

# Laboratório de Bases de Dados

Prof. Jose Fernando Rodrigues Jr

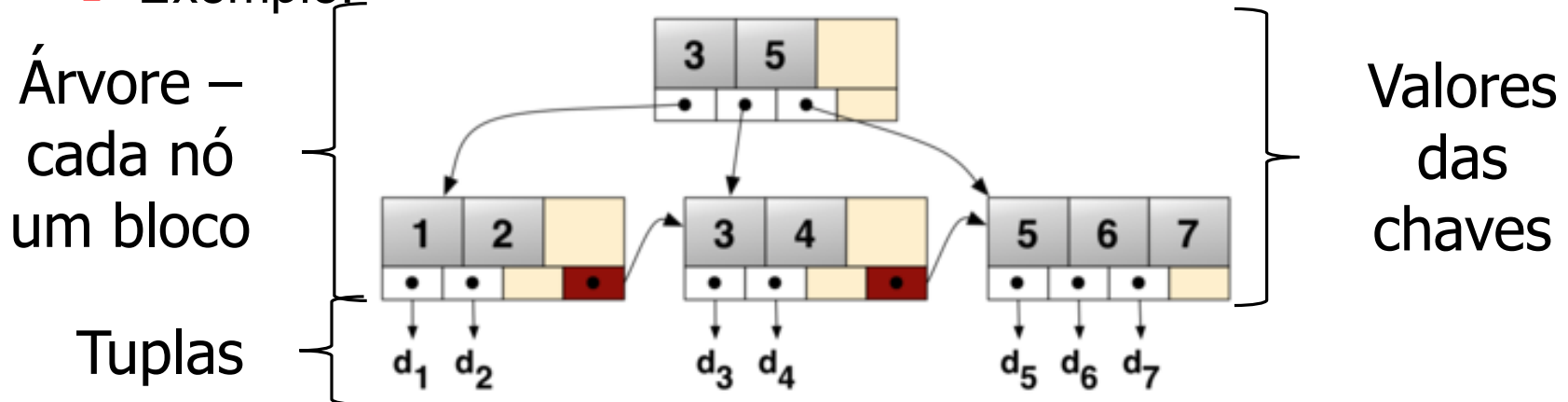
## Aula 9 – Índices



# Criação de Índices – B+Tree

## ■ Árvores B+

- A árvore B é uma **estrutura de indexação de dados** organizada de maneira hierárquica, permitindo **inserções, buscas, e remoções em tempo logarítmico**
- É a **estrutura mais usada em bancos de dados**, configuração na qual **cada nó da árvore é um bloco de disco**
- Exemplo:



# Criação de Índices – B+Tree

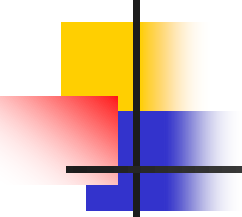
## A árvore B+:

- Mantém as chaves ordenadas permitindo varredura sequencial
- Mantém os nós (blocos) minimamente preenchidos para desempenho em inserção e remoção – evita rebalanceamento constante
- Mantém o índice balanceado por meio de um algoritmo recursivo

# Criação de Índices – B+Tree

- *O poder das árvores B (index seek)*
  - *Suponha blocos de disco de 4096 B*
  - *Chaves de 4 B e ponteiros de 8 B*
    - *Portanto:  $4096/(4+8) \sim 340$  entradas por bloco (nó da árvore)*
  - *Supondo-se uma ocupação média de 256 chaves por nó e 3 níveis*
    - *Poderiam ser indexados:  $256*256*256 \sim 16$  milhões de tuplas*
    - *Para 4 níveis, teríamos 4+ bilhões de tuplas indexadas*
  - *Isto é, qualquer tupla pode ser encontrada em apenas 4 acessos a disco (3 se a raiz for mantida em memória)*
- *Lembrando: árvores B+ podem aceitar repetição de valores, os quais serão alocados sequencialmente na varredura da árvore*

# Seletividade

- 
- *No entanto, árvores  $B$  nem sempre são recomendadas;*
  - *É necessário avaliar a seletividade do atributo*

$$\text{Seletividade} = 1 - \frac{\text{registros-na-resposta-filtrada-pelo-atributo}}{\text{total-de-registros}}$$

Exemplos assumindo distribuição uniforme:

- atributos chaves têm seletividade ~1 sempre
- um atributo de gênero (masc/fem) tem seletividade ~0,5 (pior caso)
- um atributo categórico usual (sim/não/talvez) tem seletividade ~0,777

Então, a árvore B é boa pra tudo?

R.: não, ela é útil para:

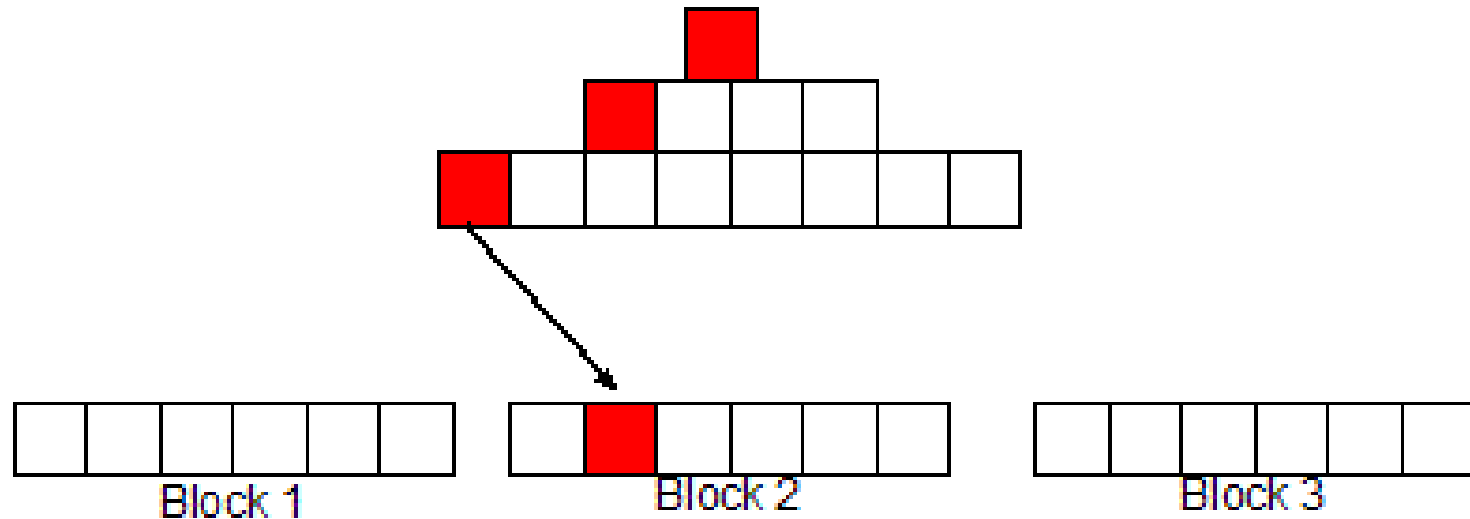
- alta seletividade, isto é, um valor de busca retorna um número pequeno de tuplas, por exemplo o CPF, relativamente ao tamanho da relação, muitos autores falam em torno de  $\sim 200$  valores distintos por valor;
- buscas envolvendo igualdade;
- envolvendo faixas de valores (range) – as árvores permitem varrer suas entradas em ordem, e a variação B+Tree permite recuperar uma faixa de valores com uma única busca, evitando ter que ordenar a tabela; por exemplo, responder consultas sobre o atributo idade, mas sem envolver todos os atributos da relação (pois seria necessário buscar tupla por tupla).

Mas isso é bom para tudo?

- ➔ Para valores com baixa seletividade (poucos valores distintos), é melhor percorrer toda a relação (full table scan);
- ➔ Para casos onde busca aleatória pode ocorrer (como em junções), é melhor usar estruturas hash (hash join), se possível.

# Limitação da B+Tree

Limitações do uso de índices B-Tree.







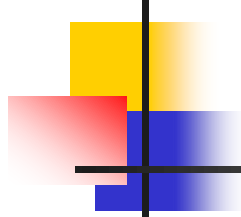
# Criação de Índices

---

- *Trade-off* em indexação

- em consultas: índices apropriados podem tornar a execução mais rápida
- em atualizações
  - índices podem acelerar a componente de consulta das atualizações, mas...
  - exigem atualizações também nos índices, o que consome tempo

# Criação de Índices – B+Tree



- A criação de um índice B-Tree em Oracle é feita da seguinte maneira:

```
CREATE INDEX index_name_ix  
ON table_name(attribute_name);
```

**Exemplo:** CREATE INDEX salario\_ix  
ON funcionarios(salario);

- Existem também índices baseados em funções (function-based):

**Exemplo:** CREATE INDEX salario\_bonus\_ix  
ON funcionarios(salario \* percent\_bonus);

- O que permite consultas indexadas como

```
SELECT * FROM funcionarios where salario * percent_bonus > 1000
```

# Criação de Índices – B+Tree

Para atributos chave (primary key e unique), a criação de índices é feita automaticamente sempre, pois chaves possuem sempre a seletividade máxima (uma tupla por valor indexado)

■ A

ma

eguinte

Exem

■ Ex

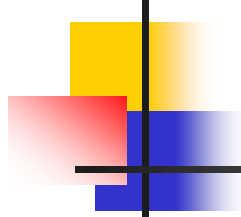
Ex

d):

■ O que permite consultas indexadas como

```
SELECT * FROM funcionarios where salario * percent_bonus > 1000
```

# Criação de Índices – Bitmap



- Mas, e se eu não tenho a seletividade adequada, isto é, possuo menos de ~200 elementos distintos no atributo de indexação?
- Resposta: Bitmap index

# Criação de Índices – Bitmap

Mas, e se eu não tenho a seletividade adequada, isto é, possuo menos de ~200 elementos distintos no atributo de indexação?

Resposta: Bitmap index

Tabela

Rowid	Nfunc	Nome	Gênero	Cor
1	10	Joao	Masculino	Pardo
2	30	Jose	Masculino	Branco
3	40	Genésio	Masculino	Amarelo
4	60	Gertrudes	Feminino	Negro
5	90	Genival	Masculino	Negro
6	100	Germina	Feminino	Branco

Índice-gênero

Valor\RowID	1	2	3	4	5	6
Masculino	1	1	1	0	1	0
Feminino	0	0	0	1	0	1

Índice-cor

Valor\Rowid	1	2	3	4	5	6
Pardo	1	0	0	0	0	0
Branco	0	1	0	0	0	1
Amarelo	0	0	1	0	0	0
Negro	0	0	0	1	1	0

# Criação de Índices – Bitmap

- E como funciona?

```
SELECT * FROM tabela WHERE genero = 'feminino'
```

# Criação de Índices – Bitmap

- E como funciona?

SELECT \* FROM tabela WHERE genero = 'feminino'

Índice-genero

Valor\RowID	1	2	3	4	5	6
Masculino	1	1	1	0	1	0
Feminino	0	0	0	1	0	1

 {4,6}

# Criação de Índices – Bitmap

- E como funciona?

SELECT \* FROM tabela WHERE genero = 'feminino'

Índice-genero

Valor\RowID	1	2	3	4	5	6
Masculino	1	1	1	0	1	0
Feminino	0	0	0	1	0	1

 {4,6}

SELECT \* FROM tabela WHERE cor = 'negro'



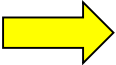
# Criação de Índices – Bitmap

- E como funciona?

SELECT \* FROM tabela WHERE genero = 'feminino'

Índice-genero

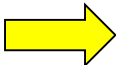
Valor\RowID	1	2	3	4	5	6
Masculino	1	1	1	0	1	0
Feminino	0	0	0	1	0	1

 {4,6}

SELECT \* FROM tabela WHERE cor = 'negro'

Índice-cor

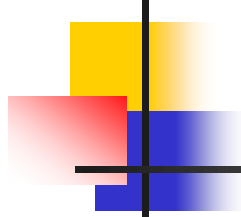
Valor\Rowid	1	2	3	4	5	6
Pardo	1	0	0	0	0	0
Branco	0	1	0	0	0	1
Amarelo	0	0	1	0	0	0
Negro	0	0	0	1	1	0

 {4,5}

# Criação de Índices – Bitmap

- Apesar das consultas serem resolvidas mais rapidamente, assim como na árvore B, será necessário ir até as tuplas para recuperar os dados → acesso aleatório
  - De fato as grandes vantagens do índice bitmap ocorrem quando há vários atributos indexados, como no exemplo
  - `SELECT * FROM tabela WHERE genero = 'feminino' AND cor = 'negro' → {4, 6} ∩ {4, 5} = {4}`
  - `SELECT * FROM tabela WHERE genero = 'feminino' OR cor = 'negro' → {4, 6} ∪ {4, 5} = {4,5,6}`
- Não há necessidade de se ir até a tupla para resolver o predicado, apenas para buscar as tuplas resposta
- Quanto maior a quantidade de atributos “bitmapped” envolvidos em uma consulta, maior é o ganho.

# Criação de Índices – Bitmap



- A criação de um índice Bitmap em Oracle é feita da seguinte maneira:

```
CREATE BITMAP INDEX index_name_ix  
ON table_name(attribute_name);
```

**Exemplo:** `CREATE BITMAP INDEX genero_bix  
ON funcionarios(genero);`



# PRÁTICA 9

---