

Introdução

6897/9895 – Organização e Recuperação de Dados
Profa. Valéria D. Feltrim

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



Comparativo:

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

Introdução

- Discos são lentos...
- Exemplo comparativo

Dispositivo	Tempo de Acesso Aproximado	Capacidade
Memória RAM DDR3-1333	10,5 ns (0,0000000105 s)	2 GB
HD SATA	8,5 ms (0,0085 s)	1 TB

- A RAM é milhares de vezes mais rápida
 - 1 milissegundo = 1 milhão de nanossegundos
 - Neste exemplo, a RAM é 809 mil vezes mais rápida que o disco
 - Os SSDs têm tempo de acesso na casa dos microssegundos (10-100)

Fazendo uma analogia

- Supondo que o acesso à RAM equivale a buscar uma informação no índice de um livro que está em suas mãos e que essa operação consome 20 segundos do seu tempo, o acesso ao disco seria equivalente a buscar a mesma informação em uma biblioteca que levaria 187,5 dias ($\sim 16.200.000$ segundos ou ~ 6 meses) para retornar a informação desejada

Introdução

- Apesar de lentos, os discos têm suas vantagens...
 - Armazenam terabytes utilizando pouquíssimo espaço físico
 - São mais baratos do que RAM
 - Retêm a informação armazenada mesmo desligados
 - **Característica essencial!**
- Por isso precisamos conhecer estruturas de dados que sejam eficientes para arquivos em disco

EDs em Arquivos

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

Projeto de EDs em arquivos

- Objetivos do projeto de ED em arquivos é **minimizar o tempo de busca e maximizar o uso do disco**:
 - O **ideal** é que toda a informação necessária possa ser obtida com apenas **1 acesso** ao disco
 - Não gostaríamos de ter de encaminhar à biblioteca uma série de requisições que demoram 187 dias cada!
 - Se esse ideal não puder ser atingido, tentaremos chegar o **mais próximo possível do ideal**
 - Uma busca binária ($O(\log_2 n)$) permite encontrar uma chave pesquisada entre 50.000 com no máximo 16 comparações, mas acessar o disco 16 vezes para buscar uma informação é tempo demais...
 - Precisamos de estruturas que permitam recuperar essa mesma chave em **dois ou três acessos**

Projeto de EDs em arquivos

- Além disso, espera-se que uma vez encontrada a chave, toda a informação relativa a essa 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
- Esses objetivos seriam relativamente simples de alcançar se a informação contida nos arquivos fosse estática
 - É mais difícil obter uma estrutura de arquivos que mantêm as qualidades desejadas para arquivos que são alterados ao longo do tempo
 - Os dados mudam, aumentam e diminuem de tamanho 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 por meio de acesso sequencial tornou-se impraticável

Um pouco de história

- Surgiram os discos, que permitiu a associação de índices aos arquivos
 - Possibilidade de acesso direto a uma região específica 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 o arquivo

Um pouco de história

- No início dos anos 60 surgiu a ideia de usar árvores como solução para a estruturação de índices
 - O problema é que as árvores tendem a crescer de maneira desigual a medida que as chaves são inseridas e removidas (ficam desbalanceadas)
- Em 1963 surgiram as árvores AVL, que são ótimas para estruturar dados na RAM
 - Pesquisadores logo pensaram em utilizar algo parecido como estruturas de dados para arquivos
 - O problema é que mesmo árvores binárias perfeitamente balanceadas exigem muitos acessos para se localizar uma dada chave
 - Era necessário encontrar uma estrutura que permitisse armazenar, em cada nó da árvore, um bloco do arquivo contendo o maior número possível de registros e que se mantivesse balanceada

Um pouco de história

- Mais de 10 anos de pesquisas foram necessários para se chegar as árvores-B
 - Excelente desempenho na busca por registros ao custo de não se poder mais acessar o arquivo sequencialmente de modo eficiente
 - A solução encontrada foi a adição de uma lista encadeada no nível inferior da árvore-B, que nesse caso 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 mais de 10 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
- A técnica de hashing é uma boa solução se estivermos trabalhando com arquivos que não mudam muito
 - Há muito tempo índices gerados com *hashing* garantem acesso eficiente a arquivos estáticos
- A técnica conhecida por hashing extensível, que pode ser aplicada a arquivos dinâmicos eficientemente, é relativamente recente (início dos anos 80)

Um pouco de história

- A Web trouxe consigo uma nova necessidade de indexação
 - Indexação de documentos (recuperação de arquivos contendo uma determinada informação)
 - Uso de índices invertidos (similares às listas invertidas usadas com índices secundários)
- A demanda por compartilhamento de arquivos também cresceu com a internet
 - *Uploads e downloads*
 - Arquivos grandes geram um tráfego pesado na rede e têm acesso demorado
 - A solução foi comprimir esses arquivos
 - Técnicas específicas para áudio, imagem e vídeo
 - Veremos alguns algoritmos gerais de compressão

Diferentes organizações de arquivos

- Organizações clássicas de arquivos que estudaremos nesta disciplina são:
 - Sequencial → acesso obrigatoriamente sequencial
 - É a organização mais simples
 - Indexado → garante o acesso indexado
 - 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* estático ou dinâmico