DBMS实验报告

SZ2016084陆艺仁

1. **题目选择**

我选择的实验题目是题目3：持久化的B+树索引机制，其功能要求为：

1. 实现一个基于主存的B+树
2. 持久化B+树，即将B+树存储在磁盘中
3. 装入已存储在磁盘上的B+树，并能够进行B+树的查询和修改
4. 保证B+树文件在磁盘中是加密的，只有内存中是明文

需考虑的问题有：

1. 申请的节点空间由DBMS管理；
2. 支持key的插入和删除，即具有维护B+树的功能
3. 保证B+树的形状不退化
4. **实验环境**

本实验的语言环境为go1.15.2，开发软件为GoLand，机器配置为Windows 10家庭中文版，Intel i7，16G内存，64位操作系统

1. **方法流程**

**3.1 数据结构**

B+树包含两种节点：

内部节点：所有的非最底层节点都是内部节点，其内部存放着关键字（索引值）以及可以指向叶节点也可以指向内部节点的指针。其索引值是叶节点中索引值的拷贝，为其指向子节点中关键字的最大或最小值。

叶节点：在B+树的最底层，且所有的叶节点都在同一层，其中存放着关键字（索引值），指向记录的指针以及指向下一个结点的指针。索引值为记录中关键字的拷贝，且有序排列，从左到右分布在叶结点中，每个叶节点关键字个数都介于m/2到m之间，形成一个有序链表式的结构，其头指针指向含最小关键字的节点。

根节点比较特别，当B+树只有一层的时候，根节点就是叶节点，其中索引值的最小个数不受到叶节点关键字个数的限制。当B+树为两层及以上，根节点为最上层的内部节点，其中指针必须为两个以上。

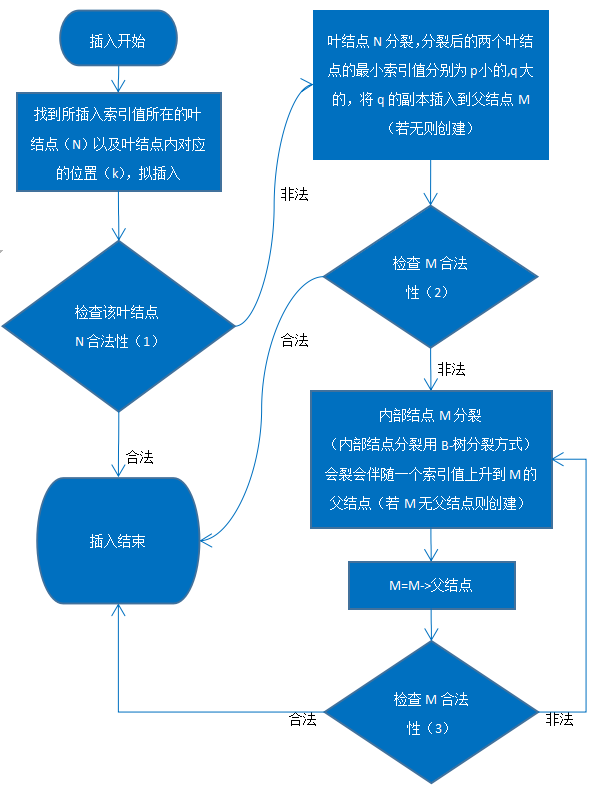
**3.2 算法流程**

**3.2.1 B+树的查找**

从根节点开始，进行二分法查找，因为节点内部关键字排列是有序的，内部节点找到对应区间后继续向下一层节点进行查找直至进入叶节点，若叶节点存在该索引值，则返回记录对应的指针，否则返回无记录。

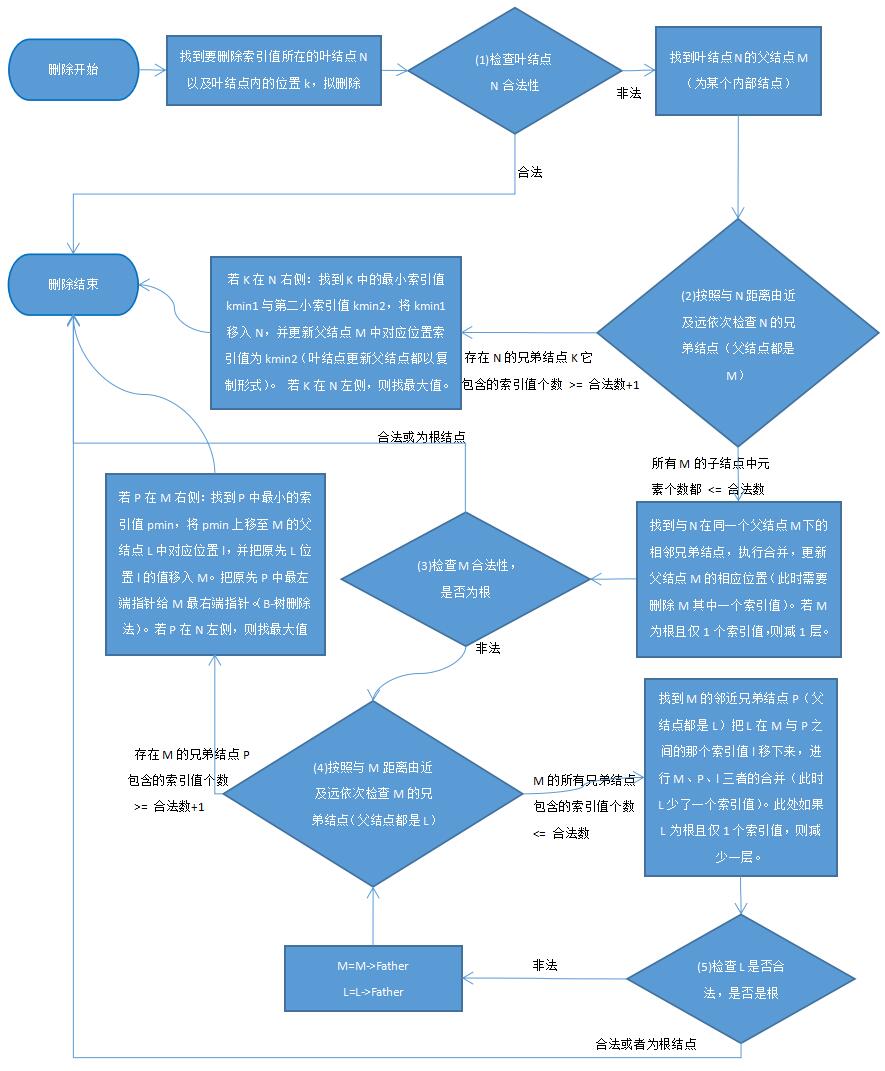
**3.2.2 B+树的插入**

B+树的插入流程较为繁琐。首先我们要在查找到叶节点中该点的对应位置，并进行插入操作，先插入，再检查合法性，如果不合法即超出了关键字存储数量，那么就分裂该节点，并将右边节点的最小关键字上移至父节点，父节点为内部节点，检查其合法性，如果超出存储数量那么父节点也要分裂，这样一层一层分裂上去直至根节点，如果根节点也不合法，那么根节点也要分裂并产生新的根节点。具体流程见下图。



**3.2.3 B+树的删除**

相比插入，删除过程更为复杂，因为存在着删除之后关键字数量不够从而合并节点的过程。其具体流程见下图。



1. **问题及解决方法**

问题1：如何实现数据落盘持久化

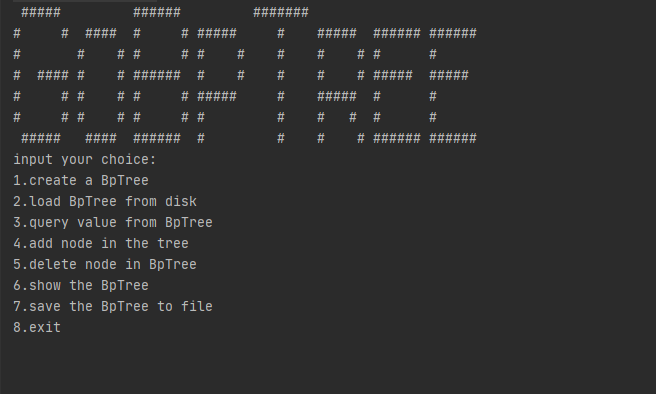
解决方法：由于是模拟磁盘的B+树索引机制，我们需要自定义一种B+树存储的格式，并且要求能从磁盘中读出并且恢复B+树，由于在内存中树与树之间是以指针和地址的形式连接起来，很明显不可以直接保存到磁盘中，这里我选择保存树的key与value，然后运行时重建B+树。

问题2：如何实现磁盘中保存的永远是加密的B+树的问题

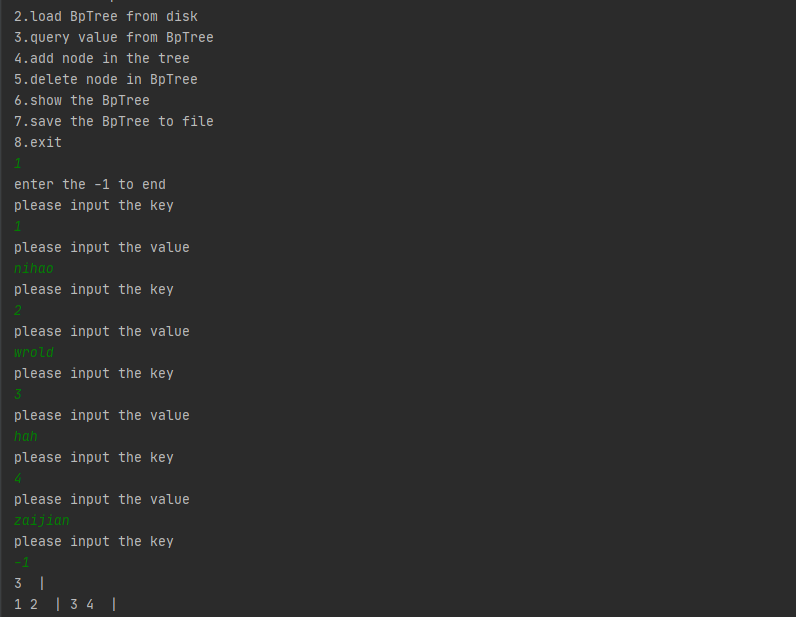
解决方法：考虑到安全性，我想要在磁盘中保存的都是加密的B+树，因此我会在整个程序中添加一个aes加密文件的部分，每次保存B+树时会调用aes加密文件，key的话可以自己设置也可以随机生成，当我们需要从一个加密的文件中恢复B+树时，我们需要首先解密文件，从文件中读取出key和value之后立即加密文件，这样可以保证文件不明文落地存储。

1. **实验结果演示**

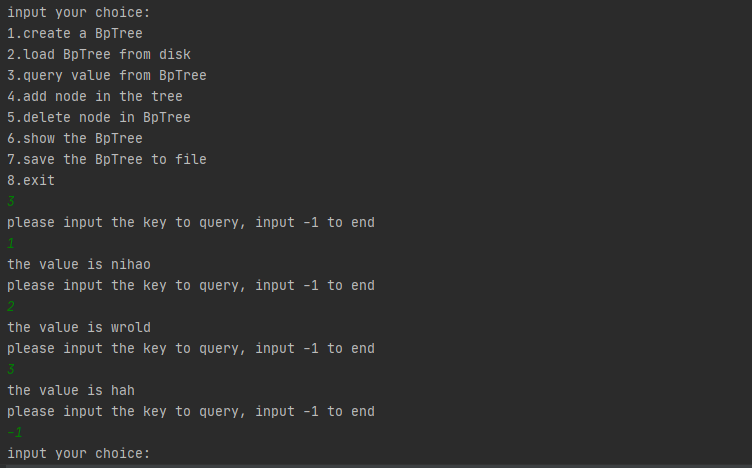
初始化：



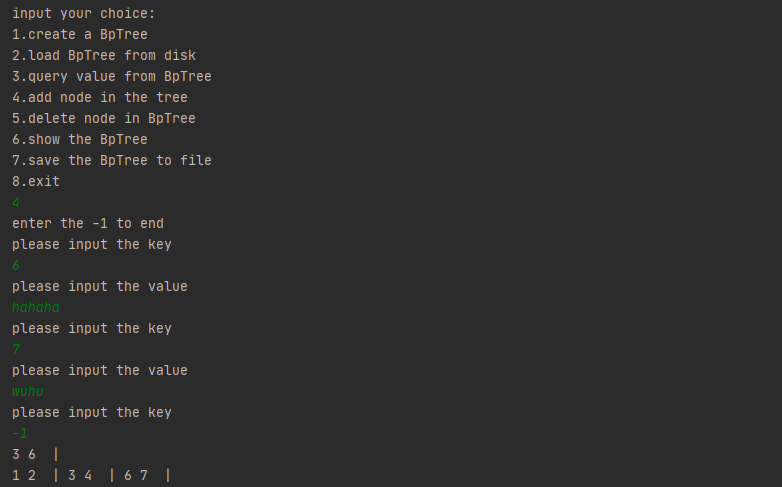
创建B+树



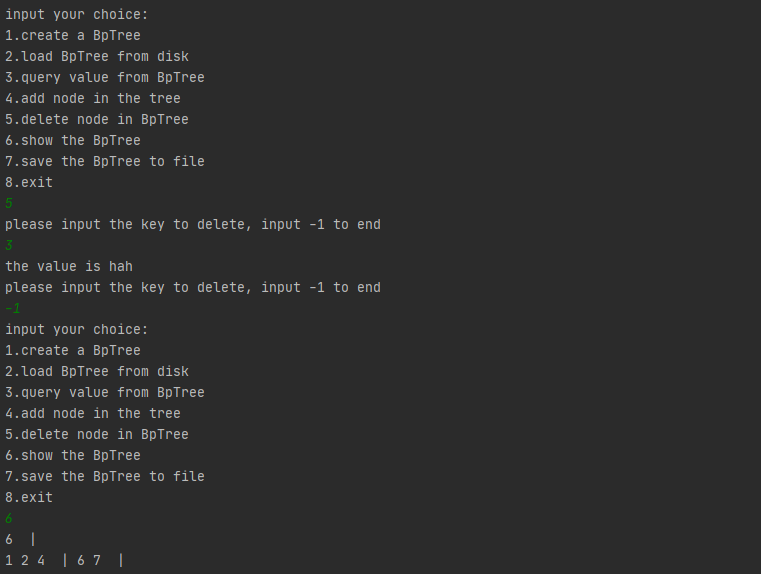
从树中查询



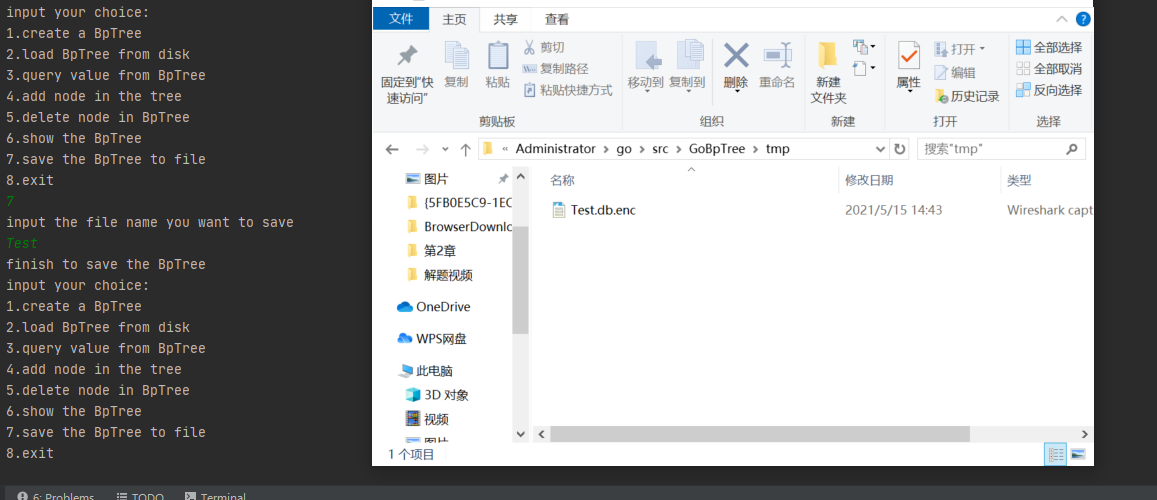
向树中添加结点

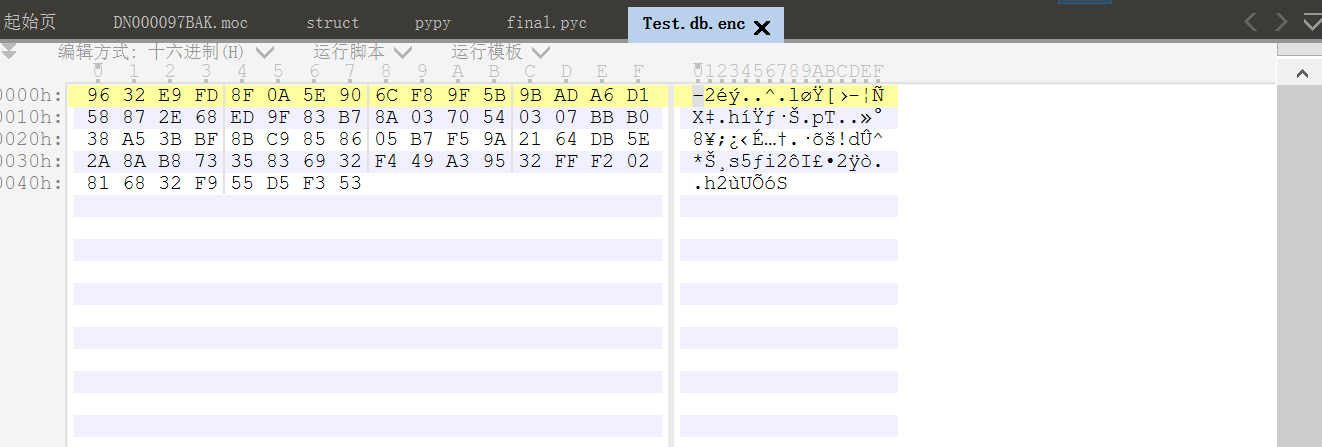


删除单个结点

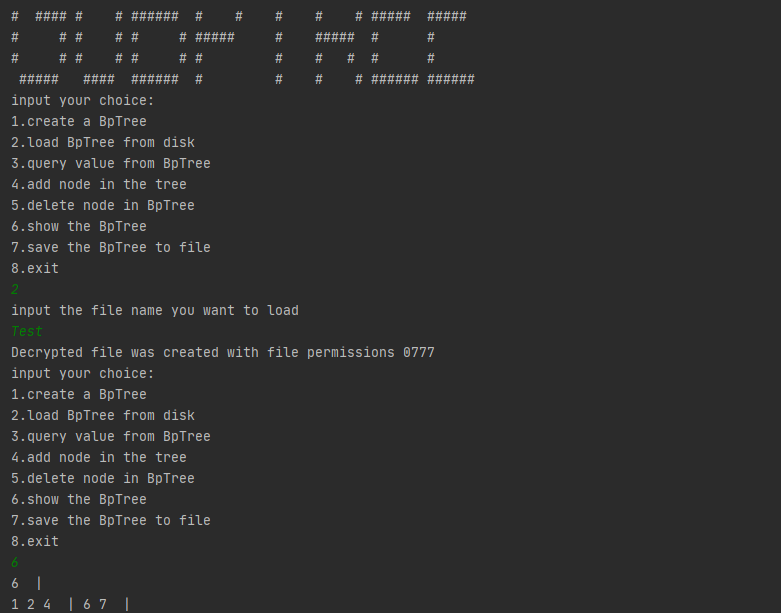


B+树落盘加密存储





从磁盘中加载B+树



1. **感想与体会**

这次DBMS实验让我对数据库原理有了新的体会，同时系统学习了B+树的建立及其功能，我的代码能力也有了很大提高。本来想制作一个精美的GUI，无奈go语言的gui实在是不人性化，考虑到本身是做安全的，因此给程序添加了一点文件加密的功能。最后感谢秦老师的教导，祝老师身体健康，工作顺利！