哈尔滨工业大学

**<<数据库系统>>**

**实验报告**

**(2020年度春季学期)**

|  |  |
| --- | --- |
| **姓名：** | **史纪元** |
| **学号：** | **1173300919** |
| **学院：** | **计算机科学与技术学院** |
| **教师：** | **史建焘** |

实验二 数据库索引及查询算法实现

## 一、实验目的

掌握B树索引查找算法，多路归并排序算法，并用高级语言实现

## 二、实验环境

Windows 10，Python 3.6，Pycharm

## 三、实验过程及结果

**1. 生成数据**

使用random.sample()函数在0~0xFFFFFFFF中随机生成10^6个整数，每个整数后接一个长度为12个字符的随机生成的字符串，生成字符串的函数为：

def random\_string(string\_length):  
 letters = string.ascii\_lowercase  
 return ''.join(random.choice(letters) for k in range(string\_length))

然后以每个条目4B整数+12B字符串的格式二进制写入data文件中，完整代码为：

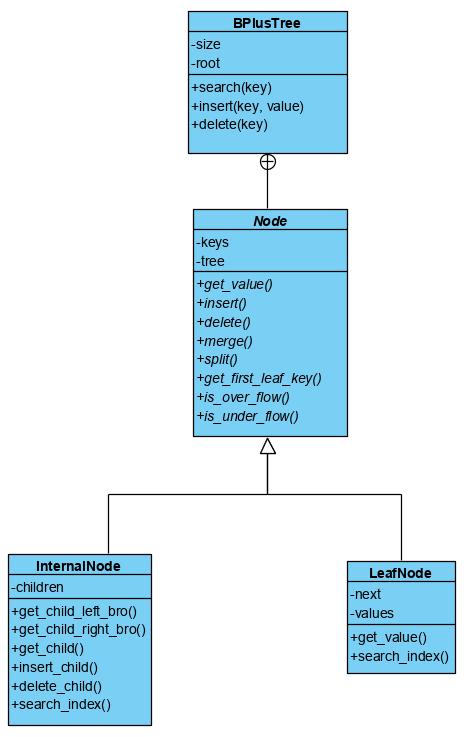
number = random.sample(range(0xFFFFFFFF), 10 \*\* 6)  
 items = []  
 for i in range(10 \*\* 6):  
 items.append((number[i], random\_string(12)))  
 with open('data', 'wb') as f:  
 for item in items:  
 f.write(item[0].to\_bytes(4, byteorder='big'))  
 f.write(item[1].encode(encoding='utf-8'))

得到16B \* 10^6大小的文本文件，用WinHex查看：

**2.B+树索引**

本次实验我选用的是B+树索引。

因为B+树的叶结点比非叶结点多了一个指向下一个叶结点的属性，因此我分别定义了B+树类BPlusTree、结点抽象类Node、内部节点类InternalNode和叶结点类LeafNode。各类的属性、方法和关系如下：



B+树初始化时，首先初始化阶数size，然后初始化一个叶结点，并将根节点置为该叶结点。B+树类的search、insert和delete方法委托给了InternalNode和LeafNode来实现。

抽象类Node定义了叶结点和内部节点通用的属性keys、tree（当前节点所属的B+树）和insert、delete、get\_value等方法。

**2.1 InternalNode类**

内部结点类继承Node抽象类，并新增了children属性，该属性用来保存一个内部节点的子结点。

函数介绍：

search\_index(key)找到self.keys中能够插入key的下标并返回该下标。

def search\_index(self, key):  
 i = 0  
 for k in self.keys:  
 if k > key:  
 break  
 i += 1  
 return i

get\_child(key)找到可能包含key的子结点并返回，非叶结点一定能找到一个可能包含key的子结点。

def get\_child(self, key):  
 i = self.search\_index(key)  
 return self.children[i]

get\_value(key)方法根据输入key返回B+树中key对应的value。具体实现为首先找到包含该key的子结点，然后再递归调用get\_value(key)方法。

def get\_value(self, key):  
 return self.get\_child(key).get\_value(key)