PROJETO FÍSICO

Profa. Dra. Maria Madalena Dias

PROJETO FÍSICO

- Introdução
- Armazenamento em Disco
  - Dispositivos de Armazenamento
  - Buffering de Blocos
  - Disposição de Registros de Arquivos em Disco
  - Operações em Arquivos
- Estruturas de Indexação de Arquivos
  - Índices Primários
  - Índices *Clustering*
  - Índices Secundários
  - Índices Multiníveis
  - Índices Multiníveis Dinâmicos (Árvores-B e Árvores-B^+)
  - Técnicas Hashing

Introdução

- As coleções de dados que compõem um banco de dados computadorizado precisam ser de alguma forma armazenadas em alguma mídia de armazenamento, para que assim os softwares SGBDs possam recuperar, atualizar e processar esses dados conforme necessário.

Dispositivos de Armazenamento

- Memória cache
  - Memórias não eram mais capazes de acompanhar a velocidade dos processadores
  - Tipo ultra-rápido de memória que serve para armazenar os dados mais frequentemente usados pelo processador
  - Extremamente caro (chega a ser algumas centenas de vezes mais cara que a memória RAM convencional)

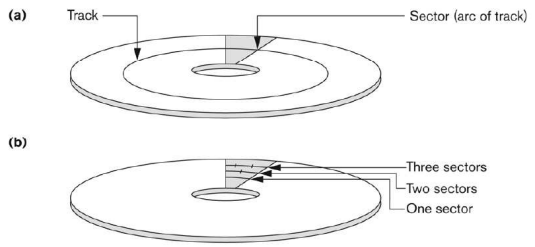
Dispositivos de Armazenamento

- Armazenamento em disco
  - São utilizados para armazenamento de grande quantidade de dados
  - Não voláteis.
  - Na unidade de disco estão presentes: cabeçote de leitura/escrita, braço mecânico, atuador e controladora de disco
  - Divisão física e divisão lógica

Dispositivos de Armazenamento

- Armazenamento em disco
  - Divisão física (não tem como mudar)
    - Trilhas
    - Cilindros
    - Setores
  - Divisão lógica (formatação)Blocos
  - Blocos são a unidade de transferência entre o disco e a memória principal

Dispositivos de Armazenamento



Dispositivos de Armazenamento

- Armazenamento em disco
  - Operação de leitura/escrita move o cabeçote do disco para o bloco a ser transferido.
  - Movimentos de rotação posicionamno setor apropriado.
  - Um endereço físico de bloco de disco consiste de:
    - Número do cilindro
    - Número da trilha
    - Número do bloco

Dispositivos de Armazenamento

- Armazenamento em disco
  - Desempenho baseado na medida de três tempos:
    - Tempo de pesquisa
    - Tempo de atraso rotacional (latência)
    - Tempo de transferência de bloco
  - Tempo de pesquisa e atraso rotacional são geralmente muitos maiores que o tempo de transferência do bloco

Dispositivos de Armazenamento

- Armazenamento em fitas magnéticas
  - Dispositivos de armazenamento de acesso sequencial
  - Os dados são armazenados em cartuchos de fitas magnéticas
  - Parecidas com as fitas de áudio e vídeo comuns
  - Cada byte disposto de forma transversal na fita
  - Acesso lento
  - Utilizadas principalmente para backup

Buffering de Blocos

- Diversos buffers podem ser reservados na memória principal para acelerar a transferência de dados
- Enquanto um buffer estiver sendo lido ou escrito a CPU pode processar os dados em outro.
- Útil para processos executados concorrentemente
- Buffering duplo: permite que leituras e escritas sejam realizadas de forma contínua em blocos consecutivos no disco, eliminando assim o tempo de pesquisa e atraso rotacional

Disposição de Registro Arquivos em Disco

- Um arquivo é uma sequência de registros
- Registro é uma coleção de valores de dados (itens de dados)
- Dois tipos de registros: tamanho fixo e tamanho variável
- Arquivos com mesmo tipo de registro e de tamanhos fixo facilitam a procura para SGBDs

Disposição de Registro Arquivos em Disco

- Registros são gravados em blocos
- Fator de blocagem (bfr): número de registros que podem ser armazenados em um bloco (B/R).
- Registros de arquivo podem ser spanned e não-spanned (unspanned)
  - Spanned: um registro pode ser armazenado em mais de um bloco
  - Não-spanned: registro não pode passar o limite do bloco

Operações em Arquivos

- As operações em um arquivo podem ser:
  - operações de recuperação
  - Operações de atualização
- Operações mais comuns:
  - Reset (Reinicializar): Reposiciona o ponteiro de arquivo, de um arquivo aberto, no seu inicio.
  - Find ou Locate (Encontrar ou Localizar): Busca o primeiro registro que satisfaça a condição de pesquisa, transfere o bloco que tem a condição de pesquisa para um buffer de memória principal e faz o ponteiro de arquivo apontar para o registro no buffer, tornando-o o registro atual.

Operações em Arquivos

- Read ou Get (Ler ou Obter): Copia o registro atual do buffer para uma variável do programa de usuário.
- FindNext (Encontrar o próximo): procura o próximo registro no arquivo que satisfaça a condição de pesquisa, transferindo o bloco que contém aquele registro para o buffer da memória principal.
- Delete (Excluir): Exclui o registro atual e no final atualiza o arquivo de disco para refletir a exclusão.
- Modify (Modificar): Modifica alguns valores de campos do registro atual e no final atualiza o arquivo no disco para refletir a modificação.

Operações em Arquivos

- Insert (Incluir): Acrescente um novo registro no arquivo por meio da localização do bloco no qual o registro deve ser incluído, transferindo aquele bloco para o buffer da memória principal, escrevendo o registro no buffer e no final escrevendo o buffer no disco para refletir a modificação.
- Close (Fechar): Finaliza o acesso ao arquivo por meio da liberação dos buffers e da execução de quaisquer outras operações de limpeza necessárias.

Estruturas de Indexação de Arquivos

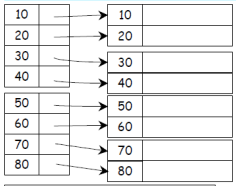
- **Índices Primários**
  - índice cuja chave especifica a ordem sequencial do arquivo
  - índice denso:
    - há uma entrada no índice para cada valor de chave que ocorre em um registro de dados
    - a entrada aponta para o primeiro registro que contém aquele valor de chave

Índices Primários

- índice esparso:
  - há uma entrada no índice apenas para alguns valores de chave
  - a entrada aponta para o primeiro registro que contém aquele valor de chave
  - para localizar um registro com chave K, procura-se a entrada E do índice com o maior valor de chave menor ou igual a K e pesquisa-se o arquivo a partir do registro apontado por E

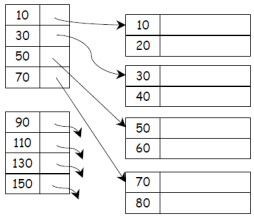
Índices Primários

Densos: uma entrada no arquivo de índices p/cada registro no arquivo de dados



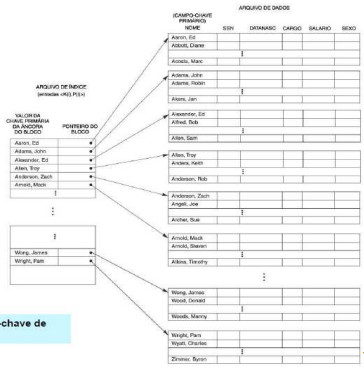
Um índice denso sobre um arquivo de dados seqüenciais

Esparsos: apenas alguns registros de dados são representados no arquivo de índices



Um índice esparso sobre um arquivo de dados seqüenciais

Índices Primários

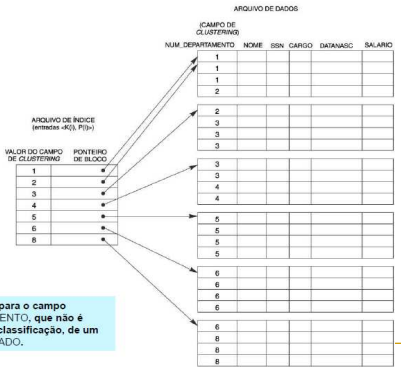


Índice primário para o campo-chave de classificação do arquivo.

Índices *Clustering* (Agrupamento)

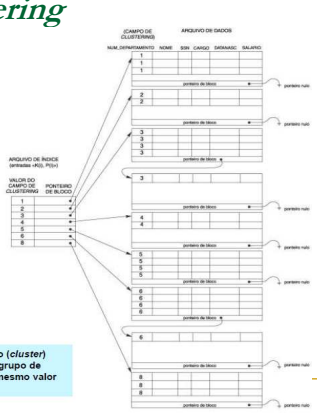
- Os registros de um arquivo são fisicamente ordenados segundo um campo que não seja um campo-chave – o qual possua um valor distinto para cada registro – chamado campo de *clustering*
- Arquivo ordenado com dois campos:
  - o primeiro é do mesmo tipo do campo de clustering do arquivo de dados;
  - o segundo é um ponteiro de bloco.
- Há uma entrada no índice de *clustering* para cada valor distinto do campo de *clustering*, contendo o valor e o ponteiro para o primeiro bloco de arquivo de dados que possui um registro com aquele valor para seu campo de *clustering.*

Índices *Clustering*



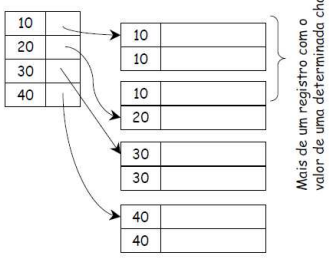
Índice *clustering* para o campo NUM_DEPARTAMENTO, **que não é campo-chave de classificação, de um arquivo** EMPREGADO.

Índices *Clustering*



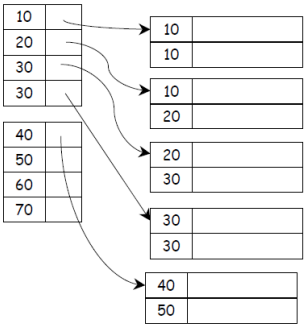
Índice *clustering* com um grupo (*cluster*) separado de blocos para cada grupo de registros que compartilhem o mesmo valor de campo *clustering*.

Índices *Clustering*



Índice denso quando são permitidas chaves de pesquisa duplicadas

Índices *Clustering*



Índice esparso indicando a nova chave de pesquisa mais baixa em cada bloco

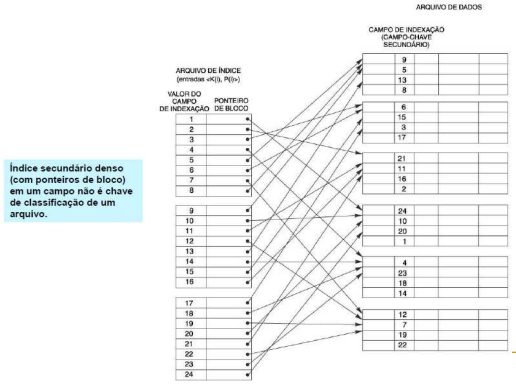
Índices Secundários

- índice cuja chave não é aquela da ordem sequencial do arquivo
- organização:
  - há uma entrada no índice para cada valor de chave que ocorre em um registro de dados
  - a entrada aponta para **todos** os registros que contêm aquele valor de chave x

Índices Secundários

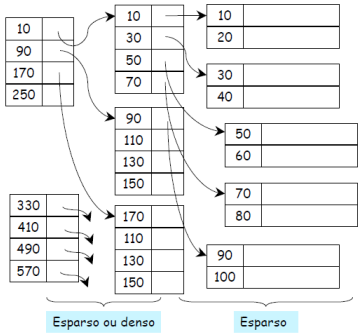
- A entrada de índice para um bloco de dados é a menor chave de pesquisa que é nova, isto é, a chave não apareceu em um bloco anterior;
- Se não há chave de pesquisa nova no bloco, então sua entrada de índice contém a única chave de pesquisa encontrada nesse bloco;
- Pode-se encontrar os registros de uma chave de pesquisa K examinando-se o índice p/a primeira entrada cuja chave é:
  - Igual a K
  - Menor que K, mas a próxima chave é maior que K
- Segue-se o ponteiro da entrada. Se for encontrado pelo menos um registro com a chave de pesquisa K, então a busca continua em blocos adicionais até encontrar todos os registros com a chave de pesquisa K.

Índices Secundários



Índice secundário denso (com ponteiros de bloco) em um campo não é chave de classificação de um arquivo.

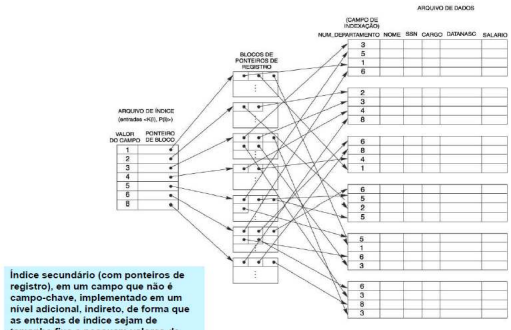
Índices Secundários



Índices Secundários sobre campos não-chave

- Vários registros do arquivo de dados podem ter o mesmo valor para o campo de indexação.
- **Opção 1**: Várias entradas no índice com o mesmo valor, uma para cada registro. Índice denso.
- **Opção** 2: Uma entrada no índice para cada valor X do campo de indexação, com lista de ponteiros. Índice não denso.
- **Opção 3**: Uma entrada no índice para cada valor X do campo de indexação, com um ponteiro para o bloco que contém a lista de ponteiros para os registros com o valor X. Índice não denso.

Índices Secundários sobre campos não-chave

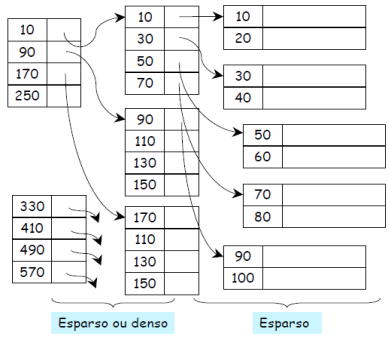


Índice secundário (com ponteiros de registro), em um campo que não é campo-chave, implementado em um nível adicional, indireto, de forma que as entradas de índice sejam de tamanho fixo e possuam valores de campo únicos.

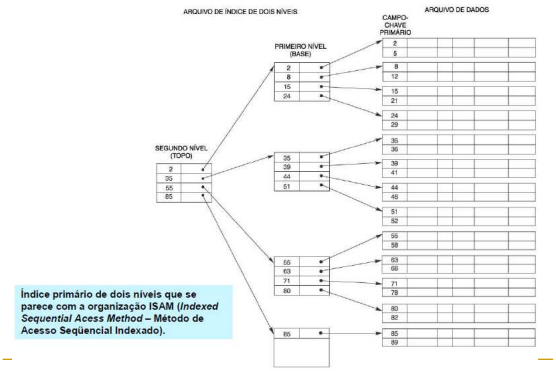
Índices Multiníveis

- Se o arquivo de índices se torna muito grande para ser armazenado em bloco de disco, é interessante indexá-lo em mais de um nível
- Vantagem: índice pequeno pode ser mantido em memória e o tempo de busca é mais baixo
- Desvantagem: muitos níveis de índices podem aumentar a complexidade do sistema (talvez seja melhor usar a árvore-B)

Índices Multiníveis



Índices Multiníveis



Índice primário de dois níveis que se parece com a organização ISAM (*Indexed Sequential Acess Method* – Método de Acesso Seqüencial Indexado).

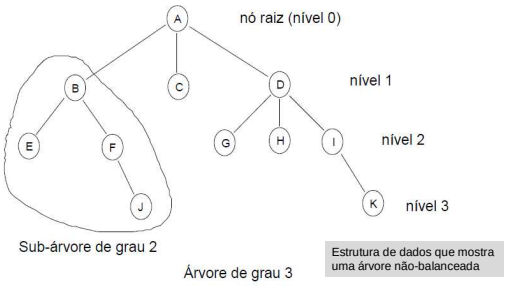
Índices Multiníveis

- Problema dos índices multiníveis:
  - índices são arquivos fisicamente ordenados, portanto, ineficientes na inserção e remoção.
- Solução:
  - Deixar algum espaço em cada um dos blocos para inserção de novas entradas.
  - Estruturas de dados: Árvores B e suas variações.

Índices Multiníveis Dinâmicos que usam Árvores-B (*B-Trees*) e Árvores-B$^+$ (*B$^+$–Trees*)

- Árvore:
  - formada por nós (ou nodos)
  - árvore de busca de ordem n
  - a raiz possui no mínimo 2 filhos
  - cada folha possui no mínimo $\lceil (n-1)/2 \rceil$ valores de chave
  - cada nó interior possui no mínimo $\lceil n/2 \rceil$ filhos
  - todos os ramos têm o mesmo comprimento

Índices Multiníveis Dinâmicos que usam Árvores-B (*B-Trees*) e Árvores-B$^+$ (*B$^+$–Trees*)

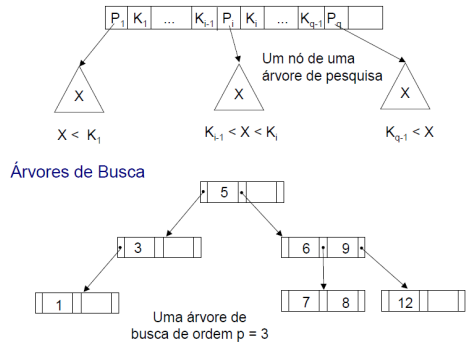


Estrutura de dados que mostra uma árvore não-balanceada

Árvores de Busca e Árvores-B

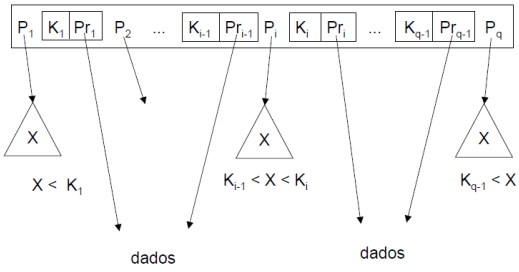
- Uma árvore de busca de ordem p é uma árvore tal que cada nó contenha pelo menos p-1 valores de busca e p ponteiros na ordem <P_1, K_1, P_2, K_2,..., P_{q-1},K_{q-1},P_q>, onde q ≤ p; cada P_i é um ponteiro para cada nó filho (ou um ponteiro nulo); e cada K_i é um valor de busca para algum conjunto ordenado de valores.
- Duas condições sempre devem ser satisfeitas em uma árvore de busca:
  - Dentro de cada nó, $K_1 < K_2 < ... < K_{q-1}$;
  - Para todos os valores X na subárvore apontada por P_i, temos K_i-1 < X < K_i para 1 < I < q; X < K_i para i = 1; e K_{i-1} < X para i = q.

Árvores de Busca e Árvores-B



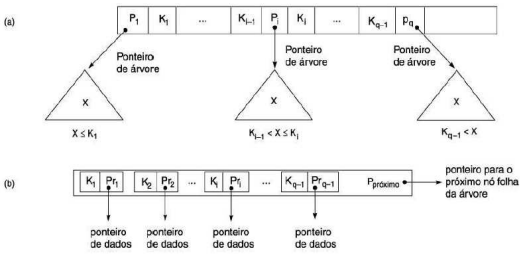
Árvores de Busca

Árvores de Busca e Árvores-B



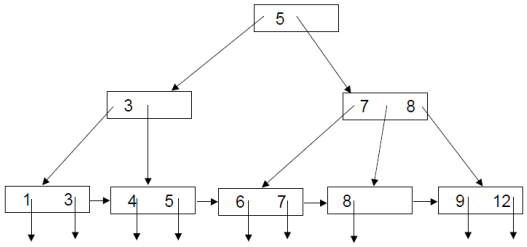
p/2 < q ≤ p

Árvores-B$^+$



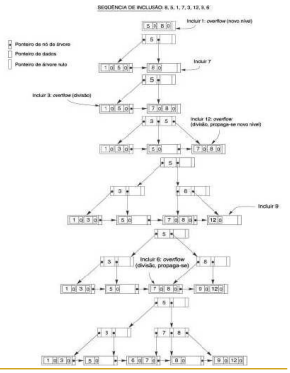
Os nós de uma árvore-B$^+$. (a) Nó interno de uma árvore-B$^+$ com q - 1 valores de busca. (b) Nó folha de uma árvore-B$^+$ com q - 1 valores de busca e q - 1 ponteiros de dados.

Árvores-B$^+$



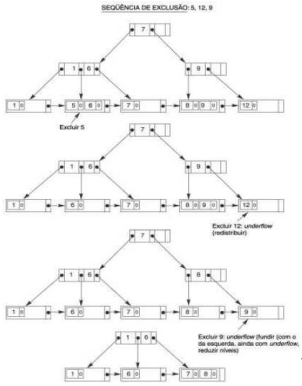
Árvores-B+

Um exemplo de inclusão em uma árvore-B* de ordem p = 3 e p_{folha} = 2.



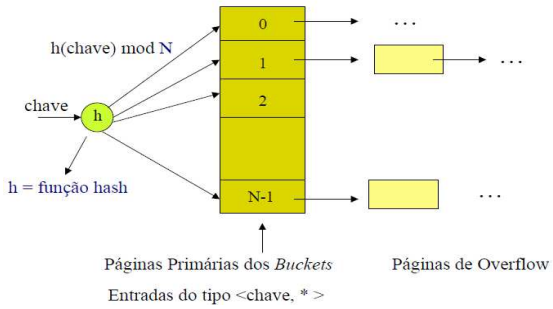
Árvores-B+

Um exemplo de remoção em uma árvore-B*.



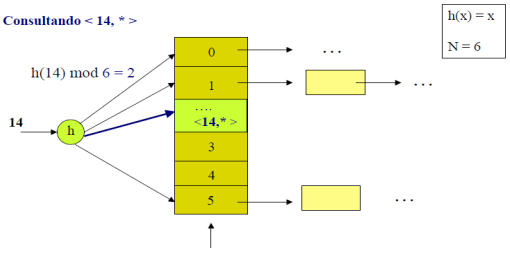
TÉCNICAS HASHING

Hash Estático



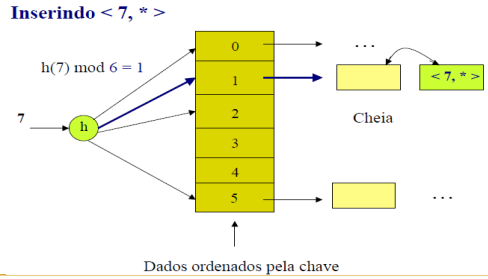
HASH ESTÁTICO

Hash Estático - Busca



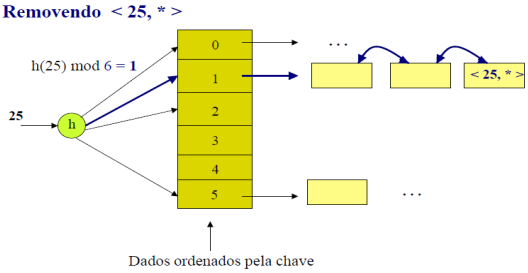
HASH ESTÁTICO

Hash Estático - Inserção



HASH ESTÁTICO

Hash Estático - Remoção



HASH ESTÁTICO

- Função Hash:
  - Componente importante da técnica Hash;
  - Deve distribuir valores das chaves de maneira uniforme nos buckets;
  - Número de buckets = N = parâmetro;
  - h(x) = a*x + b
    - a, b: parâmetros de ajuste

HASH ESTÁTICO

- Hash - Custos:
  - Páginas primárias podem ser armazenadas em
  - páginas de disco sucessivas.
  - Caso não haja overflow:
    - Busca requer 1 I/O
    - Inserção e Remoção requerem 2 I/O
  - Custo pode ser alto se existem muitas páginas de overflow.

HASH ESTÁTICO

- Desvantagens do Hash Estático:
  - Número de buckets é fixo.
  - Se arquivo encolhe muito, o espaço é desperdiçado, já que os buckets são fixos.
  - Crescimento do arquivo produz longas cadeias de páginas de overflow, prejudicando o desempenho da busca.

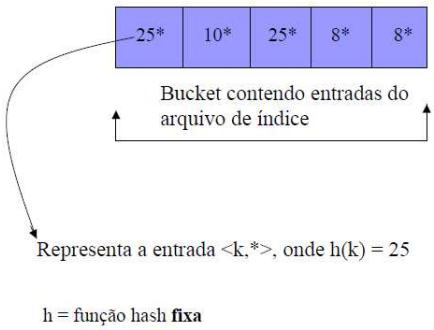
HASH ESTÁTICO

- Alternativa 1:
  - Periodicamente modificar a função hash e
  - reestruturar todo o arquivo de modo a evitar páginas
  - de overflow.
  - «rehash» toma muito tempo.
  - Indice não pode ser utilizado durante o processo de «rehash».
- Alternativa 2: Hash dinâmicos
  - Extensível.

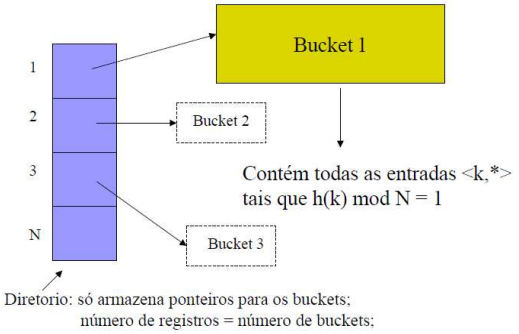
HASH EXTENSÍVEL

- Solução 1: quando algum bucket ficar cheio:
  - Dobrar o número de buckets;
  - Distribuir as entradas nos novos buckets.
  - Defeito: o arquivo todo deve ser lido e reorganizado e o dobro de páginas devem ser escritas.
- Solução 2: utilizar um diretório de ponteiros para os buckets:
  - Dobrar o número de entradas no diretório.
  - Separar somente os buckets que ficaram cheios.

HASH – Convenção de Notação



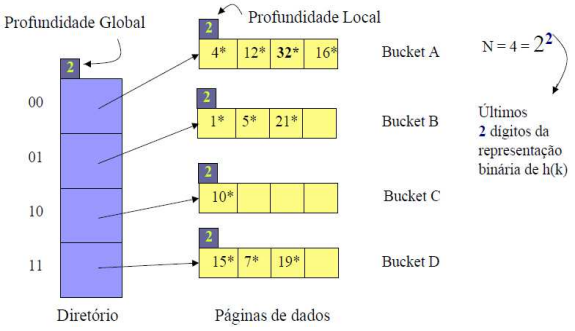
HASH – Diretórios de Buckets



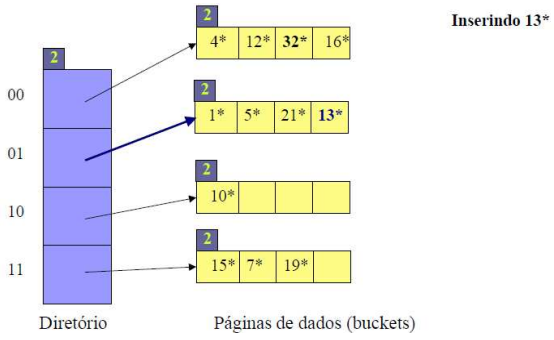
HASH – O que pode variar

- Função hash não varia.
- O número N de buckets varia.
- A medida que os buckets se enchem, estes se duplicam, e o diretório de buckets duplica.
- Resultado:
  - se um único bucket duplica, o diretório todo de buckets duplica;
  - Dois ponteiros do diretório podem apontar para o mesmo bucket;
  - Só duplicam os buckets que ficam cheios;
  - Ao contrário do hash estático, registros em buckets duplicados (decorrentes de um overflow) podem ser facilmente localizados através do novo ponteiro no diretório de buckets.

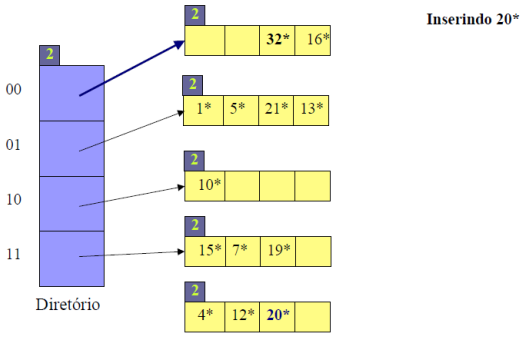
HASH - Exemplo



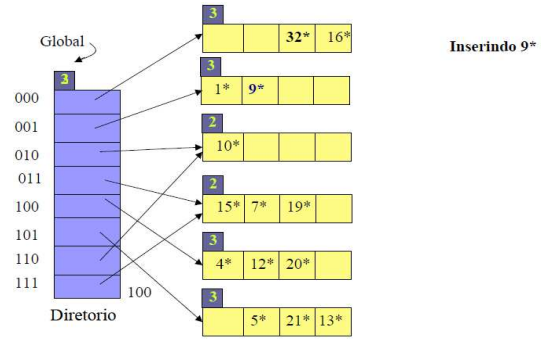
HASH – Exemplo Inserção



HASH – Exemplo Inserção



HASH – Exemplo Inserção



HASH – Regra geral para Inserção de K*

- Se Nível global = d
  - Calcula h(k);
  - Considera a entrada m do diretório, onde m = número correspondente aos d últimos dígitos da representação binária de h(k);
  - Dirige-se para o bucket indicado;
  - Se o bucket estiver cheio e nível local = d
    - Divide o bucket e duplica o diretório de buckets;
  - Se o bucket estiver cheio e nível local = d-1
    - Divide o bucket, mas não duplica diretório.

HASH – Possíveis Problemas

- Distribuição tendenciosa dos valores h(k): muitos em um único bucket.
  - Este é um problema que pode ser resolvido no momento da criação do índice: basta ajustar a função h de modo a ter uma distribuição uniforme.
- Colisão: quando existem muitas entradas <k,*> com mesmo h(k), que não cabem em uma página
  - Este é um problema que só aparece à medida que o arquivo cresce.
  - Neste caso, páginas de overflow são utilizadas.

HASH – Vantagens e Limitações

- Hash é excelente para seleção por igualdade na chave.
- Não suporta seleção range (>, <, <=, >=)
- B-Trees suportam seleção range e são quase tão boas quanto Hash para igualdade.
- Técnica de indexação Hash é muito útil na implementação do operador
- Junção, que inclui diversas seleções por igualdade. Muitos SGBDs só implementam índices estruturados por B-Trees.
- Diferença de custo entre B-Tree e Hash é significativa neste caso.

Referência

- Elmasri, R.; Navathe, S. B. Sistemas de Banco de Dados, 4a. Edição, Person Education do Brasil, 2005.