Universidade Eduardo Mondlane
Faculdade de Engenharia
Departamento de Engenharia Electrotécnica
Curso de Engenharia Informática

Base de Dados II

Msc Sérgio Mavie
Eng. Cristiliano Maculuve

Agenda

- Optimizacão de Bases de Dados
- Índices

Definicão

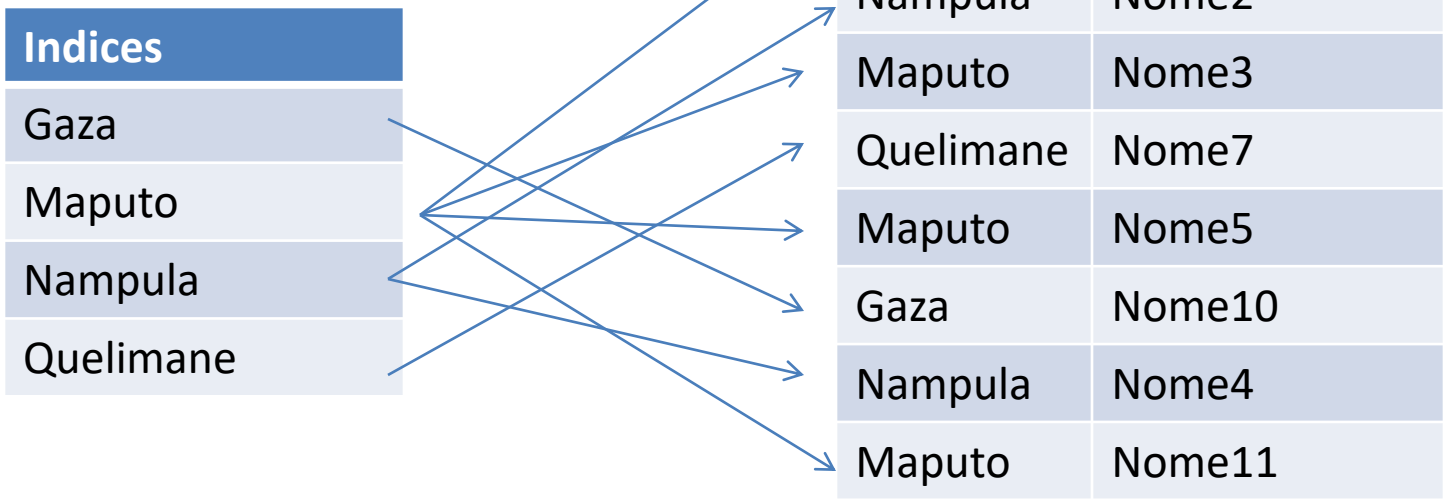
Um índice é uma organização de dados que permite acelerar o tempo de acesso as linhas de uma tabela.

“Uma abordagem semelhante ao conceito de índices em banco de da dos é o **índice remissivo, utilizado em livros**, onde os termos e os conceitos procurados freqüentemente pelos leitores são reunidos em um índice alfabético, colocado ao fi nal dele”

Motivacão

- **Uma é permitir que as linhas sejam acessadas rapida mente através do valor do atributo indexado.**
- **A segunda é facilitar a ordenação das linhas por aquele atributo.**
- **Já a terceira razão é concernente à unicidade, quando é necessário que uma coluna seja única, um índice é criado pelo SGBD com a função de assegurar que não sejam aceitos valores duplicados.**

Exemplo



Indices
Gaza
Maputo
Nampula
Quelimane

Provincia	Nome
Maputo	Nome1
Nampula	Nome2
Maputo	Nome3
Quelimane	Nome7
Maputo	Nome5
Gaza	Nome10
Nampula	Nome4
Maputo	Nome11

Busca por um inidivíduo de Maputo não implica percorrer todas as tuplas da tabela.

Extruturas de Dados

Uma query sobre uma base de dados encontrará sempre uma solução. No entanto, uma organização física dos dados adequada poderá devolver esta resposta de forma mais rápida.

Diferentes organizações físicas existem e podemos destacar:

- ✓ HEAP
- ✓ HASH
- ✓ ISAM
- ✓ BTREE

Estruturas de Dados

Heap (Pilha): acesso sequencial aos dados. Todos os dados devem ser percorridos para aceder ao dado pretendido.

Acesso sequencial →

Func	Nome	Salário
0005	Xavier	100
0015	Carlos	150
0001	Alberto	120

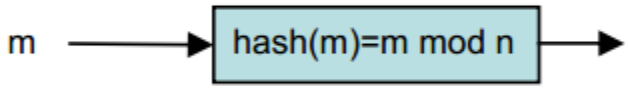
Estruturas de Dados

HASH : organização aleatória dos dados para um acesso via chave calculada na forma de uma função de hashing.

No caso das tabelas de HASH podem existir colisões.
De facto, se tomarmos n=80 então, hash(15)=hash(95)=hash(175)=…=15.
Este fenómeno chama-se uma colisão.
Neste caso, espaço deve ser encontrado para colocar os diferentes registos.
Utiliza-se usualmente uma lista de blocos para cada entrada. Estes blocos são chamados de Blocos de Overflow.

Estruturas de Dados

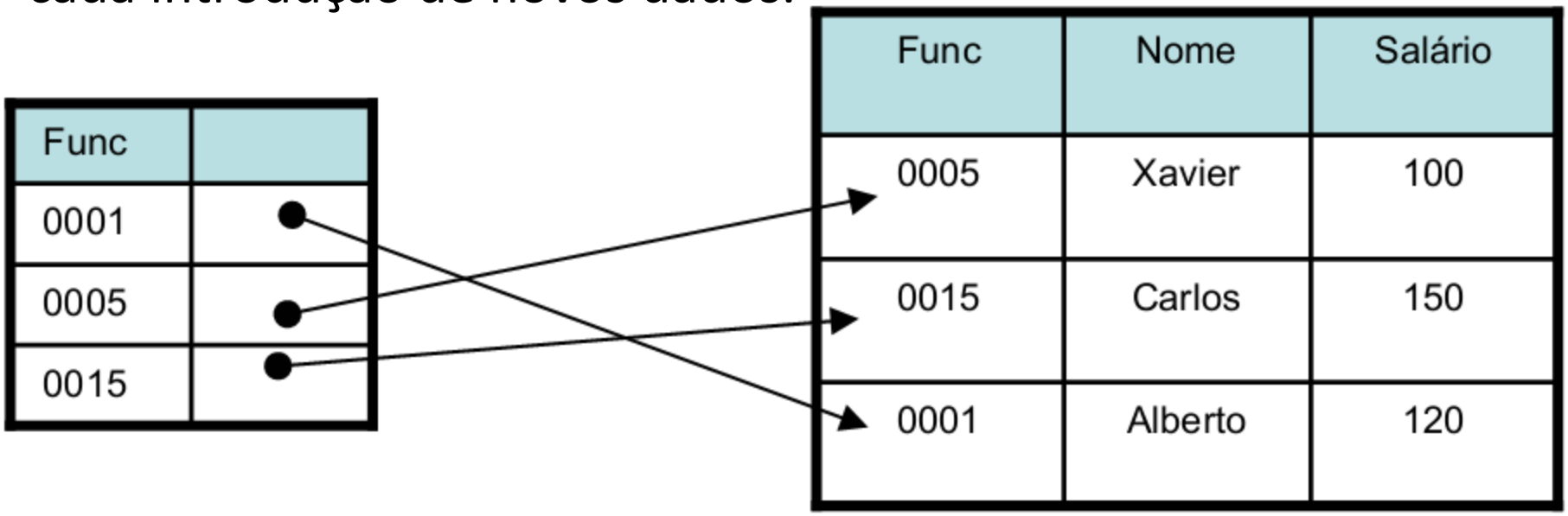
m representa o valor de procura do registo (e.g. o código do funcionário) e n representa o espaço de endereçamento definido para o ficheiro de dados.



	Func	Nome	Salário
Bloco 1	0005	Xavier	100
Bloco 2	0015	Carlos	150
Bloco 3	0001	Alberto	120
…	…	…	..
Bloco n			

Estruturas de Dados

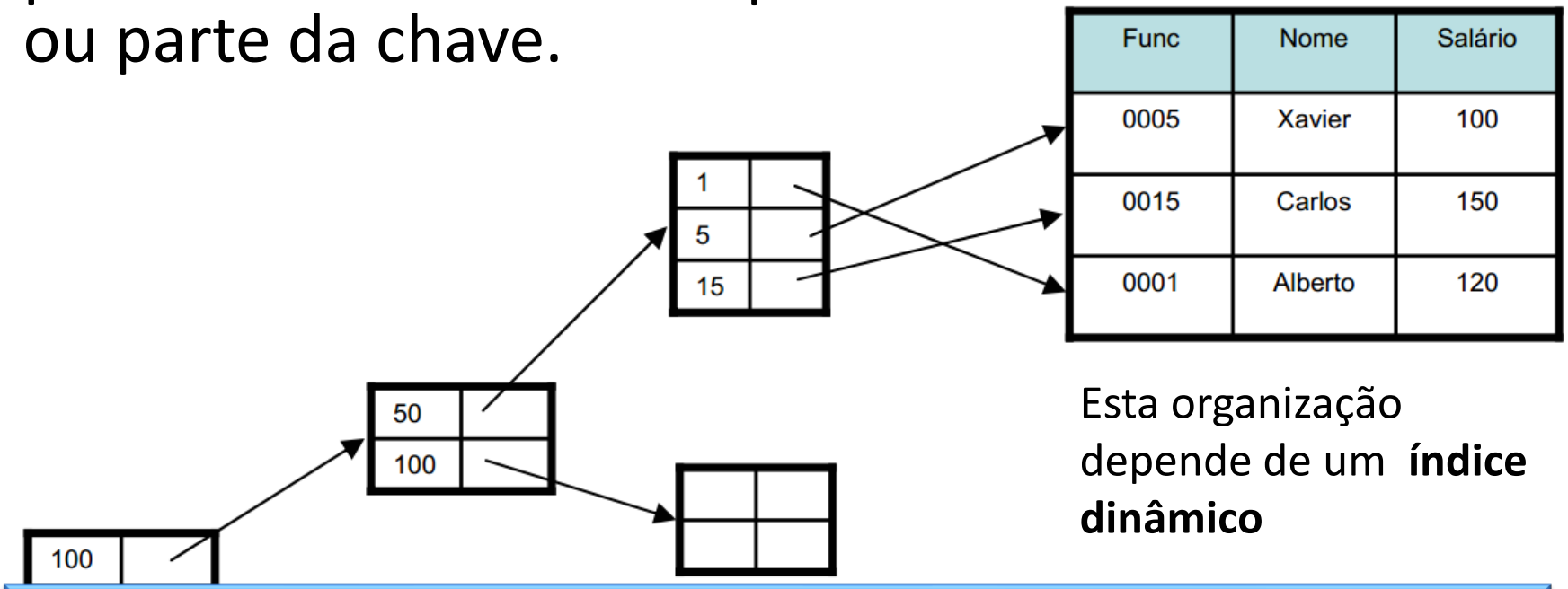
ISAM: organização indexada caracterizada por uma ordenação segundo os dados armazenados permitindo um acesso rápido sobre um valor exacto ou parte da chave. Esta organização depende de um índice estático que deve ser recalculado a cada introdução de novos dados.



Neste caso, o índice é ordenado mas a introdução de um novo funcionário implicará a reordenação do índice.

Estruturas de Dados

BTREE: organização indexada caracterizada por uma ordenação segundo os dados armazenados permitindo um acesso rápido sobre um valor exacto ou parte da chave.



Esta organização depende de um **índice dinâmico**

O índice é ordenado dinamicamente e a introdução de um novo funcionário não implicará a reordenação de todo o índice mas só uma reorganização local da árvore.

Estruturas de Dados

Tabela Resumo:

Operação	HEAP	HASH	ISAM	BTREE
Carregar a tabela	1	-	-	-
Anular duplos	-	1	1	1
Pesquisa na chave completa	-	1	2	2
Intervalos / Pattern Matching	-	-	1	1
Pesquisa Sequencial	1	3	2	2
Pesquisa sobre chave parcial	-	-	1	1
Acesso a dados ordenados	-	-	1	1
Índice cresce com a tabela	-	-	-	1
Pequena tabela	1	-	-	-
Grande tabela	-	-	-	1

- 1 para preferido
- 2 para aceitável
- 3 para medíocre
- - para não aconselhado ou impossível

Classificacão dos Índices

Primário vs. secundário

Se a chave de pesquisa contém a chave primária, o índice é chamado de primário.

Classificacão dos Índices

Clusterizado vs. não clusterizado

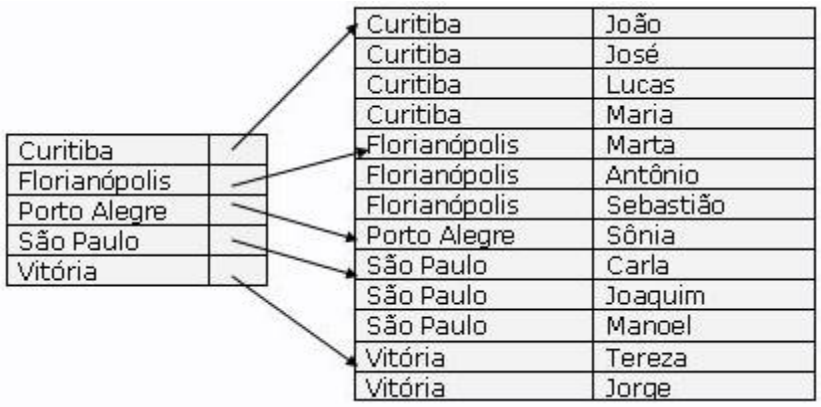
Se a ordem dos registros de dados é a mesma das entradas de dados, o índice é chamado de clusterizado.

- Um arquivo pode ser clusterizado apenas por uma chave de pesquisa
- O custo de recuperar registro de dados é fortemente dependente da clusterização

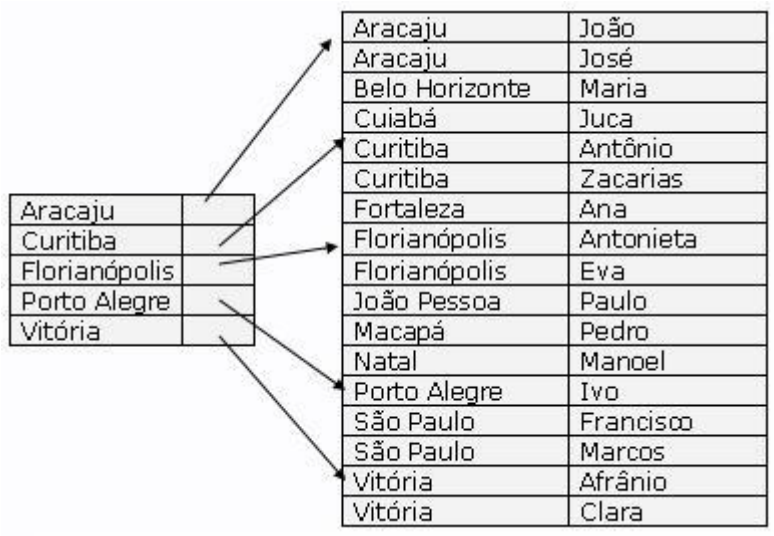
Classificacão dos Índices

Denso vs. Esparso

Se há pelo menos uma entrada de dados por valor de chave de pesquisa (em algum registro de dados), o índice é denso



Denso



Esparso

Exemplo

Suponhamos que temos a seguinte tabela:

```
CREATE TABLE pessoas (
    id INT NOT NULL AUTO_INCREMENT,
    primeiro_nome CHAR(30) NOT NULL,
    ultimo_nome CHAR(30) NOT NULL,
    idade SMALLINT NOT NULL,
    PRIMARY KEY (id)
);
```

Consulta pretendida:

```
SELECT id FROM pessoas WHERE primeiro_nome='Hugo' AND
ultimo_nome='Durães' AND idade=24;
```

Exemplo

- O primeiro passo seria criar um índice para um dos campos da claúsula WHERE (**primeiro_nome**, **ultimo_nome** ou **idade**).
- Se o índice fosse criado no campo **primeiro_nome**, o MySQL iria limitar a pesquisa aos registos para os quais o campo **primeiro_nome** fosse 'Hugo'.
- Com este resultado temporário, iria aplicar as restantes condições: primeiro iria eliminar todos os registos cujo **ultimo_nome** fosse diferente de 'Durães' e de seguida eliminaria os registos nos quais a **idade** fosse diferente de 24. Só após isto o MySQL poderia devolver os resultados da consulta.

Em que campos devem ser criados índices?

*Existem dois locais fulcrais para a criação de índices: campos referenciados na claúsula **WHERE** e campos usados na claúsula **JOIN**.

Atencão:
É necessário ter em conta o tipo de comparações que vão ser efectuadas:
O MySQL apenas usa índices para comparações do tipo < , <=, =, >, >=, BETWEEN, IN e em algumas do tipo LIKE.
Nas comparações do tipo LIKE, o MySQL não usa índices caso o primeiro caracter de pesquisa seja uma *wildcard* (% ou _).

Exemplos de criacão de Índices

```
CREATE INDEX IndicePorNome ON Pessoa
WITH STRUCTURE = BTREE,
KEY = (nome)

CREATE INDEX IndicePorCidadeEstado ON Pessoa
WITH STRUCTURE = BTREE,
KEY = (cidade, estado)

CREATE INDEX IndicePorEstadoCidade ON Pessoa
WITH STRUCTURE = BTREE,
KEY = (estado, cidade)
```

Estratégia de Indexacão

- ✓ A coluna de chave primária da tabela sempre deve ser indexada. As colunas de chave primária são freqüentemente usadas como chaves de pesquisa e em operações de ligação.
- ✓ Para as tabelas com menos de 100 linhas e com apenas algumas colunas, a indexação não é um procedimento vantajoso. Geralmente, as tabelas pequenas se ajustam facilmente ao cache do banco de dados.

Estratégia de Indexacão

- ✓ Os índices também devem ser definidos para as consultas executadas com freqüência ou para as consultas que devem recuperar dados rapidamente (consultas realizadas enquanto alguém espera um outro evento). Um índice deve ser definido para cada conjunto de atributos usados como critérios de pesquisa. Por exemplo, se o sistema precisa localizar todos os Pedidos de determinado produto, deve haver um índice na tabela Item, na coluna com números de produtos.

Estratégia de Indexacão

- ✓ Os índices devem ser definidos apenas em colunas usadas como identificadores e não em valores numéricos (saldos bancários) ou informações textuais (comentários sobre pedidos). Os valores identificadores tendem a ser atribuídos quando o objeto é criado e permanecem inalterados durante o tempo de vida do objeto.

Estratégia de Indexacão

- ✓ Os índices devem ser números simples (inteiros ou tipos de dados numéricos), e não números com pontos flutuantes. Raramente devem aparecer em seqüências de caracteres. As operações de comparação ficam bem mais simples e rápidas com números do que com seqüências de caracteres, além de fornecerem grandes volumes de dados processados em uma consulta ou em uma grande ligação. Pequenos recursos que significam eficiência. Os índices em colunas numéricas tendem também a ocupar menos espaço.

Cuidados a Ter

A **criação de índices**, com vista a indexar a informação, resulta numa **melhoraria das consultas** de informação. Mas, em contrapartida, estamos a **dificultar** os pedidos de **actualização** de informação. Logo, a definição dos índices tem de ser **bem estudada**, de modo a que seja aplicada apenas sobre **pontos críticos**.

Cuidados a Ter

- Não utilize índices para agilizar uma consulta pouco executada, a menos que ela ocorra em um ponto crítico e um máximo de velocidade seja necessário.
- Em alguns sistemas, o desempenho de atualizações e inserções é mais importante do que o de consultas.
- Lembre-se de que os índices têm um custo embutido: levam tempo para serem atualizados (uma taxa paga a cada inserção, atualização ou exclusão) e ocupam espaço em disco. Esteja certo de que usá-los será vantajoso.

Cuidados a Ter

O uso de índices provoca um aumento do espaço ocupado em disco. Assim, a criação de índices não estudada pode provocar um aumento exagerado do tamanho da informação indexada, podendo esta chegar ao seu limite físico em termos de tamanho em disco.

Referências

1. ELMASRI, R.; NAVATHE, S. B. , *Fundamentals of Database Systems*, Addison-Wesley Publishing; 2000, ISBN: 013057591
2. DATE, C. J. , *An Introduction to Database Systems*, Addison-Wesley Pub Co; 6th edition, 2000, ASIN: 020154329X
3. PEREIRA, J. L. , *Tecnologias de Base de Dados*, FCA, 3 edição, ISBN: 972-722-143-2
4. SILBERSCHATZ, A., KORTH, H. F. , SUDARSHAN, S.. *Sistemas de Bancos de Dados*. Campus, 1999.


OBRIGADO!