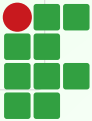


Instituto Federal Minas Gerais - Campus Bambuí
Departamento de Engenharia e Computação - DEC
Curso de Engenharia de Computação - ENGCOMP

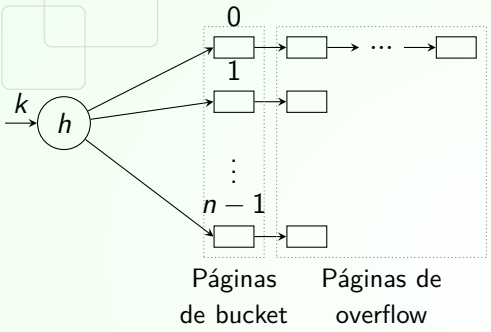
Banco de dados II

4 – Indexação baseada em hash

Marcos Roberto Ribeiro

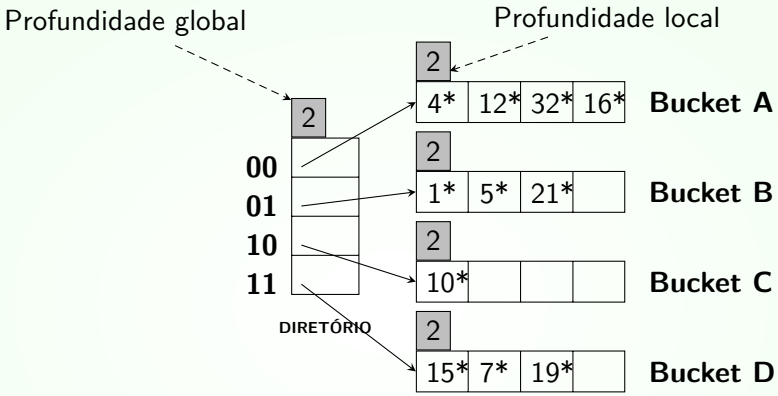
2022

Hash estático



- $h(k) = k$ **mod** n onde n é o número de buckets (ou $\log_2 n$ últimos bits de k)
- Páginas de overflow são criadas quanto não há espaço na página de bucket
- O índice é estático porque o número de páginas de bucket não muda (são alocados assim que o índice é criado)
- O problema é que podem aparecer longas cadeias de overflow causando lentidão nas pesquisas

Hash extensível



- O hash extensível utiliza um diretório de ponteiros para os buckets
- Este diretório é duplicado se ocorrer algum overflow

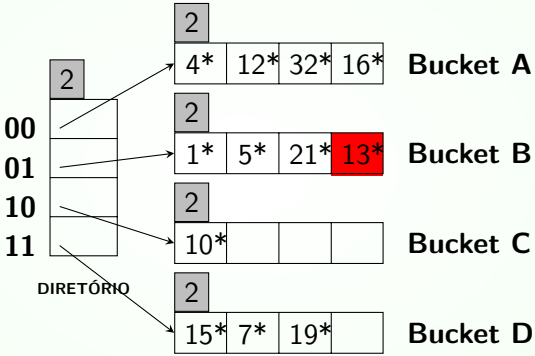
Inserções em hash extensível

- Para inserir uma entrada calculamos a função hash sobre a chave considerando os últimos n bits
- Onde n é a profundidade local
- Exemplo: Inserir a entrada 13*, temos $h(13) = 01$

- Neste caso a entrada cabe no bucket, então apenas gravamos a entrada na posição livre

Inserções em hash extensível

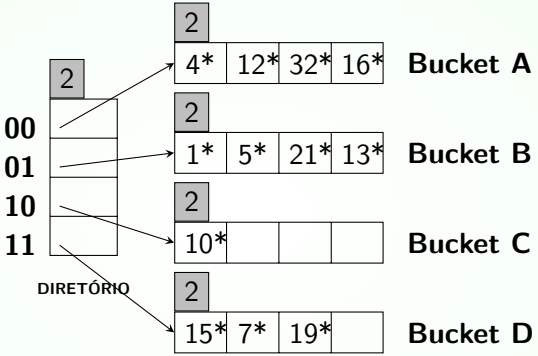
- Para inserir uma entrada calculamos a função hash sobre a chave considerando os últimos n bits
- Onde n é a profundidade local
- Exemplo: Inserir a entrada 13*, temos $h(13) = 01$



- Neste caso a entrada cabe no bucket, então apenas gravamos a entrada na posição livre

Inserção com overflow

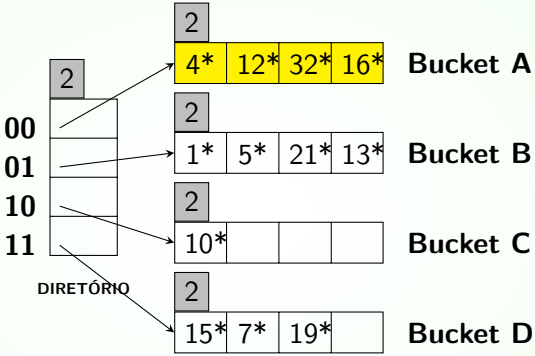
- Agora, considere a inserção da entrada 20*
- $h(20) = 00$ (bucket A)



- Entretanto, o bucket A está cheio

Inserção com overflow

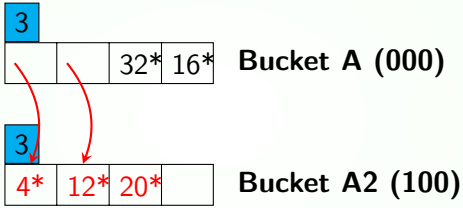
- Agora, considere a inserção da entrada 20*
- $h(20) = 00$ (bucket A)



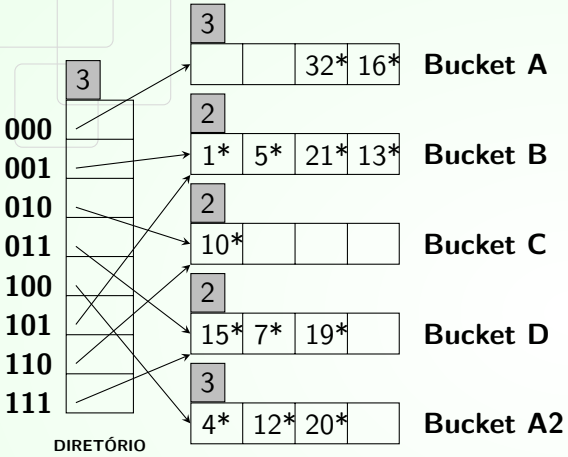
- Entretanto, o bucket A está cheio

Inserção com overflow - novo bucket

- Aumentamos a profundidade local do bucket A para 3
- Criamos o novo bucket A2 e redistribuímos os registros usando a nova profundidade (3 últimos bits)



Inserção com overflow - duplicação do tamanho do diretório



- Devemos duplicar o tamanho do diretório sempre que há uma profundidade local maior do que a profundidade global
- Os buckets que não foram divididos permanecem com a profundidade local inalterada e recebem dois ponteiros

Exclusões de buckets

- Durante as exclusões, poderíamos excluir os buckets vazios
- Porém, na prática, isto não é viável porque os arquivos tendem a crescer

Hash linear

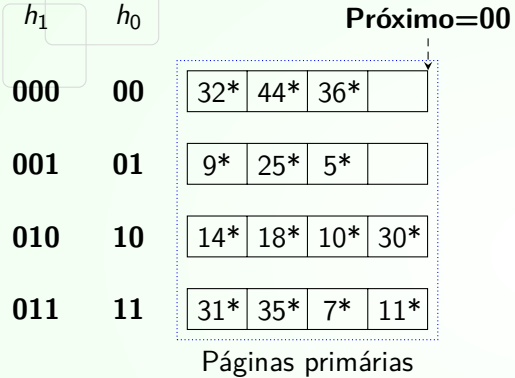
- O hash linear reduz as divisões e proporciona maior ocupação dos buckets do que o hash extensível
- Este tipo de hash utiliza uma família de funções hash h_0, h_2, h_2, ...
- O intervalo de h_{i+1} é o dobro do intervalo de h_i
- Por exemplo, supondo 32 buckets iniciais ($N_0 = 32 = 2^5$):

  $h_0(k) = k \textbf{ mod } 32 \Rightarrow d_0 = 5$ (5 bits)

  $h_1(k) = k \textbf{ mod } 64 \Rightarrow d_1 = d_0 + 1 = 5 + 1 = 6$ (6 bits, $N_1 = 64$)

  $\vdots$

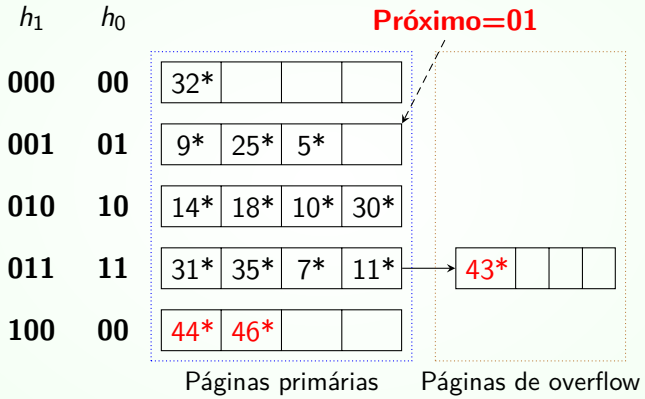
Exemplo de hash linear



- Iniciamos com 4 buckets ($N_0 = 4$) e nível 0 (zero)
- $N = N_0 * 2^{\text{nível}} = 4 * 2^0 = 1$ (número de buckets atual)
- Podem ocorrer páginas de overflow
- Dividimos o bucket apontando por **próximo** sempre que ocorre overflow
- Redistribuímos as entradas usando $h_{\text{nível}+1}$ e incrementamos o apontador **próximo**

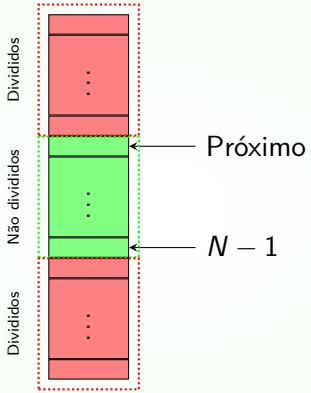
Inserção com overflow

- Ao inserir 43*, temos $h_0(43) = 11$
- Como o bucket **11** está cheio, criamos um página de overflow para incluir a entrada 43*
- Dividimos o bucket **00** apontado por próximo e incrementamos próximo para **01**

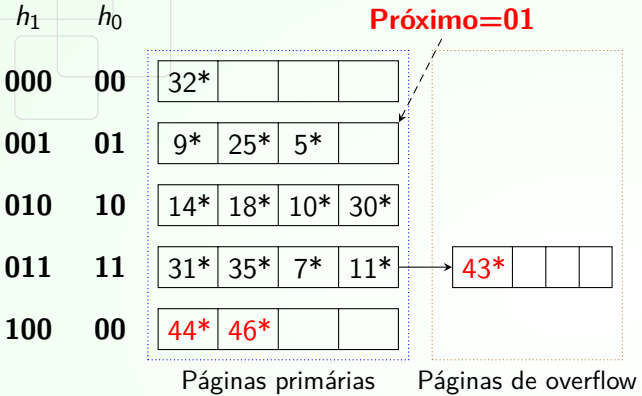


Rodadas de divisão

- O apontador próximo circula pelos buckets até que todos sejam divididos
- Quando isto acontece, passamos a usar as funções hash do próximo nível



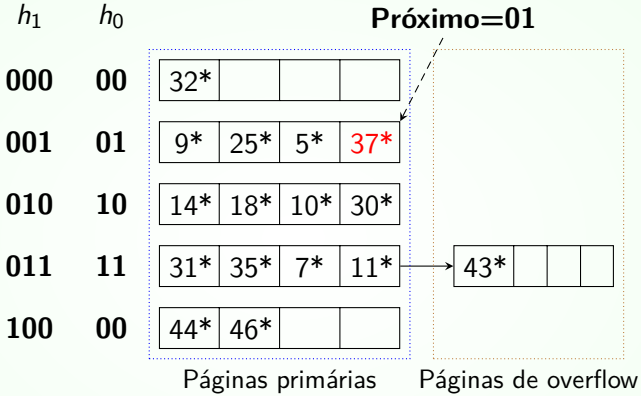
Pesquisa



- Primeiro usamos a função $h_{\text{nível}}$
- Se **próximo** $\leq h_{\text{nível}} < N_{\text{nível}}$, já encontramos o bucket. Exemplo
  $h_0(18) = 10(2)$,
  próximo $\leq 10 < 100$, (bucket 10)
- Caso contrário, aplicamos $h_{\text{nível}+1}$. Exemplo:
  $h_0(32) = 00$, $00 <$ **próximo**,
  $h_1(32) = 000$
  $h_0(44) = 00$, $00 <$ **próximo**,
  $h_1(44) = 100$

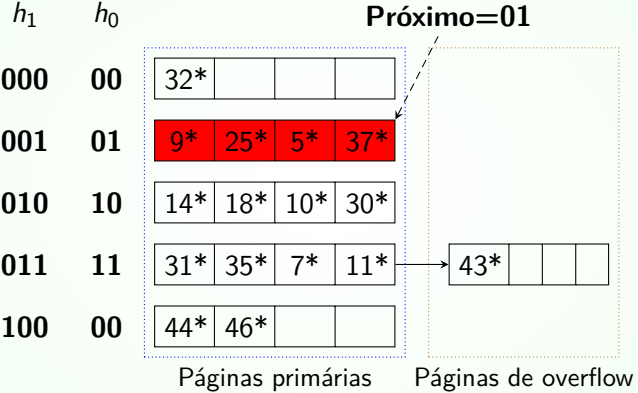
Inserção sem overflow

- Nem sempre ocorre divisão, por exemplo, na inserção da entrada 37*



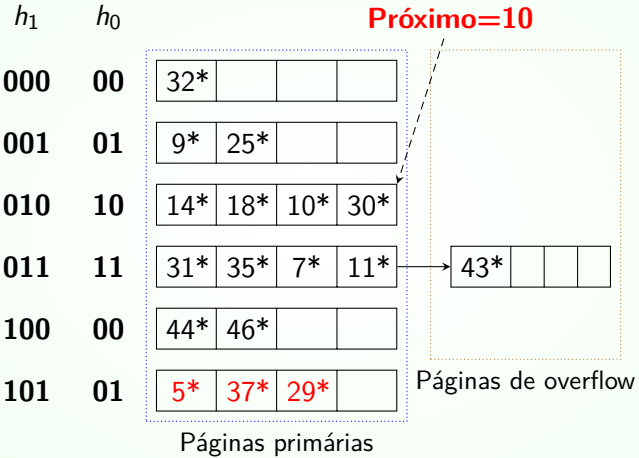
Inserção com divisão e sem overflow

- Se o bucket que causar overflow for aquele apontado pelo **próximo**, dividimos o bucket e evitamos este overflow
- Exemplo: inserir a entrada 29*, $h_0(29) = 01$ (está cheio e é o próximo)



Inserção com divisão e sem overflow

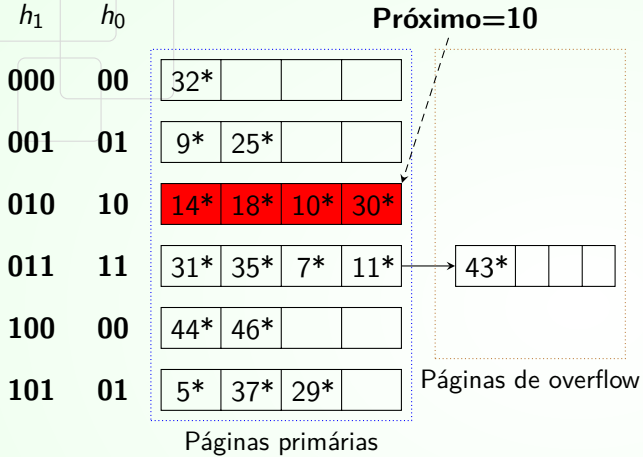
- Se o bucket que causar overflow for aquele apontado pelo **próximo**, dividimos o bucket e evitamos este overflow
- Exemplo: inserir a entrada 29*, $h_0(29) = 01$ (está cheio e é o próximo)



Exercícios

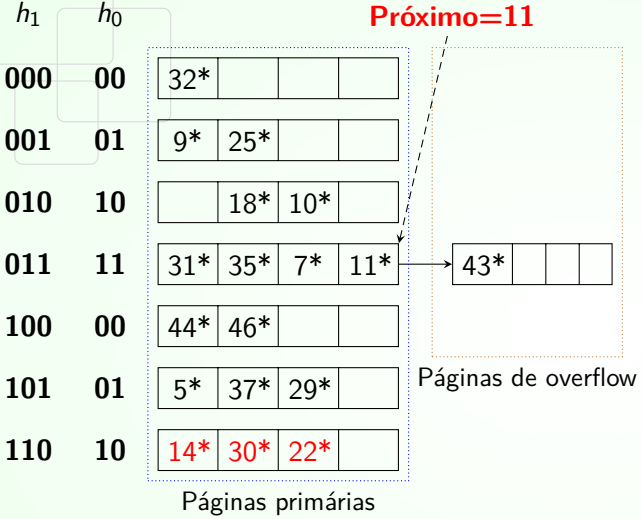
- Inserir as entradas 22*, 66* e 34*

Inserção da entrada 22*



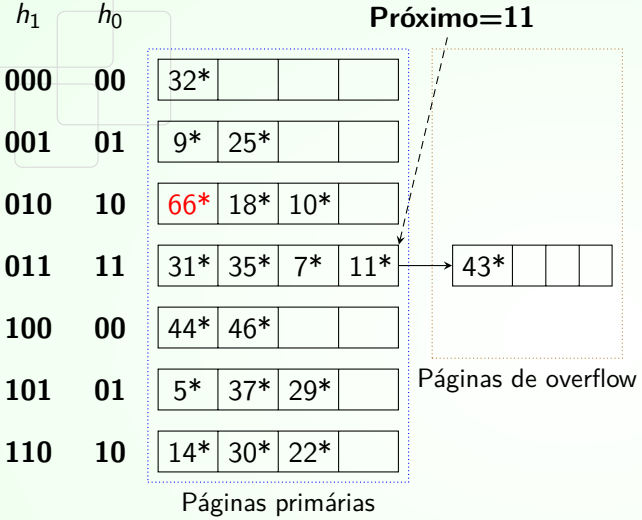
- $h_0(22) = 10$ (está cheio e é o próximo)

Inserção da entrada 22*



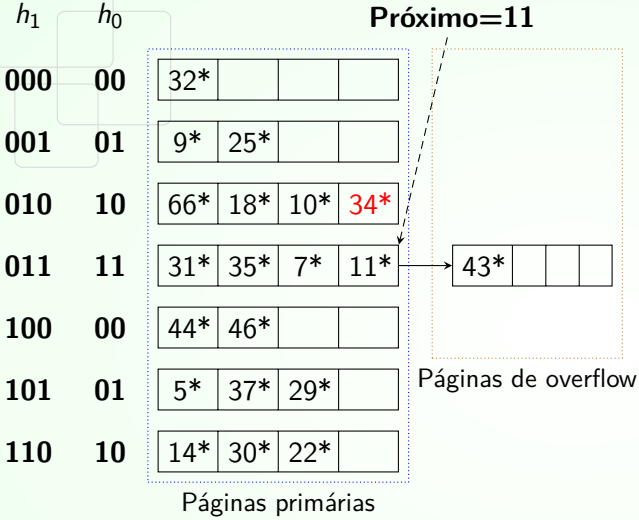
- $h_0(22) = 10$ (está cheio e é o próximo)

Inserção da entrada 66*



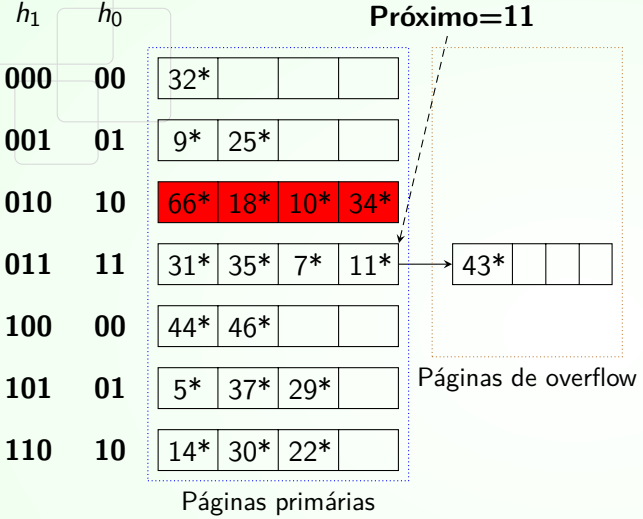
- $h_0(66) = 10$, como $01 <$ **próximo**, aplicamos h_1
- $h_1(66) = 010$

Inserção da entrada 34*



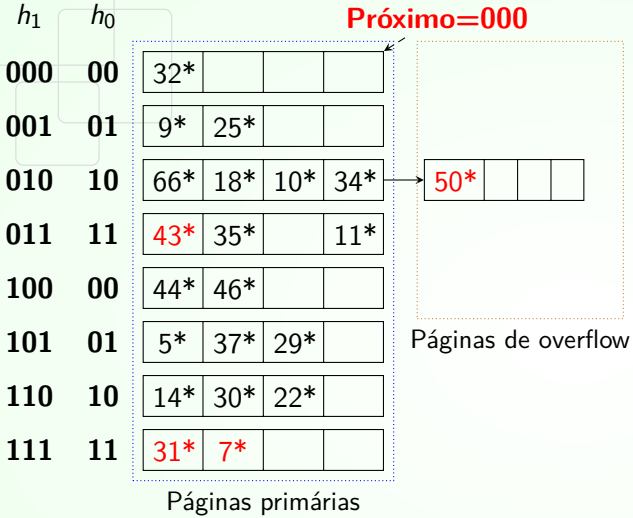
- $h_0(34) = 10$, como $01 <$ **próximo**, aplicamos h_1
- $h_1(34) = 010$

Início de nova rodada



- Vamos inserir a entrada 50*, $h_0(50) = 10$, como $01 <$ **próximo**, aplicamos h_1
- $h_1(50) = 010$, mas bucket 010 está cheio, precisamos de overflow
- O **próximo** é último bucket do nível 0, passamos então para o próximo nível e mudamos o **próximo** para o primeiro bucket novamente

Início de nova rodada



- Vamos inserir a entrada 50*, $h_0(50) = 10$, como $01 <$ **próximo**, aplicamos h_1
- $h_1(50) = 010$, mas bucket 010 está cheio, precisamos de overflow
- O **próximo** é último bucket do nível 0, passamos então para o próximo nível e mudamos o **próximo** para o primeiro bucket novamente

Referências

DATE, C. J. **Introdução a sistemas de bancos de dados**. Rio de Janeiro: Elsevier, 2004.

ELMASRI, R.; NAVATHE, S. B. **Sistemas de banco de dados**. 6. ed. São Paulo: Pearson Addison Wesley, 2011.

RAMAKRISHNAN, R.; GEHRKE, J. **Sistemas de gerenciamento de banco de dados**. 3. ed. São Paulo: McGrawHill, 2008.

SILBERSCHATZ, A.; KORTH, H. F.; SUDARSHAN, S. **Sistema de bancos de dados**. 3. ed. São Paulo: Campus, 2007.