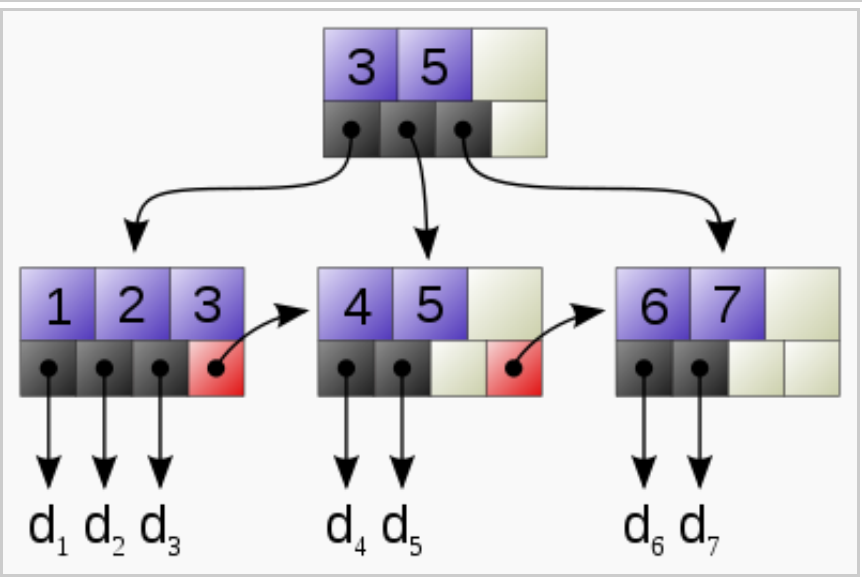
B+树

维基百科，自由的百科全书

B+ 树是一种树数据结构，通常用于数据库和操作系统的文件系统中。B+ 树的特点是能够保持数据稳定有序，其插入与修改拥有较稳定的对数时间复杂度。B+ 树元素自底向上插入，这与二叉树恰好相反。



链接键 1-7 到数据值 d_1-d_7 的简单例子。注意链接列表（红色）允许快速按顺序遍历

B+ 树在节点访问时间远远超过节点内部访问时间的时候，比可作为替代的实现有着实在的优势。这通常在多数节点在次级存储比如硬盘中的时候出现。通过最大化在每个内部节点内的子节点的数目减少树的高度，平衡操作不经常发生，而且效率增加了。这种价值得以确立通常需要每个节点在次级存储中占据完整的磁盘块或近似的大小。

B+ 背后的想法是内部节点可以有在预定范围内的可变数目的子节点。因此，B+ 树不需要象其他自平衡二叉查找树那样经常的重新平衡。对于特定的实现在子节点数目上的低和高边界是固定的。例如，在 2-3 B 树（常简称为2-3 树）中，每个内部节点只可能有 2 或 3 个子节点。如果节点有无效数目的子节点则被当作处于违规状态。

B+ 树的创造者 Rudolf Bayer 没有解释B代表什么。最常见的观点是B代表平衡(balanced)，因为所有的叶子节点在树中都在相同的级别上。B也可能代表Bayer，或者是波音（Boeing），因为他曾经工作于波音科学研究实验室。

目录

节点结构

在 B+ 树中的节点通常被表示为一组有序的元素和子指针。如果此B+树的序数（order）是m ，则除了根之外的每个节点都包含最少 $\lceil m/2 \rceil$ 个元素最多 m-1 个元素，对于任意的节点有最多 m 个子指针。对于所有内部节点，子指针的数目总是比元素的数目多一个。因为所有叶子都在相同的高度上，节点通常不包含确定它们是叶子还是内部节点的方式。

每个内部节点的元素充当分开它的子树的分离值。例如，如果内部节点有三个子节点（或子树）则它必须有两个分离值或元素 a_1 和 a_2。在最左子树中所有的值都小于 a_1，在中间子树中所有的值都在 a_1 和 a_2 之间，而在最右子树中所有的值都大于 a_2。

算法

查找

查找以典型的方式进行，类似于二叉查找树。起始于根节点，自顶向下遍历树，选择其分离值在要查找值的任意一边的子指针。在节点内部典型的使用是二分查找来确定这个位置。

插入

节点要处于违规状态，它必须包含在可接受范围之外数目的元素。

1. 首先，查找要插入其中的节点的位置。接着把值插入这个节点中。
2. 如果没有节点处于违规状态则处理结束。
3. 如果某个节点有过多元素，则把它分裂为两个节点，每个都有最小数目的元素。在树上递归向上继续这个处理直到到达根节点，如果根节点被分裂，则创建一个新根节点。为了使它工作，元素的最小和最大数目典型的必须选择为使最小数不小于最大数的一半。

删除

1. 首先，查找要删除的值。接着从包含它的节点中删除这个值。
2. 如果没有节点处于违规状态则处理结束。
3. 如果节点处于违规状态则有两种可能情况：
    1. 它的兄弟节点，就是同一个父节点的子节点，可以把一个或多个它的子节点转移到当前节点，而把它返回为合法状态。如果是这样，在更改父节点和两个兄弟节点的分离值之后处理结束。
    2. 它的兄弟节点由于处在低边界上而没有额外的子节点。在这种情况下把两个兄弟节点合并到一个单一的节点中，而且我们递归到父节点上，因为它被删除了一个子节点。持续这个处理直到当前节点是合法状态或者到达根节点，在其上根节点的子节点被合并而且合并后的节点成为新的根节点。

注解

假定 L 是节点允许拥有子节点的最小数目，而 U 是最大数目。则每个节点总是有在 L 和 U 之间（包含它们在内）个子节点，除了一个例外：根节点有从2到U（包含它们在内）个子节点。换句话说，根节点豁免于低边界限制，而拥有它自己的低边界2。这允许树持有小数目的元素。根有一个子节点没有意义，因为附着在这个子节点上的子树可以简单的附着在根节点上。允许根节点没有子节点也是不需要的，因为没有元素的树典型的表示为没有根节点。

Robert Tarjan 证明了均摊的分裂 / 合并数目是 2。

参见

- NTFS
- 数据库
- 二叉树
- B# Tree
- B树
- Bitmap index

外部链接

- B+ tree in C language (http://disl.cc.gatech.edu/courses/4420/05Spring/project/Bplustree.c)
- Open Source C++ B+ Tree Implementation (http://www.scalingweb.com/opensource.php)
- B+ tree implementation as C++ template library (http://idlebox.net/2007/stx-btree/)
- Perl implementation of B+ trees (http://search.cpan.org/~hanenkamp/Tree-BPTree-1.07)
- java/C#/python implementations of B+ trees (http://bplusdotnet.sourceforge.net)
- Your Grandmother's guide to Algorithms (http://savutils.sf.net) Java implementation of in-memory B+-Trees and other algorithms.
- Study notes for B+ Trees - Insertion and Deletion (http://www-users.itlabs.umn.edu/classes/Spring-2006/csci4707/B+Trees.pdf)
- Dr. Monge's B+ Tree index notes (http://www.cecs.csulb.edu/%7emonge/classes/share/B+TreeIndexes.html)
- Link to how BTrees work (http://www.virtualmachinery.com/btreeguide.htm)
- Evaluating the performance of CSB+-trees on Mutithreaded Architectures (http://www2.enel.ucalgary.ca/~whassane/papers/CSB+_ccece_2007.pdf)
- Effect of node size on the performance of cache conscious B+-trees (http://www.eecs.umich.edu/~jignesh/quickstep/publ/cci.pdf)
- Fractal Prefetching B+-trees (http://www.pittsburgh.intel-research.net/people/gibbons/papers/fpbptrees.pdf)
- Towards pB+-trees in the field: implementations Choices and performance (http://gemo.futurs.inria.fr/events/EXPDB2006/PAPERS/Jonsson.pdf)
- Cache-Conscious Index Structures for Main-Memory Databases (https://oa.doria.fi/bitstream/handle/10024/2906/cachecon.pdf?sequence=1)
- B+Tree Java SortedMap Implementation (http://ale3hs.aliagas.gr/content/view/23/1/)

来自“http://zh.wikipedia.org/w/index.php?title=B%2B树&oldid=25356276”
