### **一、基本原理**

首先，简单说一下B树产生的原因。B树是一种查找树，我们知道，这一类树（比如二叉查找树，红黑树等等）最初生成的目的都是为了解决某种系统中，查找效率低的问题。B树也是如此，它最初启发于二叉查找树，二叉查找树的特点是每个非叶节点都只有两个孩子节点。然而这种做法会导致当数据量非常大时，二叉查找树的深度过深，搜索算法自根节点向下搜索时，需要访问的节点也就变的相当多。如果这些节点存储在外存储器中，每访问一个节点，相当于就是进行了一次I/O操作，随着树高度的增加，频繁的I/O操作一定会降低查询的效率。

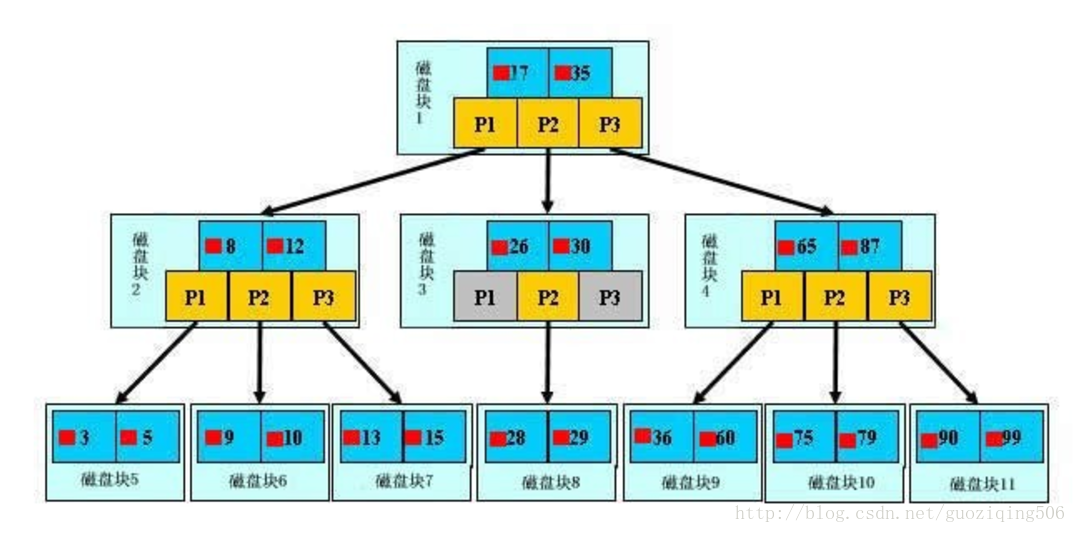
这里有一个基本的概念，就是说我们从外存储器中读取信息的步骤，简单来分，大致有两步：

1. 找到存储这个数据所对应的磁盘页面，这个过程是机械化的过程，需要依靠磁臂的转动，找到对应磁道，所以耗时长。
2. 读取数据进内存，并实施运算，这是电子化的过程，相当快。

综上，对于外存储器的信息读取最大的时间消耗在于寻找磁盘页面。那么一个基本的想法就是能不能减少这种读取的次数，在一个磁盘页面上，多存储一些索引信息。B树的基本逻辑就是这个思路，它要改二叉为多叉，每个节点存储更多的指针信息，以降低I/O操作数。

### **三、搜索算法**

这里，我直接用博客[从B树、B+树、B\*树谈到R 树](http://blog.csdn.net/v_july_v/article/details/6530142" \t "https://blog.csdn.net/guoziqing506/article/details/_blank)中的例子（因为这个例子非常好，也有现成的图示，就直接拿来用，不再自己班门弄斧了），一棵已经建立好的B树如下图所示，我们的目的是查找关键字为29的文件：



先简单对上图说明一下：

图中的小红方块表示对应关键字所代表的文件的存储位置，实际上可以看做是一个地址，比如根节点中17旁边的小红块表示的就是关键字17所对应的文件在硬盘中的存储地址。

P是指针，不用多说了，需要注意的是：指针，关键字，以及关键字所代表的文件地址这三样东西合起来构成了B树的一个节点，这个节点存储在一个磁盘块上

下面，看看搜索关键字的29的文件的过程：

1、从根节点开始，读取根节点信息，根节点有2个关键字：17和35。因为17 < 29 < 35，所以找到指针P2指向的子树，也就是磁盘块3（1次I/0操作）

2、读取当前节点信息，当前节点有2个关键字：26和30。26 < 29 < 30，找到指针P2指向的子树，也就是磁盘块8（2次I/0操作）

3、读取当前节点信息，当前节点有2个关键字：28和29。找到了！（3次I/0操作）

由上面的过程可见，同样的操作，如果使用平衡二叉树，那么需要至少4次I/O操作，B树比之二叉树的这种优势，还会随着节点数的增加而增加。另外，因为B树节点中的关键字都是排序好的，所以，在节点中的信息被读入内存之后，可以采用二分查找这种快速的查找方式，更进一步减少了读入内存之后的计算时间，由此更能说明对于外存数据结构来说，I/O次数是其查找信息中最大的时间消耗，而我们要做的所有努力就是尽量在搜索过程中减少I/O操作的次数。