

ACH2025

Laboratório de Bases de Dados

Aula 8

Indexação e Hashing – Parte 1

Professora:

➤ Fátima L. S. Nunes



Conceitos básicos

- ✓ Boa parte das consultas a BD referem-se a apenas uma parte pequena dos registros.
 - Exemplo:
 - mostrar os nomes de todos os clientes da cidade de Boituva
- ✓ O sistema se torna ineficiente se tiver que ler cada registro e verificar se atende a uma condição.
- ✓ O que precisamos?
 - Capacidade de localizar os registros desejados diretamente.
- ✓ Para atender a esta necessidade: estruturas adicionais associadas aos arquivos.



Conceitos básicos

- ✓ **Índice**: analogia com um catálogo para um livro em uma biblioteca
 - procura por autor: catálogo de autores
 - procura por título: catálogo de títulos
 - catálogo físico segue uma determinada ordem (alfabética, por exemplo)
- ✓ Em Banco de Dados ► dois tipos básicos de índices:
 - **Índices ordenados** – com base na ordem dos valores
 - **Índices *hash*** – com base na distribuição uniforme dos valores por meio de uma faixa de *buckets* (baldes).
 - O *bucket* ao qual um valor é atribuído é determinado por uma função, denominada *função hash*.



Conceitos básicos

- ✓ Há diversas técnicas para os dois tipos de índices
- ✓ Qual é a melhor?
 - Depende das aplicações específicas do BD
 - Fatores a serem avaliados???



Conceitos básicos

- ✓ Há diversas técnicas para os dois tipos de índices
- ✓ Qual é a melhor?
 - Depende das aplicações específicas do BD
 - Fatores a serem avaliados:
 - **Tipos de acesso:** encontrar registros com atributo que possui um certo valor; encontrar registros com atributos dentro de uma faixa de valores etc;
 - **Tempo de acesso:** tempo gasto para encontrar um item em particular ou um conjunto de itens;
 - **Tempo de inserção:** tempo gasto para incluir um novo item (tempo para encontrar o lugar correto e atualizar a estrutura de índice);
 - **Tempo de exclusão:** tempo gasto para excluir um item (tempo para encontrar o item a ser excluído e atualizar a estrutura de índice);
 - **Sobrecarga de espaço:** espaço adicional ocupado por uma estrutura de índice (analisar custo e benefício para se manter um índice).



Conceitos básicos

- ✓ Em geral, é interessante ter mais de um índice para um arquivo.
- ✓ Atributo (ou conjunto de atributos) usado para procurar registros em um arquivo:
 - **chave de procura** (*não confundir com superchave, chave candidata, chave primária*)
- ✓ Vários índices → **várias chaves de procura**



Conceitos básicos

- ✓ Em Banco de Dados ► dois tipos básicos de índices:
 - Índices ordenados
 - Índices *hash*



Conceitos básicos

✓ Em Banco de Dados ► dois tipos básicos de índices:

- **Índices ordenados**

- Índice primário
- Índice secundário



Índices ordenados

- ✓ Cada estrutura de índice → associada a uma chave de procura em particular
- ✓ Índice:
 - armazena os valores das chaves de procura de forma ordenada
 - a cada chave de procura associa os registros que a contêm.
- ✓ Próprio arquivo pode armazenar os registros seguindo uma determinada ordem.
- ✓ Arquivo pode ter diversos índices com diferentes chaves de procura.



Índices ordenados

✓ Índice primário:

- índice cuja chave de procura especifica a ordem sequencial (física) do arquivo.
- a chave de procura em geral é a chave primária, mas **nem sempre** isso ocorre – é errôneo utilizar o termo índice primário para designar um índice de chave primária;
- também chamado de **índice agrupado** ou **índice de agrupamento**.



Índices ordenados

✓ Índices secundários:

- chaves de procura especificam uma ordem diferente da ordem sequencial (física) do arquivo
- também chamados de índices não agrupados ou índices não de agrupamento.



Índices ordenados → Índice primário

- ✓ Consideramos que todos os arquivos são ordenados sequencialmente por alguma chave de procura → *arquivos sequenciais indexados*.
- ✓ Um dos mais antigos esquemas de índices usados em sistemas de BD.
- ✓ Projetados para aplicações que requerem tanto processamento sequencial quanto acesso aleatório a registros individuais.



Índices ordenados → Índice primário

- ✓ Exemplo: registros armazenados na ordem da chave de procura (chave de procura = atributo *cidade*)

Cidade	Nome	Endereço
Arapongas	Júlia	Rua dos Anjos, 123
Brasília	Ana	Av. Antonio, 865
Brasília	Heitor	Av. Antonio, 865
Maringá	Mariana	Rua Estreita, 89
Petrópolis	Carlos	Alameda das Rosas, 634
Petrópolis	Maria	Rua São Paulo, 432
Petrópolis	Luiza	Avenida Dom Pedro, 800
Rio Claro	Bruno	Rua Aparecida, 7600
Rondonópolis	Márcia	Rua Santa Rita, 632



Índices ordenados → Índice primário → Índices densos e esparsos

- ✓ registro de índice ou entrada de índice → valor da chave de busca e ponteiros para um ou mais registros com o mesmo valor para chave de busca.

Cidade	Nome	Endereço
Arapongas	Júlia	Rua dos Anjos, 123
Brasília	Ana	Av. Antonio, 865
Brasília	Heitor	Av. Antonio, 865
Maringá	Mariana	Rua Estreita, 89
Petrópolis	Carlos	Alameda das Rosas, 634
Petrópolis	Maria	Rua São Paulo, 432
Petrópolis	Luiza	Avenida Dom Pedro, 800
Rio Claro	Bruno	Rua Aparecida, 7600
Rondonópolis	Márcia	Rua Santa Rita, 632

Diagram illustrating a primary index structure. The index table (highlighted in green) maps the search key 'Arapongas' to the first record (Júlia, Rua dos Anjos, 123). The main data table contains records for various cities. Arrows indicate the mapping from the index entry to the corresponding record in the data table.

entrada de índice

Arapongas

Índices ordenados → Índice primário → Índices densos e esparsos

- ✓ registro de índice ou entrada de índice → valor da chave de busca e ponteiros para um ou mais registros com o mesmo valor para chave de busca.

Cidade	Nome	Endereço
Arapongas	Júlia	Rua dos Anjos, 123
Brasília	Ana	Av. Antonio, 865
Brasília	Heitor	Av. Antonio, 865
Maringá	Mariana	Rua Estreita, 89
Petrópolis	Carlos	Alameda das Rosas, 634
Petrópolis	Maria	Rua São Paulo, 432
Petrópolis	Luiza	Avenida Dom Pedro, 800
Rio Claro	Bruno	Rua Aparecida, 7600
Rondonópolis	Márcia	Rua Santa Rita, 632

Diagram illustrating a primary index structure. The index table (highlighted in pink) maps the search key 'Brasília' to two records in the main data table. The main data table has columns: Cidade, Nome, and Endereço. The index table has columns: Cidade, Nome, and Endereço. The index table shows two entries for 'Brasília', both pointing to the same record in the main data table (Ana, Av. Antonio, 865). The label 'entrada de índice' points to the index table, and 'Brasília' points to the search key.

Índices ordenados → Índice primário → Índices densos e esparsos

- ✓ registro de índice ou entrada de índice → valor da chave de busca e ponteiros para um ou mais registros com o mesmo valor para chave de busca.

entrada de índice

Brasília

Cidade	Nome	Endereço
Arapongas	Júlia	Rua dos Anjos, 123
Brasília	Ana	Av. Antonio, 865
Brasília	Heitor	Av. Antonio, 865
Maringá	Mariana	Rua Estreita, 89
Petrópolis	Carlos	Alameda das Rosas, 634
Petrópolis	Maria	Rua São Paulo, 432
Petrópolis	Luiza	Avenida Dom Pedro, 800
Rio Claro	Bruno	Rua Aparecida, 7600
Rondonópolis	Márcia	Rua Santa Rita, 632

Mas o que é este ponteiro???

Índices ordenados → Índice primário → Índices densos e esparsos

- ✓ registro de índice ou entrada de índice → valor da chave de busca e ponteiros para um ou mais registros com o mesmo valor para chave de busca.

Cidade	Nome	Endereço
Arapongas	Júlia	Rua dos Anjos, 123
Brasília	Ana	Av. Antonio, 865
Brasília	Heitor	Av. Antonio, 865
Maringá	Mariana	Rua Estreita, 89
Petrópolis	Carlos	Alameda das Rosas, 634
Petrópolis	Maria	Rua São Paulo, 432
Petrópolis	Luiza	Avenida Dom Pedro, 800
Rio Claro	Bruno	Rua Aparecida, 7600
Rondonópolis	Márcia	Rua Santa Rita, 632

Mas o que é este ponteiro???

ponteiro → identificador de um bloco do disco e um deslocamento dentro do bloco para encontrar o registro



Conceitos básicos

✓ Dois tipos básicos de índices: denso e esparso

– Índices ordenados

- Índice primário
 - denso
 - esparso
- Índice secundário



Índices ordenados → Índice primário → Índices densos e esparsos

- ✓ Dois tipos de índices ordenados: denso e esperso
- ✓ Índice denso
 - um registro de índice (ou entrada de índice) para cada valor de chave de procura.
 - registro contém chave e ponteiro para o primeiro registro com a chave de busca.
 - demais registros com mesma chave são armazenados sequencialmente após o primeiro registro.

	Cidade	Nome	Endereço
Arapongas	Arapongas	Júlia	Rua dos Anjos, 123
Brasília	Brasília	Ana	Av. Antonio, 865
	Brasília	Heitor	Av. Antonio, 865
Maringá	Maringá	Mariana	Rua Estreita, 89
Petrópolis	Petrópolis	Carlos	Alameda das Rosas, 634
	Petrópolis	Maria	Rua São Paulo, 432
Rio Claro	Petrópolis	Luiza	Avenida Dom Pedro, 800
Rondonópolis	Rio Claro	Bruno	Rua Aparecida, 7600
	Rondonópolis	Márcia	Rua Santa Rita, 632



Índices ordenados → Índice primário → Índices densos e esparsos

✓ Índice esperso

- um registro de índice é criado apenas para alguns dos valores;
- cada registro de índice contém um valor de chave de procura e um ponteiro para o primeiro registro de dados com esse valor de chave de procura;
- para localizar um registro:
 - encontra-se entrada de índice com maior valor de chave de procura que seja menor ou igual ao valor de chave de procura desejado;
 - inicia-se no registro apontado pela entrada de índice e segue-se os ponteiros até encontrar o registro desejado.



Índices ordenados → Índice primário → Índices densos e esparsos

✓ Exemplo: Procura pela cidade de Petrópolis

Cidade	Nome	Endereço
Arapongas	Júlia	Rua dos Anjos, 123
Brasília	Ana	Av. Antonio, 865
Brasília	Heitor	Av. Antonio, 865
Maringá	Mariana	Rua Estreita, 89
Petrópolis	Carlos	Alameda das Rosas, 634
Petrópolis	Maria	Rua São Paulo, 432
Petrópolis	Luiza	Avenida Dom Pedro, 800
Rio Claro	Bruno	Rua Aparecida, 7600
Rondonópolis	Márcia	Rua Santa Rita, 632



Índices ordenados → Índice primário → Índices densos e esparsos

- ✓ Vantagem índice denso?
- ✓ Vantagens índice esperso?



Índices ordenados → Índice primário → Índices densos e esparsos

✓ Vantagem índice denso?

- mais rápido

✓ Vantagens índice esperso?

- menos espaço
- menos sobrecarga de manutenção para inserções e exclusões



Índices ordenados → Índice primário → Índices densos e esparsos

✓ Vantagem índice denso:

- mais rápido

✓ Vantagens índice esparso:

- menos espaço
- menos sobrecarga de manutenção para inserções e exclusões

✓ Projetista deve escolher entre tempo de acesso e sobrecarga de espaço

✓ Boa decisão: índice esparso com uma entrada de índice por bloco. Por quê????



Índices ordenados → Índice primário → Índices densos e esparsos

✓ Vantagem índice denso:

- mais rápido

✓ Vantagens índice esperso:

- menos espaço
- menos sobrecarga de manutenção para inserções e exclusões

✓ Projetista deve escolher entre tempo de acesso e sobrecarga de espaço

✓ Boa decisão: índice esperso com uma entrada de índice por bloco. **Por quê???**

- custo predominante de processamento de uma solicitação: **carregamento do bloco do disco** para a memória principal;
- se o bloco está na memória principal, **tempo para percorrer bloco é desprezível.**



Índices ordenados → Índice primário → Índices densos e esparsos

- ✓ Boa decisão: índice esperso com uma entrada de índice por bloco. **Por quê????**
 - custo predominante de processamento de uma solicitação: carregamento do bloco do disco para a memória principal;
 - se o bloco está na memória principal, tempo para percorrer bloco é desprezível.
- ✓ Qual o problema???



Índices ordenados → Índice primário → Índices densos e esparsos

- ✓ Boa decisão: índice esperso com uma entrada de índice por bloco. **Por quê????**
 - custo predominante de processamento de uma solicitação: carregamento do bloco do disco para a memória principal;
 - se o bloco está na memória principal, tempo para percorrer bloco é desprezível.
- ✓ **Qual o problema???**
 - ✓ registros para um valor de chave podem ocupar vários blocos



Índices ordenados → Índice primário → Índices multiníveis

- ✓ Problema: índices grandes (mesmo se esparsos)
- ✓ Exemplo de cálculo:
 - 100 mil registros, com 10 registros armazenados em cada bloco
 - se tiver 1 registro de índice por bloco → índice com 10 mil registros
 - Supondo que 100 registros de índice caibam em um bloco → índice terá 100 blocos (armazenados como arquivos sequenciais em disco).
 - Se **índice pequeno** e mantido em memória principal → tempo de acesso pequeno.
 - E se o índice for grande e mantido em disco??? O que isso envolve???



Índices ordenados → Índice primário → Índices multiníveis

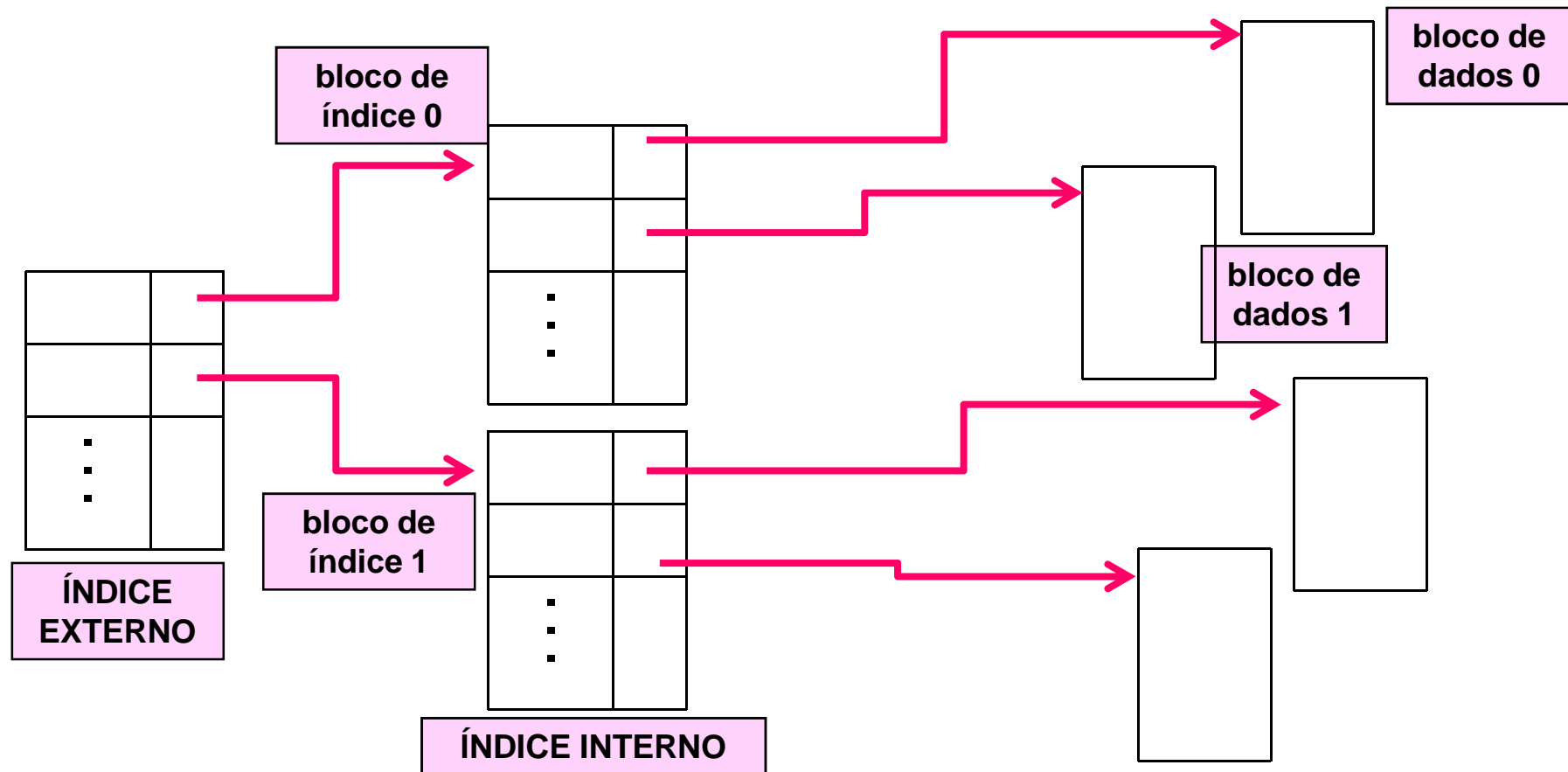
- ✓ Problema: índices grandes (mesmo se esparsos)
- ✓ Exemplo de cálculo:
 - Se **índice grande** e mantido em disco → busca exige várias leituras de bloco → mesmo com busca binária teríamos custo alto (100 blocos → $\log_2 100 =$ leitura de até 7 blocos)
 - Em um disco que leitura de bloco leva 30 milissegundos → 7 blocos gastariam 210 milissegundos

✓ Como resolver ???



Índices ordenados → Índice primário → Índices multiníveis

- ✓ Problema: índices grandes (mesmo se esparsos)
- ✓ Para minimizar este problema:
 - Trata-se o índice como arquivo sequencial
 - Constrói-se índice esparsos sobre o índice primário



Índices ordenados → Índice primário → Índices multiníveis

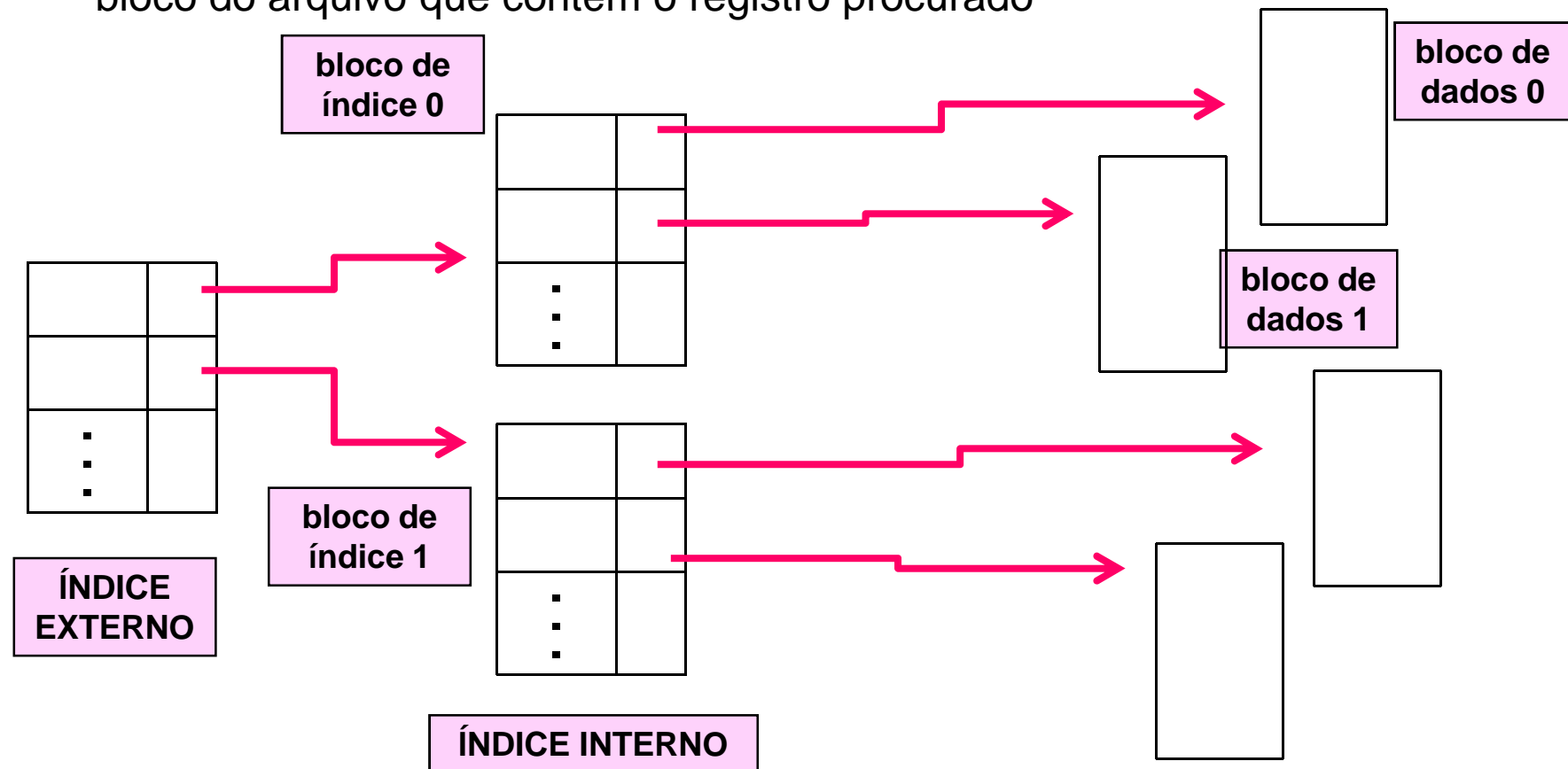
- ✓ Como localizar um registro?



Índices ordenados → Índice primário → Índices multiníveis

✓ Como localizar um registro?

- **Busca binária no índice mais externo** – procura o registro com maior valor na chave de procura que seja menor ou igual ao valor desejado → ponteiro indica bloco do índice interno
- Varre-se o bloco até encontrar o registro desejado – novamente maior valor na chave de procura que é menor ou igual ao valor desejado → ponteiro indica o bloco do arquivo que contém o registro procurado



Índices ordenados → Índice primário → Índices multiníveis

- ✓ Como localizar um registro?
 - Busca binária no índice mais externo – procura-se o registro com maior valor na chave de procura que seja menor ou igual ao valor desejado → ponteiro indica bloco do índice interno
 - Varre-se o bloco até encontrar o registro desejado – novamente maior valor na chave de procura que é menor ou igual ao valor desejado → ponteiro indica o bloco do arquivo que contém o registro procurado
- ✓ Usando dois níveis de indexação e considerando índice externo na memória, lê-se apenas 1 bloco de índice em vez dos 7 necessários no cálculo anterior



Índices ordenados → Índice primário → Índices multiníveis

- ✓ Como localizar um registro?
 - Busca binária no índice mais externo – procura-se o registro com maior valor na chave de procura que seja menor ou igual ao valor desejado → ponteiro indica bloco do índice interno
 - Varre-se o bloco até encontrar o registro desejado – novamente maior valor na chave de procura que é menor ou igual ao valor desejado → ponteiro indica o bloco do arquivo que contém o registro procurado
- ✓ Usando dois níveis de indexação e considerando índice externo na memória, lê-se apenas 1 bloco de índice em vez dos 7 necessários no cálculo anterior
- ✓ Quando o índice externo crescer muito e não caber mais na memória principal → ??????



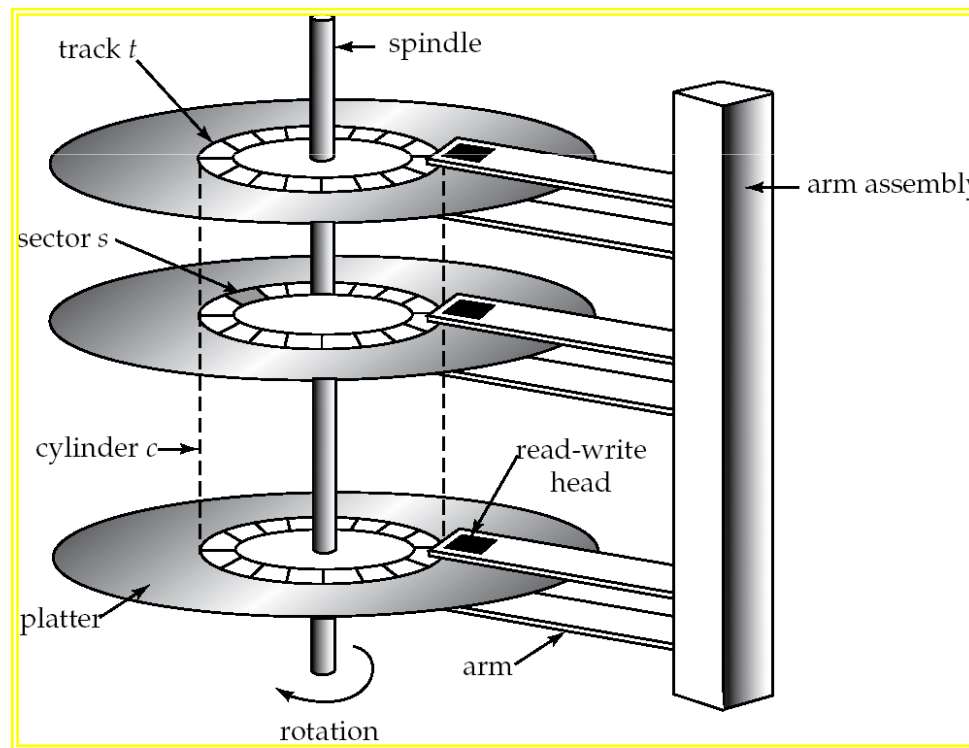
Índices ordenados → Índice primário → Índices multiníveis

- ✓ Como localizar um registro?
 - Busca binária no índice mais externo – procura-se o registro com maior valor na chave de procura que seja menor ou igual ao valor desejado → ponteiro indica bloco do índice interno
 - Varre-se o bloco até encontrar o registro desejado – novamente maior valor na chave de procura que é menor ou igual ao valor desejado → ponteiro indica o bloco do arquivo que contém o registro procurado
- ✓ Usando dois níveis de indexação e considerando índice externo na memória, lê-se apenas 1 bloco de índice em vez dos 7 necessários no cálculo anterior
- ✓ Quando o índice externo crescer muito e não caber mais na memória principal → **cria-se novo nível**



Índices ordenados → Índice primário → Índices multiníveis

- ✓ Pode-se criar quantos níveis forem necessários → índices de níveis múltiplos
- ✓ Cada nível pode corresponder a uma unidade de armazenamento físico → nível de trilha, cilindro e disco.



Índices ordenados → Índice primário → Atualização de índices

✓ Inclusão - índices densos

- Faz-se busca usando o valor da chave de procura

Cidade	Nome	Endereço
Arapongas	Júlia	Rua dos Anjos, 123
Brasília	Ana	Av. Antonio, 865
Brasília	Heitor	Av. Antonio, 865
Maringá	Mariana	Rua Estreita, 89
Petrópolis	Carlos	Alameda das Rosas, 634
Petrópolis	Maria	Rua São Paulo, 432
Petrópolis	Luiza	Avenida Dom Pedro, 800
Rio Claro	Bruno	Rua Aparecida, 7600
Rondonópolis	Márcia	Rua Santa Rita, 632

- Exemplo: quero inserir a cidade **Bauru** (não existe no índice)
- Como fica o índice acima? (fazer)

Índices ordenados → Índice primário → Atualização de índices

✓ Inclusão - índices densos

- Faz-se busca usando o valor da chave de procura

	Cidade	Nome	Endereço
Arapongas	Arapongas	Júlia	Rua dos Anjos, 123
Brasília	Brasília	Ana	Av. Antonio, 865
	Brasília	Heitor	Av. Antonio, 865
Maringá	Maringá	Mariana	Rua Estreita, 89
Petrópolis	Petrópolis	Carlos	Alameda das Rosas, 634
	Petrópolis	Maria	Rua São Paulo, 432
Rio Claro	Petrópolis	Luiza	Avenida Dom Pedro, 800
Rondonópolis	Rio Claro	Bruno	Rua Aparecida, 7600
	Rondonópolis	Márcia	Rua Santa Rita, 632

- Exemplo: quero inserir a cidade **Bauru** (não existe no índice)
- **Como fica o índice acima?** (insere ponteiro no índice e registro no arquivo de dados)

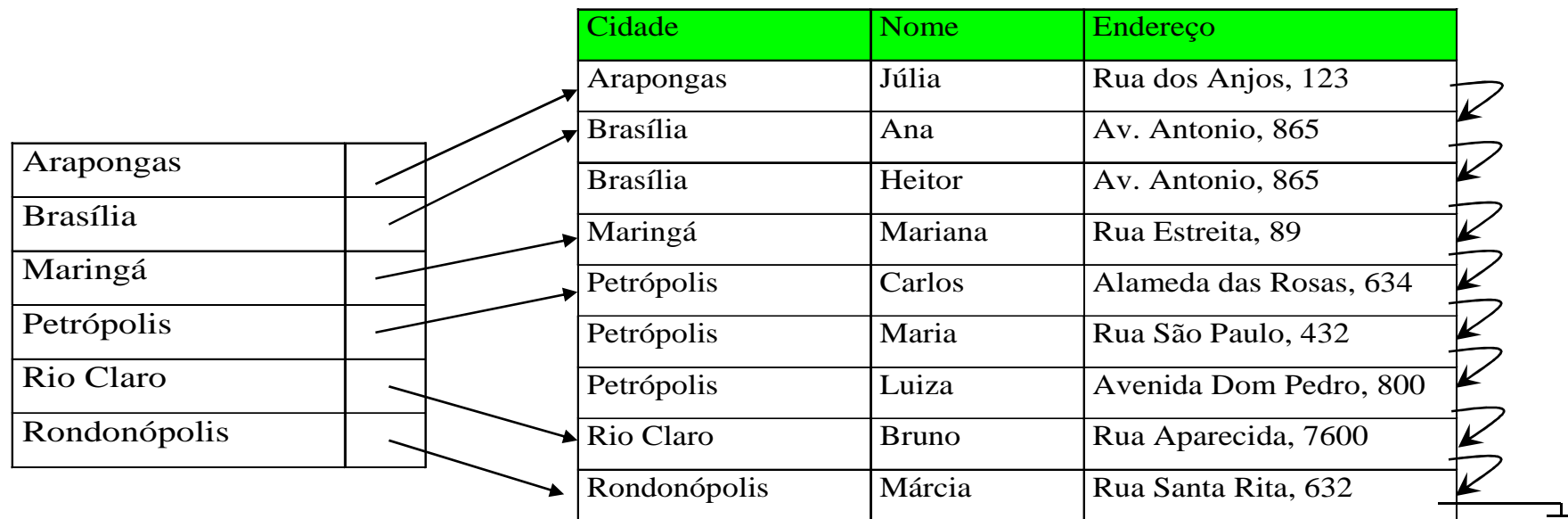


Índices ordenados → Índice primário → Atualização de índices

✓ Inclusão - índices densos

- Faz-se busca usando o valor da chave de procura

Cidade	Nome	Endereço
Arapongas	Júlia	Rua dos Anjos, 123
Brasília	Ana	Av. Antonio, 865
Brasília	Heitor	Av. Antonio, 865
Maringá	Mariana	Rua Estreita, 89
Petrópolis	Carlos	Alameda das Rosas, 634
Petrópolis	Maria	Rua São Paulo, 432
Petrópolis	Luiza	Avenida Dom Pedro, 800
Rio Claro	Bruno	Rua Aparecida, 7600
Rondonópolis	Márcia	Rua Santa Rita, 632



- Exemplo: quero inserir **novo registro para Petrópolis (já existe no índice)**
- **Como fica o índice acima?**
 - índice armazena ponteiros para todos registros com mesma chave de busca
 - índice armazena ponteiro somente para o primeiro registro com a chave de busca



Índices ordenados → Índice primário → Atualização de índices

✓ Inclusão - índices densos

- Faz-se busca usando o valor da chave de procura

	Cidade	Nome	Endereço
Arapongas	Arapongas	Júlia	Rua dos Anjos, 123
Brasília	Brasília	Ana	Av. Antonio, 865
	Brasília	Heitor	Av. Antonio, 865
Maringá	Maringá	Mariana	Rua Estreita, 89
Petrópolis	Petrópolis	Carlos	Alameda das Rosas, 634
	Petrópolis	Maria	Rua São Paulo, 432
Rio Claro	Petrópolis	Luiza	Avenida Dom Pedro, 800
Rondonópolis	Rio Claro	Bruno	Rua Aparecida, 7600
	Rondonópolis	Márcia	Rua Santa Rita, 632

- Exemplo: quero inserir **novo registro para Petrópolis**
- **Como fica o índice acima?**
 - índice armazena ponteiros para todos registros com mesma chave de busca: **somente acrescenta um ponteiro para o novo registro de dados**
 - índice armazena ponteiro somente para o primeiro registro com a chave de busca: **coloca registro de dados após outros registros com mesmo valor de chave**



Índices ordenados → Índice primário → Atualização de índices

✓ Inclusão – índices esparsos

- considerando que índice armazena uma entrada para cada bloco:

	Cidade	Nome	Endereço
Arapongas	Arapongas	Júlia	Rua dos Anjos, 123
	Brasília	Ana	Av. Antonio, 865
	Brasília	Heitor	Av. Antonio, 865
Maringá	Maringá	Mariana	Rua Estreita, 89
	Petrópolis	Carlos	Alameda das Rosas, 634
	Petrópolis	Maria	Rua São Paulo, 432
	Petrópolis	Luiza	Avenida Dom Pedro, 800
Rio Claro	Rio Claro	Bruno	Rua Aparecida, 7600
	Rondonópolis	Márcia	Rua Santa Rita, 632

- se sistema cria novo bloco → insere primeiro valor de chave no índice
- se novo registro tiver o menor valor de chave em seu bloco → atualiza a entrada de índice apontando para o bloco.
- Exemplo: 1) inserir novo bloco com chave Bauru e Bebedouro



Índices ordenados → Índice primário → Atualização de índices

✓ Remoção

- procura-se o registro a ser removido
- Índices densos
 - se registro é único que contém chave de procura → remove valor da chave de procura do índice
 - se índice armazena ponteiros para todos registros com mesmo valor de chave → só exclui o ponteiro do registro de índice
 - se índice armazena ponteiro somente para o primeiro valor da chave → sistema atualiza registro de índice para apontar para o próximo.
 - Exemplo: excluir a cidade Maringá.



Índices ordenados → Índice primário → Atualização de índices

✓ Remoção

- procura-se o registro a ser removido
- Índices esparsos
 - se índice não tiver registro com valor da chave de busca → nada precisa fazer
 - caso contrário:
 - se registro excluído é único com valor de chave → substitui registro de índice com valor para o próximo valor da chave. Se este já tiver entrada no índice, a entrada é excluída.
 - caso contrário: se registro de índice apontar para o excluído → sistema atualiza registro de índice para apontar para o próximo com mesmo valor de chave.
 - Exemplo: excluir a cidade Maringá.



Índices ordenados → Índice primário → Atualização de índices

✓ Remoção

- procura-se o registro a ser removido
- Índices multiníveis
 - extensão do processo anterior descrito: sistema atualiza os índices, começando pelo nível mais baixo.



Conceitos básicos

✓ Contextualizando

– Índices ordenados

- Índice primário

- denso

- esperso

- » um nível

- » multinível

- Índice secundário

- somente denso

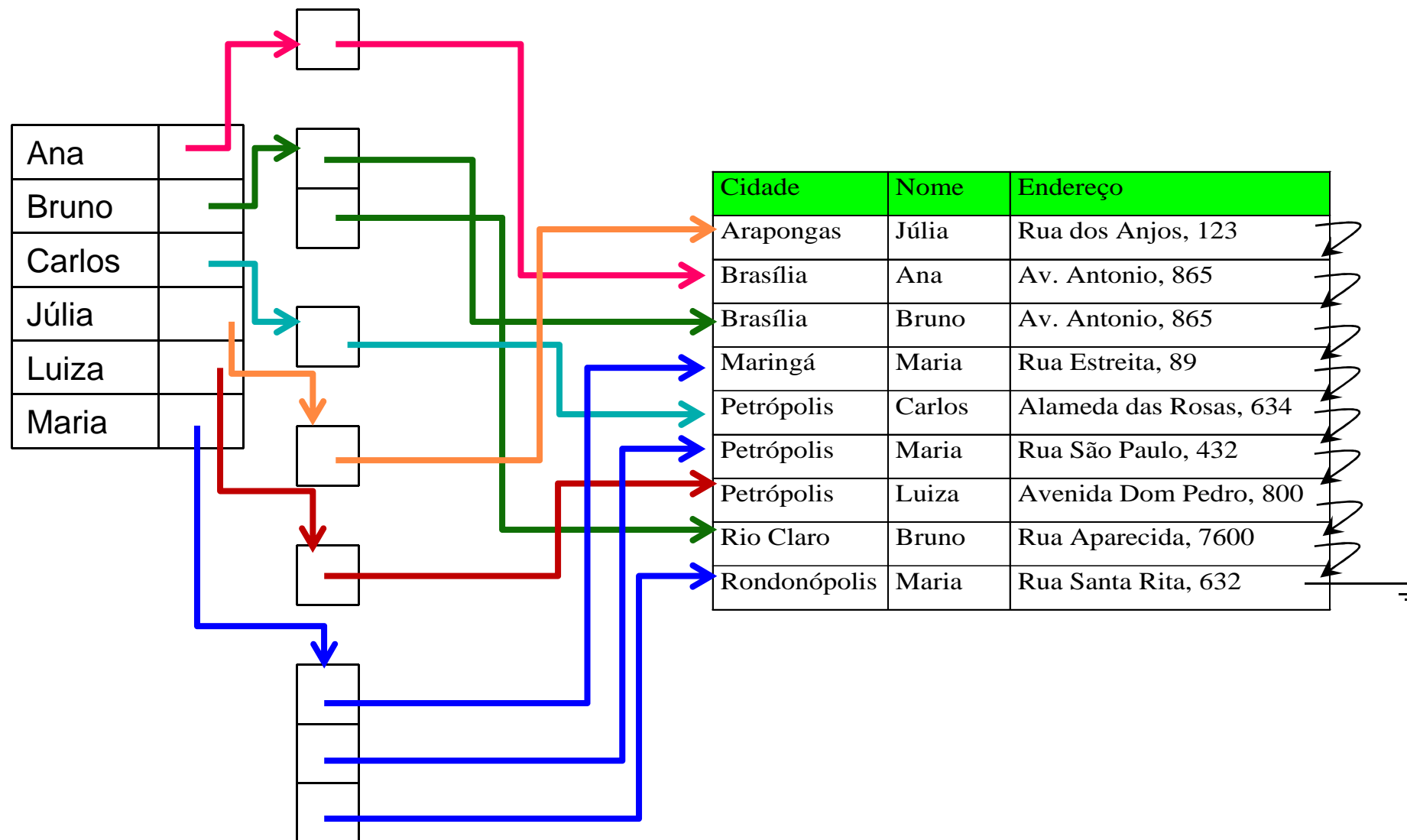


Índices ordenados → Índice secundário

- ✓ Índices de outras chaves candidatas .
- ✓ Semelhante a índice primário denso, mas os registros apontados por valores sucessivos no índice não estão armazenados de forma sequencial.
- ✓ Registros estão ordenados pela chave de procura do índice primário → registros com a mesma chave de procura podem estar em qualquer lugar do arquivo.
- ✓ Então → índice secundário **deve ter ponteiros para todos os registros** → **índices densos**.
- ✓ É possível criar um nível indireto adicional para implementar índices secundários em chaves que não são chaves candidatas → ponteiros não apontam diretamente para o arquivo, mas para um *bucket* que contém ponteiros para o arquivo.



Índices ordenados → Índice secundário



Índices ordenados → Índice secundário

- ✓ Varredura sequencial em índices secundários pode requerer a leitura do arquivo inteiro (devido à ordenação estar na ordem da chave do índice primário)
- ✓ Índice secundário devem ser densos, com uma entrada de índice para cada valor de chave de procura e um ponteiro para cada registro do arquivo
- ✓ **inserção e remoção**
 - sempre que um arquivo é modificado, *cada índice* deve ser atualizado
 - índices secundários melhoram desempenho de busca por chaves de procura diferentes do índice primário, mas exigem sobrecarga significativa na atualização do BD.
 - projetista deve estudar frequência de consulta para decidir quais índices secundários devem efetivamente ser criados.



ACH2025

Laboratório de Bases de Dados

Aula 8

Indexação e Hashing – Parte 1

Professora:

➤ Fátima L. S. Nunes

