INDEXAÇÃO E HASHING

Construção de Índices e Funções Hash

Diego Gomes Tomé - MSc. Informática
Orientador: Prof. Dr. Eduardo Almeida

October 13, 2016

Universidade Federal do Paraná

INDEXAÇÃO E HASHING

Índices

- Conceitos Básicos
- Classificação

Hashing

- Conceitos Básicos
- Funções Hash Estáticas
- Funções Hash Dinâmicas
- Resolução de Colisões

ÍNDICES

ÍNDICES - CONCEITOS BÁSICOS

Ás vezes, nós queremos recuperar os registros de uma tabela especificando os valores de um ou mais campos

- Encontrar todos os estudantes cujo curso é 'BCC'
- Encontrar todos os estudantes cujo CR > 7

Um Índice recupera os registros de forma rápida. através de uma chave de pesquisa

- Qualquer subconjunto dos campos de uma relação podem ser uma chave de pesquisa
- Chave de Pesquisa não é uma chave primária, pois não precisa necessariamente ser única

Índices - conceitos básicos

Índice

- Coleção de entradas de dados que suporta a recuperação eficiente de registros combinando uma certa condição de pesquisa.

Seletividade

- Medida da quantidade de variedade existente nos valores de uma dada coluna em relação ao número total de linhas de uma dada tabela.

Índices - classificação

Índices - classificação

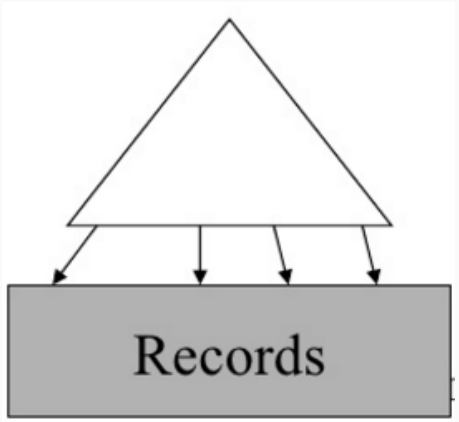
Índices podem ser:

- Agrupados (Clustered) ou Desagrupados (Unclustered)
- Denso ou Esparso
- Primário ou Secundário

ÍNDICES - CLASSIFICAÇÃO - CLUSTERED VS UNCLUSTERED

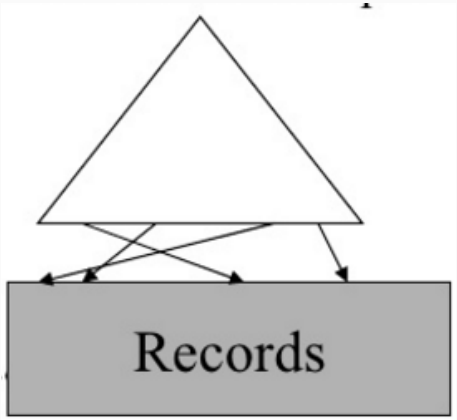
Agrupado (Clustered)

- Se a ordem dos registros e dos índices é a mesma ou próxima



Desagrupado (Unclustered)

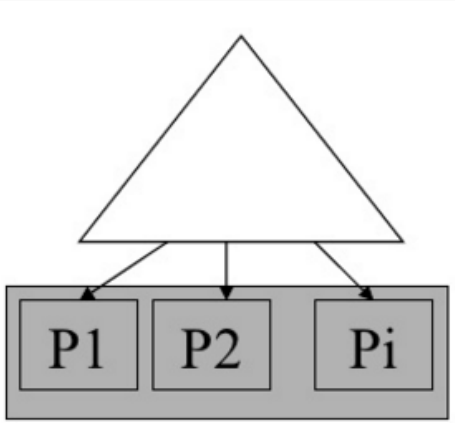
- Não existe ordem



ÍNDICES - CLASSIFICAÇÃO - DENSO VS ESPARSO

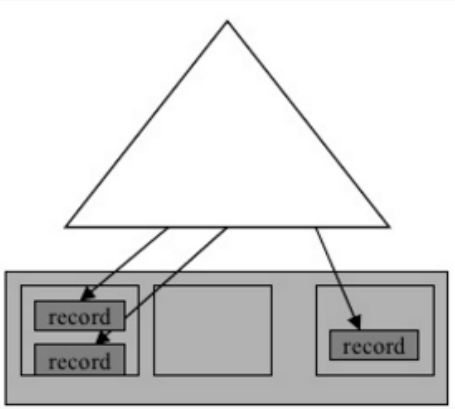
Esparso

- Um dado de entrada para cada página de arquivo
- Usa menos espaço de armazenamento, porém leva mais tempo para localizar um registro dada a sua chave



Denso

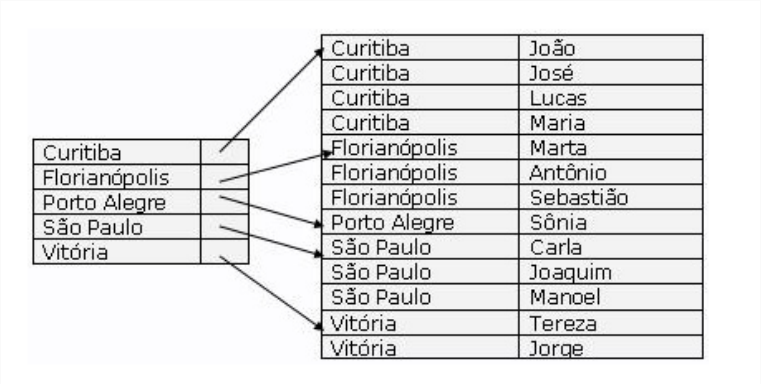
- Pelo menos um dado de entrada por chave-valor
- Seqüência de blocos contendo apenas as chaves dos registros e os ponteiros para os próprios registros



ÍNDICES - CLASSIFICAÇÃO - PRIMÁRIO VS SECUNDÁRIO

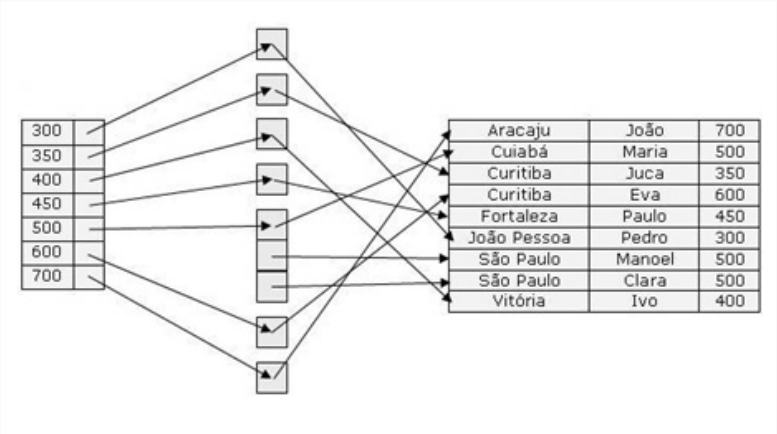
Primário

- Baseado na chave de ordenação
- Nunca contém duplicidade



Secundário

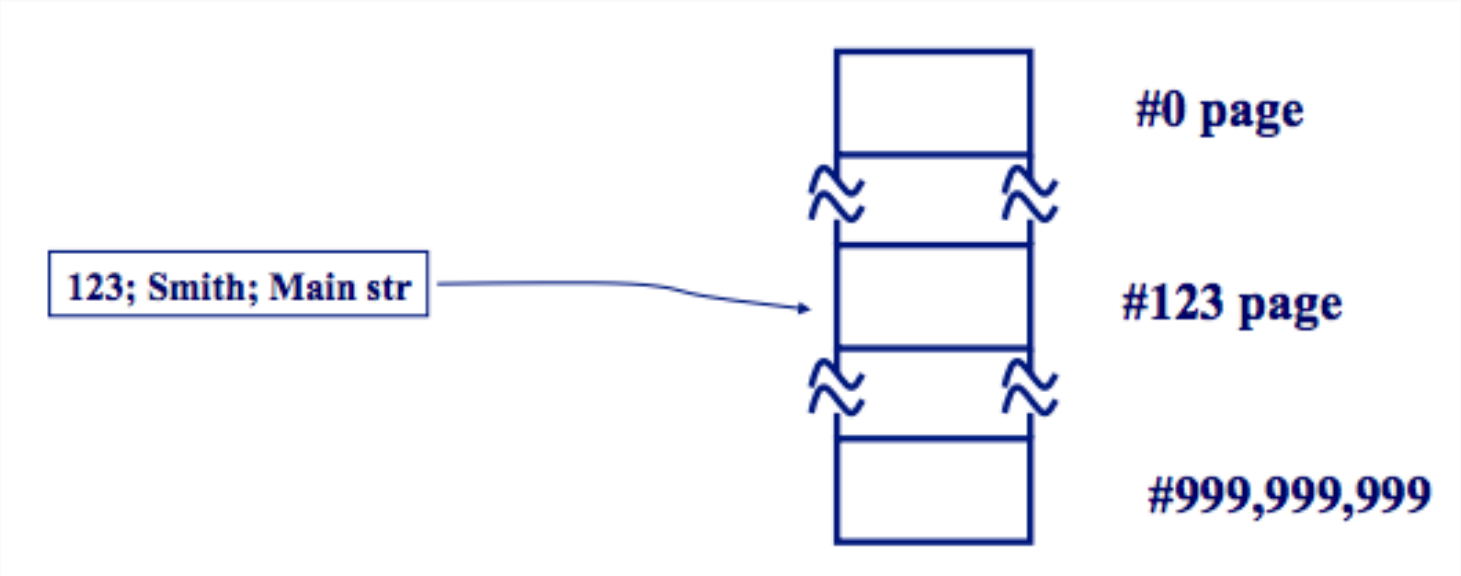
- Qualquer outro índice de campo não ordenado
- Os ponteiros não apontam diretamente para o arquivo com registros, mas para um bucket que contém ponteiros para o arquivo



HASHING

Problema

- Como Encontrar um cliente cujo id = 123 ? (Consultas pontuais)
- Considerando **999.999.999** clientes (Muitos Registros)



Bucket

- Um arquivo de hash armazena os dados em formato de bucket. Bucket é considerado uma unidade de armazenamento. Normalmente armazena um bloco de disco completo, que por sua vez pode armazenar um ou mais registros.

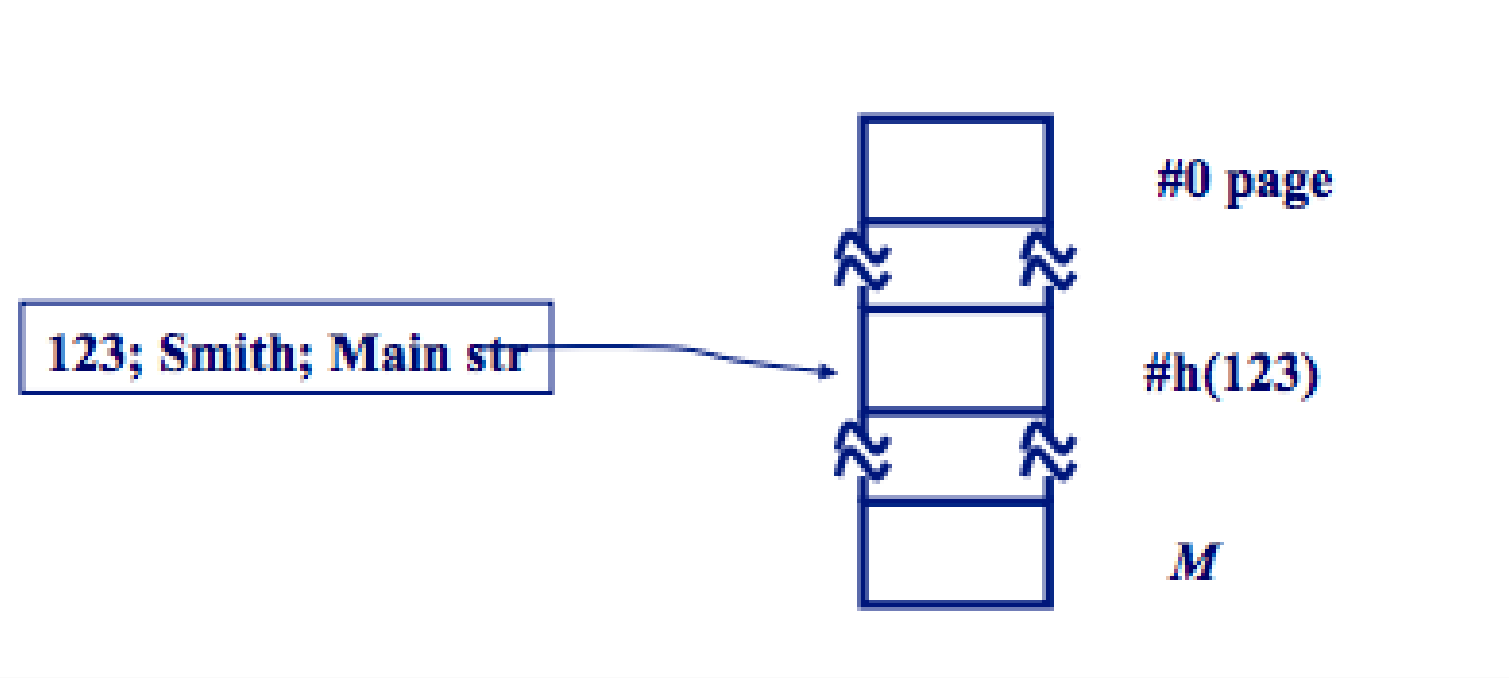
Função Hash

- A função hash, h, é uma função de mapeamento que mapeia todo o conjunto de chaves de pesquisa K para o endereço onde os registros reais são colocados. É uma função de chaves de pesquisa para endereços dos Buckets.

FUNÇÕES HASH ESTÁTICAS

Funções Hash Estáticas

- Transformar cada chave de busca de **0, 1, ..., N** em um intervalo de **0, 1, ..., M**
- Utilizando uma função hash(**key**)
- Podem ser Clustered ou Unclustered
- Tempo médio de busca O(1)



FUNÇÕES HASH ESTÁTICAS

Operações

- **Insertion** - Quando um registro é necessário para ser inserido usando função hash de estática, a função hash h calcula o endereço do bucket para a chave de pesquisa K, onde o registro será armazenado Bucket address = h(K)
- **Search** - Quando um registro deve ser recuperado, a mesma função hash pode ser usada para recuperar o endereço do bucket onde os dados são armazenados
- **Delete** - Uma operação de busca seguida por uma de deleção do dado

FUNÇÕES HASH ESTÁTICAS

Características

- A função hash mapeia uma chave para um determinado bucket
- No pior caso ela mapeia várias entradas para o mesmo bucket
- No melhor caso ela mapeia todo valor chave-pesquisa para um bucket diferente
- Funções hash devem ter distribuição uniforme e aleatória

E quando o banco de dados cresce e o número de buckets não é suficiente?

FUNÇÕES HASH DINÂMICAS

Funções Hash Dinâmicas

- A função hash é modificada dinamicamente

Hash Extensível (Extendible Hashing)

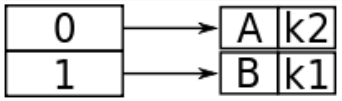
- Calcula o hash de cada chave para uma string de bits longa, mantendo o tamanho de acordo com o necessário
- Mantém um diretório com ponteiros para os buckets do hash
- A função hash gera um valor dentro de um intervalo, geralmente um b-bit = 32

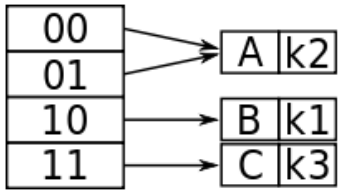
Operações

- **Insertion** - O endereço do bucket é calculado
  - Caso o bucket esteja cheio deve-se adicionar mais um bucket e adicionar mais um bit para o hash
  - Caso contrário, inserir o dado no bucket
- **Search** - Verificar a profundidade do prefixo e usar os bits para encontrar o endereço do bucket após o hash
- **Delete** - Uma operação de busca seguida por uma de deleção do dado

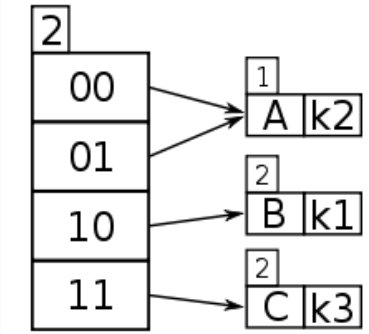
Exemplo Hash Extensível

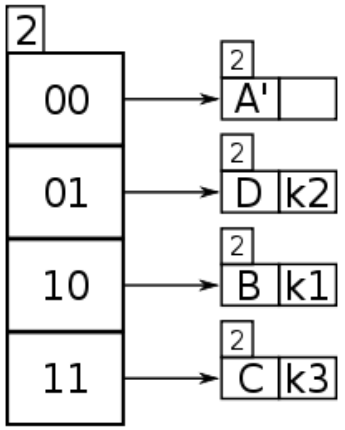
$h(k_1) = 100100$
$h(k_2) = 010110$
$h(k_3) = 110110$



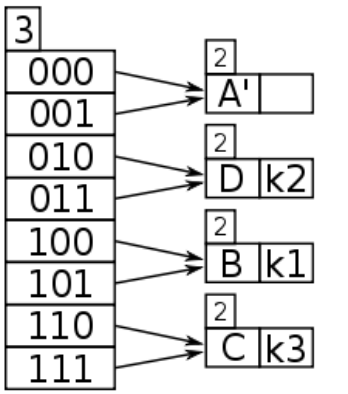


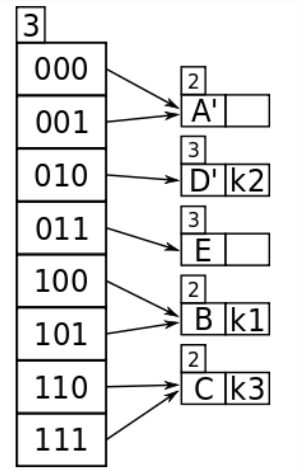
$h(k_4) = 011110$





$h(k_4) = 011110$





RESOLUÇÃO DE COLISÕES

O que é uma colisão?

- Duas chaves obtém o mesmo resultado na função hash

Como resolver uma colisão?

- Colocar o valor no próximo slot/bucket (Linear Probing)
- Rehash
- Separação em canais (Overflow chaining)

RESOLUÇÃO DE COLISÕES - LINEAR PROBING E OVERFLOW CHAINING