Index Manager 模块设计

计算机学院 软件工程 1201 尹依婷 3120102057

一、模块概述

Index Manager 负责 B+树索引的实现，实现 B+树的创建和删除（由索引的定义与删除引起）、等值查找、插入键值、删除键值等操作，并对外提供相应的接口。

B+树中节点大小应与缓冲区的块大小相同，B+树的叉数由节点大小与索引键大小计算得到。

二、主要功能

创建索引：若语句执行成功，则输出执行成功信息；若失败，必须告诉用户失败的原因。
删除索引：若语句执行成功，则输出执行成功信息；若失败，必须告诉用户失败的原因。
等值查找：根据所查找的索引和键值返回拥有该键值的记录在表中的位置（块号和块中偏移），若语句执行成功，则返回位置信息，若失败则返回 null 或抛出异常。
插入键值：在索引中插入新键值，若键值已存在，则更新相应位置信息。
删除键值：在索引中删除某个键值，若该键值不存在则什么也不做。

三、对外提供的接口

1. 创建索引
```
public void CreateBPlusTree(CString KeyType, int KeyTypeCount);
```
2. 删除索引
```
public void Drop();
```
3. 插入新的索引值
```
public void InsertValue(void* pValue, int Block, int Index);
```
4. 删除索引值
```
public void DeleteValue(void* pValue);
```
5. 等值查找
```
public bool indexmanager::FindValue(void* pValue, int& Block, int& Index)
```

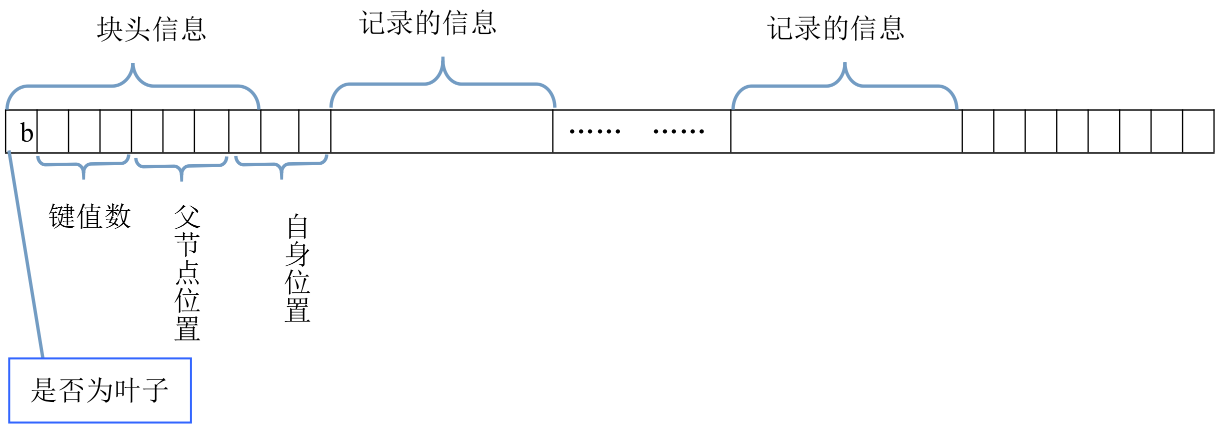
四、设计思路

Index 的实现主要依赖与 B+树的实现，首先构造结点数据结构，也是最主要的数据结构，它包含了诸多信息：是否为叶节点、父节点位置、自身位置、指向所在树的指针。

结点类

```
class Node
{
	bool m_bLeaf;
	int m_iCount;
	TKeyType* m_Key[FANOUT-1];
	int m_iPointer[FANOUT][2];
	int m_iFatherPosition[2];
	int m_iSelfPosition[2];
	int m_iKeyTypeCount;
	CFile* m_file;
	indexmanager* m_BTree;
public:
	  ……
//下文具体阐述
}
```

结点的结构



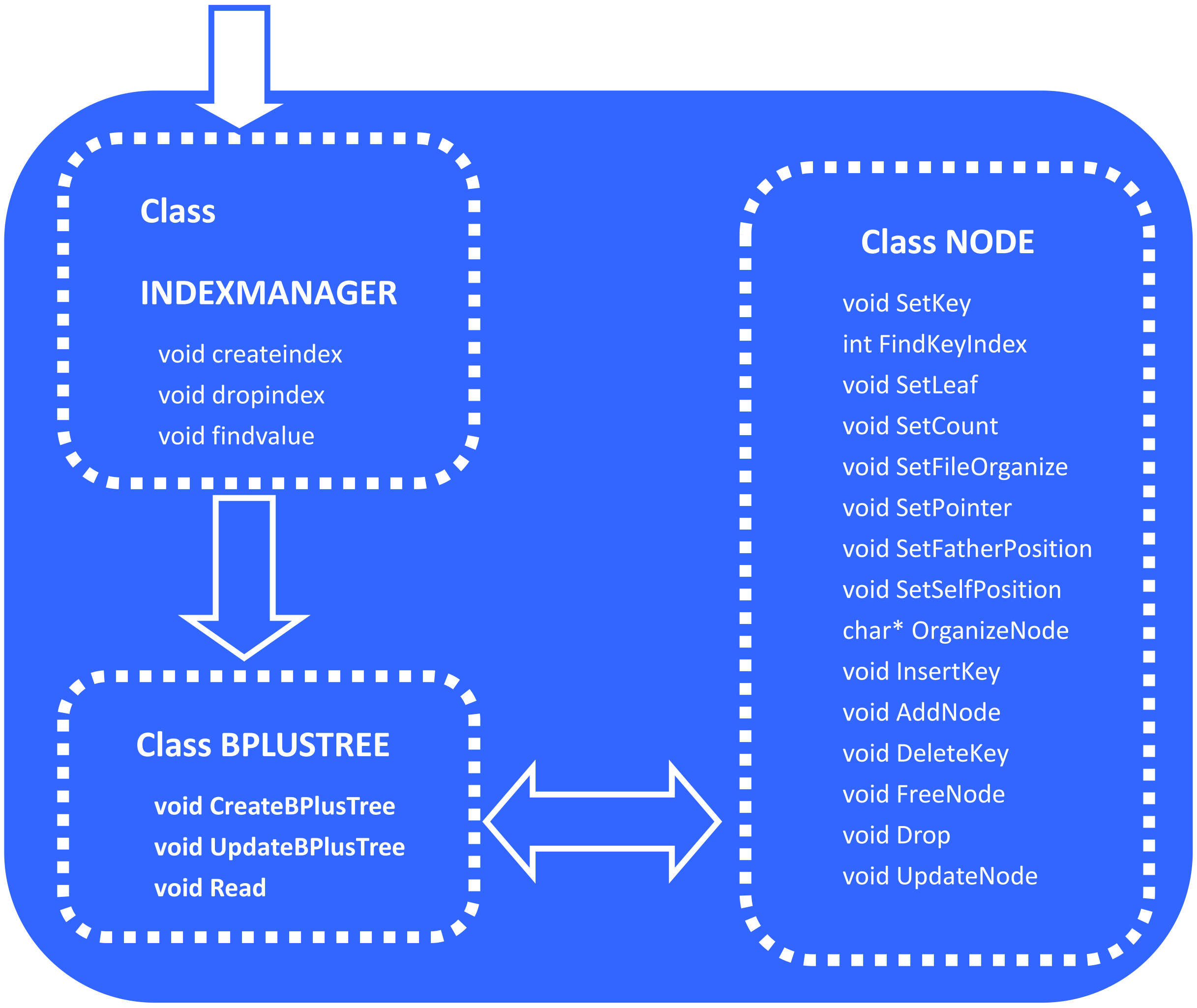
结点的初始化：

```
template <class TKeyType>
Node<TKeyType>::Node(int TypeCount, indexmanager* Tree)
{
	m_iCount = -1;
	m_bLeaf = 1;
	m_file = NULL;
	int i;
	for(i = 0; i < FANOUT; ++i) {
		m_iPointer[i][0] = -1;
		m_iPointer[i][1] = -1;
	}

	m_iFatherPosition[0] = -1;
	m_iSelfPosition[0] = -1;
```

```
    m_iFatherPosition[1] = -1;
    m_iSelfPosition[1] = -1;

    m_iKeyTypeCount = TypeCount;
    m_BTree = Tree;

    for(int j = 0; i < FANOUT - 1; ++j)
        m_Key[j] = new TKeyType[m_iKeyTypeCount];
}
```

五、整体架构

外部传入的对于 index 的操作具体由 B+树类中的操作承担，再细化到 NODE 类操作。BPLUSTREE 类主要三个函数（创建、更新、读取），而 NODE 类包括所有细节的操作，下图省略了 NODE 类读取相关位置信息的函数。



六、树类操作

创建 B+树

```
void indexmanager::CreateBPlusTree(CString KeyType, int KeyTypeCount)
{
	if(m_file == NULL)
		throw new Exception("init before create b+tree");
	m_sKeyType = KeyType;
	m_iKeyTypeCount = KeyTypeCount;
	if((m_sKeyType != "char" && m_sKeyType != "int" && m_sKeyType != "float") ||
		m_iKeyTypeCount <= 0)
		throw new Exception("error input for create B+ Tree");
	if(m_sKeyType=="char") {
		Node<char> tmpNode(m_iKeyTypeCount + 1, this);
		tmpNode.SetFileOrganize(m_file);
		tmpNode.SetCount(0);
		tmpNode.SetLeaf(1);
		tmpNode.AddNode();
		tmpNode.GetSelfPosition(m_iRoot[0], m_iRoot[1]);
	}
	m_iFirstLeaf[0]=m_iRoot[0];
	m_iFirstLeaf[1]=m_iRoot[1];

	int Size = 0;
	Size = m_sKeyType.GetLength() + 1 + sizeof(int) * 5;
	char* c = new char[Size];
	char* p = c;
	p += m_sKeyType.GetLength();
	*p = '\0';
	p++;
	*(int*)p = m_iKeyTypeCount;
	p += sizeof(int);
	((int*)p)[0] = m_iRoot[0];
	((int*)p)[1] = m_iRoot[1];
	p += (2*sizeof(int));
	((int*)p)[0] = m_iFirstLeaf[0];
	((int*)p)[1] = m_iFirstLeaf[1];
	m_file->Write(&m_iTreePosition[0], sizeof(int)*2);
	delete[] c;
}
```

更新 B+树

```
void indexmanager::UpdateBPlusTree()
```

```
{
        int tmpBlock, tmpIndex;
        GetSelfPosition(tmpBlock, tmpIndex);
        int Size = 100;
        m_file->Write(&m_iRoot[0],2*sizeof(int));
        m_file->Write(&m_iRoot[1], 2*sizeof(int));
        m_file->Write(&m_iFirstLeaf[0], 2*sizeof(int));
        m_file->Write(&m_iRoot[0], sizeof(int)*2);
}
```

读取 B+树内容

```
void indexmanager::Read()
{
        if(m_file == NULL)
                throw new Exception("error now store in b+tree");
        if(m_iTreePosition[0] < 0 || m_iTreePosition[1] < 0    )
                throw new Exception("no init position before read b+ tree");

        int Size = 0;
        m_file->Write((void *)&m_iTreePosition[0], sizeof(int));
        char* c = new char[Size];
        char* p = c;
        m_file->Write((void *)&m_iTreePosition[0], sizeof(int));
        m_sKeyType = p;
        p += m_sKeyType.GetLength() + 1;

        m_iKeyTypeCount = *(int*)p;
        p += sizeof(int);
        m_iRoot[0] = ((int*)p)[0];
        m_iRoot[1] = ((int*)p)[1];
        p += 2 * sizeof(int);
        m_iFirstLeaf[0] = ((int*)p)[0];
        m_iFirstLeaf[1] = ((int*)p)[1];
        delete[] c;
}
```

七、关键函数及代码

插入索引值

```
void indexmanager::InsertValue(void *pValue, int Block, int Index)
{
        if(m_iRoot[0] < 0 || m_iRoot[1] < 0 || m_file == NULL)
```

```
            throw new Exception("no root when find value in B+tree");
        if((m_sKeyType != "char" && m_sKeyType != "int"    && m_sKeyType != "float") ||
            m_iKeyTypeCount <= 0 )
            throw new Exception("error input for find value in b+");
        if(m_sKeyType == "TYPE") {
          //the “TYPE”refers to diffent type such as char,int,float
            Node<TYPR> tmpNode(m_iKeyTypeCount + 1, this);
            tmpNode.SetFileOrganize(m_file);
            tmpNode.SetSelfPosition(m_iRoot[0], m_iRoot[1]);
            tmpNode.Read();

            int tmpBlock, tmpIndex;
            int FindIndex = 0;
            while(!tmpNode.IsLeaf()) {
                FindIndex = tmpNode.FindGreaterKeyIndex((TYPE*)pValue);
                if(FindIndex == -1)
                    FindIndex = tmpNode.GetCount();
                tmpNode.GetPointer(FindIndex, tmpBlock, tmpIndex);
                tmpNode.SetSelfPosition(tmpBlock, tmpIndex);
                tmpNode.Read();
            }
            tmpNode.InsertKey(TYPEr*)pValue, Block, Index);
        }
}
```

删除索引值

```
void indexmanager::DeleteValue(void* pValue)
{
        if(m_iRoot[0] < 0 || m_iRoot[1] < 0 || m_file == NULL)
            throw new Exception("no root when find value in B+tree");
        if((m_sKeyType != "char" && m_sKeyType != "int" && m_sKeyType != "float") ||
            m_iKeyTypeCount <= 0 )
            throw new Exception("error input for find value in b+");
        if(m_sKeyType == "TYPE") {
            //the “TYPE”refers to diffent type such as char,int,float
            Node<TYPE> tmpNode(m_iKeyTypeCount + 1, this);
            tmpNode.SetFileOrganize(m_file);
            tmpNode.SetSelfPosition(m_iRoot[0], m_iRoot[1]);
            tmpNode.Read();
            while(!tmpNode.IsLeaf()) {
                FindIndex = tmpNode.FindGreaterKeyIndex((TYPE*)pValue);
                if(FindIndex == -1)
                    FindIndex = tmpNode.GetCount();
                tmpNode.GetPointer(FindIndex, tmpBlock, tmpIndex);
                tmpNode.SetSelfPosition(tmpBlock, tmpIndex);
                tmpNode.Read();
            }
            tmpNode.DeleteKey(tmpNode.FindKeyIndex((TYPE*)pValue));
    }
}
```

删除结点

```
void indexmanager::DropNode(int Block, int Index)
{
    if(m_file == NULL)
        throw new Exception("no root when find value in B+tree");
    if((m_sKeyType != "char" && m_sKeyType != "int"    && m_sKeyType != "float") ||
        m_iKeyTypeCount <= 0 )
        throw new Exception("error input for find value in b+");
    if(m_sKeyType == "TYPE") {
        //the “TYPE”refers to diffent type such as char,int,float
        Node<TYPE> tmpNode(m_iKeyTypeCount + 1, this);
        tmpNode.SetFileOrganize(m_file);
        tmpNode.SetSelfPosition(Block, Index);
        tmpNode.Read();
        int tmpBlock, tmpIndex;
        if(tmpNode.IsLeaf())
            tmpNode.FreeNode();
        else {
            for(int i = 0; i < tmpNode.GetCount(); ++i) {
                tmpNode.GetPointer(i, tmpBlock, tmpIndex);
                DropNode(tmpBlock, tmpIndex);
            }
            tmpNode.FreeNode();
        }
}
```

删除索引

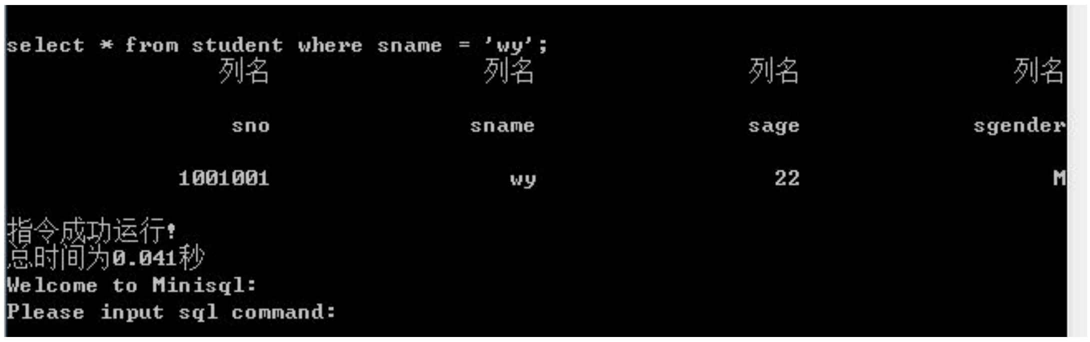
```
void indexmanager::Drop()
{
    if(m_iRoot[0] < 0 || m_iRoot[1] < 0 ||
        m_iFirstLeaf[0] < 0 || m_iFirstLeaf[1] < 0
        || m_file == NULL)
        throw new Exception("error when drop B+");
    DropNode(m_iRoot[0], m_iRoot[1]);

    m_sKeyType = "";
    m_iKeyTypeCount = 0;
    m_iRoot[0] = -1;
    m_iRoot[1] = -1;
    m_iFirstLeaf[0] = -1;
    m_iFirstLeaf[1] = -1;

    m_iTreePosition[0] = -1;
    m_iTreePosition[1] = -1;
}
```

等值查找

```
bool indexmanager::FindValue(void* pValue, int& Block, int& Index)
{
    if(m_iRoot[0] < 0 || m_iRoot[1] < 0 || m_file == NULL)
        throw new Exception("no root when find value in B+tree");
    if((m_sKeyType != "char" && m_sKeyType != "int" && m_sKeyType != "float") ||
        m_iKeyTypeCount <= 0)
        throw new Exception("error input for find value in b+");
    if(m_sKeyType == "TYPE") {
      //the "TYPE"refers to diffent type such as char,int,float
        Node<TYPE> tmpNode(m_iKeyTypeCount + 1, this);
        tmpNode.SetFileOrganize(m_file);
        tmpNode.SetSelfPosition(m_iRoot[0],m_iRoot[1]);
        tmpNode.Read();
        int tmpBlock, tmpIndex;
        int FindIndex = 0;
        while(!tmpNode.IsLeaf()) {
            FindIndex = tmpNode.FindGreaterKeyIndex((TYPE*)pValue);
            if(FindIndex == -1)
                FindIndex = tmpNode.GetCount();
            tmpNode.GetPointer(FindIndex, tmpBlock, tmpIndex);
            tmpNode.SetSelfPosition(tmpBlock, tmpIndex);
            tmpNode.Read();
        }
        FindIndex = tmpNode.FindKeyIndex((TYPE*)pValue);
        if(FindIndex < 0)
            return false;
        tmpNode.GetPointer(FindIndex, Block, Index);
    }
    return true;
}
```

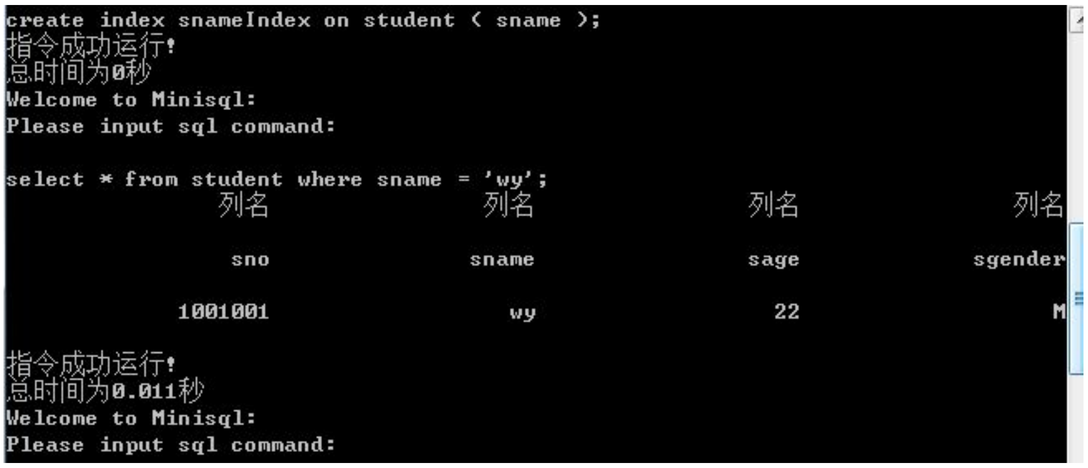
八、Index 的性能

创建 index 之前查找



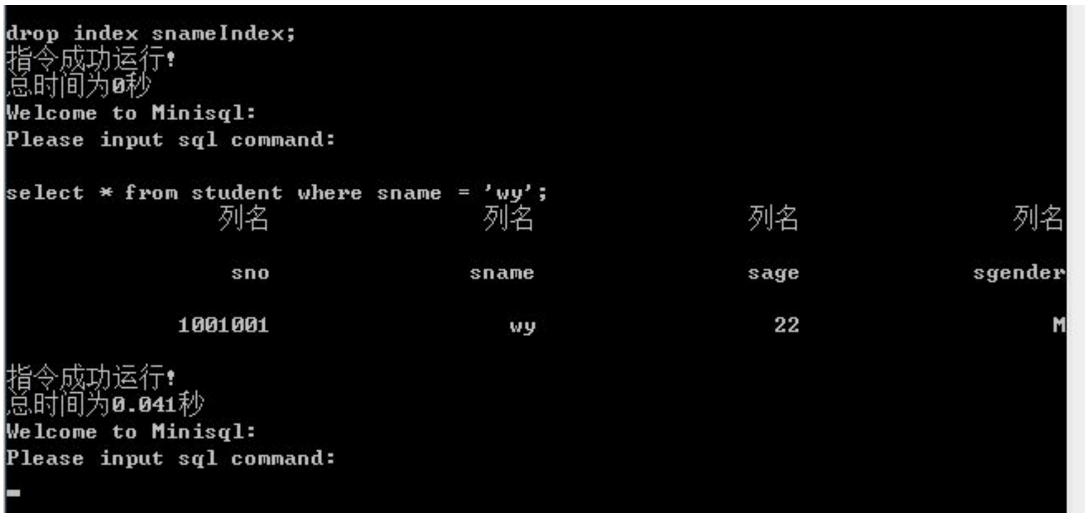
```
select * from student where sname = 'wy';
              列名                列名                列名                列名

               sno               sname               sage             sgender

           1001001                  wy                 22                   M

指令成功运行!
总时间为0.041秒
Welcome to Minisql:
Please input sql command:
```

创建 index 之后的查找



```
create index snameIndex on student ( sname );
指令成功运行!
总时间为0秒
Welcome to Minisql:
Please input sql command:

select * from student where sname = 'wy';
              列名                列名                列名                列名

               sno               sname               sage             sgender

           1001001                  wy                 22                   M

指令成功运行!
总时间为0.011秒
Welcome to Minisql:
Please input sql command:
```

删除 index 之后的查找



```
drop index snameIndex;
指令成功运行!
总时间为0秒
Welcome to Minisql:
Please input sql command:

select * from student where sname = 'wy';
              列名                列名                列名                列名

               sno               sname               sage             sgender

           1001001                  wy                 22                   M

指令成功运行!
总时间为0.041秒
Welcome to Minisql:
Please input sql command:
```