### 1、存储引擎比较



注：上面提到的B树索引并没有指出是B-Tree和B+Tree索引，

但是B-树和B+树的定义是有区别的。

在MySQL中，主要有四种类型的索引，分别为：B-Tree索引，Hash索引， Fulltext索引，R-Tree索引。

B-Tree索引是MySQL数据库中使用最为频繁的索引类型，除了Archive存储引擎之外的其他所有的存储引擎都支持B-Tree索引。Archive引擎直到MySQL 5.1才支持索引，而且只支持索引单个AUTO\_INCREMENT列。

不仅仅在MySQL中是如此，实际上在其他的很多数据库管理系统中B-Tree索引也同样是作为最主要的索引类型，这主要是因为B-Tree索引的存储结构在数据库的数据检索中有非常优异的表现。

一般来说，MySQL中的B-Tree索引的物理文件大多都是以Balance Tree的结构来存储的，也就是所有实际需要的数据都存放于Tree的Leaf Node(叶子节点) ，而且到任何一个 Leaf Node 的最短路径的长度都是完全相同的，所以我们大家都称之为 B-Tree 索引。

当然，可能各种数据库（或 MySQL 的各种存储引擎）在存放自己的 B-Tree 索引的时候会对存储结构稍作改造。如 Innodb 存储引擎的 B-Tree 索引实际使用的存储结构实际上是 B+Tree，也就是在 B-Tree 数据结构的基础上做了很小的改造，在每一个Leaf Node 上面出了存放索引键的相关信息之外，还存储了指向与该 Leaf Node 相邻的后一个 LeafNode 的指针信息（增加了顺序访问指针），这主要是为了加快检索多个相邻 Leaf Node 的效率考虑。

InnoDB是Mysql的默认存储引擎(Mysql5.5.5之前是MyISAM）

可能对于没有了解过索引的猿友这样看这篇文章十分吃力，这类猿友有必要先对Mysql索引有个大体的了解。

接下来我们先看看B-树、B+树的概念。弄清楚，为什么加了索引查询速度会加快？

### 2、B-树B+树概念

B树，即二叉搜索树：

1. 所有非叶子结点至多拥有两个儿子（Left和Right）；
2. 所有结点存储一个关键字；
3. 非叶子结点的左指针指向小于其关键字的子树，右指针指向大于其关键字的子树；

如：



B-树，是一种多路搜索树（并不是二叉的）：

1. 定义任意非叶子结点最多只有M个儿子；且M>2；
2. 根结点的儿子数为[2, M]；
3. 除根结点以外的非叶子结点的儿子数为[M/2, M]；
4. 每个结点存放至少M/2-1（取上整）和至多M-1个关键字；（至少2个关键字）
5. 非叶子结点的关键字个数=指向儿子的指针个数-1；
6. 非叶子结点的关键字：K[1], K[2], …, K[M-1]；且K[i] < K[i+1]；
7. 非叶子结点的指针：P[1], P[2], …, P[M]；其中P[1]指向关键字小于K[1]的子树，P[M]指向关键字大于K[M-1]的子树，其它P[i]指向关键字属于(K[i-1], K[i])的子树；
8. 所有叶子结点位于同一层；

如：（M=3）



B-树的搜索，从根结点开始，对结点内的关键字（有序）序列进行二分查找，如果命中则结束，否则进入查询关键字所属范围的儿子结点；重复，直到所对应的儿子指针为空，或已经是叶子结点；

B-树的特性：

1. 关键字集合分布在整颗树中；
2. 任何一个关键字出现且只出现在一个结点中；
3. 搜索有可能在非叶子结点结束；
4. 其搜索性能等价于在关键字全集内做一次二分查找；
5. 自动层次控制；

由于限制了除根结点以外的非叶子结点，至少含有M/2个儿子，确保了结点的至少利用率。

所以B-树的性能总是等价于二分查找（与M值无关），也就没有B树平衡的问题；

由于M/2的限制，在插入结点时，如果结点已满，需要将结点分裂为两个各占M/2的结点；删除结点时，需将两个不足M/2的兄弟结点合并；

B+树，B+树是B-树的变体，也是一种多路搜索树：

1. 其定义基本与B-树同，除了：
2. 非叶子结点的子树指针与关键字个数相同；
3. 非叶子结点的子树指针P[i]，指向关键字值属于[K[i], K[i+1])的子树（B-树是开区间）；
4. 为所有叶子结点增加一个链指针；
5. 所有关键字都在叶子结点出现；

如：（M=3



B+的搜索与B-树也基本相同，区别是B+树只有达到叶子结点才命中（B-树可以在非叶子结点命中），其性能也等价于在关键字全集做一次二分查找；

B+的特性：

1. 所有关键字都出现在叶子结点的链表中（稠密索引），且链表中的关键字恰好是有序的；
2. 不可能在非叶子结点命中；
3. 非叶子结点相当于是叶子结点的索引（稀疏索引），叶子结点相当于是存储（关键字）数据的数据层；
4. 更适合文件索引系统；