Fragmentação e Reutilização de Espaço

6897/9895 – Organização e Recuperação de Dados

Profa. Valéria D. Feltrim

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
- Um dos registros foi modificado e o novo registro é maior do que o original
- O que fazer com os dados extra?
  1. Colocar o que sobra no final do arquivo e criar um ponteiro a partir do espaço do registro original para a extensão do registro
  2. Reescrever todo o registro no final do arquivo, deixando um "buraco" na posição do registro original
- Cada solução tem suas desvantagens
  - 👎 Na 1ª solução, o trabalho de processar o registro passa a ser mais complexo e mais lento do que era originalmente
  - 👎 Na 2ª solução, o arquivo desperdiça espaço (fragmentação)

Fragmentação em arquivos

- A organização do arquivo se deteriora a medida que ele vai sendo modificado
  - **Fragmentação** em nível de arquivo
    - **Interna**: espaço desperdiçado dentro do registro
    - **Externa**: espaço desperdiçado fora do registro
      - 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 deteriorar a organização do arquivo

Recuperação de espaços não utilizados

- As questões de manutenção do arquivo se tornam complicadas quando:
  - Atualizamos registro de tamanho **variável**
    - O novo registro é menor → atualizar no mesmo lugar gera desperdício de espaço (fragmentação interna)
    - O novo registro é maior → remove o registro antigo e insere o novo no fim do arquivo (fragmentação externa)
  - Removemos um registro de tamanho **fixo ou variável**
    - A remoção é **LÓGICA**
    - Coloque um caractere especial (por exemplo, * ou $) no início do registro indicando que ele não é mais válido
    - Deixe um campo reservado para sinalizar o *status* do registro

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

Reutilização estática

1. Arquivo original (Registros de tamanho fixo com campos de tamanho variável)

   Ames|John|123 Maple|Stillwater|OK|74075|.....
   Morrison|Sebastian|9035 South Hillcrest|Forest Village|OK|74820|.....
   Brown|Martha|625 Kimbark|Des Moines|IA|50311|.....

2. Remoção do segundo registro

   Ames|John|123 Maple|Stillwater|OK|74075|.....
   *| rrison|Sebastian|9035 South Hillcrest|Forest Village|OK|74820|.....
   Brown|Martha|625 Kimbark|Des Moines|IA|50311|.....

3. Após a compactação

   Ames|John|123 Maple|Stillwater|OK|74075|.....
   Brown|Martha|625 Kimbark|Des Moines|IA|50311|.....

Reutilização dinâmica

- Reutilização **dinâmica**
  - A recuperação dos espaços disponíveis é feita na inserção de novos registros
  - Para reutilizar o espaço de um registro removido de forma rápida, precisamos
    - Saber imediatamente se existem espaços disponíveis no arquivo
    - Uma maneira de "pular" diretamente para esses espaços, caso existam
  - Tratamento diferenciado para registros de tamanho fixo e de tamanho variável

- A reutilização dinâmica pode ser usada como forma de retardar a reutilização estática (compactação)

Reutilização dinâmica

- Registros de tamanho fixo
  - Utilize uma **lista encadeada** contendo todos os registros removidos → **lista de disponíveis**
  - Quando os registros têm tamanho fixo, podemos usar uma **pilha** → **Pilha de Espaços Disponíveis (PED)**
  - Os ponteiros da PED são RRNs
  - Onde a PED fica armazenada?
    - No próprio arquivo de registros

Reutilização dinâmica

- **PED** (**P**ilha de **E**spaços **D**isponíveis)
  - Começamos o arquivo com um cabeçalho, contendo o topo da PED, ou seja, o RRN do último registro removido
  - Se *topo(PED) = -1*, então a pilha está vazia
  - Quando um registro é removido, ele é marcado e inserido na PED
    - Seu RRN vai para o topo da PED e um ponteiro para o registro removido antes dele é colocado no espaço que acabou de ser liberado
    - O espaço físico que o registro removido ocupava continua na mesma posição física de antes, mas logicamente passa a fazer parte da PED

Como os registros estão em disco, **os ponteiros da PED são RRNs** e não ponteiros no sentido tradicional

PED

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

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

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

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

Reutilização dinâmica

- Registros de tamanho variável
  - Tratamento similar, porém não podemos mais usar uma pilha → Lista de Espaços Disponíveis **(LED)**
  - Os ponteiros da LED são *byte-offsets*
  - Não sabemos mais se o 1º espaço da LED será suficiente para armazenar o registro que está sendo inserido
    - Precisamos de um algoritmo que busque na LED um espaço adequado
  - A LED pode ser gerenciada usando diferentes estratégias

LED

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

Suponha que cada registro é precedido por 2 bytes que armazenam o seu tamanho e que o cabeçalho ocupe 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 (*Morrison*) → *byte-offset* **46**

Cabeça(LED) → **46**

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

Reutilização dinâmica

- LED
  - A inserção de um espaço na LED
    - Podemos inserir o espaço liberado na cabeça da LED
  - Na inserção de um novo registro:
    - Busca-se o primeiro 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 o espaço adequado for encontrado, então ele é removido da LED e reutilizado na inserção do novo registro
    - Senão, 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|**ison|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|

Sobra de 37 bytes

- Para reduzir a <u>fragmentação interna</u>, o espaço que sobra pode retornar para a LED

Fragmentação interna

- **Antes da inserção**

Cabeça(LED) → 46

....40 Ames|John|123 Maple|Stillwater|OK|74075|**64*|-1|**ison|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) → 46

....40 Ames|John|123 Maple|Stillwater|OK|74075|**35∗|−1**.....**27Ham|Al|28 Elm|Ada|OK|70332|45**Brown|Martha|625 Kimbark|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 disponível 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?
  - **Minimizar a fragmentação antes que ela ocorra**, adotando uma das seguintes estratégias 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 sempre são inseridos na cabeça da lista
  - 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 foi encontrado, o novo registro é escrito
      - O espaço que sobra pode voltar para a LED
      - **O primeiro espaço grande o suficiente será adequado**
    - 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 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 foi encontrado, o novo registro é escrito
      - O espaço que sobra pode voltar para a LED
      - **O espaço encontrado na LED será o menor espaço disponível que é adequado para o novo registro**
      - **A sobra será a menor possível**
    - 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, o desperdício é o menor possível
  - Desvantagens
    - Sobras pequenas demais → esses espaços se acumularão no início da LED
    - A manutenção da LED mais lenta devido à inserção ordenada

Estratégias de gerenciamento da LED

- Pior ajuste (*worst fit*)
  - **A LED é ordenada em ordem decrescente** de 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 grande o suficiente
    - Se o primeiro espaço da lista for adequado, o novo registro é escrito
      - O espaço que sobra pode voltar para a LED
      - **O espaço encontrado na LED será o maior espaço disponível**
      - **A sobra será a maior possível, aumentando as chances de reutilização**
    - Se o primeiro espaço 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: olha-se apenas o elemento que está na cabeça da LED
      - Se o primeiro elemento não 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

Exercício

Cada registro é precedido por um campo de **2 bytes** que armazena o seu tamanho.
O cabeçalho do arquivo ocupa **4 bytes**.

- Suponha o arquivo abaixo:

```
LED = -1
0   4             34            56             81                115
... 28SOUZA|... |20LOPES|... |23CASTRO|... |32SILVEIRA|... |22ALVES|...
```

- Usando a estratégia *worst fit* para gerenciar a LED, mostre como fica o arquivo depois das seguintes operações, nesta ordem:
  a) Remoção do registro de chave CASTRO
  b) Remoção do registro de chave SOUZA
  c) Remoção do registro de chave LOPES
  d) Remoção do registro de chave ALVES
  e) Inserção do registro de chave BARROS com 36 bytes
  f) Inserção do registro de chave CRUZ com 15 bytes