# IndexManagement

**刘轩铭**

## 主要功能

- 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。
- B+树对值的类型（type）支持int，float和char三种。

## 接口说明

```
API:
    bool CreateIndex(Table &table, Index &index);
    bool CreateIndex(Table &table);
    bool HasIndex(std::string &TableName, std::string &AttribName);
    bool DropIndex(std::string &TableName, std::string &AttribName);
    void SaveIndex();
    bool LoadIndex(std::string &TableName, std::string &AttribName, int
type);
    bool InsertItem(std::string &value, Pointer pointer, int type);
    bool DeleteItem(std::string &value, int type) ;
    bool DeletePointers(std::vector<Pointer> &pointers, int type);
    bool SelectItem(Condition &cond, int type, std::vector<Pointer>
&pointers);
    Pointer FindPointer(std::string& value, int type);
    bool Find(std::string& value, int type);
```

- 本模块提供Index的相关操作给顶层模块使用。
  - CreateIndex(Table &table, Index &index);

    CreateIndex(Table &table);

    顶层模块获取并解析table和index的相关信息，然后调用该API进行索引的创建。

    其中第一个API是在用户指定某属性为索引的情况下进行创建，后者是在初次建立表的时候对主码创建索引。
  - LoadIndex(std::string &TableName, std::string &AttribName, int type);

    由于内存中每个时刻都只有一棵B+树生存，所以当每次调用索引相关函数时，都应该通过指定表和属性的名字，以及值的类型，进行B+树的导入。
  - HasIndex(std::string &TableName, std::string &AttribName);

    DropIndex(std::string &TableName, std::string &AttribName);

    void SaveIndex();

    顶层模块获取相关信息对索引进行各项操作。其中SaveIndex这个API应该在数据库退出的时候调用，保存当前的B+树到文件。
  - InsertItem(std::string &value, Pointer pointer, int type);

DeleteItem(std::string &value, int type) ;

SelectItem(Condition &cond, int type, std::vector &pointers);

通过索引增删查键值。

## 模块说明

- 模块定义

```
typedef int Pointer;
```

Pointer作为一个标记，指明将要读取的数据在数据文件里面的位置（也就是第几个，偏移量）

- 为了实现B+树索引的功能，首先需要实现对B+树的创建，删除，查找，插入等功能。其底层实现由Node类进行完成。

```
class Node{
public:
    int MaxChildrens;
    int numElements;
    T *values;
    Pointer *element;
    Node **children;
    Node *parent;
    Node *before;
    Node *next;

public:
    Node() {};
    Node(int _MAXChildrens = defaultMaxChildrens);
    void Save(FILE *fp);
    void Load(FILE *fp);
    ~Node();
```

模块的底层由Node类进行管理，对应B+树中的一个节点，其成员变量分别是节点的相关信息。定义的函数Save和Load分别用来递归地对节点进行数据的储存和读取。

- 之后实现B+树的功能，具体功能由BPlusTree类进行实现

```
class BPlusTree{
private:
    Node<T> *root;
    string TableName;
    string AttribName;
    int MaxChildrens;

public:
    BPlusTree() {};
    BPlusTree(std::string &FileName);
    BPlusTree(std::string &_TableName, std::string &_AttribName, int
_MaxChildrens);
    bool Find(const T &data, Pointer &pointer);
    Pointer FindPointer(const T& data);
    int  FindLess(T &data, bool CanEqual, vector<Pointer> &pointers);
    int  FindLarger(T &data, bool CanEqual, vector<Pointer> &pointers);
```

```
    int  FindNonEqual(T &data, vector<Pointer> &pointers);
    int  FindBetween(T &data1, bool CanEqual1, T &data2, bool CanEqual2,
std::vector<Pointer> &pointers);
    int  CalNodeNum(Node<T> *node);
    string GetSaveFileName();
    void Insert(const T &data, const Pointer &pointer);
    void SplitLeafNode(Node<T> *node);
    void SplitNode(Node<T> *node);
    void Delete(T &data);
    void DoSave(Node<T> *node, FILE *fp);
    void Save(const char *filename);
    void Load(const char *filename);
    bool DeletePointers(vector<Pointer> &pointers);
    void Update(T &data_old, T &data_new, Pointer pointer_new);
    int PrintAll();
    int PrintAllreverse();
    int PrintAll(std::ofstream &fout);
};
```

以上部分通过BPlusTree类实现B+树地相关操作。其中public部分的函数可以开放给IndexManager模块作为API调用。为了降低程序的耦合度，Index文件的保存和读取也在这一部分进行实现。

```
class IndexManager
{
public:
    std::string bptIntName;
    std::string bptFloatName;
    std::string bptStringName;
    BPlusTree<int> *bpt_INT;
    BPlusTree<float> *bpt_FLOAT;
    BPlusTree<string> *bpt_STRING;

public:
    IndexManager();
    bool CreateIndex(Table &table, Index &index);
    bool CreateIndex(Table &table);
    bool HasIndex(std::string &TableName, std::string &AttribName);
    bool DropIndex(std::string &TableName, std::string &AttribName);
    void ResetBptInt(bool save);
    void ResetBptFloat(bool save);
    void ResetBptString(bool save);
    void SaveIndex();
    bool LoadIndex(std::string &TableName, std::string &AttribName, int
type);
    bool InsertItem(std::string &value, Pointer pointer, int type);
    bool DeleteItem(std::string &value, int type) ;
    bool DeletePointers(std::vector<Pointer> &pointers, int type);
    bool SelectItem(Condition &cond, int type, std::vector<Pointer>
&pointers);
    Pointer FindPointer(std::string& value, int type);
    bool Find(std::string& value, int type);
    ~IndexManager();
};
```

在IndexManager中，通过调用BPlusTree的API，以及相应的索引信息，来完成对索引的建立，删除，保存，导入等功能。对于每个API，上层的API模块和Interpreter模块只需要给出相应的参数信息，便可以得到返回的结果。