Fragmentação e Reutilização de Espaço

Organização e Recuperação de Dados

Profa. Valéria

UEM – CTC – DIN

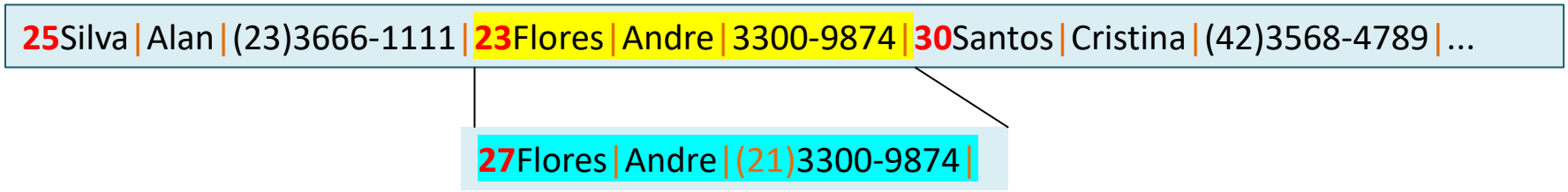
Slides preparados com base no Cap. 5 livro FOLK, M.J. & ZOELLICK, B. *File Structures*. 2nd Edition, Addison-Wesley Publishing Company, 1992, e nos slides disponibilizados pelo Prof. Pedro de Azevedo Berger (DCC/UnB)

Fragmentação em arquivos

- Suponha um arquivo com registros de tamanho variável

25Silva|Alan|(23)3666-1111|23Flores|Andre|3300-9874|30Santos|Cristina|(42)3568-4789|...

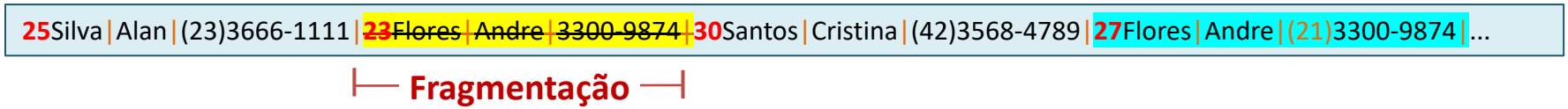
- Suponha também que um dos registros foi modificado e o novo registro é maior do que o original



- O que fazer?
  - Reescrever parte do arquivo para abrir espaço não é uma opção

Fragmentação em arquivos

- A solução mais comum é remover logicamente o registro antigo e gravar o registro novo no fim do arquivo



- O espaço ocupado pelo registro removido é um tipo de fragmentação
  - Arquivos crescem dinamicamente conforme fazemos novas escritas, mas não encolhem com as remoções

Fragmentação em arquivos

- Devido à fragmentação, a organização do arquivo pode se deteriorar a medida que ele vai sendo modificado
  - Como ficaria a busca em um arquivo fragmentado?
- **Fragmentação** em nível de arquivo
  - **Interna**: espaço perdido dentro de um registro
  - **Externa**: espaço perdido fora dos registros
    - Não confundir com fragmentação do disco, que é gerenciada pelo S.O.
- Modificações no arquivo são ocasionadas por:
  - Inserção de novos registros
  - Atualização de registros
  - Remoção de registros

Ações que podem gerar fragmentação

Fragmentação em arquivos

- Situações que geram fragmentação:
  - Atualização de registro de tamanho **variável**
    - Se o novo registro é menor, atualize no mesmo lugar → fragmentação interna
    - Se o novo registro é maior, remova o registro antigo e insira o novo no fim do arquivo → fragmentação externa
  - Remoção de registro de tamanho **fixo ou variável**
    - A remoção é **LÓGICA** → o registro deve ser marcado como removido de alguma forma
      - Insira um caractere especial (por exemplo, '*') no início do registro removido para indicar que ele não é mais válido
      - Reserve um campo adicional para sinalizar a remoção

Reutilização estática

- O que fazer com o espaço ocupado por fragmentação?
- Reutilização **estática (compactação)**
  - De tempos em tempos, recupere todos os espaços de uma só vez
    - Copie os registros válidos para um novo arquivo e libere a arquivo antigo → mais fácil
    - Compactar no mesmo lugar, lendo e regravando apenas os registros válidos → mais demorado, mas requer menos espaço em disco

Arquivo original:

25Silva|Alan|(23)3666-1111|~~**23***lores|Andre|3300-9874~~|**30**Santos|Cristina|(42)3568-4789|**27**Flores|Andre|(21)3300-9874|...

Fragmentação

Arquivo compactado:

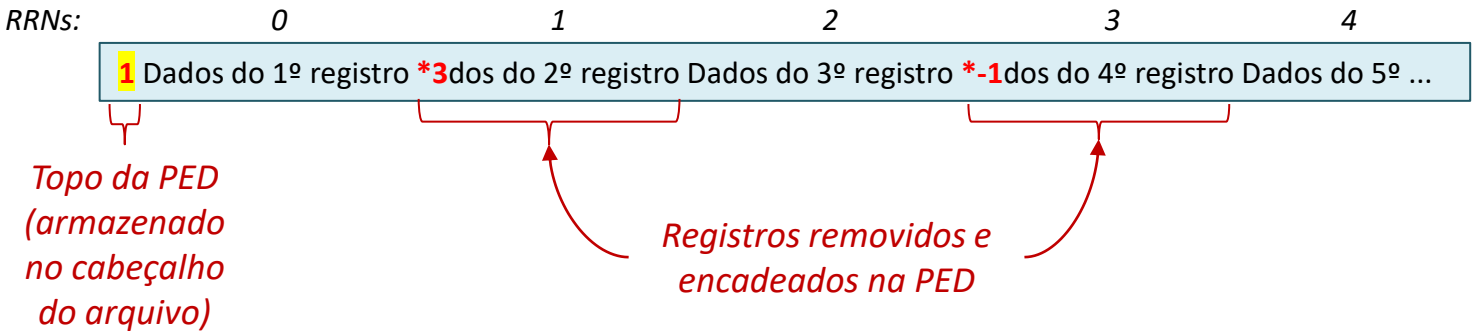
25Silva|Alan|(23)3666-1111|**30**Santos|Cristina|(42)3568-4789|**27**Flores|Andre|(21)3300-9874|...

Reutilização dinâmica

- Reutilização **dinâmica**
  - Utilize os espaços de fragmentação na inserção de novos registros
  - Pode ser usada como forma de retardar a reutilização estática (compactação)
- Para poder reutilizar o espaço de um registro removido de forma rápida, precisamos:
  - Saber rapidamente se existem espaços disponíveis no arquivo
  - Poder “saltar” diretamente para esses espaços, caso existam
  - Para isso, precisamos armazenar o **endereço dos espaços disponíveis**
- A estrutura de dados utilizada para armazenar os endereços disponíveis pode variar dependendo se o arquivo armazena registros de tamanho fixo ou variável

Reutilização dinâmica

- Arquivos com registros de tamanho **fixo**
  - Podemos usar uma **pilha** → **Pilha de Espaços Disponíveis (PED)**
  - Os ponteiros da PED são os RRNs dos registros removidos
- Onde a PED ficará armazenada?
  - No próprio arquivo de registros



Reutilização dinâmica

- Criação e manutenção da **PED**
  - Armazenamos o topo da PED no cabeçalho do arquivo
  - O topo da PED guarda o RRN do último registro removido
  - Se *topo(PED) = -1*, a pilha está vazia (não há registros removidos)
  - Quando um registro é removido, ele é marcado e inserido na PED
    - Gravamos o caractere de remoção no início do 1º campo
    - O RRN do registro será o novo o topo da PED e um ponteiro para o topo antigo (registro removido antes dele) é colocado no espaço que acabou de ser liberado, logo após o caractere de remoção
    - O espaço físico que o registro removido ocupava continua na mesma posição física de antes, mas logicamente passa a integrar a PED

Atenção: os ponteiros da PED são RRNs e não ponteiros de memória

PED

- Após a remoção dos registros de RRN 3 e 5 → Topo(PED) = **5**

	0	1	2	3	4	5	6
5	Nina…	Fred…	Nick…	*-1	Ted…	*3	Julie…

- Após a remoção do registro de RRN 1 → Topo(PED) = **1**

	0	1	2	3	4	5	6
1	Nina…	*5	Nick…	*-1	Ted…	*3	Julie…

- Após a inserção de dois registros novos → Topo(PED) = **3**

	0	1	2	3	4	5	6
3	Nina…	Novo1	Nick…	*-1	Ted…	Novo2	Julie…

Reutilização dinâmica

- Arquivos com registros de tamanho **variável**
  - Tratamento similar, porém utilizamos uma **lista** → **Lista de Espaços Disponíveis (LED)**
  - Os ponteiros da LED são os *byte-offsets* dos registros removidos
  - Não sabemos mais se o 1º espaço da LED será suficientemente grande para armazenar o registro que está sendo inserido
    - Precisaremos buscar na LED por um espaço adequado

LED

- **LED** (**L**ista de **E**spaços **D**isponíveis)

Suponha que cada registro seja precedido por 2 bytes que armazenam o seu tamanho e que **o cabeçalho ocupa os 4 primeiros bytes do arquivo**.

- Arquivo original

Cabeça(LED) → -1

....**40**Ames|John|123 Maple|Stillwater|OK|74075|**64**Morrison|Sebastian|
9035 South Hillcrest|Forest Village|OK|74820|**45**Brown|Martha|625 Kim
bark|Des Moines|IA|50311|

- Após a remoção do 2º registro → *byte-offset* **46**

Cabeça(LED) → **46**

....**40**Ames|John|123 Maple|Stillwater|OK|74075|**64*** **-1**son|Sebastian|
9035 South Hillcrest|Forest Village|OK|74820|**45**Brown|Martha|625 Kim
bark|Des Moines|IA|50311|

Reutilização dinâmica

- Criação e manutenção da LED
  - Armazenamos a cabeça da LED no cabeçalho do arquivo
  - A cabeça da LED guarda o *byte-offset* do último registro removido
  - Quando um registro é removido, ele é marcado e inserido na cabeça da LED
  - Para reutilizar um espaço disponível na inserção de um novo registro:
    - Encontre o 1º espaço na LED tal que |registro| ≤ |espaço|
      - Pode acontecer de se pesquisar a LED inteira e não se achar um espaço adequado (espaço adequado = grande o suficiente)
    - Se um espaço adequado for encontrado, então ele é removido da LED e reutilizado na inserção do novo registro
    - Senão não houver um espaço adequado, o novo registro é inserido no final do arquivo e a LED não é modificada

Exemplo

- **Antes da inserção**

Cabeça(LED) → 46

....40 Ames|John|123 Maple|Stillwater|OK|74075|**64*** **-1**son|Sebastian|
9035 South Hillcrest|Forest Village|OK|74820|**45**Brown|Martha|625 Kim
bark|Des Moines|IA|50311|

- **Após inserção de um registro de <u>27 bytes</u>**

Cabeça (LED) → **-1**

....40 Ames|John|123 Maple|Stillwater|OK|74075|**64Ham|Al|28 Elm|Ada
|OK|70332|.....45**Brown|Martha|625 Kim
bark|Des Moines|IA|50311|

37 bytes de fragmentação interna

- Podemos reduzir a <u>fragmentação interna</u> fazendo com que o espaço que sobra de uma inserção retorne para a LED

Fragmentação interna

- **Antes da inserção**

Cabeça(LED) → **46**

....40 Ames|John|123 Maple|Stillwater|OK|74075|**64*** **-1**son|Sebastian|
9035 South Hillcrest|Forest Village|OK|74820|**45**Brown|Martha|625 Kim
bark|Des Moines|IA|50311|

- **Após inserção de um registro de <u>27 bytes + 2 bytes (tamanho) = 29 bytes</u>**

Cabeça(LED) → **75**

....40 Ames|John|123 Maple|Stillwater|OK|74075|**27Ham|Al|28 Elm|Ada
|OK|70332|35*** **−1**..... **45**Brown|Martha|625 Kim
bark|Des Moines|IA|50311|

Fragmentação interna

- **Antes da inserção**

Cabeça(LED) → **46**

....40 Ames|John|123 Maple|Stillwater|OK|74075|**64*** **-1**son|Sebastian|9035 South Hillcrest|Forest Village|OK|74820|**45**Brown|Martha|625 Kim bark|Des Moines|IA|50311|

De forma alternativa, podemos deixar a sobra no começo do espaço. Assim o offset da sobra será o mesmo do espaço original.

Cabeça(LED) → 46

....40 Ames|John|123 Maple|Stillwater|OK|74075|**35*** **–1**..... ..... **27Ham|Al|28 Elm|Ada|OK|70332|45**Brown|Martha|625 Kim bark|Des Moines|IA|50311|

Fragmentação interna

- **Antes da inserção**

Cabeça(LED) → **46**

....40 Ames|John|123 Maple|Stillwater|OK|74075|**64*** **-1**son|Sebastian|9035 South Hillcrest|Forest Village|OK|74820|**45**Brown|Martha|625 Kim bark|Des Moines|IA|50311|

De forma alternativa, podemos deixar a sobra no começo do espaço. Assim o offset da sobra será o mesmo do espaço original.

Cabeça(LED) → 46

....40 Ames|John|123 Maple|Stillwater|OK|74075|**35*** **–1**..... ..... **27Ham|Al|28 Elm|Ada|OK|70332|45**Brown|Martha|625 Kim bark|Des Moines|IA|50311|

Se o espaço que sobrou for tão pequeno a ponto de não vir a ser utilizado por outro registro, teremos **fragmentação externa**

- O espaço está na LED, mas é muito pequeno para ser reutilizado

Fragmentação externa

- Estratégias para combater a fragmentação externa
  - Gerar um novo arquivo quando a fragmentação ficar intolerável (compactação)
  - Concatenar espaços adjacentes na LED não é viável
    - PROBLEMA: como encontrar os espaços adjacentes?
  - **A melhor opção é minimizar a fragmentação antes que ela ocorra**, adotando uma **estratégia de gerenciamento** da LED
    - Primeiro ajuste
    - Melhor ajuste
    - Pior ajuste

Estratégias de gerenciamento da LED

Estratégias de gerenciamento da LED

- Primeiro ajuste – *"first fit"*
  - É o que vimos nos exemplos até aqui
  - **A LED não é ordenada** → os espaços disponíveis sempre são inseridos na cabeça da LED
  - Na inserção de um novo registro:
    - A LED é percorrida até que |espaço| ≥ |registro| ou até que o final da LED seja atingido
    - Se um espaço adequado for encontrado, o novo registro é escrito
      - **O primeiro espaço grande o suficiente será adequado**
      - O espaço que sobra (grande ou pequeno) pode voltar para a LED
    - Se o final da LED foi atingido, o novo registro é escrito no fim do arquivo

Estratégias de gerenciamento da LED

- Primeiro ajuste – *"first fit"*
  - Vantagem
    - A inclusão de espaços na LED é rápida
  - Desvantagens
    - Na inserção de um novo registro, a LED sempre precisa ser percorrida (muitas vezes sem sucesso)
    - Sobras pequenas demais
      - Sobras muito pequenas tendem a não ser reutilizadas

Estratégias de gerenciamento da LED

- Melhor ajuste – *"best fit"*
  - A LED é mantida ordenada em **ordem crescente do tamanho dos espaços disponíveis**
  - Novos espaços devem ser incluídos de forma ordenada na LED
    - Esse esforço pode ser significativo!
  - Na inserção de um novo registro:
    - A LED é percorrida até que |espaço| ≥ |registro| ou até que o final da LED seja atingido
    - Se um espaço adequado for encontrado, o novo registro é escrito
      - **O espaço encontrado na LED será o menor espaço disponível que é adequado para o novo registro**
      - **A sobra será a menor possível**
      - O espaço que sobra não retorna para a LED
    - Se o final da LED é atingido, o novo registro é escrito no final do arquivo

Estratégias de gerenciamento da LED

- Melhor ajuste – *"best fit"*
  - Vantagem
    - Como é reutilizado o menor espaço da LED que suporte o registro novo, a fragmentação interna será a menor possível
  - Desvantagens
    - Se as sobras forem transformadas em fragmentação externa, elas podem ser pequenas demais → esses espaços se acumularão no início da LED, tornando a busca mais demorada
    - A manutenção da LED é mais lenta devido à inserção ordenada

Estratégias de gerenciamento da LED

- Pior ajuste – *"worst fit"*
  - A LED é mantida ordenada em **ordem decrescente do tamanho dos espaços disponíveis**
  - Novos espaços devem ser incluídos na LED de forma ordenada
  - Na inserção de um novo registro:
    - Reutiliza-se o espaço que está na cabeça da LED, se ele for adequado
      - **O espaço na cabeça da LED será o maior espaço disponível**
      - **A sobra será a maior possível, aumentando as chances de reutilização**
      - O espaço que sobra é reinserido para a LED
    - Se o espaço da cabeça da LED não for adequado, o novo registro é escrito no fim do arquivo

Estratégias de gerenciamento da LED

- Pior ajuste – *"worst fit"*
  - Vantagens
    - A busca na LED é rápida, pois olha-se apenas o elemento da cabeça
      - Se o espaço da cabeça for grande o bastante para o novo registro, nenhum dos outros será
    - As sobras serão os maiores possíveis, aumentando as chances de nova reutilização
  - Desvantagem
    - A manutenção da LED mais lenta devido à inserção ordenada

Reduzindo a fragmentação externa

- Estratégias de gerenciamento só se aplicam arquivo contém registros com tamanho variável

- Qual é a melhor estratégia?
  - Depende da aplicação
  - Se o espaço é perdido por fragmentação interna → Melhor ajuste
  - Se o espaço é perdido por fragmentação externa → Pior ajuste