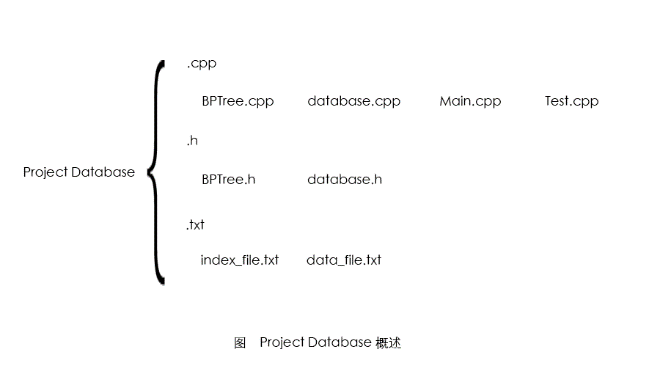
**MYDATABASE开发文档**

软件工程 罗安琪 515030910431

1. 概述

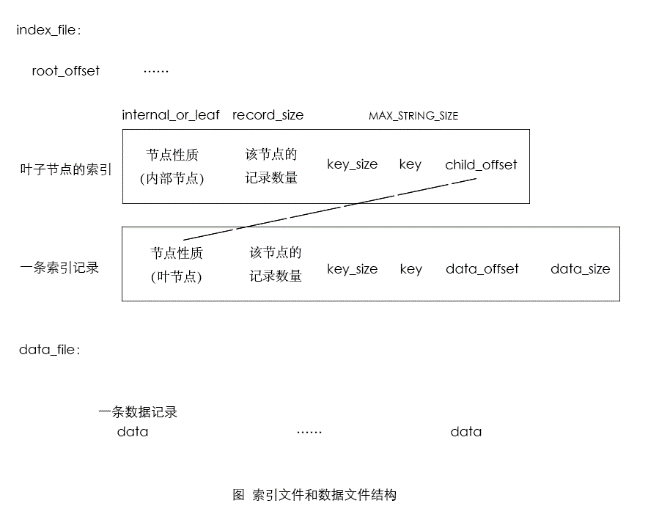
Project组成：



整个Project包含四个cpp文件、两个头文件和两个记录文件。内容包括B+树（BPTree.cpp BPTree.h）、数据库(database.cpp database.h)的实现以及增删查改的操作入口(Main.cpp)，和为了测试性能而写的随机生成数据（包含计时功能）的Test操作（Test.cpp）。

1. **总体架构**

1.记录文件

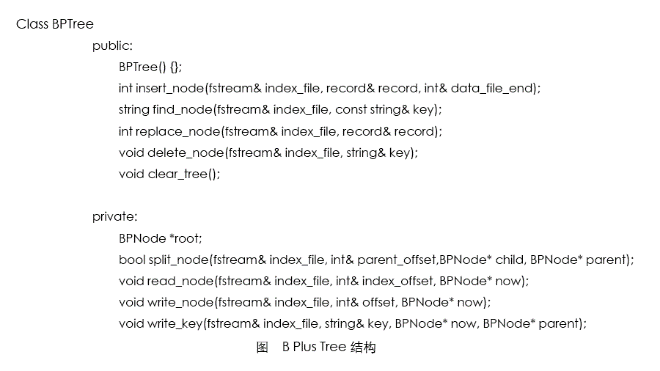
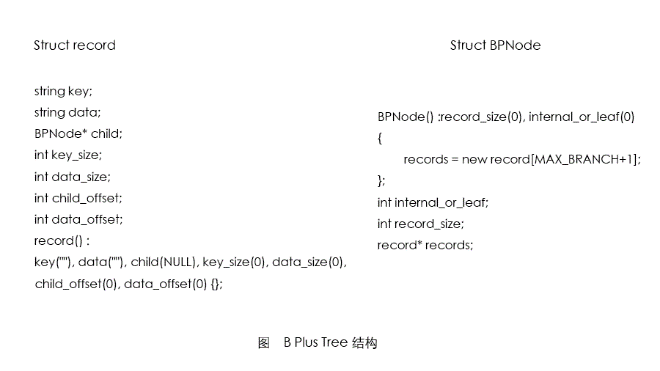
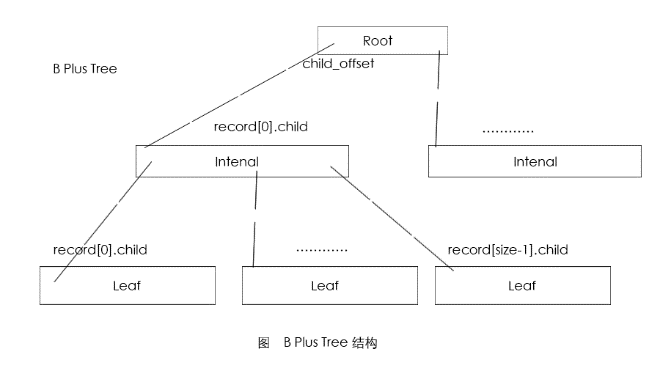


文件的存储和读取均是以二进制的形式。有索引(index\_file.txt)和数据(data\_file.txt)两个文件。

索引采用B+树的数据结构存储。数据文件为线性存储。

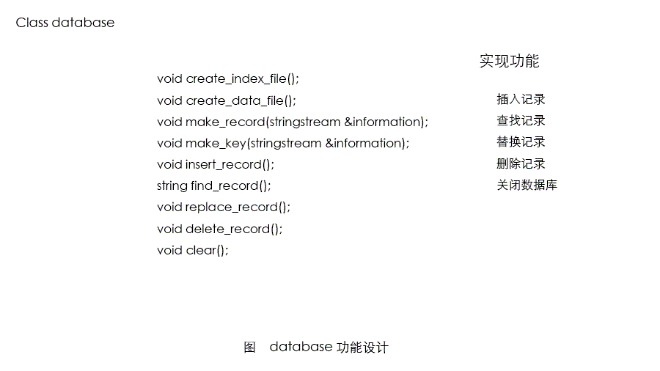
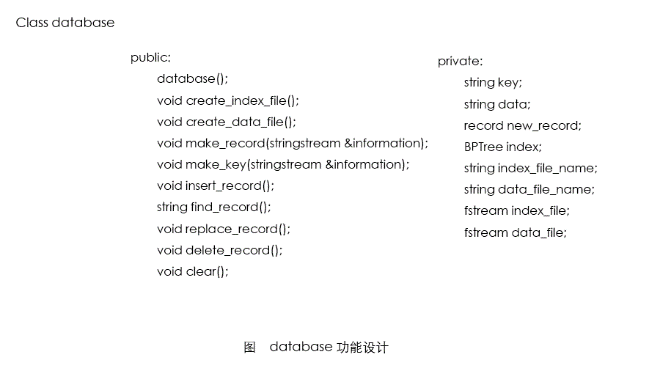
程序定义了DEGREE,MAX\_BRANCH,MIN\_BRANCH来辅助构成合法的B+树，定义了MAX\_STRING\_SIZE来限制记录的key的长度。

2. B Plus Tree



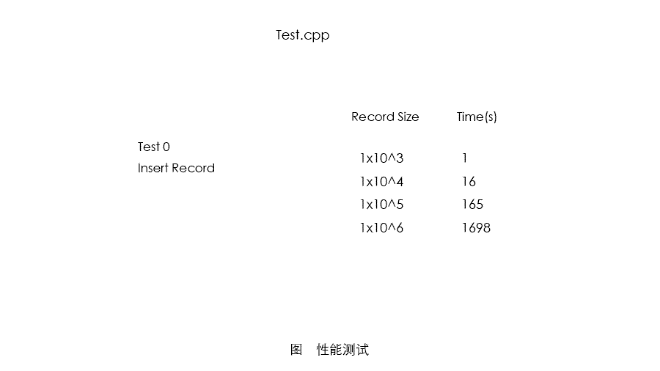
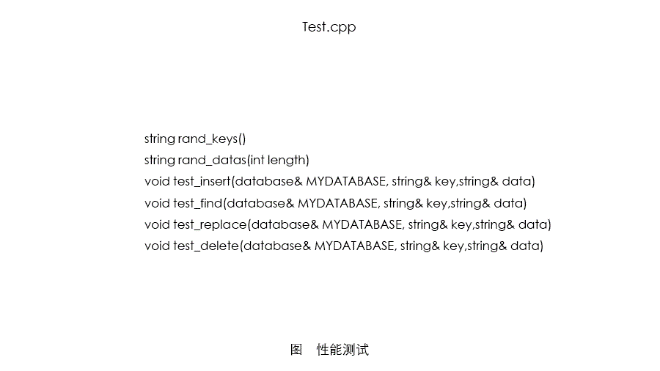
B+树中用到了两个自己定义的结构，一个是record，用于存储单条记录的信息，另一个是BPNode，用于存储B+树的节点。

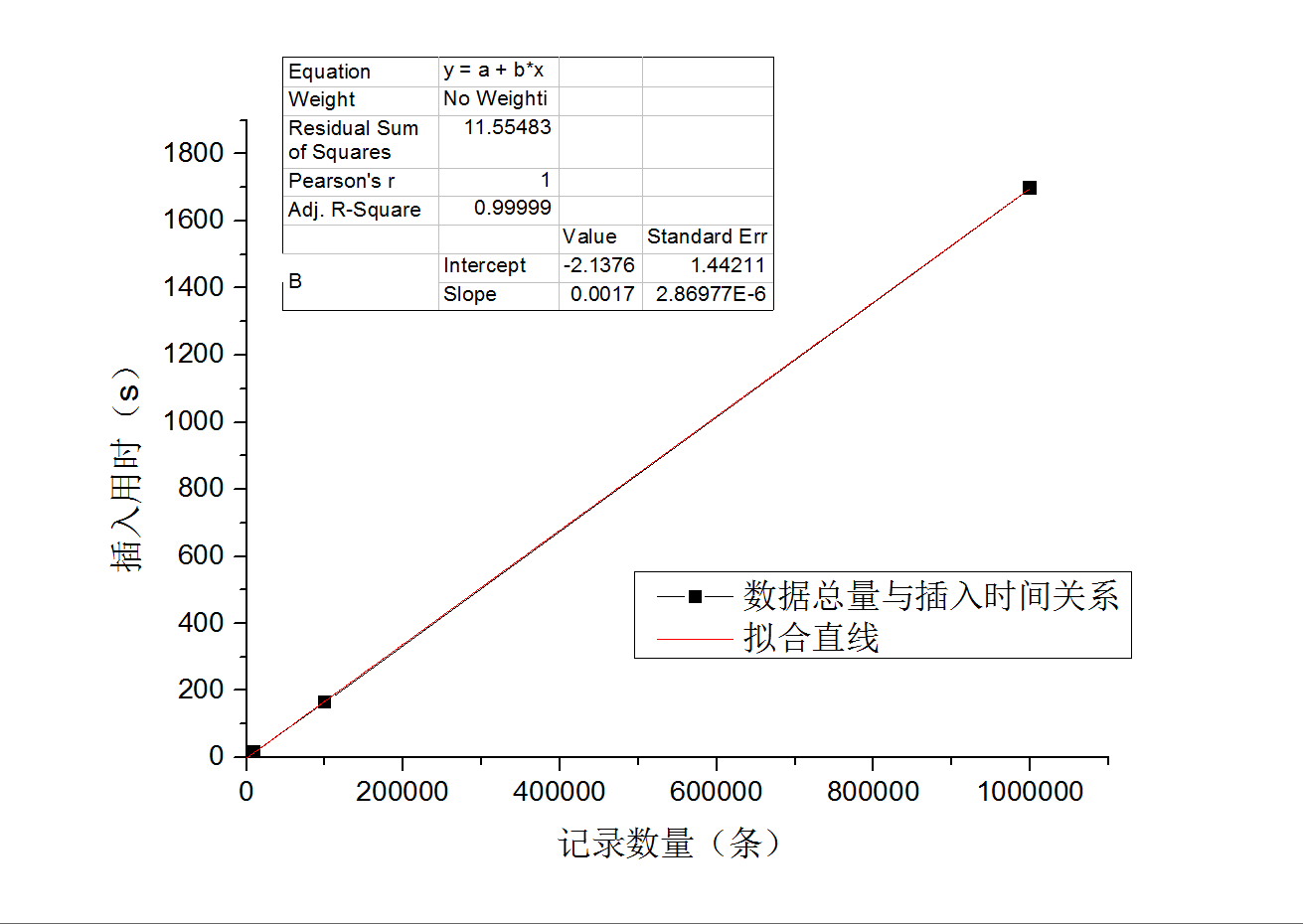
3. Database



Database中实现了插入记录、查找记录、替换记录和删除记录的功能。

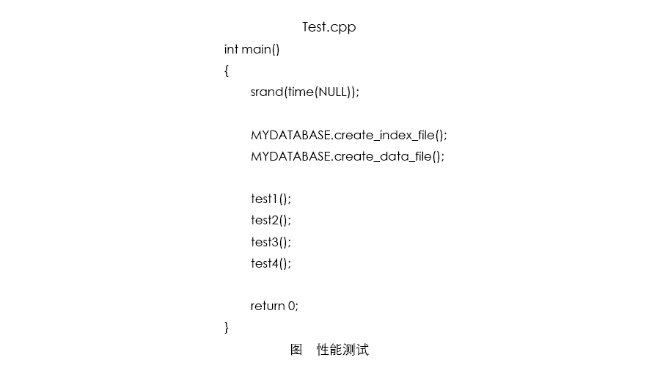
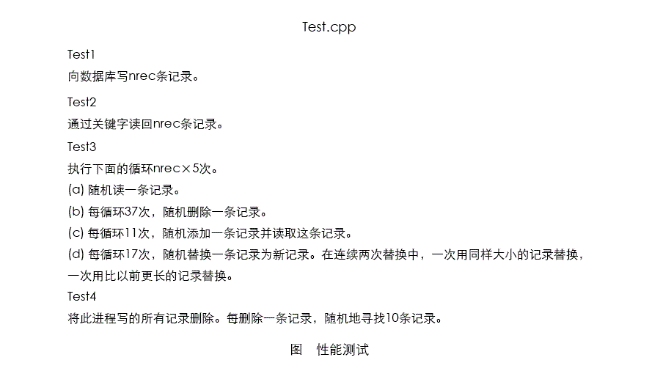
1. **性能测试**



程序耗时间的原因有两点，一是B+树的读写，测试中只涉及到写入。写入时应该是将一个节点的信息合并成一个string写入文件，这样一次只用进行一次写入操作，可以大大节省时间。这里由于时间原因未对此处作出优化，因而程序耗时较长。

此外程序中进行了较多的比较操作，有的地方是两个节点的比较，或者是节点内部某信息的比较，都比较花时间，而且有些反复使用的量应该通过中变量先保存，这样也可以减少时间开销。

由于此测试时，B+树都同步存进了内存，故时间和数据几乎线性相关，如果关闭后重新使用数据库，从文件中读取而重构B+树，随着数据量的增加而导致B+树层数增加，时间应该不再呈线性。由于本程序读写耗时较长，未能做测试验证。



通过性能测试调整了B+树的DEGREE数目，比较粗略地测试了一下，最后选定的65。

1. **综述**

完成Project的过程中，一开始先了解了B+树，在开始写后不久发现如果给节点增加一个父节点的指针会方便很多，但考虑后还是打算按常见的B+树模式写。然而写的过程中，由于此Project要求从磁盘文件里重构B+树，并且实现内存和文件中B+树的同步，父指针的缺少给我关于程序的逻辑思考带来了一定障碍，最后是选择的用vector存储重构时的节点路径。

网络上B+树的定义版本较多，书写过程中由于考虑不同，并没有太多可以参考的地方，自己从无到有地书写，困难重重。并且写代码的方法还不够掌握，导致debug花了巨量时间。本来打算做的功能和优化都只得放弃。

**不足之处：**

1. 未成功实现B+树的delete操作，虽然写了借节点和合并节点两函数，但在测试时发现会有一些错误产生，考虑到时间因素采用了较简单的替代方法。
2. 索引和数据文件的大小持续增大，而没有对其作出控制。
3. 磁盘读写操作较多，并没有利用磁盘读取的特性以减少读取磁盘所用的时间。
4. 代码还可以继续优化，有些地方操作太繁复，可以更精简。
5. 没有对可能的错误输入作出较完备的预估和处理。