A solução apresentada acima requer um protocolo bem simples e ainda assim permite um nível razoável de concorrência. Essa solução pode ser melhorada em relação ao nível de concorrência com a utilização de protocolos mais sofisticados. Por exemplo, o processo modificador pode fazer uma "reserva" em cada página acessada e mais tarde modificar a reserva para travamento-exclusivo, caso o processo modificador verifique que as modificações a serem realizadas na estrutura da árvore deverão se propagar até a página com reserva. Essa solução aumenta o nível de concorrência, desde que as páginas com reserva possam ser lidas por outros processos.

Os tipos de travamentos referidos aqui são aplicados ao nível físico do banco de dados. Em um banco de dados cujo acesso aos dados é realizado por meio de uma árvore B*, a unidade de transferência dos dados da memória secundária para a memória principal é a página. Desse modo, os protocolos de travamento são aplicados nesse nível.

A implementação dos travamentos descritos acima pode ser obtida usando **semáforos**. De acordo com Dijkstra (1965), um semáforo é um inteiro não negativo que pode ser modificado somente pelas operações *wait* e *signal*, assim descritas: *wait* (s): **when** $s > 0$ **do** $s := s - 1$; e *signal* (s): $s := s + 1$. A operação $s := s + 1$ é indivisível, isto é, somente um processo consegue realizá-la de cada vez. Por exemplo, se dois processos A e B quiserem realizar *signal* (s) ao mesmo tempo para $s = 3$, ao final $s = 5$. Se a operação $s := s + 1$ não é indivisível e as duas operações atribuem o resultado 4 a s, o resultado final pode ser 4 (e não 5). Outra referência sobre semáforos, bem como sua aplicação para sincronizar processos concorrentes, pode ser encontrada em Lister (1975).

Outro importante aspecto a ser considerado em um ambiente de processamento concorrente é o problema de **deadlock**. O *deadlock* ocorre quando dois processos estão inserindo um registro cada um em páginas adjacentes que estejam cheias. Nesse caso, cada um dos processos fica esperando pelo outro eternamente. Lister (1975) mostra que o *deadlock* pode ser evitado pela eliminação de dependências circulares entre processos e recursos. Essa condição pode ser satisfeita com o uso da estrutura em árvore para ordenar todas as solicitações para acessar o banco de dados. Basta que os algoritmos usem as operações de travamento de cima para baixo, isto é, da página raiz para as páginas folha. Bayer e Schkolnick (1977) provaram que as soluções apresentadas são livres de *deadlock*.

6.3.4 Considerações Práticas

A árvore B é simples, de fácil manutenção, eficiente e versátil. A árvore B permite acesso sequencial eficiente, e o custo para recuperar, inserir e retirar registros do arquivo é logarítmico. O espaço utilizado pelos dados é no mínimo 50% do espaço reservado para o arquivo. O espaço utilizado varia com a aquisição e liberação da área utilizada, na medida em que o arquivo cresce ou diminui de tamanho. As árvores B crescem e diminuem automaticamente, e nunca existe necessidade

de uma reorganização completa do banco de dados. O emprego de árvores B em ambientes nos quais o acesso concorrente ao banco de dados é necessário é viável e relativamente simples de ser implementado.

Um bom exemplo de utilização prática de árvores B* é o método de acesso a arquivos da IBM, chamado VSAM (Keehn e Lacy, 1974; Wagner, 1973). VSAM é um método de acesso a arquivos de aplicação geral que permite tanto o acesso sequencial eficiente bem como as operações de inserção, retirada e recuperação em tempo logarítmico. Comparado à organização **sequencial indexado**, o método VSAM oferece as vantagens da **alocação dinâmica** de memória, garantia de utilização de no mínimo 50% da memória reservada ao arquivo e nenhuma necessidade de reorganização periódica de todo o arquivo. O VSAM é considerado uma evolução do antigo ISAM, que utiliza o método sequencial indexado (vide Seção 6.2).

Análise Pelo que foi visto anteriormente, as operações de inserção e retirada de registros em uma árvore B sempre deixam a árvore balanceada. Além do mais, o caminho mais longo em uma árvore B de ordem m com N registros contém no máximo cerca de $log_{m+1}N$ páginas. De fato, Bayer e McCreight (1972) provaram que os limites para as alturas máxima e mínima de uma árvore B de ordem m contendo N registros são:

$$\log_{2m+1}(N+1) \leq altura \leq 1 + \log_{m+1}\left(\frac{N+1}{2}\right).$$

O custo para processar uma operação de recuperação de um registro cresce com o logaritmo base m do tamanho do arquivo. Para ter uma ideia do significado da fórmula acima, considere a Tabela 6.1. Uma árvore B de ordem 50, representando um índice de um arquivo de um milhão de registros, permite a recuperação de qualquer registro com quatro acessos ao disco, no pior caso. Na realidade, o número de acessos no caso médio é três.

Tabela 6.1 Número de acessos a disco, no pior caso, para tamanhos variados de páginas e arquivos usando árvore B

Tamanho da página	Tamanho do arquivo				
	1.000	10.000	100.000	1.000.000	10.000.000
10	3	4	5	6	7
50	2	3	3	4	4
100	2	2	3	3	4
150	2	2	3	3	4

A altura esperada de uma árvore B não é conhecida analiticamente, pois ninguém foi capaz de apresentar um resultado analítico. A partir do cálculo analítico do número esperado de páginas para os quatro primeiros níveis, contados das pá-