SCC 503 - ALGORITMOS E ESTRUTURAS DE DADOS II

Hashing em arquivos

Prof.: Leonardo Tórtoro Pereira
leonardop@usp.br

Monitor.: Henrique Gomes Zanin
henrique.zanin@usp.br

Índice Hash

- A ideia por trás do hashing é oferecer uma função h, chamada função de hash ou função de randomização, que é aplicada ao valor do campo chave de um registro e gera o endereço do bloco de disco no qual o registro está armazenado.
- Ou seja, funciona de maneira análoga ao *hash* em RAM

Problema: colisões

➔ Como tratar?

Abordagens convencionais

- ➔ Hashing aberto: Partindo da posição ocupada indicada pelo endereço de hash, o programa verifica as posições subsequentes em ordem, até que uma posição não usada (vazia) seja encontrada.
- ➔ Hashing fechado: Utiliza-se de uma lista ligada para lidar com conflitos nas posições
- ➔ Hashing múltiplo: O programa aplica uma segunda função de *hash* se a primeira resultar em uma colisão. Se houver outra colisão, o programa utiliza o endereçamento aberto ou aplica uma terceira função de *hash* e, depois, usa o endereçamento aberto, se necessário.

Diferenças

- Hashing interno: Hashing operado em RAM
- Hashing externo: O hashing para arquivos de disco
  - O espaço de endereços de destino é feito em buckets, cada bucket acomoda um conjunto de registros.
  - Um bucket é um bloco de disco ou um cluster de blocos de disco contíguos (em princípio, acessível com um *seek*).
  - A função de hashing mapeia uma chave em um número de bucket relativo
  - Uma tabela mantida no cabeçalho do arquivo converte o número do bucket relativo para o endereço do bloco de disco correspondente

Hashing externo estático

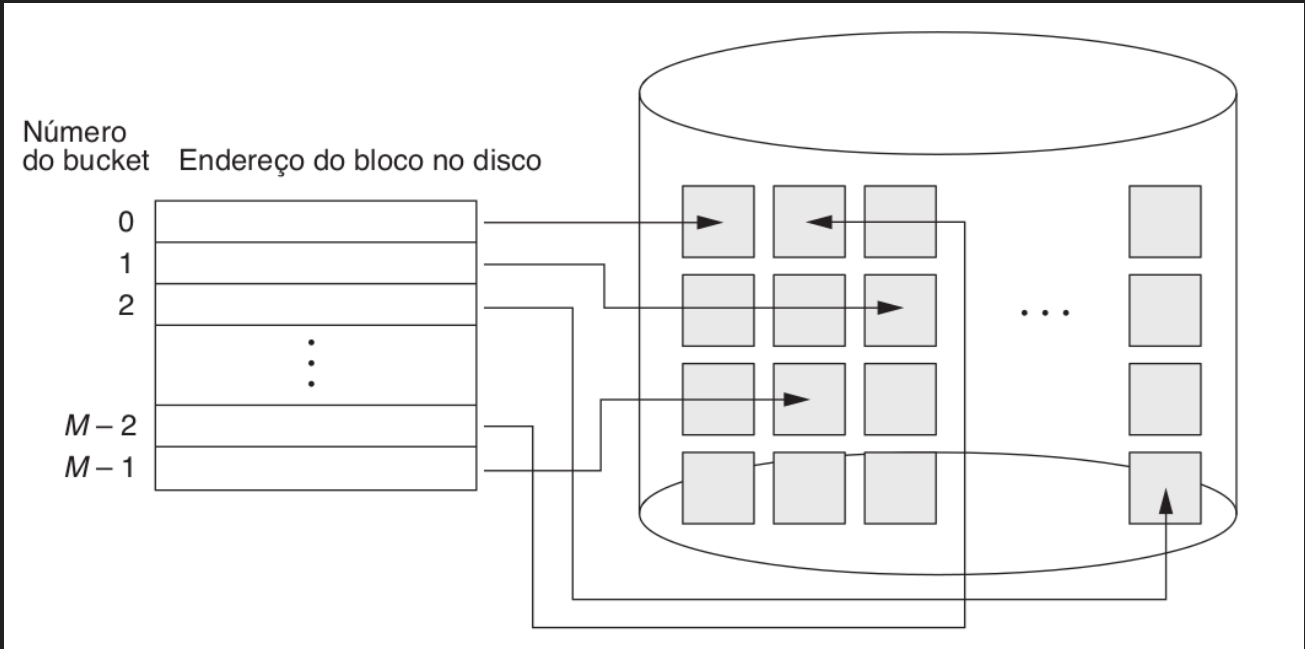


Figura extraída de: Elmasri, R., et al. *Fundamentals of Database Systems*

Hashing externo estático

➔ O hashing estático envolve o uso de um número fixo de *buckets*, definido na criação do índice.

- Isso é um problema para arquivos dinâmicos, será inviável expandir o domínio do *hash*.
- A escolha de um valor M (tamanho da tabela hash) pequeno demais causaria muitas *colisões*
- Um número grande demais seria um desperdício de espaço

Overflow em hash externo estático

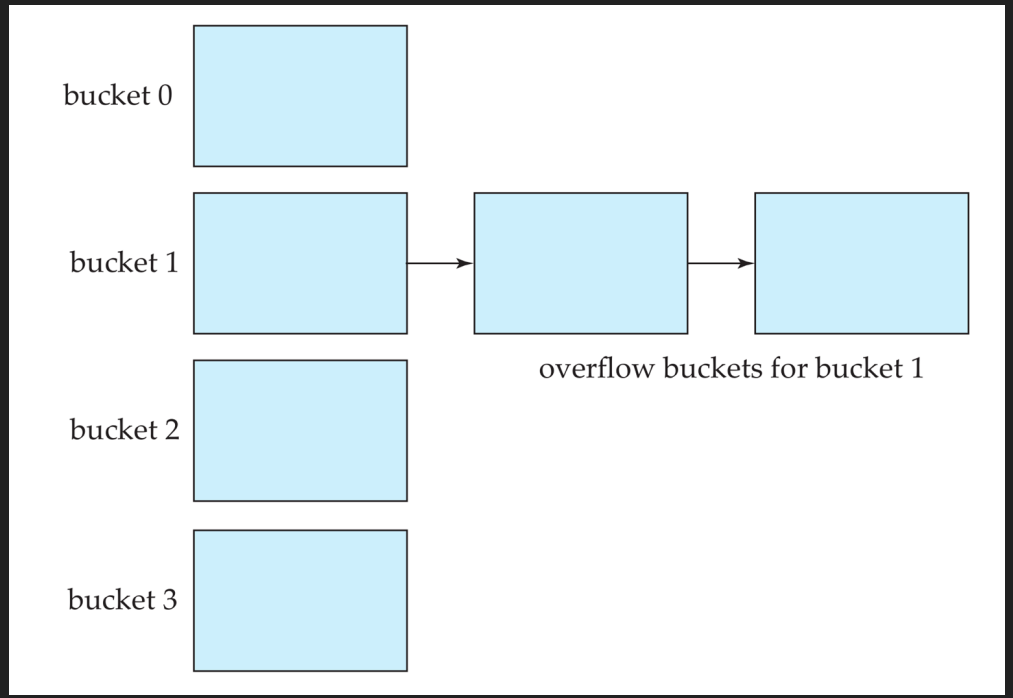


Figura extraída de: Silberschatz, Abraham, Henry F. Korth, and Shashank Sudarshan. *Database system concepts*.

Overflow em hash externo estático

- ➔ O encadeamento excessivo torna a recuperação do registro custosa. Aumenta o número de seeks e de bloqueios do processo
- ➔ Longos encadeamentos em poucos blocos representam uma função hash ruim(não uniforme)
- ➔ Também chamado de overflow chaining

Remoção e busca

- ➔ Buscas de campos diferentes do campo de hash são operadas de modo sequencial
- ➔ A exclusão de registro pode ser implementada pela remoção do registro de seu bucket. Se o bucket tiver uma cadeia de overflow, podemos mover um dos registros de overflow para o bucket e substituir o registro excluído.

Observações quanto aos tipos de hashing

- Por que não usar um hashing aberto para evitar a lista encadeada?
  - Hashing aberto é muito usado em compiladores para a construção da tabela de símbolos. Hashing fechado é extremamente útil em banco de dados.
  - MOTIVO: Remoção em hashing aberto é um problema
  - Analise o problema, há muitas inserções e remoções? O hashing aberto é útil em compiladores pois há apenas inserções e busca

Hash indices

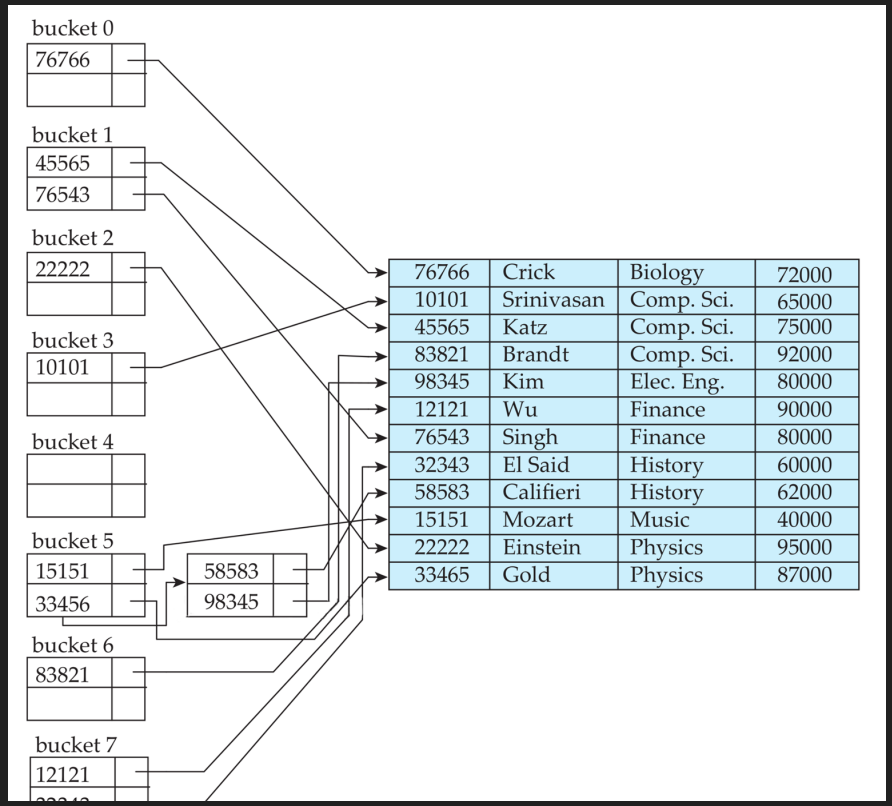


Figura extraída de: Silberschatz, Abraham, Henry F. Korth, and Shashank Sudarshan. *Database system concepts*.

Hashing Extensível

Hashing extensível

- No hashing extensível, um tipo de diretório —um array de 2^d endereços de bucket —é mantido, onde d é chamado de profundidade global do diretório. O valor inteiro correspondente aos primeiros d' bits (ordem alta) de um valor de hash é utilizado como um índice para o array para determinar uma entrada de diretório, e o endereço nessa entrada determina o bucket em que os registros correspondentes são armazenados.

Hashing extensível

Observe que é útil para organizar chaves não únicas

dept_name	h(*dept_name*)
Biology	0010 1101 1111 1011 0010 1100 0011 0000
Comp. Sci.	1111 0001 0010 0100 1001 0011 0110 1101
Elec. Eng.	0100 0011 1010 1100 1100 0110 1101 1111
Finance	1010 0011 1010 0000 1100 0110 1001 1111
History	1100 0111 1110 1101 1011 1111 0011 1010
Music	0011 0101 1010 0110 1100 1001 1110 1011
Physics	1001 1000 0011 1111 1001 1100 0000 0001

Permite encadeamento de buckets em casos de overflow

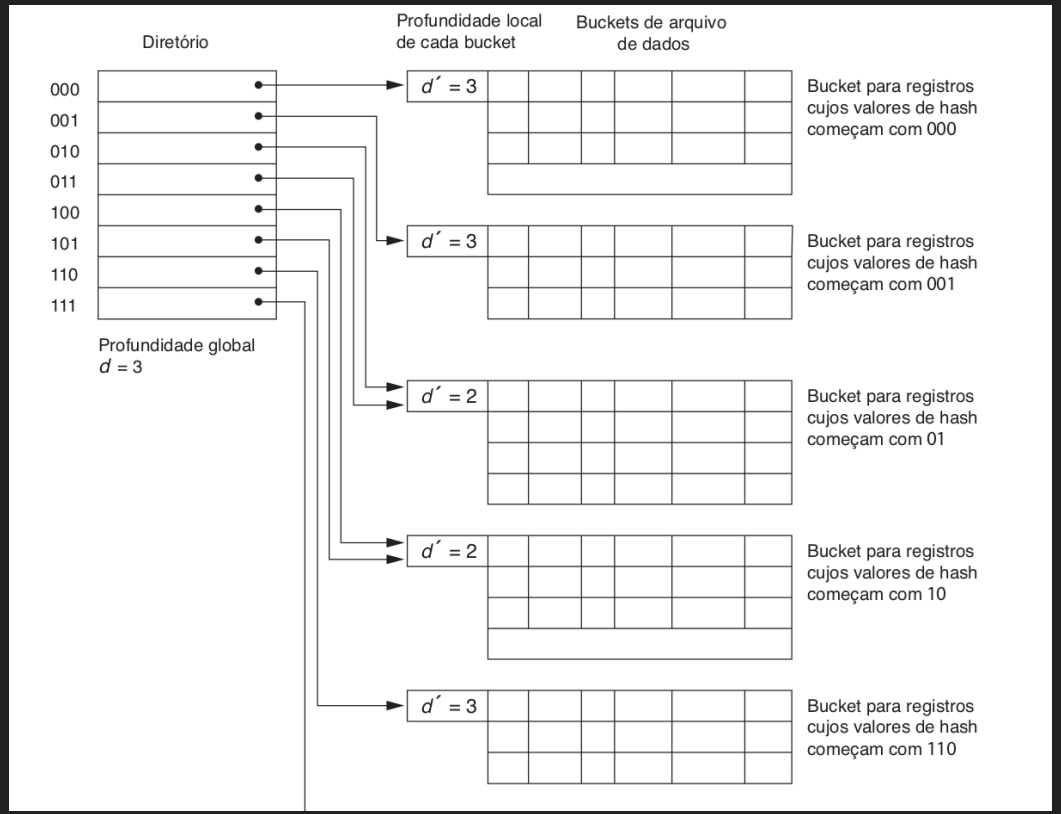


Figura extraída de: Elmasri, R., et al. *Fundamentals of Database Systems*

Hashing Dinâmico

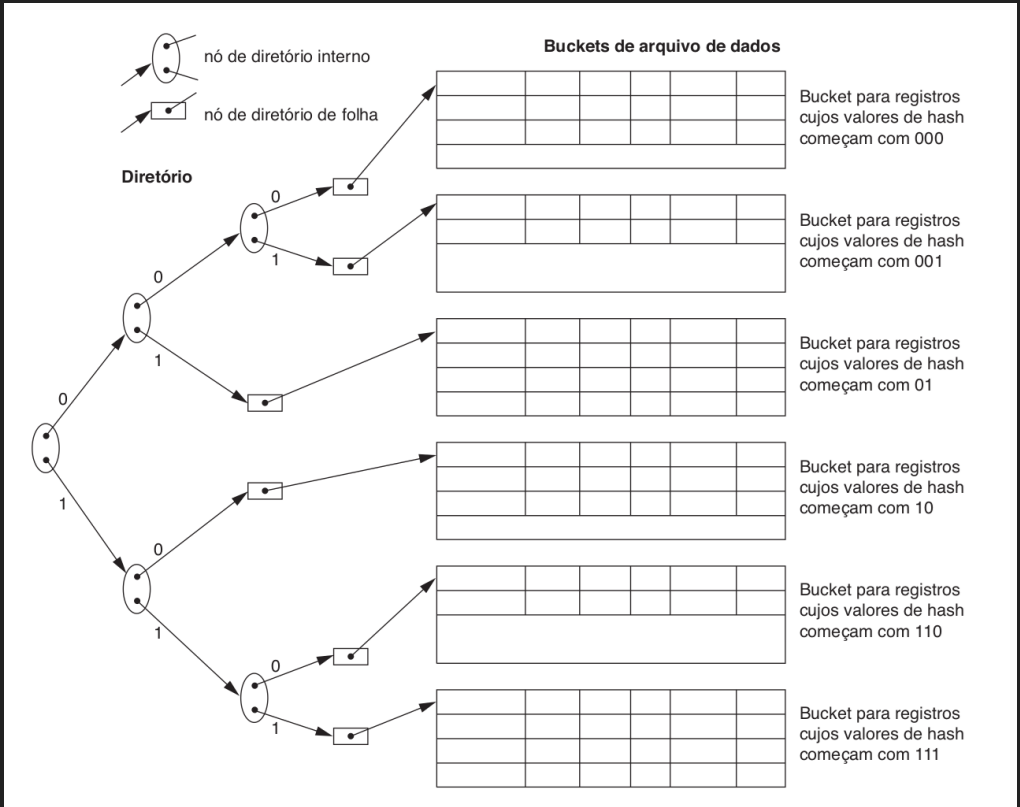


Figura extraída de: Elmasri, R., et al. *Fundamentals of Database Systems*

Hashing estático vs dinâmico

- A principal vantagem do hash extensível é a não degradação da performance quando o arquivo cresce
- Economiza-se espaço por não necessitar reservar buckets para crescimento futuro
- Possui a desvantagem de criar um nível adicional de indireção, já que o deve-se acessar a tabela de endereços de buckets antes de acessar o bucket

Bora fechar arquivos!!!

Mas antes… Uma pequena reflexão

Queries

OBSERVEM:

```
SELECT nUSP, nome, nota FROM alunos WHERE nUSP = 1044;
```

```
SELECT nUSP, nome, nota FROM alunos WHERE nota > 5 and nota < 8;
```

Hash indices

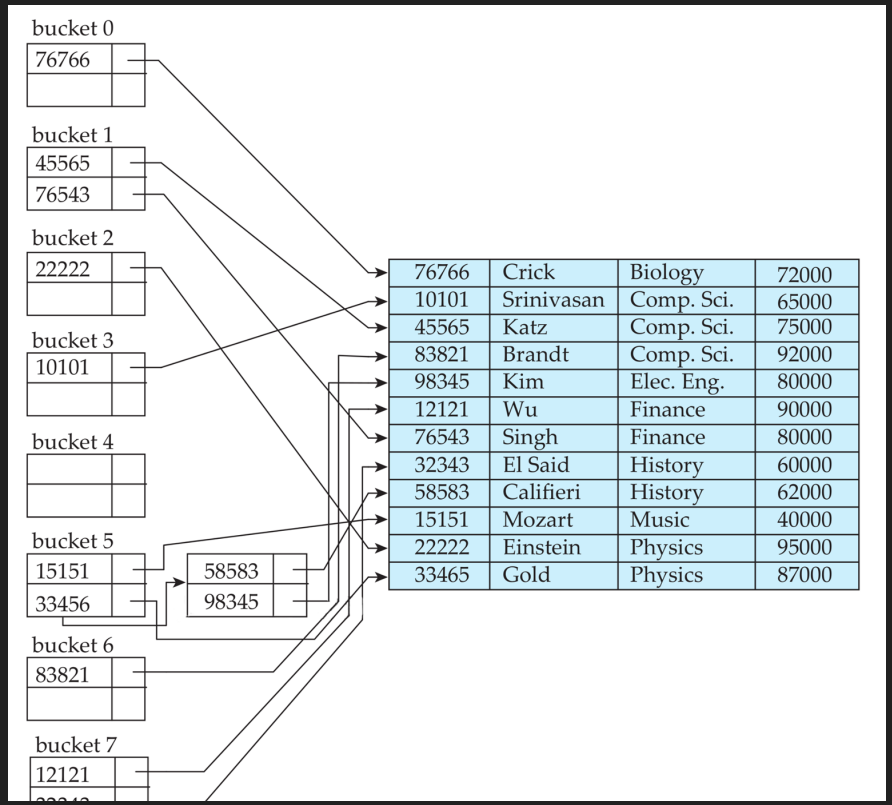
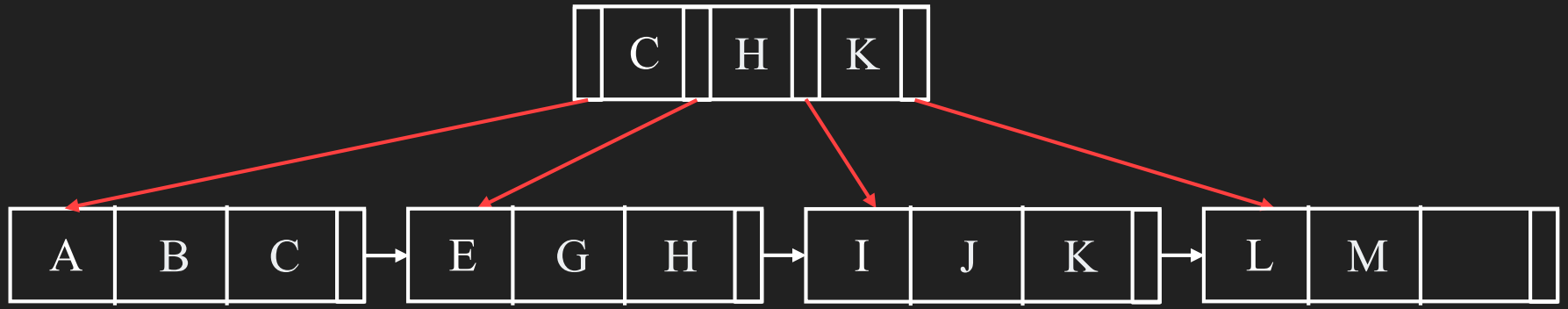


Figura extraída de: Silberschatz, Abraham, Henry F. Korth, and Shashank Sudarshan. *Database system concepts*.

Árvores B+



Referências

- Silberschatz, Abraham, Henry F. Korth, and Shashank Sudarshan. Database system concepts.
- Elmasri, R., et al. Fundamentals of Database Systems