由于组员分外勤劳，写出了带重根的情况的删除算法在B+树的实现。所以，应该会抛弃之前的假删除，当然要看时间是否允许。

在进行整合时发觉对于查找结果的处理不是特别好，因为直接以字符串形式返回结果的话，不管在处理得出结果上，还是在在拿到结果之后的处理上都比较麻烦。所以，希望用类似jdbc中的方法返回数据结果感觉可能会更好一点。

因此，又新建了ShabbyResultSet类。在向DataFile类给出初步搜索结果，也就是指针时，DataFile类会返回一条裸的数据。

在StorageEngine中将所有的裸数据打包，返回ShabbyResultSet的对象。通过调用ShabbyResultSet的方法，自助的访问各条数据。

采用类似ResultSet的方法也就意味着可以忽视属性的名称，但是必须知道每个数据的长度和偏移量。

问题：

在仍未真正开始进行数据写入而仅停留在对一个索引树的插入时，发现性能严重问题，完全无法达到目标，对于10k的长度为32的字符串（实际使用长度均为11）插入需要数分钟之久，一个有1.6k结果的范围查找需要90毫秒左右。（且是在固态硬盘中进行，包括生成随机字符串的时间）。要寻找性能瓶颈。

发现之前对于文件的写入，在每次写入方法中都要进行一次文件的打开与关闭（使用写回模式）。实验改用写同步模式，即进行写操作时就对文件进行修改，并让文件在对象仍指向其的时候始终处于打开状态。性能明显提升。

修改：

添加范围搜索方法和显示全部数据方法。

范围搜索方法：select from <tableName> <attributeName> between [ key1 key2 ]

将搜索闭区间key1，key2之间的所有数据。

显示全部数据：select from <tableName>

将指令注释掉：在行首加##