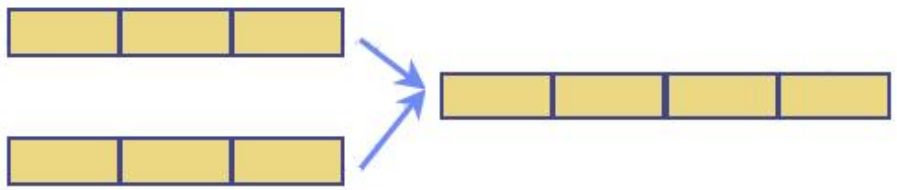
Processos Co sequenciais

Helton Hideraldo Bíscaro

Apresentação

Processos Co sequenciais



- Envolvem o processamento coordenado de duas ou mais listas de entrada seqüenciais, de modo a produzir uma única lista como saída.
- Dois dos principais tipos de resultados de saída são:
  - **matching** (intersecção) de itens duas ou mais listas
  - **merging** (concatenação, ou união) de itens duas ou mais listas
- A operação de união, por exemplo, é a base do processo de ordenação de arquivos muito grandes, cujo índice não cabe na memória principal

Processos Co sequenciais

Exemplos de Aplicações:

- **matching** (intersecção) em um sistema de contas bancárias
  - Arquivo mestre - Contas bancárias: n°conta; nome cliente; saldo $\longrightarrow$ ordenado pelo número da conta
  - Arquivo de transação - Atualizações em contas: n°conta; crédito/débito
  - matching $\longrightarrow$ Atualiza arquivo mestre para cada registro com número da conta que aparece no arquivo de transações
- **merging** (união) $\longrightarrow$ unir duas listas mantendo a ordem alfabética
  - Ordenar arquivos grandes
  - quebrar o arquivo em partes pequena, ordenar cada parte e então fazer a união.

Implementação de Processos Co sequenciais

Dadas duas listas de nomes de pessoas, queremos produzir uma lista com os nomes comuns a ambas, assumindo que cada uma das listas originais não contém nomes repetidos e que está ordenada em ordem crescente.

Lista 1

1. Adriana
2. Carlos
3. Cid
4. Davi
5. Fábio
6. Gabriel
7. Tânia

Lista 2

1. Adriana
2. Anderson
3. André
4. Beatriz
5. Bruno
6. Carlos
7. Davi
8. Deise
9. Fábio
10. Gabriel
11. Gisele

Saída

1. Adriana
2. Carlos
3. Davi
4. Fábio
5. Gabriel

Implementação de Processos Co sequenciais

Lê um nome de cada lista e os compara:

1. Se os nomes são iguais → copiar o nome para a saída e avançar para o próximo nome de cada Lista.
2. Se o nome da Lista1 é menor → avançar na Lista1 (ler próximo nome).
3. Se o nome da Lista1 é maior → avançar na Lista2

Gerenciar as condições:

- Fim de arquivo ;
- Fim da lista → usar High value.

Implementação de Processos Co sequenciais

União de duas listas (merging de nomes em duas listas)

Dadas duas listas de nomes de pessoas, queremos produzir uma lista todos os nomes de ambas (sem repetição), assumindo que cada uma das listas originais está ordenada em ordem crescente.

Lista 1

1. Adriana
2. Carlos
3. Cid
4. Davi
5. Fábio
6. Gabriel
7. Tânia

Lista 2

1. Adriana
2. Anderson
3. André
4. Beatriz
5. Bruno
6. Carlos
7. Davi
8. Deise
9. Fábio
10. Gabriel
11. Gisele

Saída

1. Adriana
2. Anderson
3. André
4. Beatriz
5. Bruno
6. Carlos
7. Cid
8. Davi
9. Deise
10. Fábio
11. Gabriel
12. Gisele
13. Tânia

Implementação de Processos Co sequenciais

Lê um nome de cada lista e os compara:

1. Se os nomes são iguais → copiar o nome para a saída e avançar para o próximo nome de cada lista.
2. Se o nome da Lista1 é menor → copiar o nome da Lista 1 para a saída e avançar na Lista1 (pegar o próximo nome).
3. Se o nome da Lista1 é maior → copiar o nome da Lista2 para a saída e avançar na Lista2

Gerenciar as condições:

- Fim de arquivo ;
- Fim da lista → usar High value.

Implementação de Processos Co sequenciais

Pontos importantes a serem considerados nos algoritmos:

- **Inicialização** $\longrightarrow$ como abrir os arquivos e inicializar as informações para o processo funcionar corretamente;
- **Sincronização** $\longrightarrow$ como avançar adequadamente em cada arquivo;
- Gerenciamento de **condições de fim-de-arquivo e fim das listas**;
- Reconhecimento de **erros** $\longrightarrow$ nomes duplicados ou fora de ordem.

Implementação de Processos Co sequenciais

Pontos importantes a serem considerados nos algoritmos:

- **Inicialização** $\longrightarrow$ como abrir os arquivos e inicializar as informações para o processo funcionar corretamente;
- **Sincronização** $\longrightarrow$ como avançar adequadamente em cada arquivo;
- Gerenciamento de **condições de fim-de-arquivo e fim das listas**;
- Reconhecimento de **erros** $\longrightarrow$ nomes duplicados ou fora de ordem.

Implementação de Processos Co sequenciais

Procedimento de inicialização - função input()

- Abre arquivos de entrada e de saída;
- Seta o flag como TRUE $\rightarrow$ Não existe mais nome;
- Inicializa as variáveis que armazenam o nome anterior (uma para cada lista) com um valor que garantidamente é menor $<$LOW VALUE $>$ que qualquer valor de entrada.
- Isso garante que a rotina de entrada não precise tratar a leitura dos 2 primeiros registros como um caso especial.

Implementação de Processos Co sequenciais

Sincronização: Comentários sobre o algorítmo de interseção

- O procedimento co seqüencial para intersecção é baseado num único loop, que controla a continuação do processo através de um flag;
- Dentro do loop tem-se um comando condicional triplo para testar cada uma das condições acima, e o controle volta ao loop principal a cada passo da operação;
- O corpo do procedimento principal não se preocupa com EOF ou com erros na seqüência de nomes. Para garantir uma lógica clara ao procedimento principal, esta tarefa é deixada para a função Input(), que verifica a ocorrência de fim de arquivo, bem como a ocorrência de nomes duplicados ou fora de ordem

Implementação de Processos Co sequenciais

Gerenciamento de condições de fim-de-arquivo e fim de listas

Fim de Arquivo:

- Match: Quando o fim de um dos arquivos é alcançado → parar o programa
- Merge: Quando o fim de um dos arquivos é alcançado → percorrer o outro arquivo até o fim.

Fim de Lista:

- Usando um **high value** → Armazenamento do <high value> no item corrente para a lista que terminou.

Implementação de Processos Co sequenciais

Algorítmo para Intercalação (merging)

Mudanças no procedimento de inicialização

- mantém o flag em TRUE enquanto existir entradas em qualquer um dos arquivos;
- não fica lendo do arquivo que já terminou → solução: seta NOME (1 ou 2, dependendo do arquivo que terminou antes) para um valor que não pode ocorrer como entrada válida, e que seja maior em valor que qualquer entrada válida **high value**;
- usa também uma variável que indica se a outra lista já terminou.

Implementação de Processos Co sequenciais

Resumo do Modelo Co sequencial → Suposições feitas e comentários

- Dois ou mais arquivos de entrada devem ser processados simultaneamente para produzir um ou mais arquivos de saída. Em alguns casos, o arquivo de saída pode ser igual a um dos arquivos de entrada.
- Cada arquivo é ordenado sobre um ou mais campos chave, e todos os arquivos são ordenados do mesmo modo sobre esses campos. Não é necessário que as estruturas dos registros sejam iguais em todos os arquivos.
- Em alguns casos, deve existir um "valor alto "para a chave, que seja maior do que qualquer chave válida, e um "valor baixo "que seja menor do que qualquer chave válida. A existência desses valores especiais para a chave não é absolutamente necessária, mas ajuda no gerenciamento de condições especiais de início e final de arquivo.

Implementação de Processos Co sequenciais

Resumo do Modelo Co sequencial → Suposições feitas e comentários

- Registros são processados de acordo com a ordem lógica de ordenação. A ordem física não é relevante ao modelo, mas na prática pode ser importante para a implementação; a ordenaçao física pode ter grande impacto na eficiência do processamento.
- Para cada arquivo existe um único registro corrente, cuja chave é aquela disponível no loop de sincronização principal. O modelo não proíbe que se olhe para a frente ou para trás no arquivo, desde que feito fora do loop e sem afetar a estrutura de sincronização.
- Registros podem ser manipulados somente em memória principal. Um programa não pode alterar um registro diretamente no arquivo em disco.

Exemplo: Livro de contabilidade

Problema: projetar um programa para lançar contas em um Livro, como parte de um sistema de contabilidade

- Dois arquivos estão envolvidos nesse processo:
- Arquivo mestre: Livro 1 resumo mensal do balanço contábil para cada livro de contabilidade mantido pelo sistema.
- Arquivo de transação: Livro 2 - contém as transações mensais a serem lançadas no arquivo mestre
- Quando o livro 2 (arquivo de transação) está completo para um dado mês, seu conteúdo deve ser lançado no livro 1
- O lançamento envolve a associação de cada transação com sua conta no arquivo mestre.

Exemplo: Livro de contabilidade

Livro 1

Nº conta	Título da conta	Jan	Fev	Mar
101	Conta verificação 1	1.032,00	2.114,00	5.219,00
102	Conta verificação 2	543,00	3.094,17	1.321,20
510	Gasto veículo	57,00	105,25	138,37
540	Gasto escritório	195,00	307,00	510,00
550	Aluguel	500,00	500,00	500,00

Exemplo: Livro de contabilidade

Livro 2

Nº conta	Nº cheque	descrição	débito/crédito
101	1271	Gasto Veículo	-79,00
510	1271	regulagem reparo	79,00
101	1272	aluguél	-500,00
550	1272	aluguel abril	500,00
102	670	Gasto escritório	-32,00
540	670	Cartucho impressora	32,00
101	1273	Gasto veículo	-31,00
510	1273	Troca de óleo	31,00

Exemplo: Livro de contabilidade

Relatório

101 Conta verificação 1

1271	02/04/2006	Gasto Viagem	-79,00
1272	03/04/2006	Aluguél	500,00
1273	04/05/2006	Gasto Viagem	-31,00
Balanço anterior:	5.129,00	Novo balanço	4.609,00

101 Conta verificação 2

-			
-			
-			
Balanço anterior:		Novo balanço	

Exemplo: Livro de contabilidade

Como implementar o processo de lançamento de contas?

Passos:

- Ordenar o arquivo de transações
- Usar o número da conta como chave para relacionar as transações com os registros do arquivo mestre
- Matching + Merging
  - Se no resultado deve ser listado todas as contas mesmo sem transações no mês
  - matching: contas que não batem constituem um erro
  - Merging: combinar as contas com transação(se houver)
  - Se no resultado deve ser listado apenas as contas com transações no mês
  - fazer matching pela chave entre os dois arquivos
  - depois fazer merging do resultado do match com o arquivo de transações

Processos Co Sequenciais

Extensão do modelo para Intercalação em Múltiplas Vias (Multiway Merging)

- **Intercalação em K-vias** (K-way merge)
  - K arquivos de entrada ordenados são intercalados, gerando um único arquivo de saída.
  - O valor K é a ordem da intercalação.
- Árvore de seleção
  - A intercalação em K-vias funciona bem para K até 8, aproximadamente. Para $K > 8 \rightarrow$ Muito caro $\rightarrow$ usar árvore de seleção.
  - Reduzimos o tempo para encontrar a chave com o menor valor utilizando uma estrutura de dados para guardar a menor chave encontrada em cada loop do processo.

Processos Co Sequenciais

Um algoritmo para intercalação em K-vias

1. decidir qual dos dois nomes disponíveis na entrada tem o menor valor,
2. enviar este nome para o arquivo de saída,
3. avançar no arquivo que possua o menor nome.
4. Caso existam entradas comuns às listas, ambos os arquivos são avançados.
5. Dada uma função min( ), que retorna o menor valor de um conjunto de nomes, não existe razão para restringir o número de arquivos de entrada a 2. O procedimento utilizado pode ser estendido para manipular 3 ou mais listas
6. A parte cara desse processamento é a série de testes para verificar em quais listas o nome ocorreu, de forma a determinar de quais listas os próximos novos valores devem ser lidos. Como o nome pode ter ocorrido em várias listas, cada um dos testes deve ser executado em cada ciclo do loop.

Processos Co Sequenciais

Usando vetores para a implementação:

- das listas podem ser acessadas através de um vetor LIST[i] que contém os nomes de cada lista de dados.
- e os dados (nomes, por exemplo) podem ser acessados através de um vetor de nomes NOME[i], que dá o nome corrente associado a cada lista de entrada.
- O flag é controlada pelo procedimento de entrada