# Index Manager说明文档

作者：2016级软件工程

赵丰年

## 一、Index Manager 主要功能

Index Manager负责索引文件的管理，主要功能有：

1. 为指定表的指定属性建立索引；
2. 删除索引；
3. 通过索引查找某键值；

## 二、模块总体设计思路

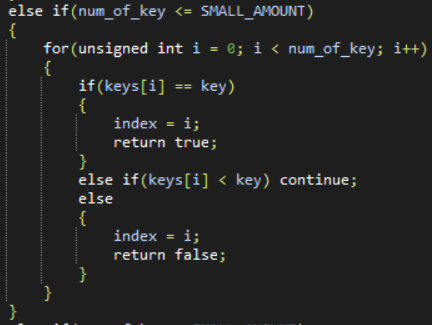
整个索引文件的结构基础是B+树，首先要实现B+树节点类型的定义，包括叶子节点（信息真实存放的地方）与非叶子结点。创建节点类，其中包括针对节点操作的函数。再是创建B+树类，其中包括针对整个树结构操作的函数，以及与Buffer Manager交互的子函数（信息的读写）。最后创建Index Manager类，一个Index Manager应当与一个表（Table）相关联，负责管理这张表所有属性对应的索引文件。当Index Manager类被创建（参数为表（class Table）类），构建该表所有属性的对应类型的B+树（使用map，将表名与索引配对），每一棵B+树类创建时，读取索引文件内容（重构）。当Index Manager被析构时，每一棵B+树将文件写回。整个写回与读取的操作需要与Buffer Manager交互，程序运行结束时，索引文件被写回。

## 三、具体实现

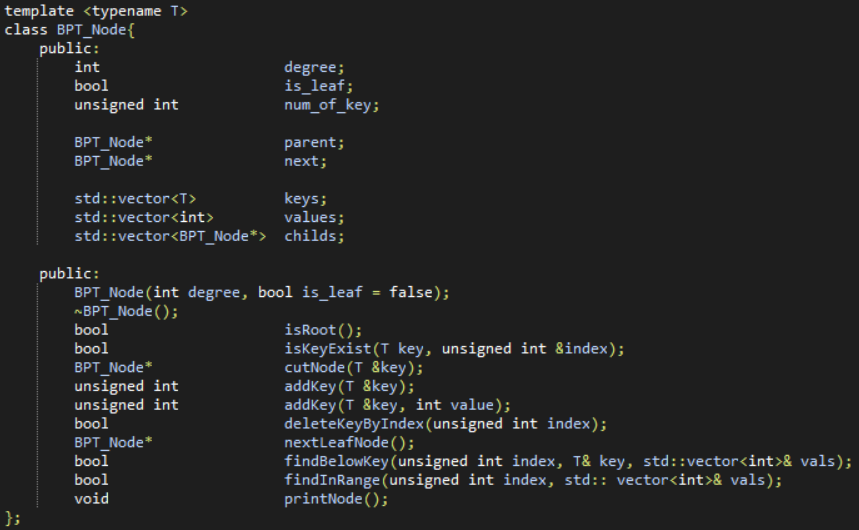
1. **宏定义**



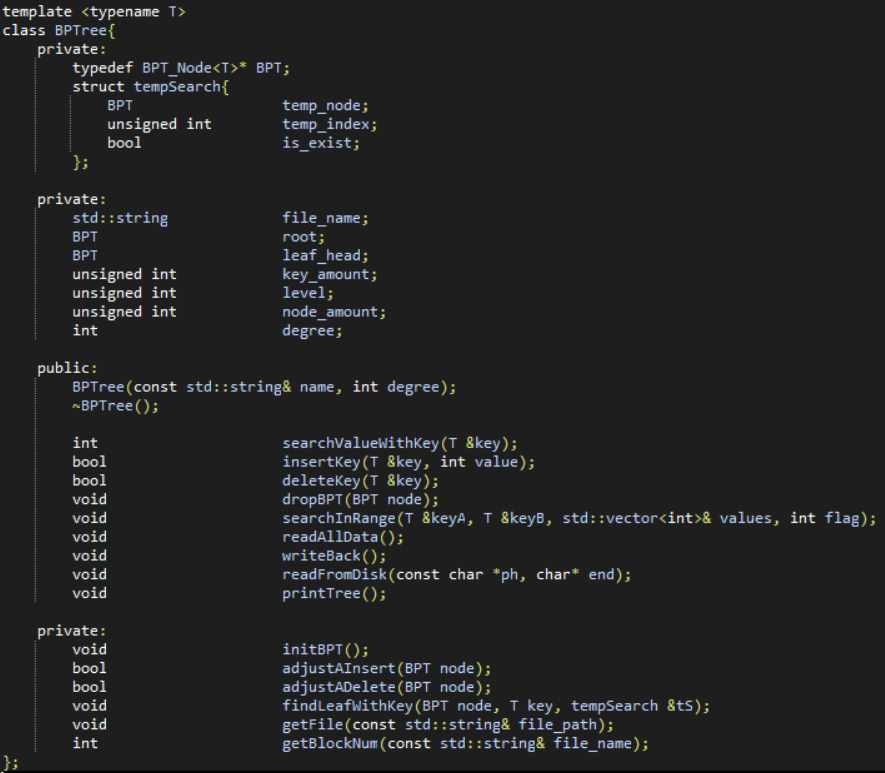
在B+树的查询操作中，少于该数量时，将直接遍历搜索：



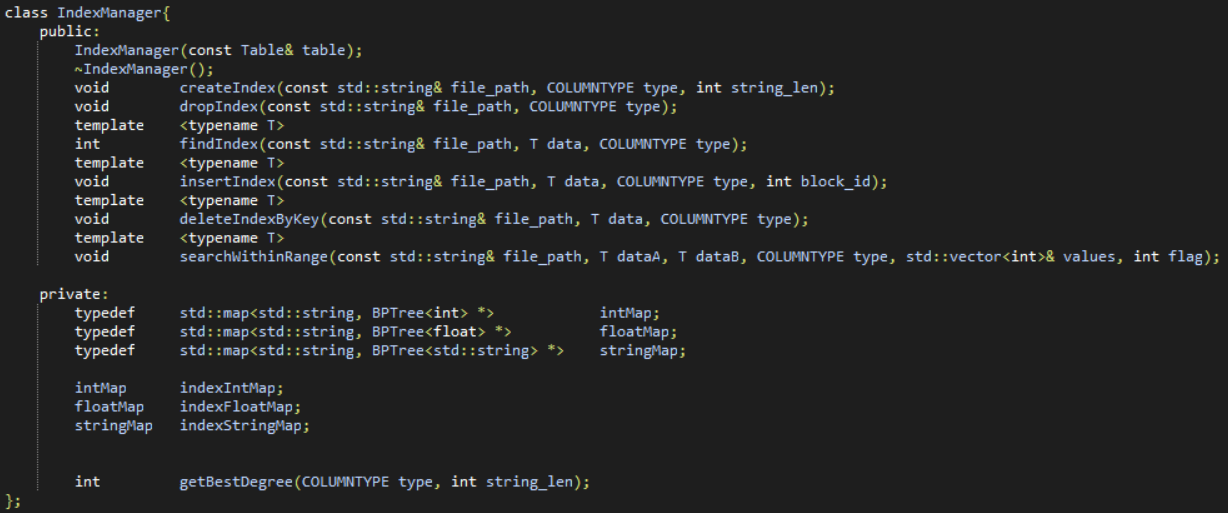
1. **类定义**



B+树节点类定义如上（在本程序中，values用于存储叶子节点所在的block\_id）



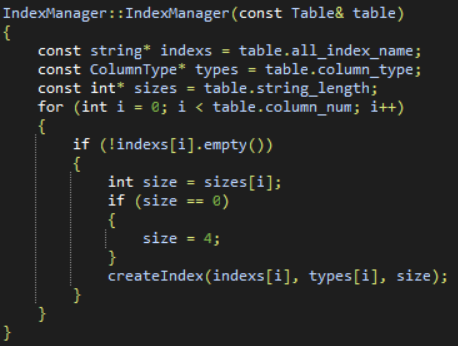
B+树类定义如上（file\_name即是索引文件名，一个索引文件对应一棵B+树）



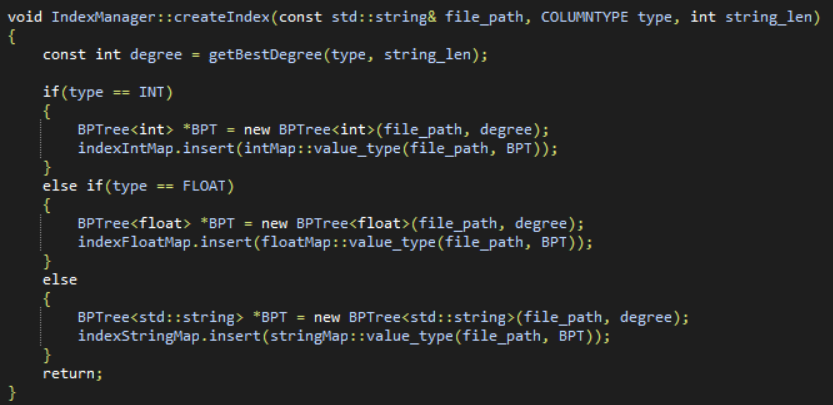
Index Manager类定义如上

1. **主要函数及其功能描述**

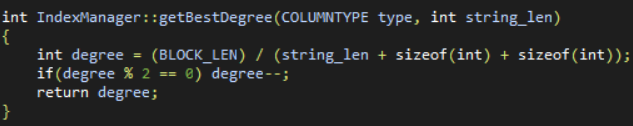
我们将忽略B+树节点操作的部分函数，该部分属于B+树结构调整的技术细节，较为冗余。



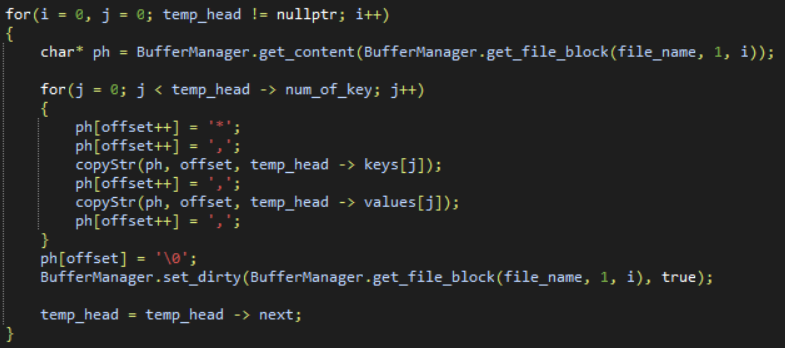
首先是Index Manager的构造函数，有且仅有这一个，我们不允许没有参数的构造函数，因为索引文件必须与一张表关联。构造函数需要一个表类型参数输入，它将会为每一个属性建立索引。建立索引的函数（createIndex（））细节如下：



首先参数表中的file\_name 即是此时index的名字，并存储在表信息中，type为对应属性的类型，可能为INT，FLOAT，也可能是任意长度的字符串，第三个参数便是用于传输属性类型长度的，用于确定合适的度（degree），以保证块（block）的合理利用。根据不同的类型，创建不同类型键值的B+树，并与该表名配对，添加到相应的map中。以下为度的计算方式：

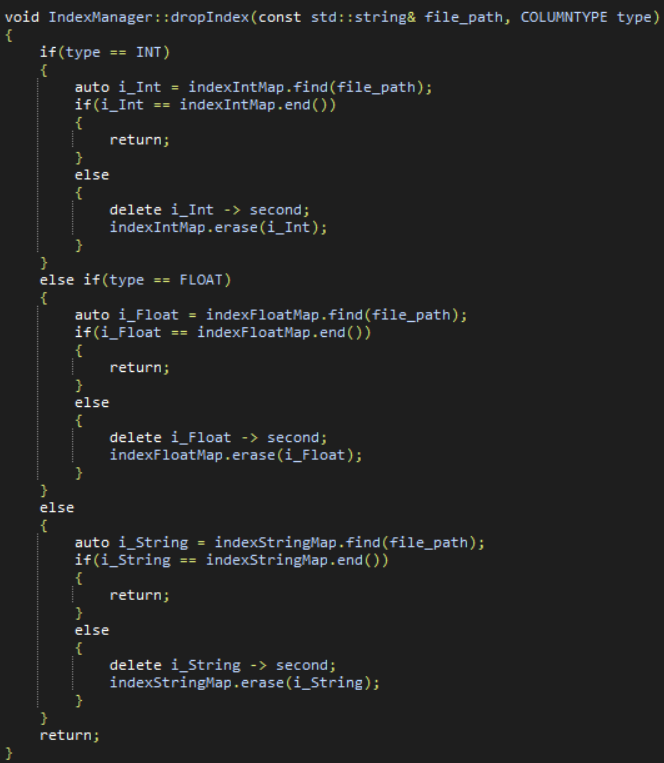


根据叶子节点的序列化方式，每个数据的实际长度需要增加8个字节，也就是两倍INT类型的大小。以下是叶子节点序列化的方式：



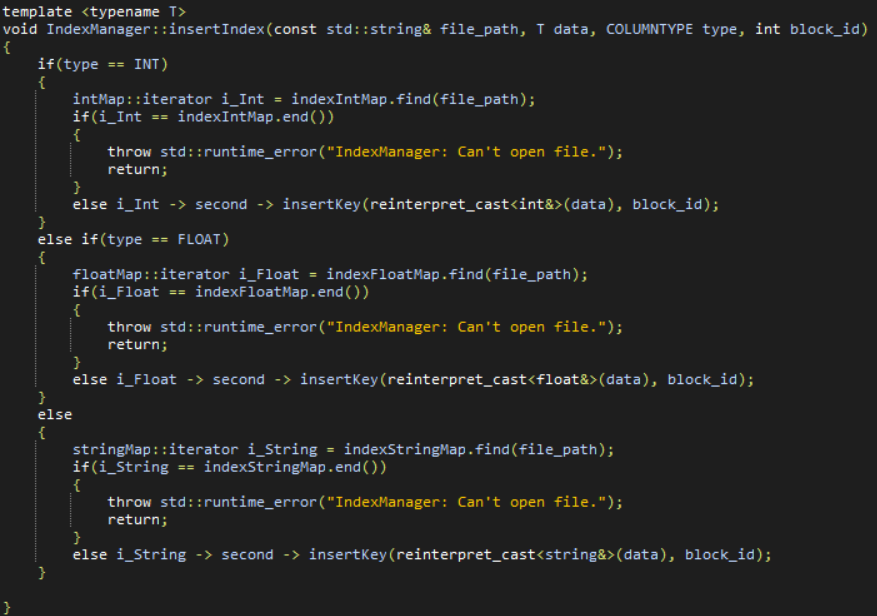
在合适的度下，每个叶子节点被存储在一个block中，用‘\*’作为每个数据的“头”，值与值之间用‘，’分隔，**这意味着我们要避免出现包含‘，’与‘\*’的键值**。

以下为删除索引的实现：



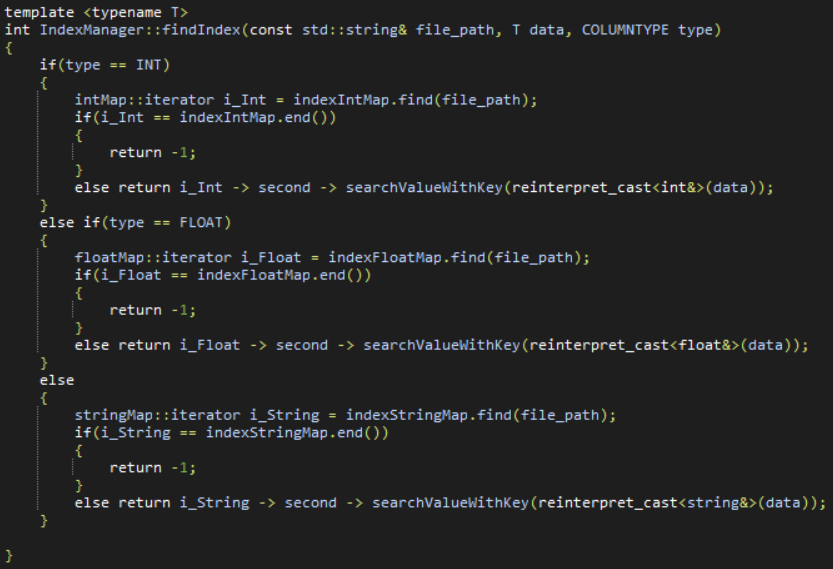
输入索引文件名，及index关联属性的类型，该函数会寻找该表对应索引，并析构所有对应的B+树类。

以下为向索引中插入键值的函数：



输入索引文件名，键值类型，值，与储存位置的块偏移量（value），函数将会将该值插入到相应的索引文件中（B+树中）

以下为通过索引查找键值的函数：



输入索引文件名，键值类型，值，函数将会返回该值所在索引文件的块偏移量（value）