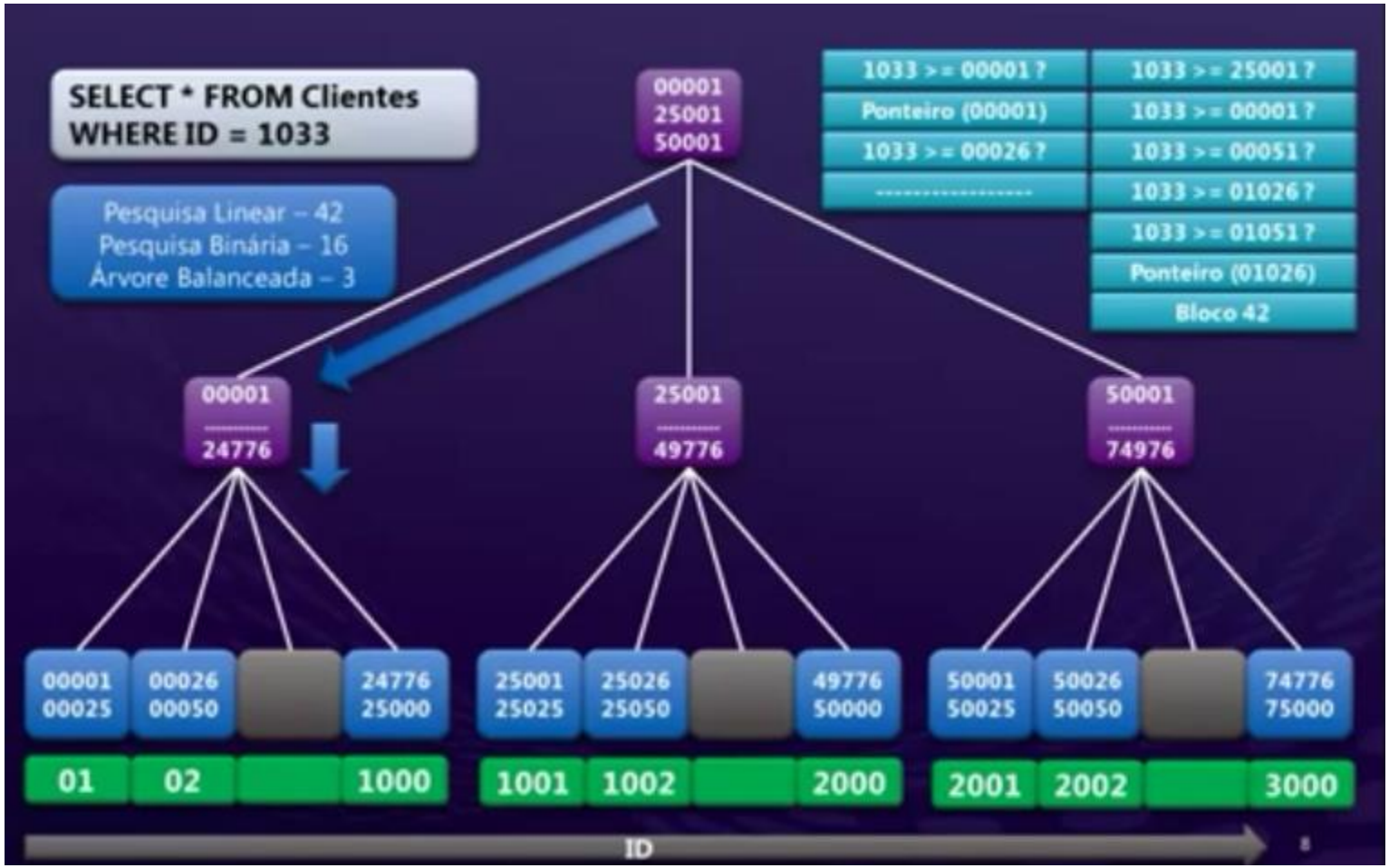
Nó Intermediário

Nó folha



INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA

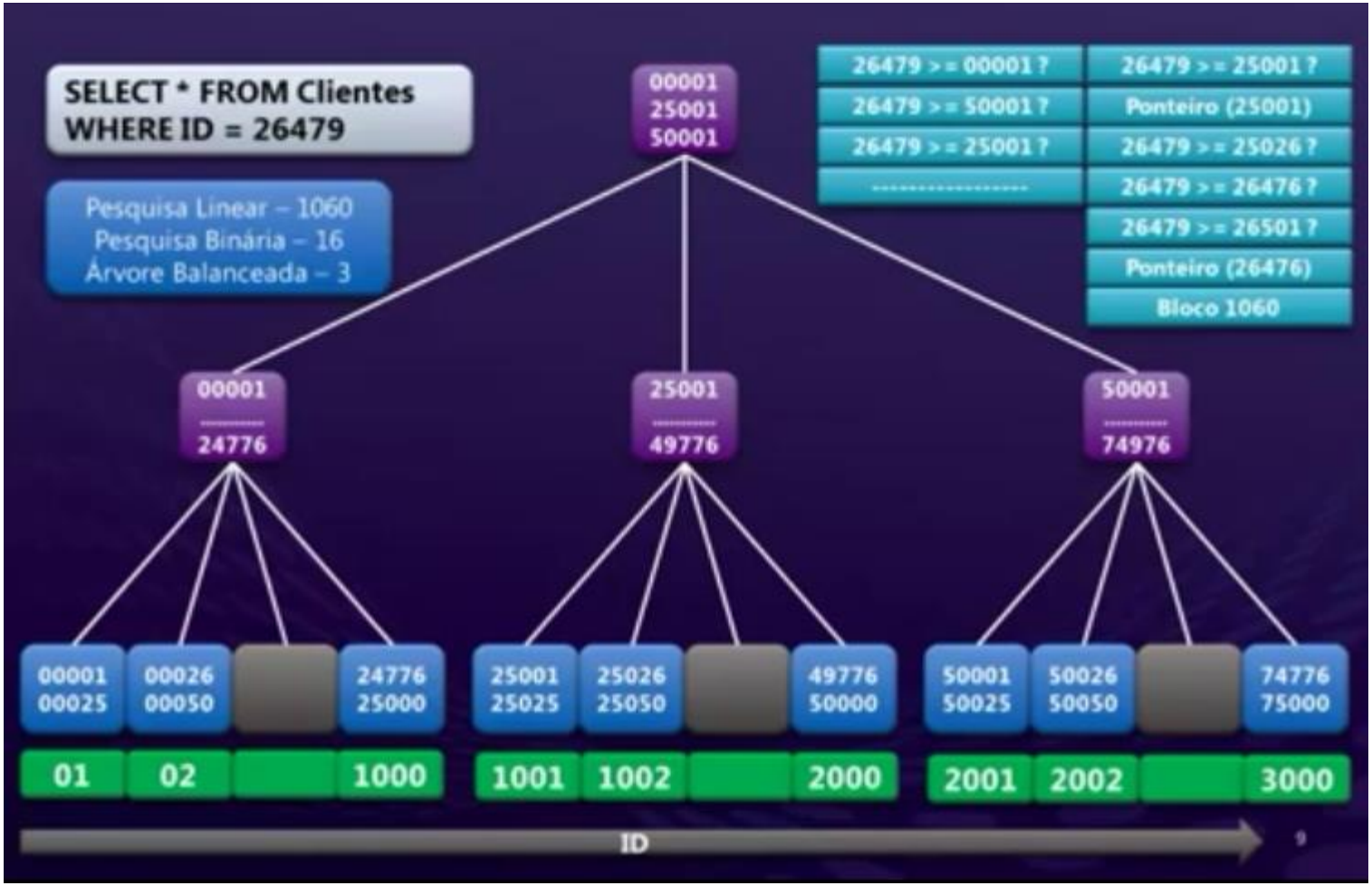
Teste de mesa 1 do algoritmo - BTREE



Resumo: BTREE foi 14 vezes mais rápida que a pesquisa linear(sequencial) e 5 vezes mais rápida que a pesquisa binária!

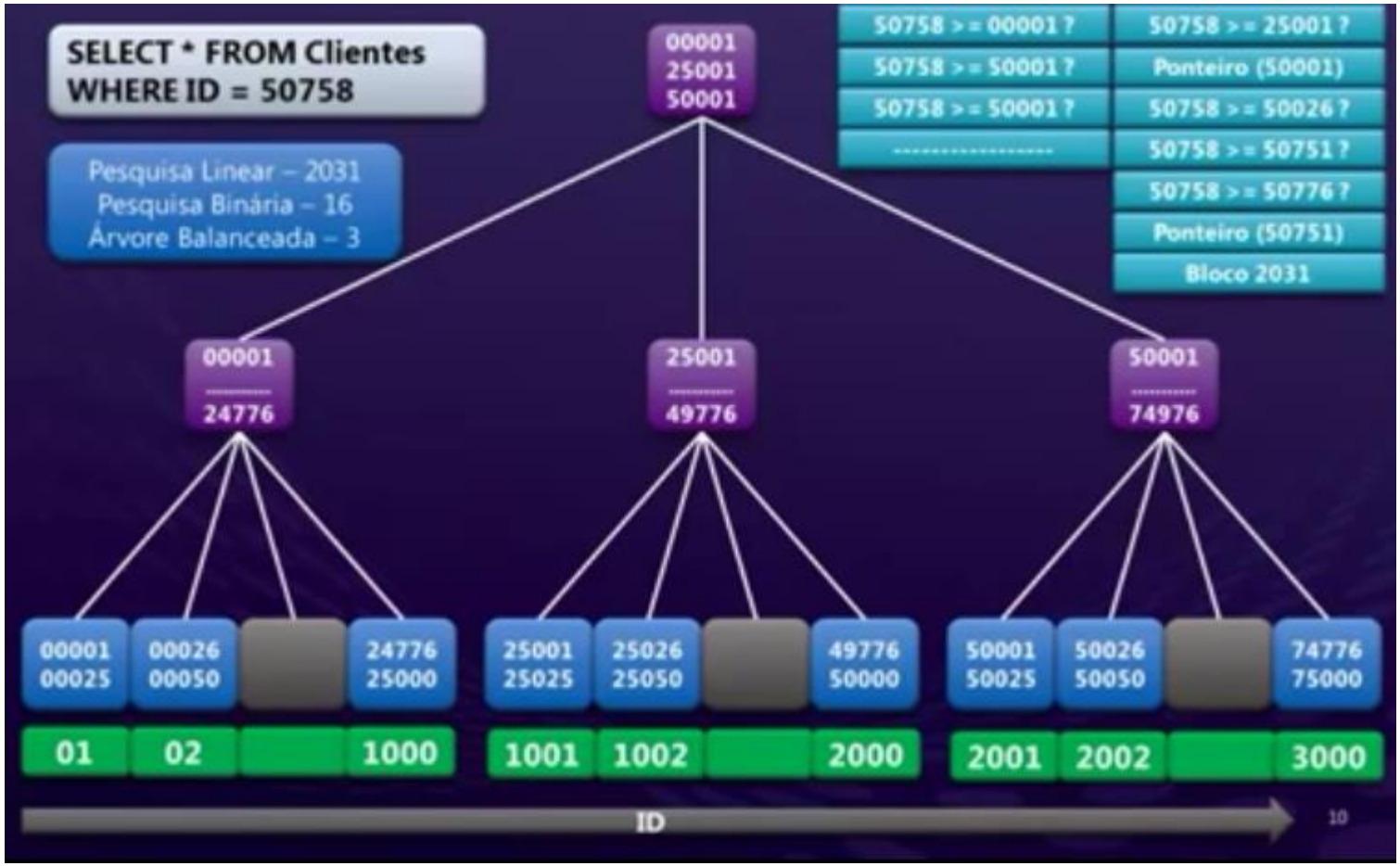
INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA

Teste de mesa 2 do algoritmo - BTREE



Resumo: BTREE foi 353 vezes mais rápida que a pesquisa linear(sequencial) e 5 vezes mais rápida que a pesquisa binária!

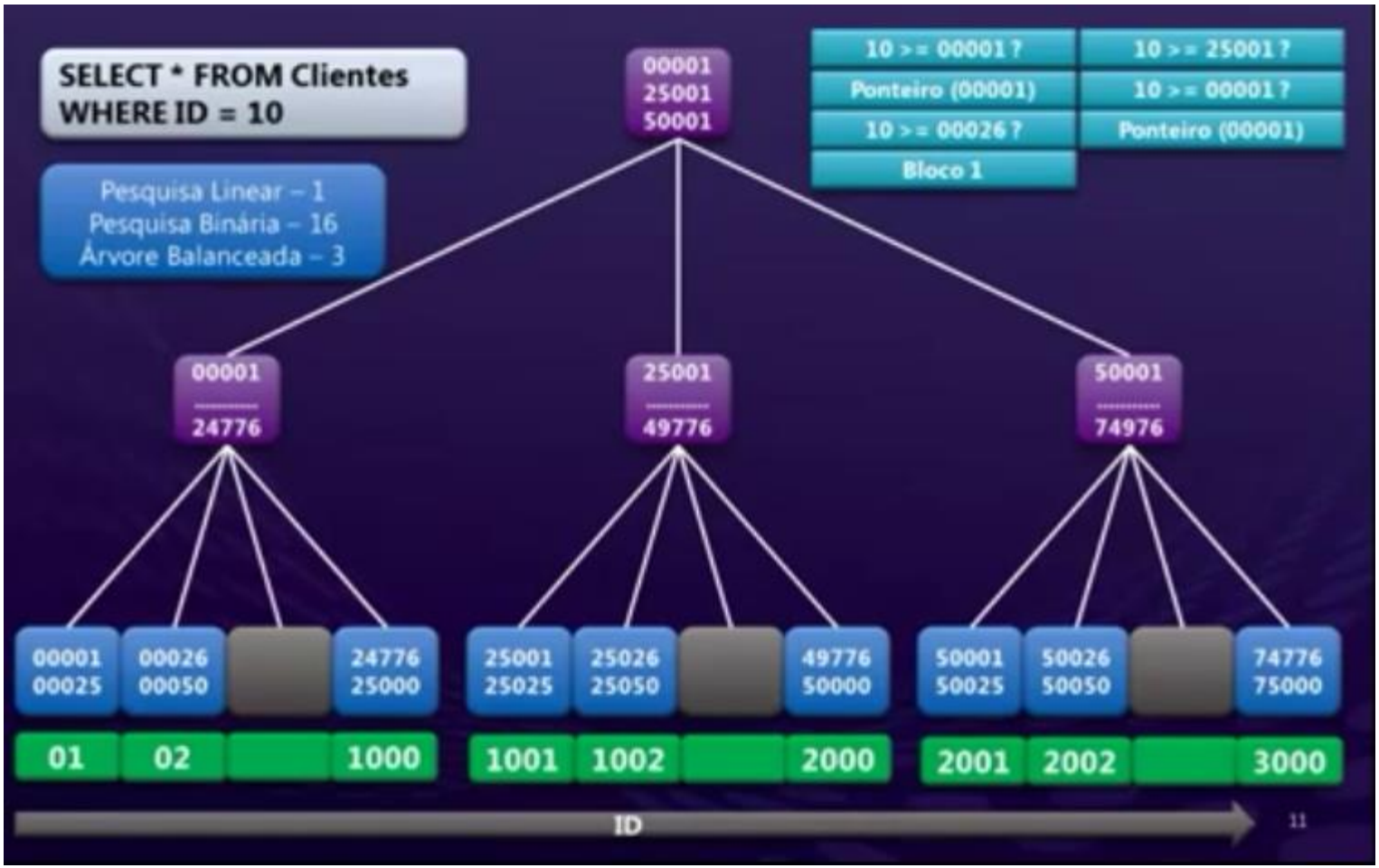
Teste de mesa 3 do algoritmo - BTREE



Resumo: BTREE foi 677 vezes mais rápida que a pesquisa linear(sequencial) e 5 vezes mais rápida que a pesquisa binária!

INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA

Teste de mesa 4 do algoritmo - BTREE



Resumo: Para registros iniciais, a pesquisa sequencial é superior a BTREE e a pesquisa binária!