

转载

索引 ----- 二叉树、平衡二叉树、b-tree、b+tree详解

2018年05月03日 13:53:03 qq_36098284 阅读数 4595 [更多](#)

本文是转载+自我梳理

主要讲的是索引中使用树这种数据结构是怎么存储的。以及从二叉树开始的4种树的应用。

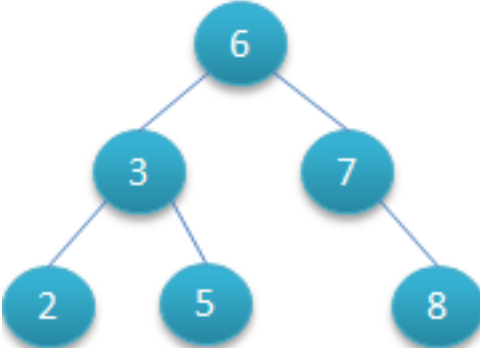
我个人认为参考价值最大的就是b+tree和b-tree都是具体怎么存数据的（相当于优化部分）

B+树索引是B+树在**数据库**中的一种实现，是最常见也是数据库中使用最为频繁的一种索引。B+树中的B代表平衡（balance），而不是二叉（binary），因为从最早的平衡二叉树演化而来的。在讲B+树之前必须先了解二叉查找树、平衡二叉树（AVLTree）和平衡多路查找树（B-Tree），B+树即由这些树逐步优化

二叉查找树

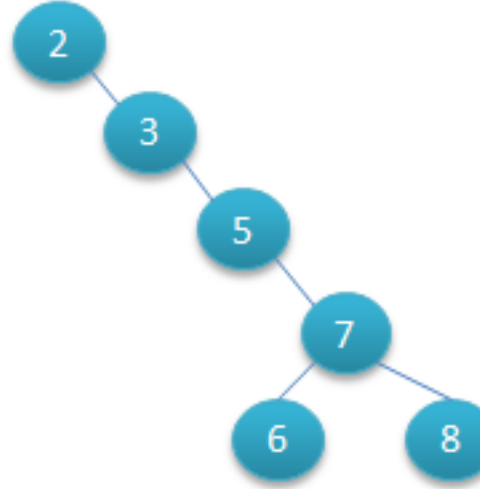
二叉树具有以下性质：左子树的键值小于根的键值，右子树的键值大于根的键值。

如下图所示就是一棵二叉查找树，



对该二叉树的节点进行查找发现深度为1的节点的查找次数为1，深度为2的查找次数为2，深度为n的节点的查找次数为n，因此其平均查找次数为 (1+2+2+3+3+3)/6 = 2.3次

二叉查找树可以任意地构造，同样是2,3,5,6,7,8这六个数字，也可以按照下图的方式来构造：

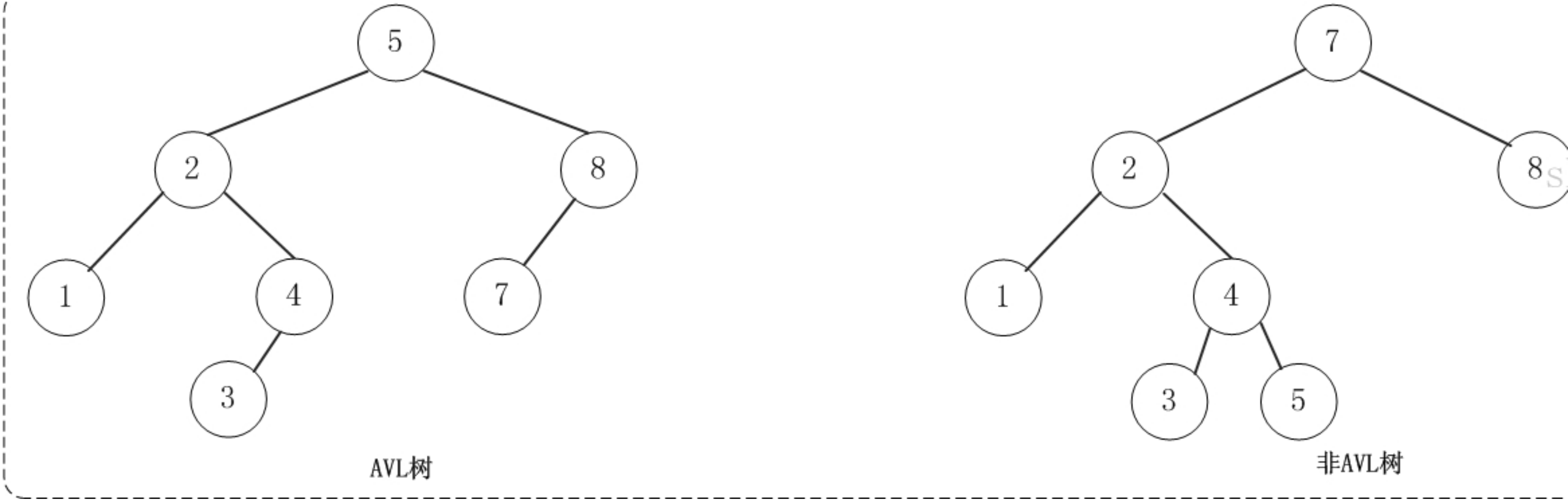


但是这棵二叉树的查询效率就低了。因此若想二叉树的查询效率尽可能高，需要这棵二叉树是平衡的，从而引出新的定义——平衡二叉树，或称AVL树。

2.平衡二叉树（AVL Tree）

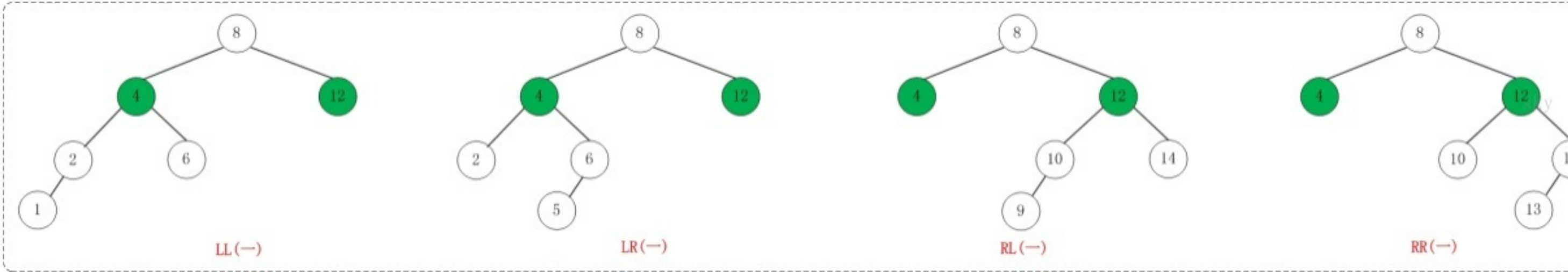
2.1 什么是AVL Tree

平衡二叉树（AVL树）在符合二叉查找树的条件下，还满足任何节点的两个子树的高度最大差为1。下面的两张图片，左边是AVL树，它的任何节点的平衡度差<=1；右边的不是AVL树，其根节点的左子树高度为3，而右子树高度为1；



2.2 平衡二叉树的4种失衡状态

如果在AVL树中进行插入或删除节点，可能导致AVL树失去平衡，这种失去平衡的二叉树可以概括为四种姿态：**LL（左左）、RR（右右）、LR（左右）、RL（右左）**。它们的示意图如下：



这四种失去平衡的姿态都有各自的定义：

LL：LeftLeft，也称“左左”。插入或删除一个节点后，根节点的左孩子（Left Child）的左孩子（Left Child）还有非空节点，导致根节点的左子树高度比右子树高度高2，AVL树失去平衡。

RR：RightRight，也称“右右”。插入或删除一个节点后，根节点的右孩子（Right Child）的右孩子（Right Child）还有非空节点，导致根节点的右子树高度比左子树高度高2，AVL树失去平衡。

LR：LeftRight，也称“左右”。插入或删除一个节点后，根节点的左孩子（Left Child）的右孩子（Right Child）还有非空节点，导致根节点的左子树高度比右子树高度高2，AVL树失去平衡。

RL：RightLeft，也称“右左”。插入或删除一个节点后，根节点的右孩子（Right Child）的左孩子（Left Child）还有非空节点，导致根节点的右子树高度比左子树高度高2，AVL树失去平衡。

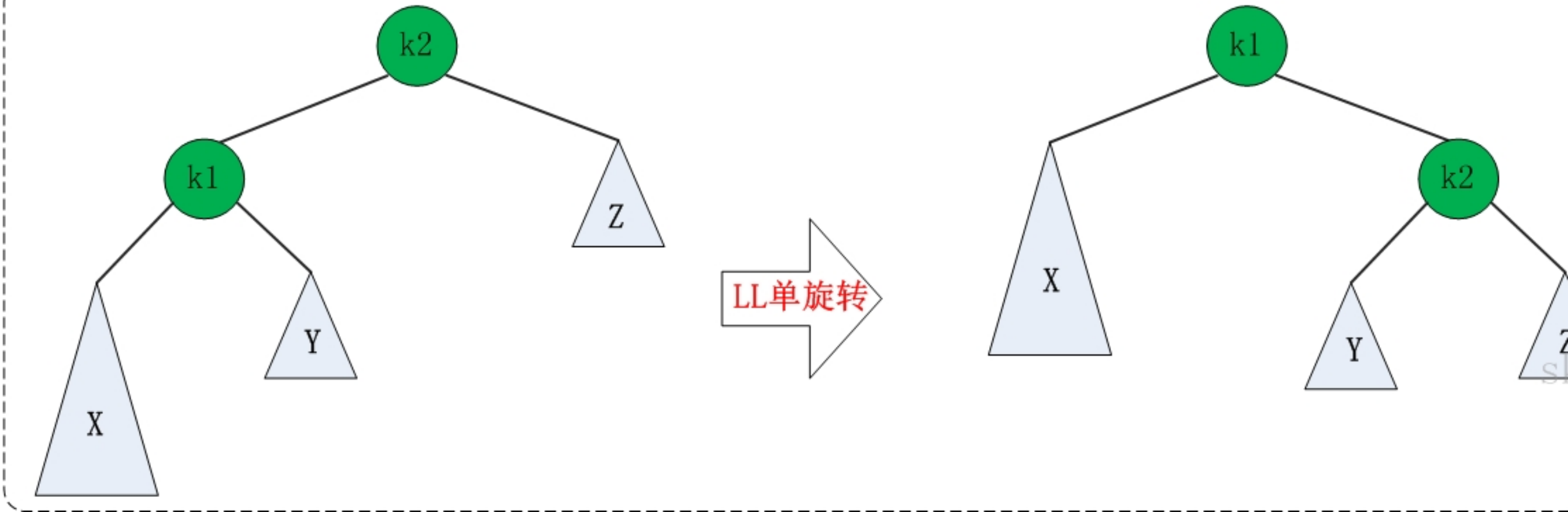
2.3 失衡状态 ----> 平衡状态

AVL树失去平衡之后，可以通过旋转使其恢复平衡。下面分别介绍四种失去平衡的情况下对应的旋转方法。

LL的旋转。LL失去平衡的情况下，可以通过一次旋转让AVL树恢复平衡。步骤如下：

1. 将根节点的左孩子作为新根节点。
2. 将新根节点的右孩子作为原根节点的左孩子。
3. 将原根节点作为新根节点的右孩子。

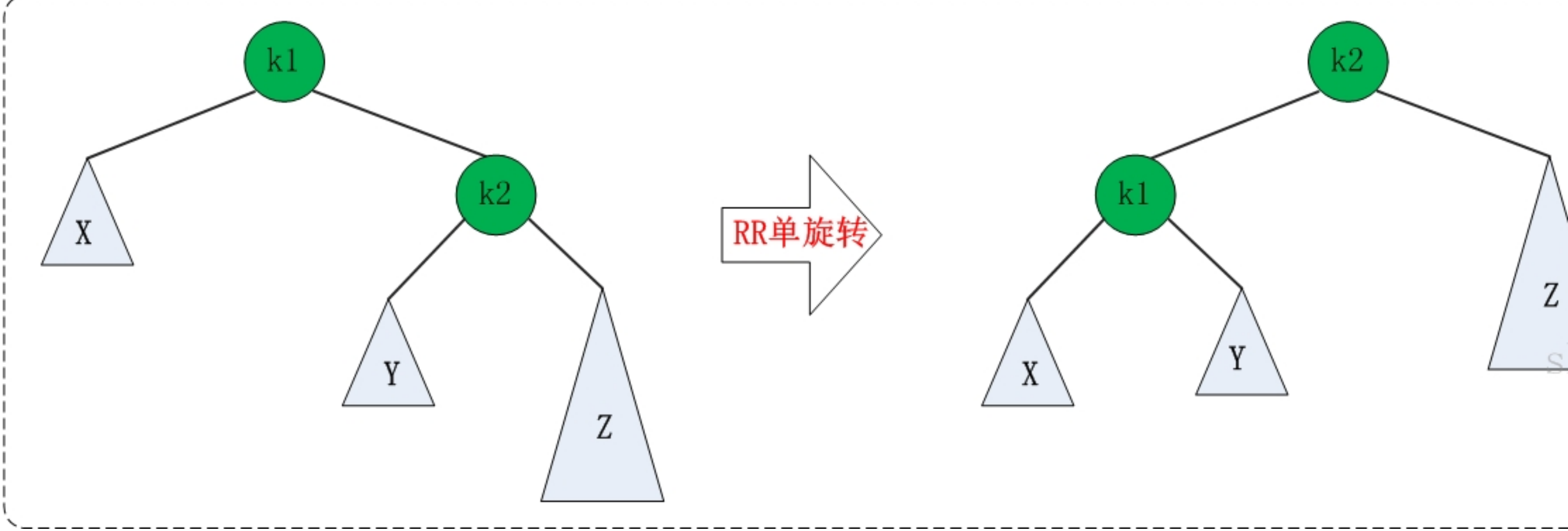
LL旋转示意图如下：



RR的旋转：RR失去平衡的情况下，旋转方法与LL旋转对称，步骤如下：

1. 将根节点的右孩子作为新根节点。
2. 将新根节点的左孩子作为原根节点的右孩子。
3. 将原根节点作为新根节点的左孩子。

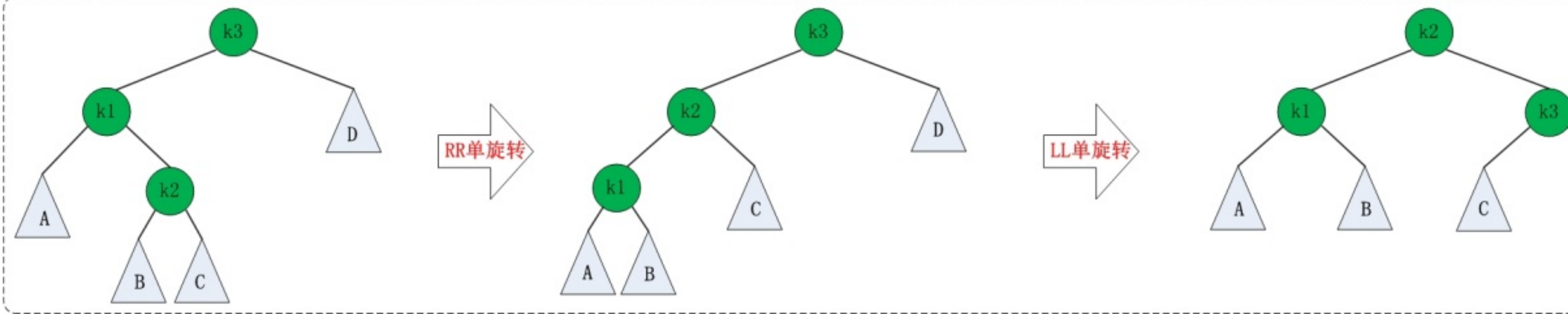
RR旋转示意图如下：



LR的旋转：LR失去平衡的情况下，需要进行两次旋转，步骤如下：

1. 围绕根节点的左孩子进行RR旋转。
2. 围绕根节点进行LL旋转。

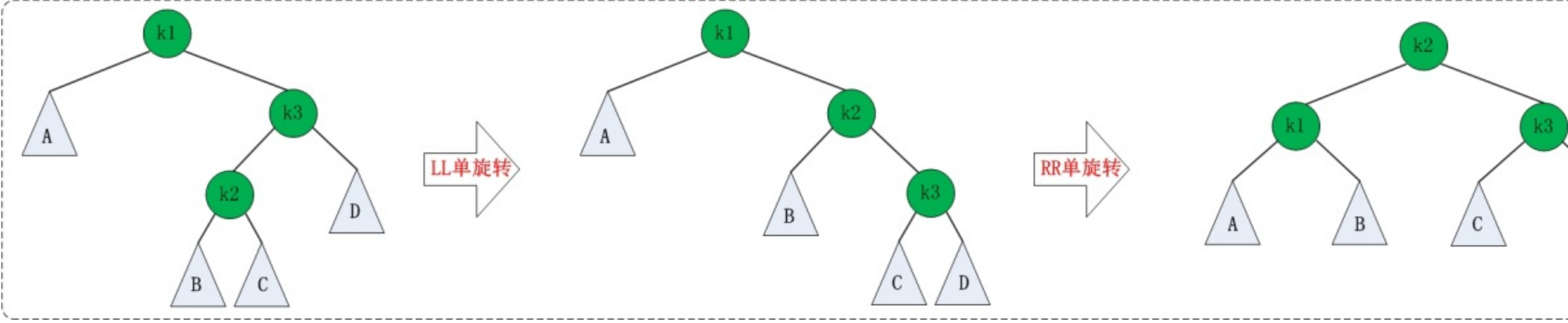
LR的旋转示意图如下：



RL的旋转：RL失去平衡的情况下也需要进行两次旋转，旋转方法与LR旋转对称，步骤如下：

1. 围绕根节点的右孩子进行LL旋转。
2. 围绕根节点进行RR旋转。

RL的旋转示意图如下：



3.平衡多路查找树（B-Tree）

3.1什么是B-Tree

B-Tree是为磁盘等外存储设备设计的一种平衡查找树。因此在讲B-Tree之前先了解下磁盘的相关知识。

系统从磁盘读取数据到内存时是以磁盘块（block）为基本单位的，位于同一个磁盘块中的数据会被一次性读取出来，而不是需要什么取什么。

InnoDB存储引擎中有页（Page）的概念，页是其磁盘管理的最小单位。InnoDB存储引擎中默认每个页的大小为16KB，可通过参数innodb_page_size将页的大小设置为4K、8K、16K，在MySQL中可通过如下命令查看页的大小：

```
mysql> show variables like 'innodb_page_size';
```

1

而系统一个磁盘块的存储空间往往没有这么大，因此InnoDB每次申请磁盘空间时都会是若干地址连续磁盘块来达到页的大小16KB。InnoDB在把磁盘数据读入内存时，会以页为基本单位，在查询数据时如果一个页中的每条数据都能有助于定位数据记录的位置，这将会减少磁盘I/O次数，提高查询效率。

B-Tree结构的数据可以让系统高效的找到数据所在的磁盘块。

3.2举例描述数据是如何使用b-tree存储的

为了描述B-Tree，首先定义一条记录为一个二元组[key, data]，key为记录的键值，对应表中的主键值，data为一行记录中除主键外的数据。对于不同的记录，key值互不相同。

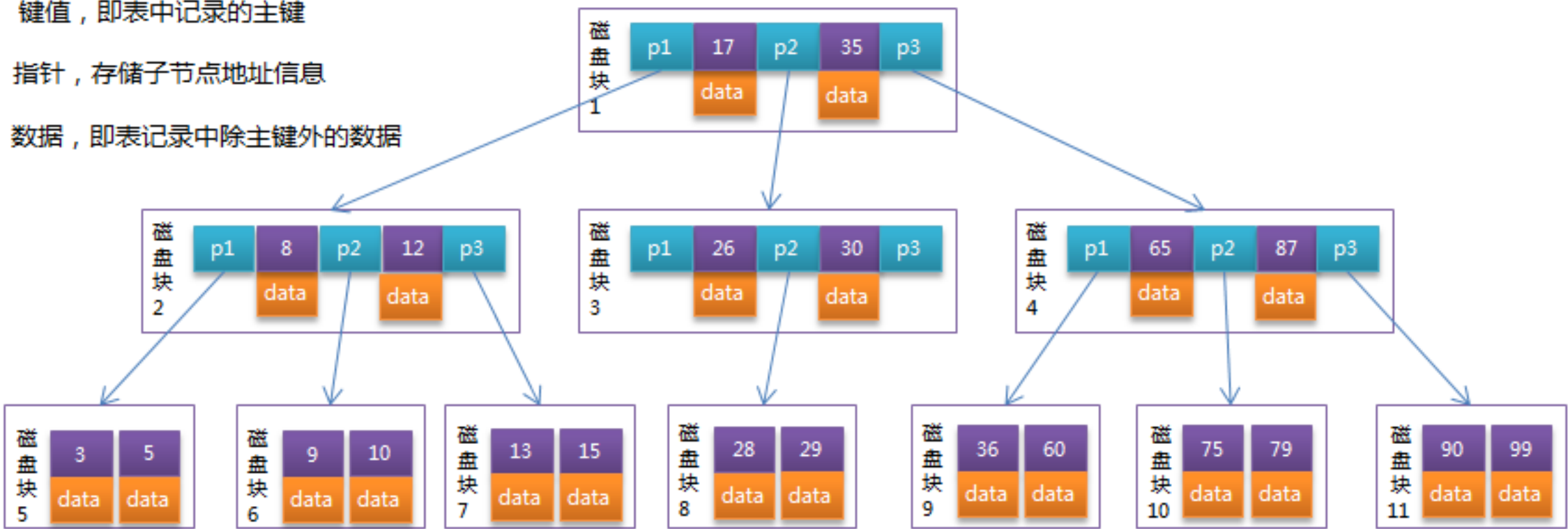
一棵m阶的B-Tree有如下特性：

1. 每个节点最多有m个孩子。
2. 除了根节点和叶子节点外，其它每个节点至少有Ceil(m/2)个孩子。
3. 若根节点不是叶子节点，则至少有2个孩子
4. 所有叶子节点都在同一层，且不包含其它关键字信息
5. 每个非终端节点包含n个关键字信息（P0,P1,...Pn, k1,...kn）
6. 关键字的个数n满足：ceil(m/2)-1 <= n <= m-1
7. ki(i=1,...n)为关键字，且关键字升序排序。
8. Pi(i=1,...n)为指向子树根节点的指针。P(i-1)指向的子树的所有节点关键字均小于ki，但都大于k(i-1)

B-Tree中的每个节点根据实际情况可以包含大量的关键字信息和分支，如下图所示为一个3阶的B-Tree：

- 键值，即表中记录的主键
- 指针，存储子节点地址信息
- data

数据，即表记录中除主键外的数据



每个节点占用一个盘块的磁盘空间，一个节点上有两个升序排序的关键字和三个指向子树根节点的指针，指针存储的是子节点所在磁盘块的地址。两个关键字将节点的数据域划分为三个范围域，这三个范围域对应三个指针指向的子树的数据的范围域。以根节点为例，关键字为17和35，P1指针指向的子树的数据范围为小于17，P2指针指向的子树的数据范围为17~35，P3指针指向的子树的数据范围为大于35。

3.3如何使用b-tree查找数据

模拟查找关键字29的过程：

1. 根据根节点找到磁盘块1，读入内存。【磁盘I/O操作第1次】
2. 比较关键字29在区间（17,35），找到磁盘块1的指针P2。
3. 根据P2指针找到磁盘块3，读入内存。【磁盘I/O操作第2次】
4. 比较关键字29在区间（26,30），找到磁盘块3的指针P2。
5. 根据P2指针找到磁盘块8，读入内存。【磁盘I/O操作第3次】
6. 在磁盘块8中的关键字列表中找到关键字29。

分析上面过程，发现需要3次磁盘I/O操作，和3次内存查找操作。由于内存中的关键字是一个有序表结构，可以利用二分法查找提高效率。而3次磁盘I/O操作是决定整个B-Tree查找效率的决定因素。B-Tree相对于AVLTree缩减了节点个数，使每次磁盘I/O取到内存的数据都发挥了作用，从而提高了查询效率。

4.B+Tree

4.1 介绍

B+Tree是在B-Tree基础上的一种优化，使其更适合实现外存储索引结构，InnoDB存储引擎就是用B+Tree实现其索引结构。

4.2 如何优化？

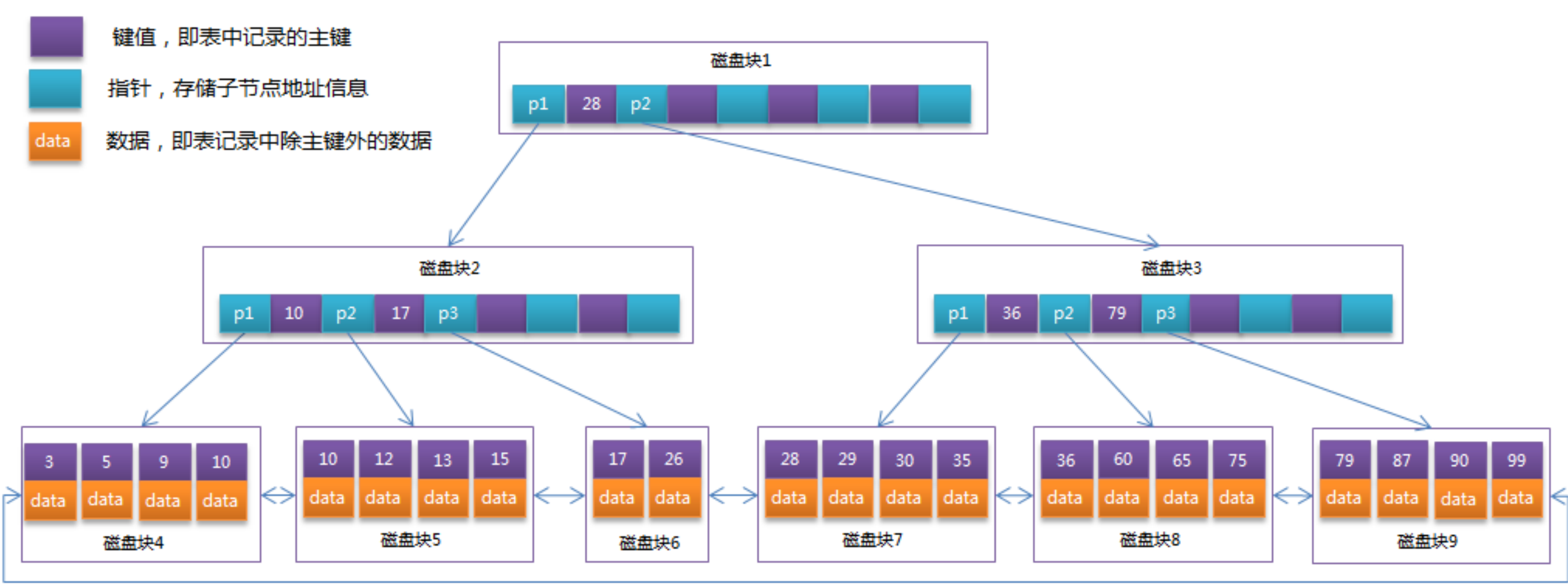
从上一节中的B-Tree结构图中可以看到每个节点中不仅包含数据的key值，还有data值。而每一个页的存储空间是有限的，如果data数据较大时将会导致每个页（即一个页）能存储的key的数量很小，当存储的数据量很大时同样会导致B-Tree的深度较大，增大查询时的磁盘I/O次数，进而影响查询效率。在B+Tree中数据记录节点都是按照键值大小顺序存放在同一层的叶子节点上，而非叶子节点上只存储key值信息，这样可以大大加大每个节点存储的key值数量，降低B-Tree的高度。

B+Tree相对于B-Tree有几点不同：

1. 非叶子节点只存储键值信息。
2. 所有叶子节点之间都有一个链指针。
3. 数据记录都存放在叶子节点中

4.3 举例

将上一节中的B-Tree优化，由于B+Tree的非叶子节点只存储键值信息，假设每个磁盘块能存储4个键值及指针信息，则变成B+Tree后其结构如下图所示：



说明：


- 1.通常在B+Tree上有两个头指针，一个指向根节点，另一个指向关键字最小的叶子节点。
2. 而且所有叶子节点（即数据节点）之间是一种链式环结构。
- 3.因此可以对B+Tree进行两种查找运算：

3.1一种是针对主键的范围查找和分页查找

3.2 另一种是从根节点开始，进行随机查找。
- #### 4.4 效率的推算
- 可能上面例子中只有22条数据记录，看不出B+Tree的优点，下面做一个推算：
- InnoDB存储引擎中页的大小为16KB，一般表的主键类型为INT（占用4个字节）或BIGINT（占用8个字节），指针类型也一般为4或8个字节，也就是说一个页（B+Tree中的一个节点）中大概存储16KB/(8B+8B)=1K个键值（因为是估值，为方便计算，这里的K取值为 $\lceil 10 \rceil^3$ ）。也就是说一个深度为3的B+Tree索引维护 $10^3 * 10^3 * 10^3 = 10$ 亿 条记录。
- 实际情况中每个节点可能不能填满，因此在数据库中，B+Tree的高度一般都在2~4层。mysql的InnoDB存储引擎在设计时是将根节点常驻内存的，也就是说在查找某一键值的行记录时最多只需要1~3次磁盘I/O操作。
- #### 4.5聚集索引 & 辅助索引
- 数据库中的B+Tree索引可以分为聚集索引（clustered index）和辅助索引（secondary index）。
- 上面的B+Tree示例图在数据库中的实现即为聚集索引，聚集索引的B+Tree中的叶子节点存放的是整张表的行记录数据。
- 辅助索引与聚集索引的区别在于：
- 辅助索引的叶子节点并不包含行记录的全部数据，而是存储相应行数据的聚集索引键，即主键。
- 当通过辅助索引来查询数据时，InnoDB存储引擎会遍历辅助索引找到主键，然后再通过主键在聚集索引中找到完整的行记录数据。
- 转载：<https://blog.csdn.net/ifollowrivers/article/details/73614549>
- 二叉树、平衡二叉树、红黑树、B-树、B+树、B*树、T树之间的详解和比较

阅读数 1939

=====||欢迎讨论技术... 博文 | 来自： 随意的风...
- 想对作者说点什么



a1454635626：想问下针对 B+树存储时，每个数据块里的主键是连续的，我们所用数据库时，他的每个数据块存储的数据 主键也还会是连续的吗

(1年前

#2楼)

查看回复(1)



金士顿：写的真棒

(1年前

#1楼)

查看回复(1)

二叉树，完全二叉树，满二叉树，平衡二叉树的区别

阅读数 1万+

度：指的是一个节点拥有子节点的个数。如二叉树的节点的最大度为2。深度：数的层...

博文

来自：

yuwushu...

二叉树、平衡二叉树、B- tree、B+ tree 基本概念

阅读数 4943

1二叉树二叉树binarytree是指每个节点最多含有两个子树的树结构。特点：1.所有节...

博文

来自：

时光钟摆

快速理解平衡二叉树、B-tree、B+tree、B*tree

阅读数 2万+

快速理解平衡二叉树、B-tree、B+tree、B*tree

博文

来自：

jacke121...

广告

关闭

二叉树，平衡二叉树，红黑树，B-树、B+树、B*树的区别

阅读数 1万+

二叉查找/搜索/排序树 BST (binarysearch/sorttree)或者是一棵空树；或者是具有下列...

博文

来自：

wyqwillia...

平衡二叉树总结

阅读数 1471

目录平衡二叉树相关概念以及性质平衡二叉树的类结构LL型失衡以及解决办法RR型失...

博文

来自：

博客已搬...

平衡二叉树（AVL树）

阅读数 141

Mark：https://blog.zhenlanghuo.top/2017/08/22/AVL平衡二叉树的实现/

博文

来自：

zhaohon...

二叉树、平衡二叉树原理及实例（一）

阅读数 1205

最近闲来无事，研究了一下二叉树。怪了，非平衡二叉树，两三个小时就搞定了生成...

博文

来自：

newbie的...

二叉排序树和平衡二叉树

阅读数 1940

一、二叉排序树

1、定义：二叉排序树（BST）也称二叉查找树。是一棵空树或...

博文

来自：

菜鸟程序...

索引--- 二叉树、平衡二叉树、b-tree、b+tree详解(强烈..._CSDN博客

二叉树、平衡二叉树、B- tree、B+ tree 基本概念 - 时..._CSDN博客

索引--- 二叉树、平衡二叉树、b-tree、b+tree详解(强烈..._CSDN博客

二叉树、平衡二叉树、B- tree、B+ tree 基本概念 - 时..._CSDN博客

索引--- 二叉树、平衡二叉树、b-tree、b+tree详解(强烈..._CSDN博客

二叉树、平衡二叉树、B- tree、B+ tree 基本概念 - 时..._CSDN博客

平衡二叉树

阅读数 1442

平衡二叉树，是一种二叉排序树，其中每个结点的左子树和右子树的高度差至多等于1...

博文

来自：

baidu_18...

快速理解平衡二叉树、B-tree、B+tree、B*tree - CSDN博客

快速理解平衡二叉树、B-tree、B+tree、B*tree - Akaks..._CSDN博客

快速理解平衡二叉树、B-tree、B+tree、B*tree - CSDN博客

Java 泛型，你了解类型擦除吗？ 阅读数 4万+

泛型，一个孤独的守门者。大家可能会有疑问，我为什么叫做泛型是一个守门者。这... 博文 | 来自： frank 的...

树和二叉树和平衡二叉树 阅读数 58

树型结构是一类非常重要的非线性结构。直观地，树型结构是以分支关系定义的层次... 博文 | 来自： csdnoyns...

剑指offer—关于判断二叉树是否为平衡二叉树 阅读数 9105

判断一颗二叉树是否为平衡二叉树，当这到面试题摆在我们眼前的时候，我们需要进... 博文 | 来自： 宇哲

二叉搜索树与平衡二叉树 阅读数 4487

二叉树也是一种树，适用与以上的全部操作，但二叉搜索树能 够实现数据的快速查找 ... 博文 | 来自： Gouhailia...

二叉树、平衡二叉树、完全二叉树、满二叉树 阅读数 7万+

基本概念结点的层次（Level）从根开始定义，根为第一层，根的孩子为第二层。二叉... 博文 | 来自： Terry的IT...

[推动全社会公益氛围形成，使公益与空气和阳光一样触手可及。](#)
公益缺你不可，众多公益项目等你PICK——百度公益 让公益像「空气和阳光」一样触手可及！
gongyi.baidu.com

广告

[推动全社会公益氛围形成，使公益与空气和阳光一样触手可及。](#)
公益缺你不可，众多公益项目等你PICK——百度公益 让公益像「空气和阳光」一样触手可及！
gongyi.baidu.com

广告

【数据结构】 ----平衡二叉树怎么自己画？ 阅读数 2万+

【数据结构】平衡二叉树怎么自己画？ 是什么？ 要了解平衡二叉树，先得了解什... 博文 | 来自： 何新生(D...

一分钟搞懂各种树， 完全二叉树、平衡二叉树、二叉查找树， B- tree， B+ ... 阅读数 621

度：指的是一个节点拥有子节点的个数。如二叉树的节点的最大度为2。深度：树的层... 博文 | 来自： DamonU...

关于索引的B tree B-tree B+tree B*tree 详解结构图 阅读数 1万+

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）； ... 博文 | 来自： superhos...

二叉树、平衡二叉树原理及实例（二） 阅读数 142

这一篇将分享我理解并整理的非平衡二叉树到平衡二叉树的原理及代码。目录一、左... 博文 | 来自： newbie的...

快速理解平衡二叉树、B-tree、B+tree、B*tree（转载） 阅读数 80

https://blog.csdn.net/jacke121/article/details/78268602这篇文章相当精辟，有助于理... 博文 | 来自： weixin_4...

有关数据库的索引的实现

- 问答

索引 - 二叉树 阅读数 343

AOAPCI:BeginningAlgorithmContests(RujiaLiu)112-TreeSumming548-tree297-Quadt... 博文 | 来自： 潜水的程...

数据库常见索引解析（B树，B-树，B+树，B*树，位图索引，Hash索引） 阅读数 1万+

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）； ... 博文 | 来自： wI044090...

按序号索引二叉树 阅读数 4043

理论上，一个平衡的二叉树，可以在O(logn)时间内，按中序遍历的序号号（或者... 博文 | 来自： whinah的...

SQL索引问题 阅读数 523

fromhttp://www.cnblogs.com/lixiaolun/p/5058304.html数据库索引的存储结构一般是B... 博文 | 来自： basycai...

数据结构与算法——图解平衡二叉树及代码实现 阅读数 6383

平衡二叉树介绍平衡二叉树，是一种二叉排序树，其中每一个节点的左子树和右子树... 博文 | 来自： 李四老师

平衡二叉树的旋转

阅读数 1952

AVLTree高度平衡的搜索二叉树一棵平衡树，或是空树，或是具有以下性质的二叉搜...[博文](#) | 来自：[deeplan_...](#)

二叉树中完全二叉树、满二叉树、二叉排序树、平衡二叉树的区别和联系

阅读数 1353

1，完全二叉树：只有最下面的两层结点度小于2，并且最下面一层的结点都集中在该...[博文](#) | 来自：[zlhzh11...](#)

平衡二叉树和AVL

阅读数 843

1 概述 对于一棵二分搜索树，如果我们的数据是顺序添加到二分搜索树中，它就...[博文](#) | 来自：[Mamba28](#)

Android性能优化 -- Systrace工具

阅读数 7758

Systrace简介 一般来说，我们的机器以60帧/秒显示时，用户会感觉机器很流畅，如...[博文](#) | 来自：[Kitty_Lan...](#)



教你这样做牛肉汤,做出来的是非常的好喝的哦

加盟牛肉店



打算留学看这里！高品质留学服务,为您量身定制专业留学申请规划

留学中介

广告

广告

【数据结构之二叉树】（二）B+树比B树更适合做文件索引的原因

阅读数 3997

原因：相对于B树，（1）B+树空间利用率更高，可减少I/O次数，因为B+树的内部...[博文](#) | 来自：[cangchen...](#)

MySQL索引之B+树索引

阅读数 3547

B+树索引是是目前关系型数据库系统中查找最为常用和最为有效的索引，B+树的索引...[博文](#) | 来自：[lanxingla...](#)

按序号索引二叉树的应用

阅读数 1581

主要是快速计数。可以从Index得到相应结点，也就可以从相应结点得到Index。如果...[博文](#) | 来自：[whinah的...](#)

对于二叉树的非递归遍历（非常好记的三种方式）

阅读数 6310

显然，我们需要用一个stack来模拟递归时的函数调用。对于三种遍历，我们都使用pu...[博文](#) | 来自：[foreveyki...](#)

c#次横坐标不显 c#dem文件 c#免安装版反编译工具 c# 深度 递归 c#网页如何调试 c# 添加自定义的属性 c#中去除窗体边框 dll ida修改c# c#实现打印功能 c# 线程结束时执行

没有更多推荐了，[返回首页](#)



qq_36098284

私信

关注

[TA的个人主页 >](#)

原创112

粉丝41

喜欢49

评论36

等级：

博客5

访问：13万+

积分：2535

排名：2万+

勋章：



最新文章

- java 学习笔记（9） Exception Handling
- java 学习笔记（8） Polymorphism & abstract class
- 计算机网络--运输层（3） TCP
- 计算机网络--运输层（2）可靠数据传输协

议

分类专栏

	java	50篇
	算法分析与设计	20篇
	C++	1篇
	系统	5篇
	安装	2篇

展开

归档

2019年9月	12篇
2019年8月	19篇
2019年7月	5篇
2019年4月	8篇
2019年3月	17篇
2019年2月	1篇
2018年11月	2篇
2018年9月	7篇

展开

热门文章

spring自动扫描包详解

阅读数 15805

怎么理解三维数组？

阅读数 9193

'mvn-v' 不是内部或外部命令，也不是可运行的程序 或批处理文件。

阅读数 8726

WIN10下怎么找到MYSQL数据库中存储数据的位置。（默认路径）

阅读数 7426

处理 Driver class not found

阅读数 6289

最新评论

java（10）（&&am...

qq_36098284：[reply]Ssuper_X[/reply] 好的，已改正，感谢

java（10）（&&am...

Ssuper_X：||一真即真，不是两个true才为真

neo4j（4）--- 使用报...

weixin_43863672：你这两个路径写的是一样的

处理 Driver class n...





weixin_41262685：tql 66666

'mvn-v' 不是内部或外部命令...


qq_36098284：[reply]weixin_43919304[/reply] 有用就好



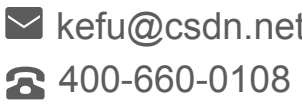
CSDN企业招聘

 QQ客服  kefu@csdn.net
 客服论坛  400-660-0108
 工作时间 8:30-22:00

[关于我们](#) | [招聘](#) | [广告服务](#) | [网站地图](#)


 百度提供站内搜索 京ICP备19004658号
 ©1999-2019 北京创新乐知网络技术有限公司
 司

网络110报警服务 经营性网站备案信息
北京互联网违法和不良信息举报中心
中国互联网举报中心 家长监护 版权申诉



工作时间 8:30-22:00

[关于我们](#) | [招聘](#) | [广告服务](#) | [网站地图](#)

 百度提供站内搜索 京ICP备19004658号
 ©1999-2019 北京创新乐知网络技术有限公司
 司

网络110报警服务 经营性网站备案信息
北京互联网违法和不良信息举报中心
中国互联网举报中心 家长监护 版权申诉