# B+树索引的实现

# 设计思想：

B+树的非叶子节点和叶子节点的存储信息是不同的，非叶子节点存储索引和节点指针，叶子节点存储索引、索引指向的记录的位置信息和右边相邻节点的指针。B+树设置一个索引参数N,表示一个除根节点外节点内至少有N/2个索引。如果节点的索引有M个，则要有M+1个节点指针。

B+树的建立：把要建立索引的文件选择一个属性作为索引读入到内存中，然后递归建树，每一层，根据设置的索引参数把索引数组分割，把分割出来的索引数组的第一个索引设置为这个节点的索引和每个分割出来的索引数组作为儿子节点，然后递归处理。把每个节点都写入一个文件，作为索引文件存入磁盘中。

B+树的查找：设置一个合理的索引参数使得B+树有根节点、内部节点和叶子节点这三层。一个节点内的索引是有序的，故可以使用二分查找。对于单点查询，先进行一个磁盘I/O，把根节点读入内存，进行一次二分查找，找到这个索引所在的下一节点的节点指针，在进行一次磁盘I/O，把节点读入到内存，在进行一次二分查找，得到这个索引所在的叶子节点的指针，在进行一次磁盘I/O，把这个叶子节点读入内存。在叶子节点上进行二分查找，得到这个索引指向的记录位置的信息（记录所在的数据库文件，所在文件的行号等信息）。B+树还可以进行范围查询，查询思想是先单点查询区间的最左值，找到最左值所在的节点，然后看区间的右节点是否超过这个节点的最大索引，如果没超过，说明这个区间在这个节点里可以获得，通过线性遍历找到区间所有的索引；如果超出这个节点的最大索引，则通过节点右边相邻节点指针从磁盘中取出右边相邻的叶子节点，进行同样的操作后即可获得区间的所有索引。

B+树的插入：插入过程中，按照单点查找的步骤找到该索引应该插入的叶节点，这时两种情况。第一种情况是这个节点没有满，直接插入就行了。第二种情况是这个节点满了，就要分裂这个节点，然后向上更新索引。如果发现父节点也满了就要分裂父节点，同样的操作直到根节点。

B+树的删除：删除过程中按照单点查找的步骤找到该索引所在的叶节点，这时有两种情况，第一种情况是删除这个节点不会破坏B+树节点索引数量的限制，直接删除。第二种情况是删除这个节点会破坏了B+树节点索引数量的限制，这时若相邻的叶节点的索引数量高于规定的最小值，则从这里拿一个索引补上，否则与相邻的节点合并，这时要递归向上删除父节点的一个儿子节点。同时注意更新父节点的索引值。

根据B+树的定义和操作可以把B+树实现出来。

# **实现方案**：

## 数据结构：

struct BplusTreeNode {

bool isleaf; //是否是叶子节点

int index[N]; //索引

BplusTreeNode\* children[N + 1]; //儿子节点指针

BplusTreeNode\* next; //叶子节点的相邻节点指针

PositionMessage data; //记录所在位置的信息

}；

## 文件结构：

数据库文件：

数据库要建立B+树索引的表的csv文件

源程序文件：

BplusTree.h //B+树头文件定义B+树节点和操作。

BplusTree.cpp //实现B+树的建树、查找、插入和删除操作。

索引文件：

建树时根据节点生成的索引文件，记录着B+树一个节点的信息。