# 《DBMS》实验报告二

**报告人：**冷友方；王大伟；罗传文；赵丹；杨婧如.

**学院：**信息学院

**日期：**2016年11月8日

## 1 B+树节点结构定义

1) 首先定义B+树的度数Degree，我们定义为3。

2) 然后定义B+树的节点结构BPKeyNode：

|  |
| --- |
| typedef struct BPKeyNode  {  int nodeId;  int keyNumInNode;  int isLeaf;  KeyDataType keyInNode[2\*Degree];  Pointer children[2\*Degree];  Pointer nextBrother;  }BPKeyNode,\*P\_BPKeyNode; |

其中nodeId 用于记录这个节点在文件的中的编号，keyNumInNode 用于记录这个节点有多少个关键字，isLeaf 用于判断是否为页节点，keyInNode[2\*Degree]是存放 KEY 值的数组，children[2\*Degree]是存放指针（数据地址）的数组，用于记录每个孩子在文件的第几个位置，nextBrother为记录下一个兄弟的指针。

3) B+树的结构

|  |
| --- |
| typedef struct BPKeyTree  {  P\_BPKeyNode treeRoot;  int rootFileId;  int nodeNum;  int keyNum;  char treeFileName[100];  FILE \*pof;  int fistLeaf;  }BPKeyTree,\*P\_BPKeyTree; |

其中rootFileId用于记录根节点的在文件中的标号，即id，nodeNum用于记录共有多少个节点，keyNum用于记录共有多少个key，treeFileName[100]用于存储B+树的节点文件的名字，\*pof为打开写入name文件时，指向文件的指针，fistLeaf为最小的数据所在的叶节点。

以上关于 B+树的定义在文件”bPlusTreeIndex.h”中，下面介绍 B+树的查找插入删除操作的具体实现。

## 2索引的初始建立

索引的初始建立是以索引的插入函数为基础，先创建一个索引文件（即创建一个空树），然后向该 B+树中依次插入每一个 key 值，以此方法来初使建立所需要的索引文件。

## 3索引的查找

B+树索引的查找是 B+树索引中最基本的操作，它是插入、删除操作的基础。在索引查找方面，我们实现了等值查询和范围查询两种查询功能，下面依次进行介绍。

### 3.1 等值查询

等值查询过程如下（查找键值 K）：①基础：若处于叶节点，我们在其键值中查找。若第 i 个键是 K，则第 i个指针项记录了 K 对应数据的地址；②归纳：若处于内部节点（假定键值依次为 K1, K2, …Kn），定位至 Ki ≦ K < Ki+1，则为第 i+1 个子节点，如果 i=0，则为第 1 个节点。我们从根节点开始逐层应用这个查找过程。就能找到所要查找的键值。

在每一层的查找过程中，由于对应块已读入到内存中，可以使用内存的查找算法，我们采用的是从最小值开始进行顺序查找，给出key值，查找对应的id，并返回。如果不存在该节点，返回一个负数.

### 3.2 范围查询

对于范围[a, b]查找，我们通过一次查找来找出键 a，到达可能出现a的叶节点，然后在该叶节点中查找a或大于a的键，这些键均有指针指向对应的记录在范围[a, b]内，继续跟踪指向下一个叶节点的指针来继续查找。：

## 4 索引的插入

插入一个新键到B+树中时，首先要对 B+树有一个搜索过程，判断新键在已有 B+树中是否存在，如果存在，则出错返回（我们所建的是没有重复键值的 B+树）；否则执行正常的插入操作。搜索过程基于 search 的核心代码，在从根到叶节点的移动过程中用数组保存了路径上的地址（块号）信息，即记录搜索路径上当前“磁盘块”的孩子节点的块号，以便于迅速获取各层的块信息。

具体过程如下：首先将元素插入到树中；然后搜索整个树，找到插入的位置，如插入的关键字大于root的最大关键字，则需要该祖辈的关键字；将关键字以及其子树插入当前节点；若节点的个数过多则分裂该节点。

在插入过程进行一下四种情况的处理：

1. 叶子结点有空闲空间。这是最简单的情况，直接在搜索失败处插入新键的键值和其代表数据的数据地址。
2. 叶子结点没有足够的空闲空间（此时叶子结点存储已满），我们将对叶子结点进行分裂操作。对于父节点看来，需要在其中插入一个新键-指针对。第二种情况正是对应了父节点有空闲空间容纳调整的键-指针对，将调整的键-指针对插入父节点相应位置即可。
3. 内部节点也没有空闲空间，那么要对内部节点进行分裂操作，同时调整内部结点的父节点的键-指针对，这是一个递归的过程。
4. 插入操作引起 B+树的层次增加 1，这时就需要新建一个根结点，这意味着整棵树都进行了调整。

## 5 索引的删除

删除过程是插入过程的逆过程。在插入过程中的结点分裂对应着删除过程的结点合并，以避免删除元素后达不到最小充满度的要求。首先应找到该关键字所在结点，并从中删除之，若该结点为最下层的非终端结点，且其中的关键字数目不少于ceil(n/2)，则删除完成，否则要进行“合并”结点的操作。假若所删关键字为非终端结点中的Key，则可以指针所指子树中的最小关键字替代这个键值，然后在相应的结点中删去最小关键字。对于插入过程的四种情况，分别有四种删除情况的处理：

1. 叶子结点有足够的键-指针对，删除后不至于达不到最小充满度要求，直接将键及其对的指针删除。当块首元素被删除时，分情况需要相应调整父节点对应位置的键值。
2. 叶子结点删除一个键-指针对之后少于最小充满度，需要从邻居结点中借一个元素，前提是邻居结点有多余元素可借。我们默认优先从左邻居中借。当块首元素被删除时，分情况需要相应调整父节点对应位置的键值。内部结点借元素情况与叶子结点类似，只是内部结点在向邻居借元素时，要求父节点对应位置的元素下移到该节点，然后将借来的元素上提至父节点，最后要调节过程涉及块的指针。
3. 邻居结点无元素可借时，需要与其中一个邻居结点合并。默认优先与左边的结点合并。合并时，将右侧的结点合并到左侧结点，然后删除右侧结点对应的块。此时需要调整父节点，删除右侧块对应父节点的键-指针对。这是一个递归过程。内部结点删除操作与叶子结点处理类似，也需要向上删除待调整的键-指针对。
4. 当删除到达根节点时，有可能使 B+树的层次减 1，这时要删除原根结点，标识新根结点的块号。

## 6 文件结构及主要函数说明

|  |  |  |
| --- | --- | --- |
| 文件 | 函数 | 说明 |
| bPlusTreeIndex.h | BPKeyNode | 节点结构体定义 |
|  | BPKeyTree | 树的结构体定义 |
| bPlusTreeIndex.c | saveNodeToFile | 保存节点到文件 |
|  | getNodeFromFile | 从文件读取结点 |
|  | displayNode | 显示节点详细信息 |
|  | createIndexBPKeyTree | 创建B+树,并进行相应的初始化 |
|  | initIndexBPKeyTree | 初始化BPTree，打开相应文件pof；为root分配内存可以存储一个节点的内存，并读入根节点 |
|  | divideBPKeyNode | 节点的分裂 |
|  | insertBPNodeNotFull | 在节点中插入一个key，要求kNode节点不是满的 |
|  | equalSearch | 等值查询，给出key值，查找对应的id，并返回。如果不存在该节点，返回一个负数 |
|  | rangeSearch | 范围查找，key值大于等于lowValue，小于等于highValue。返回范围内的个数. |
|  | insertKeyInBPKeyTree | 向树种插入节点 |
|  | replaceKeyInBPKeyTree | Key替换，将oldkey替换为newkey |
|  | adjustWhenDel | 当删除key值时，调整树 |
|  | delKeyInBPNode | 删除节点中的一个key |
|  | delKeyInBPTree | 删除树中的一个key |
|  | endBPKeyTree | 将建立的树结束 |

## 7 实验中遇到的问题

### 7.1 索引中的keynum范围溢出问题

在代码实现时，索引树的结构体中定义了一个变量keyNum，用于存储整个索引中包含的key的数量，类型为Int，未考虑到Int只有4个字节，最大只能存储232个key的范围上界问题。如果换成long型，虽然存储的范围会扩大，仍然会有上界问题，目前暂未考虑到好的方法解决此问题。

### 7.2 范围查找返回值问题

range查找时，仅返回了匹配到的key的数量，可满足将来sql中的select count(\*) from的要求，但不能满足select \* from问题。

**解决方法：**需要新增一个函数，用来返回所有匹配到的key值。