

**大规模信息系统构建导论个人报告**

姓 名：任重先

学 号：3190105258

专 业：软件工程

班 级：软工1902

2022年 5月 22日

## 个人开发过程与主要贡献

在本次大规模分布式minisql的开发中，我主要负责minisql部分的recordmanager与buffermanager的开发与设计，该部分也是minisql中最难的部分，在本次开发过程中，我主要是在五月初的前半个月中对该部分进行开发，实现了一个基于LRU算法的内存管理buffermanager以及一个负责表中数据存取的recordmanager并调用catalogmanager、indexmanager搜索正确的存取位置，并向外提供接口，在完成该两部分调试后，我又参与了后期的debug环节，与同组的同学首先保证minisql的完成，之后与其他两位同学整合，最终实现了大规模分布式minisql的开发。

## 个人模块设计

### 2.1 buffermanager

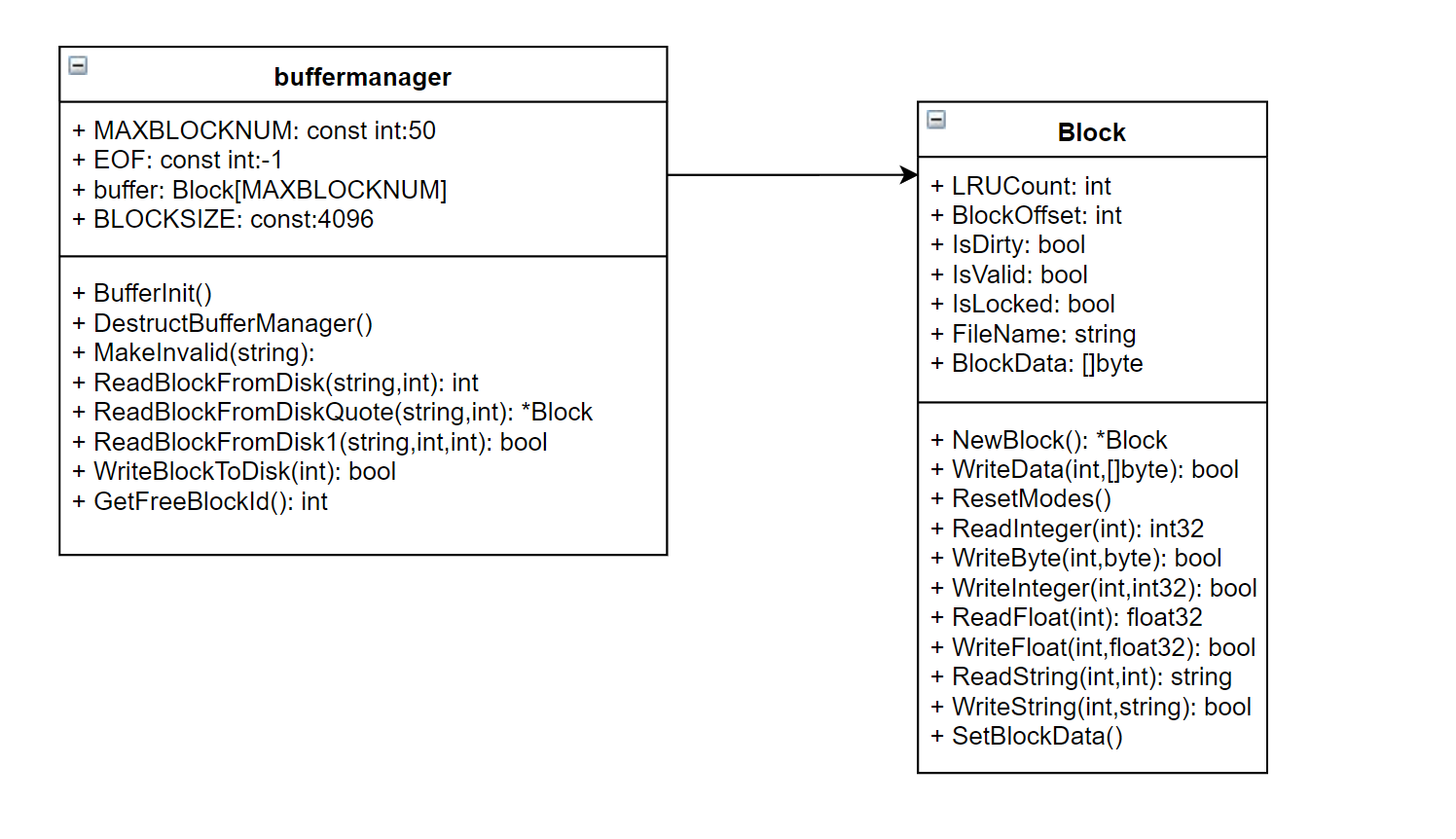
Buffermanager主要负责缓冲区内存的管理，主要功能包括：

* + 实现读写功能，可以将完成修改的block写入文件，或者从文件中读出block
  + 实现缓冲区的LRU算法，当缓冲区所有block都为invalid时，利用LRU算法将最久未使用的block写入文件
  + 对未修改的block在写入时，不需要写入，保留修改即可
  + 对正在使用的block不允许替换出去，避免出现冲突
  + 实现了不同数据类型的对块级别的写入读取操作，保证系统不会出现读取写入问题。

为了提高磁盘I/O的效率，缓冲区与文件系统交互基本单位是block，块的大小应该为文件系统和磁盘交互的整数倍，一般可以是4KB或者8KB，在本系统中设置为4KB，对于每一个块会在缓存池中预分配4096KB，对于每个region而言预分配了50个block作为缓冲池，在大数据测试中buffermanager的LRU算法得到了很好的检验，实现效果良好。

该部分向外提供了获得一个block的接口，当要获得的block已经在缓存池中则直接分配，如果不在缓存池中，则从文件中读入一个block，并向外提供了int,float,string三种类型在块级别上的读写。

以下是我设计的UML类图：



Buffermanager类图

Block类中的函数方法主要是在块层级上的操作，包括字符串、int、float的读写，以及块状态的修改。

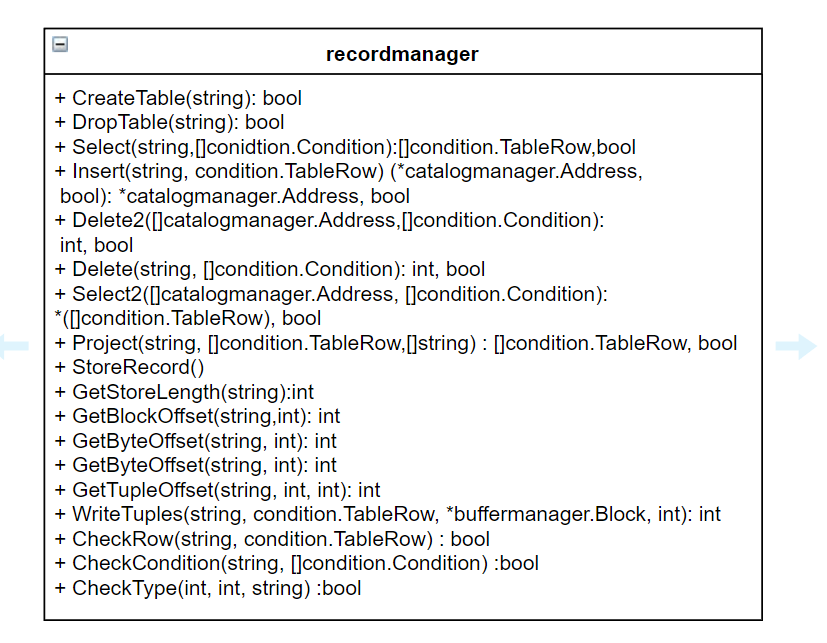
Buffermanager类中主要负责向其他模块提供获得块的接口主要通过调用ReadBlockFromDiskQuote方法，该方法会对当前缓存池中是否存在需要获得的块决定是否需要从文件中读入新的块，因此有可能调用ReadBlockFromDisk1从文件中获得一个新块。此外buffermanager还提供了可能的写入块、析构函数、获得当前缓存池中没有被使用块的信息等等。

### 2.2 recordmanager

Recordmanager负责管理与记录表中数据的数据文件，主要实现了数据文件的创建于删除（由表的定义与删除引起）、记录的插入、删除与查找操作，并对外提供相应的接口。其中插入支持在index上的插入，查找支持index搜索，实现了常规的sql语句操作。

在该部分设计了表数据的存储方式，该方式也是我们本次minisql的一大亮点，对于每个block，我们首先将前四位设置为freebyteoffset，初始值为-1，表示当前块中没有之前delete留存下的空余位置，若该freebyteoffse不为-1，则表示可以通过这四位索引值确定当前block中之前被删除空余下来的位置，优先在该位置写入信息，那么在select语句中是如何确定块中的所有位置，是否是一条有效记录可以被查找呢？我们采用对每一条记录开头使用-1作为状态位，如果该部分是有效数据，那么该位的值为-1，如果该部分数据为被delete后留存的空闲空间的话，首位不为-1，而改为前四个byte储存一个freebyteoffset，指向下一个空闲块位置，这样一来就解决了如果分割数据，以及如果进行delete的操作。

该部分的UML图如下：



## 个人心得

本次开发是一次很有挑战的经历，在这一过程中，我对于sql数据库的底层设计有了更深层次的理解，对于LRU算法也有了一个很好的认知，本次经历算是对于我在minisql中存在的一些疑问有了更深层次的理解，同时我对于分布式系统的开发，也有了一定的理解，在debug过程中也真正知晓了etcd,rpc等的工作方式，是一次很棒的开发体验。。