Introdução

Organização e Recuperação de Dados

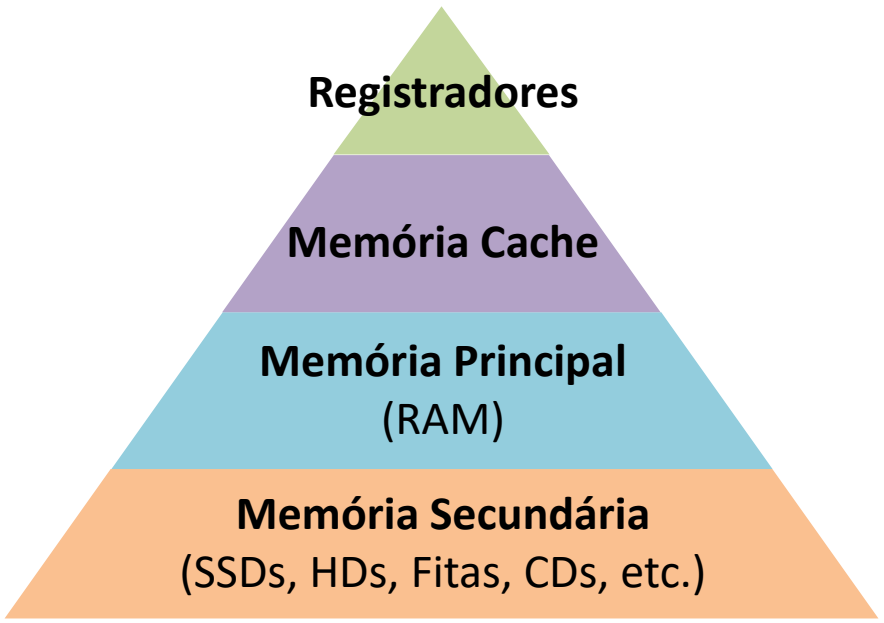
Profa. Valéria

UEM – CTC – DIN

Slides preparados com base no Cap. 1 do livro FOLK, M.J. & ZOELLICK, B. *File Structures*. 2nd Edition, Addison-Wesley Publishing Company, 1992.

Hierarquia de memória

Capacidade de armazenamento



Velocidade de acesso e Custo

Comparativo:

	Memória RAM	Memória secundária
Volatilidade	Volátil	Não volátil
Tamanho	Menor	Maior
Custo	Alto	Baixo
Acesso	Rápido	Lento

Introdução

- Dispositivos de memória secundária são lentos...

Dispositivo	Tempo de Acesso Aproximado	Capacidade
HD SATA	8,5 **ms** (0,0085 **s**)	1 TB
SSD SATA	50 **µs** (0,00005 **s**)	240 GB
Memória RAM DDR3-1333	10,5 **ns** (0,0000000105 **s**)	8 GB

  - A RAM é milhares de vezes mais rápida do que outros dispositivos
    - 1 ms = 1.000 µs = 1.000.000 ns
    - Neste exemplo, a RAM é ~809 mil vezes mais rápida que o HD e ~5 mil vezes mais rápida que o SSD

Analogia

- Imagine que um acesso à RAM fosse equivalente a buscar uma informação no índice de um livro que está em suas mãos e que essa operação consumisse 20 segundos do seu tempo
- O acesso a um dispositivo secundário seria equivalente a buscar a mesma informação em uma biblioteca que levaria dias para retornar a mesma informação
  - HD → ≈ 187 dias (≈ 6 meses)
  - SSD → ≈ 1,15 dias

Introdução

- Apesar de lentos, os discos têm suas vantagens...
  - Armazenam terabytes utilizando pouquíssimo espaço físico
  - São baratos em comparação com a RAM
  - Retêm a informação armazenada mesmo desligados
    - **Característica essencial!**

- Por isso precisamos conhecer estruturas de dados que sejam eficientes para arquivos
  - Arquivos são as estruturas utilizadas para armazenamento de informações em memória secundária

EDs em arquivos

- Um bom projeto de ED permite o acesso eficiente a toda a capacidade do dispositivo sem que as aplicações fiquem esperando demais
- O objetivo da disciplina de ORD é justamente estudar que estruturas são essas
  - **Estudaremos formas clássicas de organização de dados em arquivos que possibilitam sua manutenção e recuperação de forma mais eficiente**

EDs em arquivos

- Os objetivos das EDs para uso em arquivos é **minimizar o tempo de busca** e **maximizar o uso do dispositivo**:
  - O **ideal** é que toda a informação necessária possa ser obtida com apenas **1 acesso** ao disco
    - Não gostaríamos de ter de enviar à biblioteca várias requisições que demoram 187 dias cada uma para retornar!
  - Se esse ideal não puder ser atingido, tentaremos chegar o **mais próximo possível do ideal**
    - Uma busca binária ($O(\log_2 n)$) consegue recuperar uma chave pesquisada entre 50.000 com até 16 comparações, mas acessar um disco 16 vezes para buscar uma informação é muito demorado
    - Precisamos de estruturas que permitam recuperar essa mesma chave em **dois ou três acessos**

EDs em arquivos

  - Além disso, espera-se que uma vez que a chave seja encontrada, toda a informação relativa à chave possa ser recuperada sem acessos adicionais
    - Por ex., se precisamos do título, editora, autores, número de identificação, etc. de um certo livro, é preferível obter toda essa informação de uma só vez, em vez de procurar em outros lugares que levem a novos acessos ao dispositivo
- Esses objetivos seriam relativamente simples de alcançar se a informação contida nos arquivos fosse estática
  - O desafio é manter o desempenho da ED para arquivos que são alterados ao longo do tempo
    - Os dados mudam, aumentam e diminuem a medida em que informações são alteradas, adicionadas ou removidas

Um pouco de história

- Os primeiros projetos de ED para arquivos assumiam que os dados estariam armazenados em fitas
  - Acesso obrigatoriamente sequencial
  - Custo de acesso diretamente proporcional ao tamanho do arquivo
- Rapidamente os arquivos cresceram de modo que o processamento apenas por acesso sequencial tornou-se impraticável
  - Necessidade de acesso direto

Um pouco de história

- Surgiram os discos, que permitiu a associação de índices aos arquivos
  - Possibilidade de acesso direto a uma região especifica do disco (acesso aleatório)
  - Pesquisa no arquivo de índice + acesso direto no arquivo de dados
- Os mesmos problemas dos arquivos sequenciais ocorrem com os índices lineares
  - Quando crescem demais, esses índices se tornam difíceis de manter, principalmente no caso de arquivos dinâmicos, nos quais as chaves mudam o tempo todo
- Daí para frente, o que mudou foi a forma de indexar

Um pouco de história

- No início dos anos 60 surgiu a ideia de usar árvores como índices
  - O problema é que as árvores binárias tendem a crescer de maneira desigual a medida que as chaves são inseridas e removidas (ficam desbalanceadas)
- Em 1963 surgiram as árvores AVL, que mantém o balanceamento mesmo com inserções e remoções
  - Pesquisadores logo pensaram em utilizar algo parecido para arquivos
  - O problema é que mesmo árvores binárias perfeitamente balanceadas exigem muitos acessos para se localizar uma dada chave
  - Era necessária uma estrutura que permitisse armazenar em cada nó da árvore um bloco de registros e que se mantivesse balanceada

Um pouco de história

- No início dos anos 70 foram propostas as árvores-B
  - Bom desempenho de busca ao custo de não poder mais acessar o arquivo sequencialmente de modo eficiente
  - Para garantir o acesso sequencial eficiente, a solução foi adicionar de uma lista encadeada no nível inferior da árvore-B, estrutura que foi batizada de árvore-B^+
  - Em termos práticos, as árvores-B garantem o acesso a um registro mantido em um arquivo com milhões de entradas com apenas três ou quatro acessos ao disco e garantem esse desempenho mesmo após inserções e remoções
- Por anos as árvores-B formaram a base da maioria dos sistemas de arquivos comerciais

Um pouco de história

- Mas o ideal é apenas 1 acesso
- Um *hashing* é uma boa solução se estivermos trabalhando com arquivos que não mudam muito
  - Há muito tempo são empregados para o acesso direto a arquivos estáticos
- No início dos anos 80 foi proposto o *hashing extensível*, estrutura dinâmica que pode ser utilizada com arquivos de forma mais eficiente

Um pouco de história

- A Web trouxe consigo uma nova necessidade de indexação
  - Indexação e recuperação de documentos contendo determinada informação
    - Uso de índices invertidos (similares às listas invertidas usadas com índices secundários)
- Também aumentou a demanda por compartilhamento de arquivos
  - *Uploads* e *downloads*
    - Arquivos grandes geram um tráfego pesado na rede e têm carregamento demorado
  - A solução foi comprimir esses arquivos
    - Algoritmos gerais de compressão são muito anteriores a isso
    - Surgimento de técnicas específicas para a compressão de áudio, imagem e vídeo

Diferentes organizações de arquivos

- Organizações clássicas de arquivos que estudaremos nesta disciplina:
  - Sequencial → acesso obrigatoriamente sequencial
    - É a organização mais simples que se pode ter
  - Indexado → garante o acesso indexado (direto a um registro)
    - Arquivo sequencial + índices lineares ou árvores-B
  - Sequencial indexado → garante tanto o acesso indexado quanto o sequencial
    - Arquivo sequencial em bloco + índice linear ou árvore-B (árvore-B+)
  - Direto → acesso direto a uma posição do arquivo sem a necessidade de um índice
    - Arquivo de *hashing*