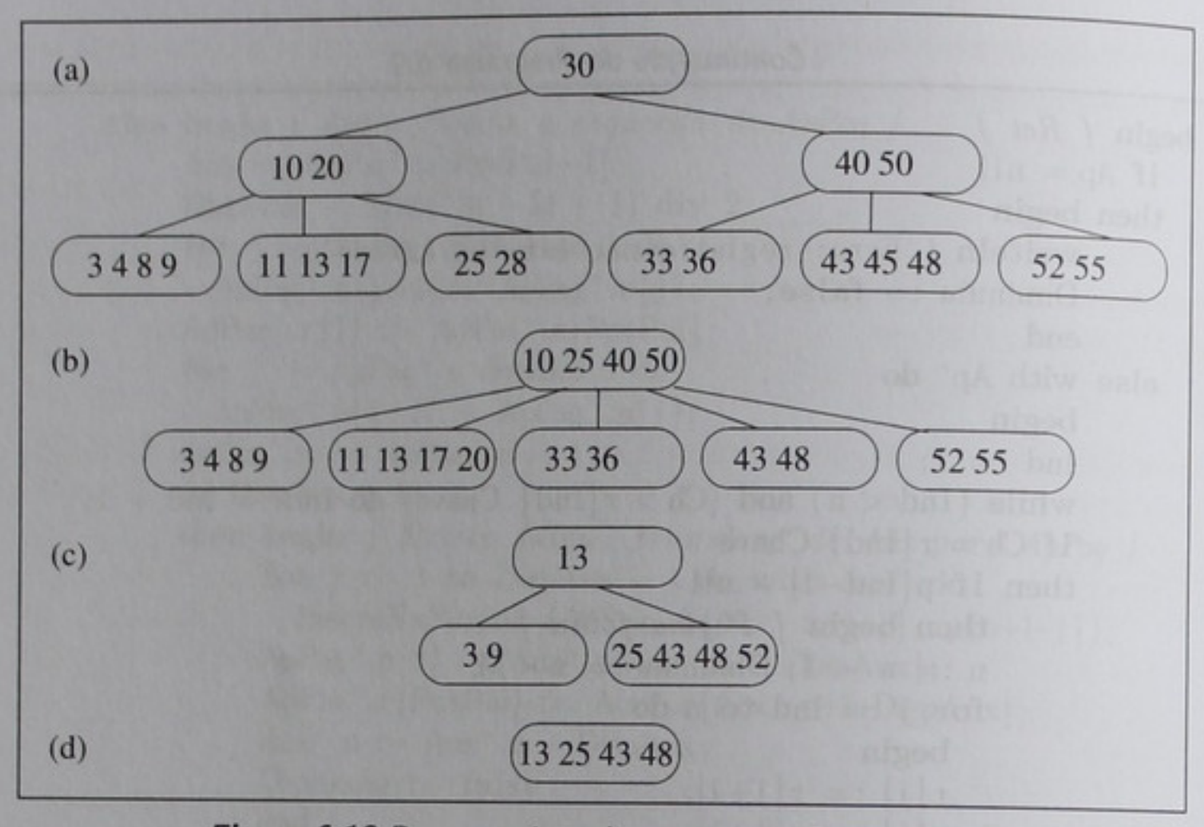

Figura 6.13 *Decomposição de uma árvore B de ordem 2.*

6.3.2 Árvores B*

Existem várias alternativas para implementação da árvore B original. Uma delas é a árvore B*. Em uma **árvore B***, todos os registros são armazenados no último nível (páginas folha). Os níveis acima do último nível constituem um índice cuja organização é a organização de uma árvore B.

A Figura 6.14 mostra a separação lógica entre o índice e os registros que constituem o arquivo propriamente dito. No índice só aparecem as chaves, sem nenhuma informação associada, enquanto nas páginas folha estão todos os registros do arquivo. As páginas folha são conectadas da esquerda para a direita, o que permite um acesso sequencial mais eficiente do que o acesso via índice. Além do acesso sequencial mais eficiente, as árvores B* apresentam outras vantagens sobre as árvores B, como a de facilitar o acesso concorrente ao arquivo, conforme veremos adiante.

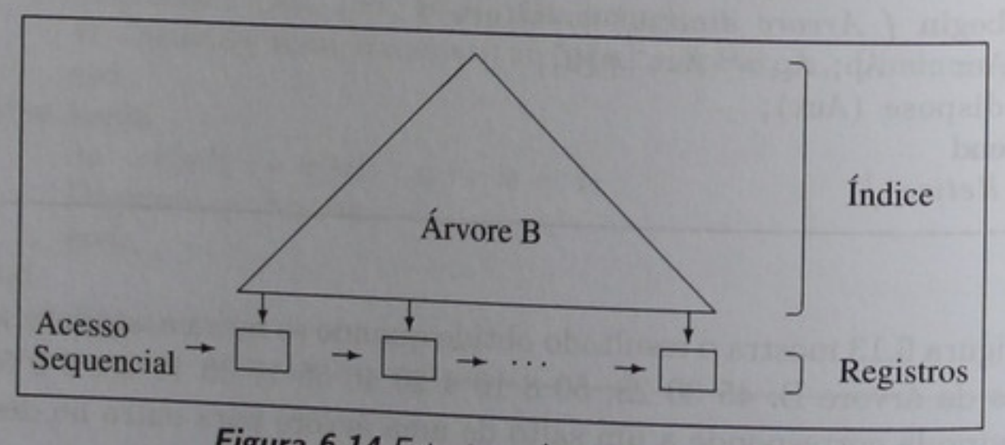

Figura 6.14 *Estrutura de uma árvore B*.*

Para recuperar um registro, o processo de pesquisa inicia-se na raiz da árvore e continua até uma página folha. Como todos os registros residem nas folhas, a pesquisa não para se a chave procurada for encontrada em uma página do índice. Nesse caso, o apontador à direita é seguido até que uma página folha seja encontrada. Esta característica pode ser vista na árvore B* da Figura 6.15, em que as chaves 29, 60 e 75 aparecem no índice e em registros do arquivo. Os valores encontrados ao longo do caminho são irrelevantes desde que eles conduzam à página folha correta.

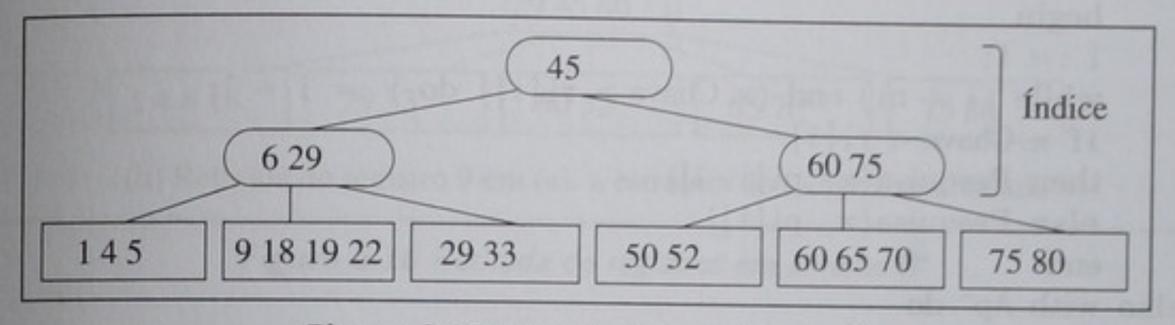

Figura 6.15 *Exemplo de uma árvore B*.*

Como não há necessidade do uso de apontadores nas páginas folha, é possível utilizar esse espaço para armazenar uma quantidade maior de registros em cada página folha. Para isso, devemos utilizar um valor de m diferente para as páginas folha. Isto não cria nenhum problema para os algoritmos de inserção, pois as metades de uma página que está sendo particionada permanecem no mesmo nível da página original antes da partição (algo semelhante acontece com a retirada de registros).

A estrutura de dados árvore B* é apresentada no Programa 6.10.

Programa 6.10 *Estrutura do dicionário para árvore B**

```
type
  TipoRegistro   = record
                     Chave: TipoChave;
                     { outros componentes }
                   end;
  TipoApontador = ^TipoPagina;
  TipoIntExt    = (Interna, Externa);
  TipoPagina    = record
                    case Pt: TipoIntExt of
                      Interna: (ni: 0..mm;
                                ri: array [1..mm] of TipoChave;
                                pi: array [0..mm] of TipoApontador);
                      Externa: (ne: 0..mm2;
                                re: array [1..mm2] of TipoRegistro);
                   end;
  TipoDicionario = TipoApontador;
```