1、索引需要占用****磁盘空间****，因此在创建索引时要考虑到磁盘空间是否足够

2、创建索引时需要****对表加锁****，因此实际操作中需要在业务空闲期间进行

B树：2-3树和2-3-4树是B树的特例

B树的所有叶子节点都在同一层。

有两个子节点的节点叫二节点，二节点要么有两个子节点，要么没有子节点。

有三个子节点的节点叫三节点。三节点要么有三个子节点，要么没有子节点。

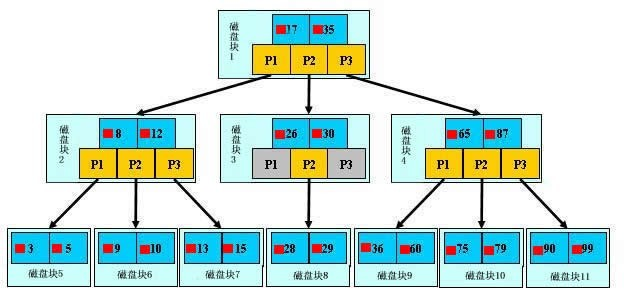
B树最大节点的数字（包含了几个小节点）叫为阶。

B+树非叶子节点只存储索引信息，不存储数据；叶子节点最右边的指针指向下一个相邻的叶节点，所有的叶节点组成了有序链表。

为了提升效率，要尽量减少磁盘IO的次数。实际过程中，磁盘并不是每次严格按需读取，而是每次都会预读。磁盘读取完需要的数据后，会按顺序再多读一部分数据到内存中，预读可以提高I/O效率.预读的长度一般为页(page)的整倍数（4K）,MySQL(默认使用InnoDB引擎),将记录按照页的方式进行管理,每页大小默认为16K(这个值可以修改)。

B+树更适合外部存储(一般指磁盘存储),由于内节点(非叶子节点)不存储data，所以一个节点可以存储更多的内节点，每个节点能索引的范围更大更精确。也就是说使用B+树单次磁盘IO的信息量相比较B树更大，IO效率更高。mysql是关系型数据库，经常会按照区间来访问某个索引列，B+树的叶子节点间按顺序建立了链指针，加强了区间访问性，所以B+树对索引列上的区间范围查询很友好。

**B树是为了磁盘或其他设备而设计的多岔（相对于二岔）平衡查找树。**，所以B树又叫平衡多路查找树。一颗m阶的B树的特性为：  
1：树中每个节点含有最多m个孩子(m > 2)。  
2：除根节点和叶子节点外，其他每个中间节点都至少有[ceil(m/2)]个孩子，ceil为向上取整。（不是很明白。。。）



3：所有叶子节点都出现在同一层，且叶子节点不包含任何关键字信息(可以看做是外部接点或查询失败的接点，实际上这些结点不存在，指向这些结点的指针都为null)  
4：每个非叶子节点包含有n个关键字信息(n，P0，K1，P1，K2，P2，......，Kn，Pn)，其中：  
a) Ki (i=1...n)为关键字，且关键字按顺序升序排序K(i-1)< Ki。  
b) Pi为指向子树根的接点，且指针P(i-1)指向子树种所有结点的关键字均小于Ki，但都大于K(i-1)。  
c) 关键字的个数n必须满足： [ceil(m / 2)-1]<= n <= m-1。  
这三天摘抄自文末参考，大致理解一下就行。

比如，我们通过上面那张btree结构来查找29这个元素，查找过程为：  
（1）根据根节点找到文件目录的跟磁盘块1，将其中的信息装入到内存中【磁盘IO操作第1次】  
（2）此时内存中有两个文件名17,35和三个存储其他磁盘页面地址的数据（指针），根据算法我们发现17 < 29 <35，因此我们找到指针p2  
（3）根据指针p2我们找到磁盘块3，并将其中信息装入到内存中【磁盘IO操作第2次】  
（4）此时内存中有两个文件名26，30和三个存储其他磁盘页面地址的数据（指针），根据算法我们发现26 <29<30，因为我们找到指针p2  
（5）根据指针p2我们定位到磁盘块8，并将其中信息装入内存【磁盘IO操作第3次】  
（6）此时内存中有两个文件名28，29。根据算法我们查找到文件29，并定位了该文件内存的磁盘地址。