SISTEMAS DE BANCO DE DADOS 2

AULA 11

Índice

Vandor Roberto Vilardi Rissoli

APRESENTAÇÃO

- Fundamentos de Índices
- Instruções SQL de Índices
- Tipos de Índices
- Índices com *Hash*
- Pesquisa sobre Índices (Estruturas)
- Referências

ÍNDICES

Nos arquivos sequencias o compromisso de manter os registros fisicamente ordenados pelo valor da chave de ordenação, acarreta uma série de cuidados.

A operação de inserção, por exemplo, pode conduzir à necessidade do uso de áreas de extensão, além de periódicas reorganizações do BD.

Com o objetivo de conseguir um acesso eficiente aos registros, pode ser acrescentada uma nova estrutura para uso do BD, sendo ela definida como **ÍNDICE**.

ÍNDICES

O uso de índice é conveniente aos processos que precisam de um desempenho mais ágil, mesmo para os registros que não se encontrem ordenados fisicamente.

Exemplo:

Para uma consulta que deseje encontrar todas as contas somente da agência de **Brasília** têm-se que:

→ refere-se a uma porção (ou fração) de todos os registros de contas;

é **ineficiente** para o sistema ler todos os registros para localizar somente os que são desta agência;

seria **eficiente** um acesso direto a esses registros.

Assim torna-se interessante acrescentar uma nova estrutura que permita essa forma de acesso **mais eficiente**.

ÍNDICES

O atributo, ou o conjunto de atributos, usado para procurar um registro em um arquivo é chamado de **chave de procura** ou **chave de acesso**.

Esta definição de chave difere das definições de chaves estudadas até agora (candidata, primária, ...). Por meio das chaves de procura será possível acrescentar em um arquivo várias chaves de acesso, sendo elas disponibilizadas de acordo com a necessidade existente na situação.

Geralmente, é interessante ter **mais que um índice** para um arquivo de dados.

ÍNDICES

Exemplo:

- Imagine o catálogo de livros em uma biblioteca. Caso você deseje encontrar um livro de um autor específico, esta pesquisa será realizada sobre o catálogo de autores.
- O resultado desta pesquisa fornecerá um cartão que identificará onde encontrar o livro.
- Para ajudar na pesquisa sobre o catálogo, a biblioteca mantém os cartões em ordem alfabética.
- Com isso não será necessário pesquisar todos os cartões do catálogo para encontrar o cartão desejado.
- Em uma biblioteca são diversos os catálogos de procura (autor, título, assunto, entre outros).

ÍNDICES

Essa nova estrutura adicional consiste na definição de índices para os arquivos de dados, definindo os arquivos indexados.

- nos arquivos indexados podem existir tantos índices quantas forem as chaves de acesso aos registros;
- um índice irá consistir de uma entrada para cada registro do arquivo, sendo que as entradas encontram-se ordenadas pelo valor da chave de acesso;
- cada índice é formado pelo valor de um atributo chave do registro e pela sua localização física no interior do arquivo;
- a estrutura com a tabela de índices é também armazenada e mantida em disco.

ÍNDICES

ÍNDICES COM SQL

Um índice é construído sobre um ou mais atributos de uma relação (tabela). Em SQL o índice é criado por meio da instrução CREATE INDEX

Forma Geral

CREATE [UNIQUE] INDEX <nome_índice_idx>
ON <nome_tabela> (<atributo(s)>);

Exemplo:

CREATE INDEX Estados_Nome_**IDX**
ON Estados(nome);

ÍNDICES

A cláusula UNIQUE é opcional, mas quando for utilizada na criação de um índice ela assegurará que não existirão valores duplicados no índice que será criado.

Exemplo: Suponha para tabela **Cidade** a criação de um índice que não permita a duplicidade para o índice relacionado ao nome da cidade.

```
CREATE UNIQUE INDEX Cidade_Nome_IDX
    ON Cidade(nome);
```

Um índice único é criado automaticamente quando uma instrução SQL cria uma chave primária em uma tabela.

ÍNDICES

Geralmente, os banco de dados avisam quando se tenta criar um índice em uma tabela que já exista sobre o mesmo atributo. Isso ocorre porque o índice já está disponível para aquela tabela e não precisa se criar outro índice igual.

Esta coluna já está indexada.

No entanto, uma tabela pode possuir vários índices diferentes envolvendo o atributo ou atributos que serão chaves de pesquisas importantes.

```
CREATE UNIQUE INDEX Empregado_Nome_Nascer_IDX
    ON Empregado(nome, nascer);
```

ÍNDICES

APAGANDO UM ÍNDICE EM SQL

A instrução DDL DROP INDEX realiza a operação de apagar um índice do BD, mas no **MySQL** a remoção do índice acontece pela alteração da tabela do índice.

Exemplo:

```
ALTER TABLE ESTADOS
    DROP INDEX Estados_Nome_IDX;

ALTER TABLE EMPREGADO
    DROP INDEX Empregado_Nome_Nascer_IDX;
```

A mesma instrução DDL apaga índices, independente deles serem UNIQUE ou envolverem mais que um atributo em sua construção (índice composto).

ÍNDICES

Ainda assim, existirão grandes arquivos de índices que não seriam manipulados tão eficientemente (como no exemplo anterior da "biblioteca").

Por isso, algumas técnicas mais sofisticadas de índices são adotadas.

Dois tipos básicos de índices:

Ordenados: baseiam-se na ordenação dos valores

Hash: baseiam-se na distribuição uniforme dos valores determinados por uma função (*função de hash*)

ÍNDICES

Não existe uma técnica melhor, sendo cada técnica mais adequada para aplicações específicas.

Cada técnica deve ser avaliada sobre alguns fatores:

- tipos de acesso: encontrar registros com um atributo determinado ou dentro de uma faixa de valores
- tempo de acesso: tempo gasto para encontrar um item de dados ou um conjunto de itens (tuplas)
- tempo de inserção: tempo gasto para incluir um novo item de dados, incluindo o tempo de localização do local correto e a atualização da estrutura de índice
- tempo de exclusão: tempo gasto para excluir um item de dados, sendo incluído o tempo de localização do registro, além do tempo de atualização do índice
- sobrecarga de espaço: espaço adicional ocupado pelo índice, compensando este sacrifício pela melhoria no desempenho

ÍNDICES

ÍNDICES ORDENADOS

Para conseguir acesso aleatório rápido sobre os registros de um arquivo, pode-se usar uma estrutura de índice.

- Cada índice esta associado a uma chave de procura (armazena a chave de procura de forma ordenada e associa a ela os registros que possuem aquela chave);
- Os registros em um arquivo indexado podem ser armazenados (eles próprios) em alguma ordem;
- Um arquivo pode ter diversos índices, com diferentes chaves de procura, estando ele ordenado sequencialmente pelo seu *índice primário*;
- Normalmente, os índices primários em um arquivo são as chaves primárias, embora isso nem sempre ocorra

ÍNDICES

Índice Primário

Este índice consiste em um arquivo ordenado cujos registros são de tamanho fixo, contendo dois campos:

1º- mesmo tipo do campo chave no arquivo de dados

2º- ponteiro para o bloco do disco

O índice primário é um índice seletivo, pois possui entradas para um subconjunto de registros (bloco), ao invés de possuir uma entrada para cada registro do arquivo de dados.

Como o índice primário mantém ordenado os registros no arquivo, os processos de **inserção e remoção** são um grande "problema" para este tipo de estrutura.

ÍNDICES

ÍNDICE ORDENADO

Há dois tipos de índices ordenados (denso e esparso):

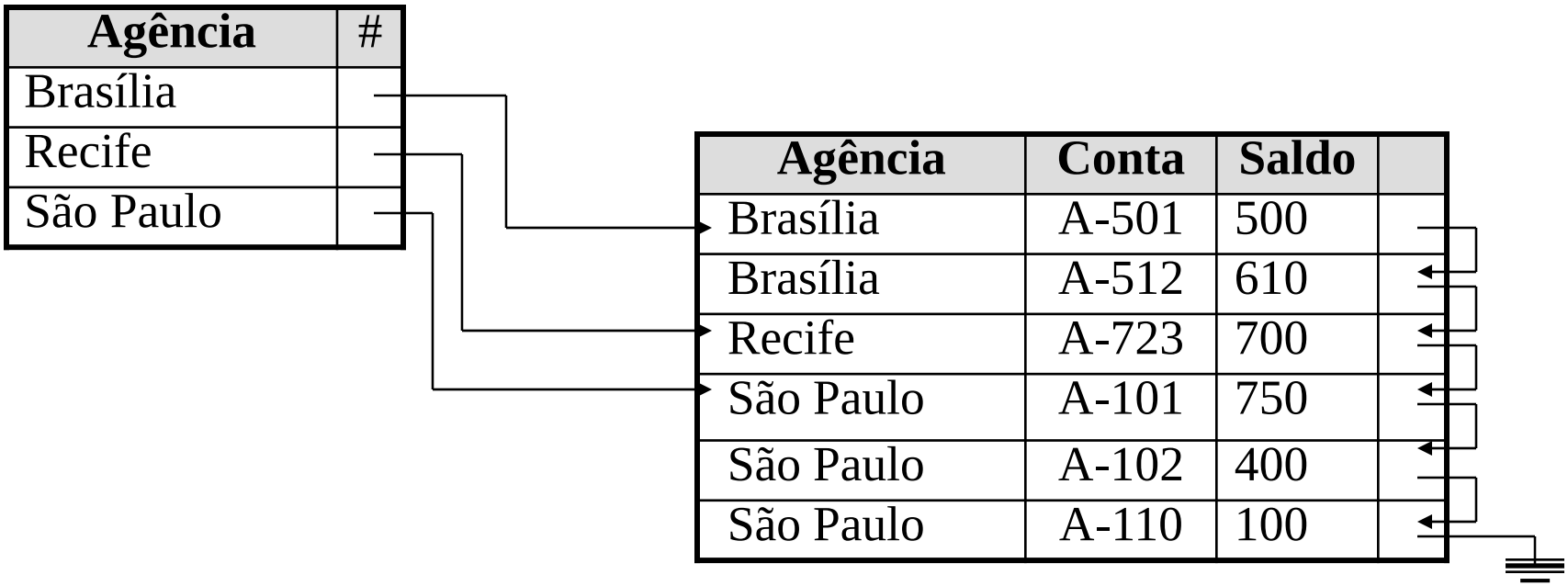
Índice Denso

- Um registro de índice aparece para cada valor distinto da chave de procura no arquivo;
- O registro contém o valor da chave de procura e um ponteiro para o primeiro registro de dados com esse valor;
- Alguns autores usam esta expressão para identificar quando um registro de índice aparece para cada registro no arquivo de dados.

ÍNDICES

Exemplo:

Índice denso para as agências bancárias.



ÍNDICES

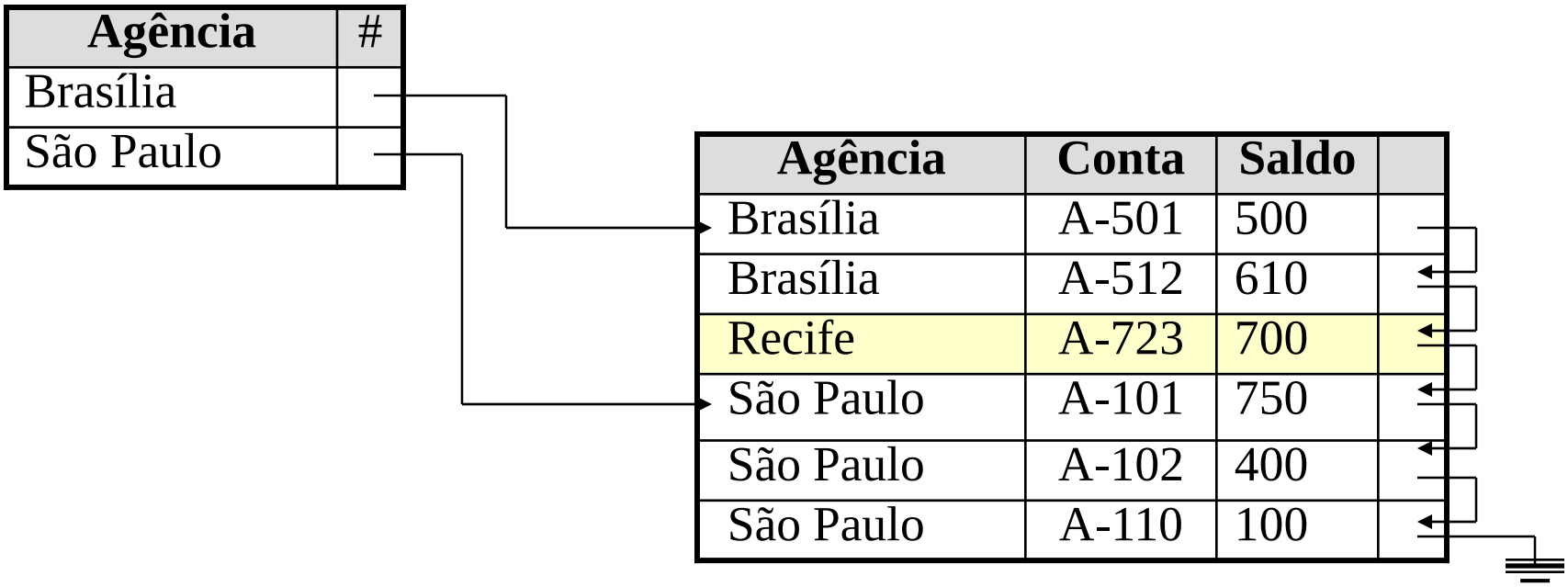
Índice Esparso

- Um registro de índice é criado apenas para alguns dos valores do arquivo de dados;
- Assim como os índices densos, o registro contém o valor da chave de procura e um ponteiro para o primeiro registro de dados com esse valor;
- Localiza-se a entrada do índice com o maior valor da chave de procura que seja menor ou igual ao valor da chave de procura que se deseja. Inicia-se no registro apontado e segue-se pelos ponteiros até localizar o registro procurado.

ÍNDICES

Exemplo:

Índice esparso para as agências bancárias.



Uma nomenclatura também utilizada é a de índice de *cluster*, onde tem-se os registros de um arquivo, fisicamente ordenados por um campo não chave, podendo serem distintos ou não.

ÍNDICES

Índice Secundário

Consiste de um arquivo ordenado que não usa o mesmo campo de ordenação como índice.

São os índices cujas chaves de procura especificam uma ordem diferente da ordem sequencial do arquivo, podendo seus valores serem distintos para todos os registros ou não.

Os índices secundários melhoram o desempenho das consultas que usam chaves diferentes da chave de procura do índice primário, mas impõem uma sobrecarga significativa na atualização do BD.

O projetista do BD decide quais índices secundários são desejáveis baseado na estimativa da frequência de consultas e atualizações.

ÍNDICES

ÍNDICES DE NÍVEIS MÚLTIPLOS

- **Arquivo de índice muito grande** para ser eficiente;
- Registros de índices são menores que os registros de dados;
- Índices grandes são armazenados como arquivos sequenciais em disco.

→ Uma busca em um índice **grande** também é **muito onerosa**.

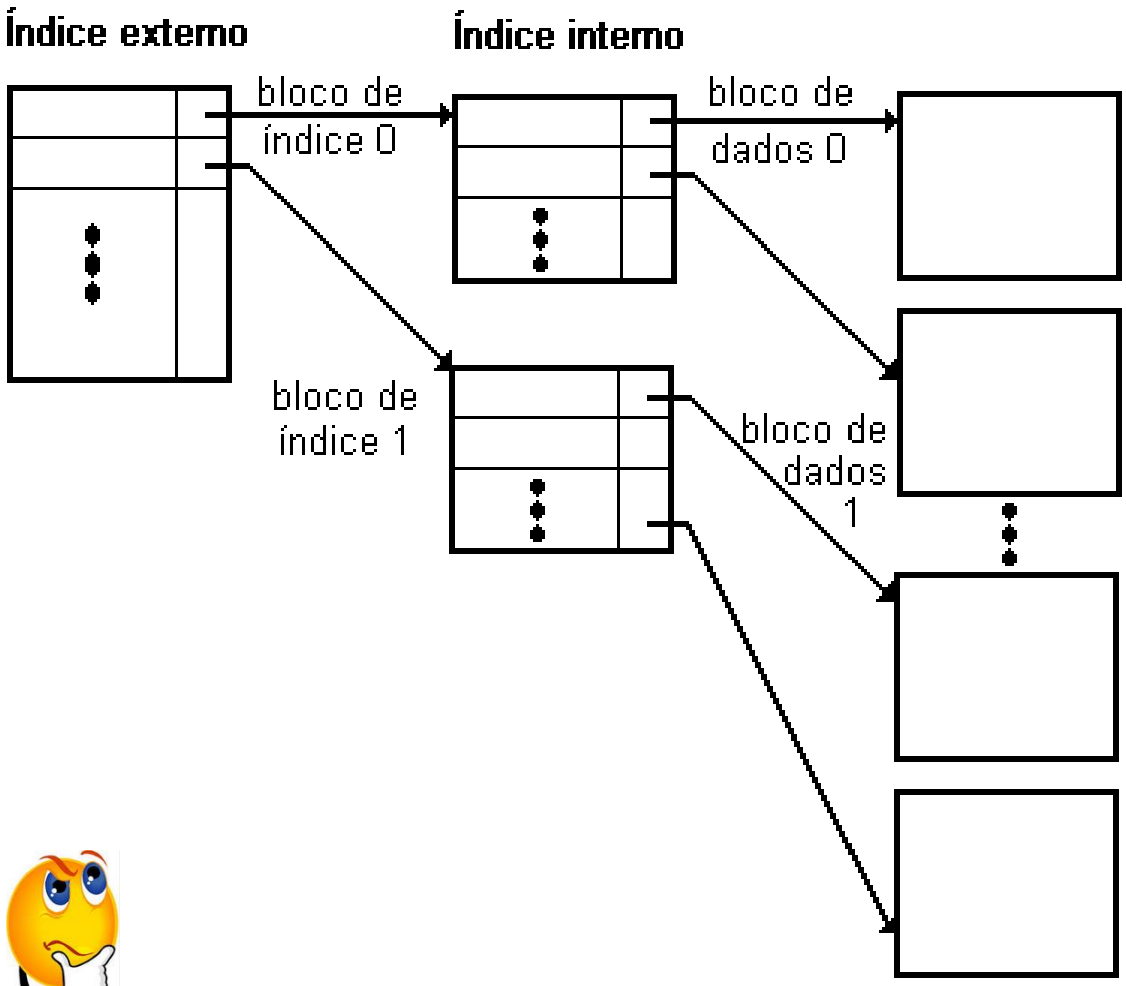
ÍNDICES

Uma possível solução na agilização de grandes Índices

1. Criação de índice ESPARSO como índice primário;
2. Realização de **busca binária** sobre o índice mais externo (encontrar maior valor da chave de procura que seja menor ou igual ao valor desejado);
3. O ponteiro indica o bloco do índice interno que será "varrido" até se encontrar o maior valor da chave de procura que seja menor ou igual ao valor desejado;
4. O ponteiro neste registro indica o bloco do arquivo que contém o registro procurado.

ÍNDICES

Observe a representação a seguir:



→ **Utiliza-se dois níveis de indexação;**

- caso o nível mais externo ainda seja muito grande;
- não cabendo totalmente na memória principal;
- pode ser criado um novo nível;
- criar quantos níveis forem necessários.

ÍNDICES

BUSCA ou PROCURA usando Níveis Múltiplos

- Requer um número significativamente menor de operações de E/S que a busca binária;
- Cada nível do índice corresponde a uma unidade de armazenamento físico (trilha, cilindro ou disco);

→ Os níveis de índices múltiplos estão relacionados as estruturas de árvores, como as árvores binárias usadas na indexação de memória.

ÍNDICES

ATUALIZAÇÃO DE ÍNDICE

Independente do tipo de índice utilizado, ele deve ser atualizado sempre que um registro for inserido ou removido do arquivo de dados.

REMOÇÃO

-Localizar o registro;

-Remover o registro;
(se houver)

-Atualizar o índice;
(denso ou esparso)

INSERÇÃO

-Localizar o registro;
(usa chave de procura)

-Incluir os dados (registro)

-Atualizar o índice
(denso ou esparso)

ÍNDICES

ORGANIZAÇÃO DE ARQUIVOS COM HASHING

Para uma localização de dados eficiente, em arquivos sequenciais, é necessário o acesso a uma estrutura de índice ou o uso da busca binária, que acarreta em mais operações de E/S.

Outra alternativa seria a aplicação de técnicas de *hashing*, em que os arquivos seriam organizados por meio de uma **função de *Hash***.

- Com o uso destas técnicas seria evitado o acesso a uma estrutura de índice;
- O *hashing* também proporciona um meio para construção de índices;

ÍNDICES

A organização de arquivos ***hashing*** oferece:

- Acesso direto ao endereço do bloco de disco que contém o registro desejado;
- Aplica-se uma função sobre o valor da chave de procura do registro para conseguir-se a identificação do bloco de disco correto;
- Utiliza-se do conceito de ***bucket*** (balde) para representar uma unidade de armazenamento de um ou mais registros;
- Normalmente equivalente a um bloco de disco, porém ele pode denotar um valor maior ou menor que um bloco de disco.

ÍNDICES

FUNÇÕES HASH

Uma função *Hash* ideal distribui as chaves armazenadas uniformemente por todos os *buckets*, de forma que todos os *buckets* tenham o mesmo número de registros.

No momento do projeto, não se sabe precisamente quais os valores da chave de procura que serão armazenados no arquivo.

Deseja-se que a função de *hash* atribua os valores da chave de procura aos *buckets* atentando a uma distribuição:

- **UNIFORME**
- **ALEATÓRIA** (o valor de *hash* não será correlacionado a nenhuma ordem visível externamente – alfabética por exemplo – parecendo ser aleatória)

ÍNDICES - *Exemplos de função Hash*

Observe a escolha de uma função *hash* sobre o arquivo conta, usando a chave de procura nome da agência.

- Decidiu-se a existência de 27 *buckets* (um por estado);
- Função simples, mas não distribui uniformemente os dados (SP tem mais conta que MA), além de não ser aleatória;

Neste outro exemplo, aplica-se as características típicas das funções de hash.

- Efetuam cálculos sobre a representação binária interna dos caracteres da chave de procura;
- Uma função simples calcularia a soma das representações binárias dos caracteres de uma chave de procura, retornando o módulo da soma pelo número de *buckets*;

ÍNDICES

CONCLUSÃO SOBRE USO DE HASH

O emprego das funções de *Hash* requerem um projeto cuidadoso.

→Função *Hash* "**RUIM**": pode resultar em procuras que consumam um tempo proporcional a quantidade de chaves no arquivo;

→Função *Hash* "**BOA**": oferece um tempo de procura médio (constante e pequeno), independente da quantidade de chaves de procura no arquivo;

ÍNDICES

OVERFLOW DE BUCKET

Até o momento supôs-se que os *buckets* sempre tem espaço para inserir um novo registro, porém isso pode não acontecer. Quando o *bucket* não possuir espaço suficiente, diz-se que ocorreu um *Overflow de Bucket*.

Este *overflow* pode acontecer por várias razões, sendo algumas delas apresentadas a seguir:

→*Buckets* insuficientes: conhecer o total de registro quando a função de *hash* for escolhida;

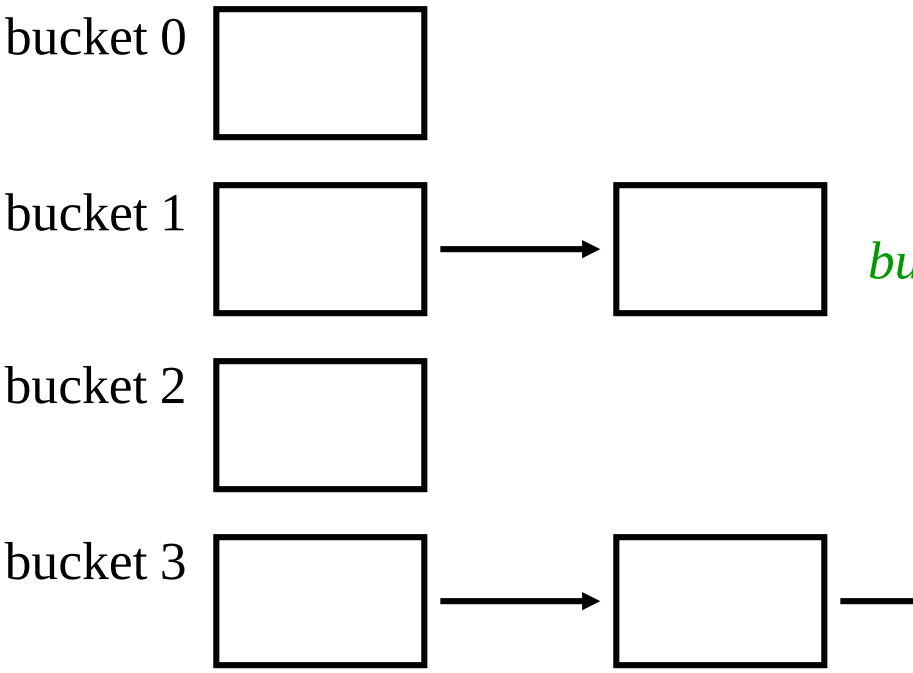
→Desequilíbrio (*skew*): pode acontecer por duas razões:

- registros múltiplos com a mesma chave de procura;
- distribuição não uniforme realizada pela função de *hash* escolhida.

ÍNDICES

Baseado em alguns cálculos é possível reduzir a probabilidade de *overflow* de *buckets*, processo este conhecido como **fator de camuflagem** (ou fator de *fudge*).

Algum espaço será desperdiçado no *bucket*, cerca de 20% ficará vazio, mas o *overflow* será reduzido. Ainda sim, o overflow de *bucket* poderá acontecer, por exemplo:



bucket de ***overflow*** para o *bucket* 1

Encadeamento (ou lista) de ***overflow*** para o *bucket* 3

ÍNDICES

Cuidados com o algoritmo de procura para o *overflow*:

- A função de *hash* é usada sobre a chave de procura para identificar qual o *bucket* do registro desejado;
- Todos os registros do *bucket* devem ser analisados, quando a igualdade com a chave de procura acontecer, inclusive nos *bucket* de *overflow*;

Existem outras variações na aplicação das técnicas de *hashing* (*hashing* aberto, linear *probing*, ...), porém a técnica descrita anteriormente é preferida para banco de dados (aspectos referentes aos problemas com a remoção).

ÍNDICES

Desvantagens desta Técnica

- A função de *hash* deve ser escolhida até o momento no qual o sistema será implementado;

- Como a função de *hash* mapeia os valores da chave de procura para um endereço fixo de *bucket*, ela não poderá ser trocada facilmente, caso o arquivo que esteja sendo indexado aumente muito ou diminua;

ÍNDICES

ATIVIDADE DE PESQUISA

TRABALHOS (livro do KORTH capítulo 11 da 3ª edição):

1- Arquivos de Índice **Árvore B+** (pág.346 – 11.3)
2- Arquivos de Índice **Árvore B** (pág.356 – 11.4)
3- Índices **Hash** (pág.362 – 11.5.2)
4- Arquivos **Grid** (pág.374 – 11.9.1 e 11.9.2 + pesquisa adicional)

Atividade (painel integrado):

- desenvolver o estudo dos temas pelo grupo
- discutir e anotar os conteúdos mais relevantes
- trocar os membros de cada grupo que explicaram aos novos companheiros de grupo o material estudado.

ÍNDICES

INDEXAÇÃO ORDENADA x HASHING

- Cada esquema representa vantagens e desvantagens para determinadas situações (inclusive para os arquivos *heap* – arquivos que não são ordenados de nenhuma maneira em particular);

- Tipo de consulta também é um fator importante:
  - Igualdade (... *where atributo = valor*)
    - em *hash* o tempo médio de procura é uma constante independente do tamanho do BD;
  - Faixa de valores (... *where atributo < valor*)
    - na ordenada localiza-se o valor e retorna-se todos os valores seguintes do *bucket* até atingir o valor limite, enquanto que o *hash* é aleatório e terá que localizar cada valor;

Exercício de Fixação

1) Acesse a base de dados indicada por seu professor e realize uma **análise** sobre o desempenho das consultas já existentes para esta base de dados. Inclua no *script* que armazena estas consultas um comentário SQL específico para cada consulta logo abaixo da instrução SQL (SELECT) esclarecendo sobre a **necessidade ou NÃO** da criação de **índice** para melhorar o desempenho de cada consulta, especificada também no M.A.L do projeto analisado.

   Esse comentário justificará a necessidade da criação de um ou mais índices para o melhor desempenho da execução de cada consulta. Caso verifique que a consulta **NÃO** precisa de novo índice para melhorar o desempenho o comentário deverá explicar porque cada uma delas não precisaria (note que este esclarecimento também é específico para cada uma das consultas que estão disponíveis no *script* de consultas).

Exercício de Fixação

... continuando exercício 1.

As soluções que indicarem a necessidade de índice para melhoria do desempenho de cada consulta, o comentário logo abaixo da instrução SELECT, deverá explicar tal necessidade e incluir também o comando SQL que cria o ÍNDICE que melhorará o desempenho da respectiva consulta.

Fundamentando o comentário por consulta deverá ainda existir uma média de **5 TEMPOS** da consulta realizada **SEM** o índice e outra média de outros **5 TEMPOS** da consulta **COM** índice como comentário neste mesmo *script* ainda (crie uma tabela mostrando o resultado de cada um dos tempos e logo abaixo a média aritmética dos 5 tempos analisados por situação de **SEM** e de **COM** novo índice). Nas consultas indicadas como sem a necessidade de novo índice serão também mostrados os 5 tempos e a média sem o novo índice.

Referência de Criação e Apoio ao Estudo

Material para Consulta e Apoio ao Conteúdo

- ELMASRI, R. e Navathe, S. B., Fundamentals of Database Systems, Addison-Wesley, 3rd edition, 2000
  - Capítulo 6

- SILBERSCHATZ, A. & Korth, H. F., Sistemas de Banco de Dados
  - Capítulo 11

- Universidade de Brasília
  - https://sae.unb.br/cae/conteudo/unbfga/

    (escolha no menu superior a opção **Labor. Banco Dados (SBD2)** seguida da opção **Índices**)