|  |  |  |
| --- | --- | --- |
| 2020春《数据库系统》实验报告 | | |
| 实验2：缓冲区管理器 | | |
| 姓名：陶飞宇 | 学号：1170300720 | 班级：1737301 |

## 1.实验分析

本次实验以现有框架为基础，通过以利用现有函数来实现核心函数的方法来完成缓冲区管理器的建立，需要对框架的各部分特别是函数的用法有清楚的认识才能顺利的完成任务。

## 2.实验过程

首先需要知道实验框架中，在构建函数内，给了我们两张关键的哈希表：

hashtable：存储着file（文件）、PageNo（磁盘页号）、frameNo（缓冲区框号）三者的映射关系，并配有insert、lookup和remove提供增删改的功能。

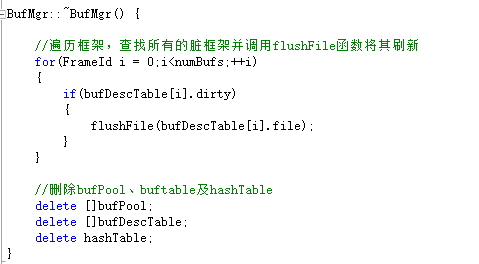
bufDescTable：存储着缓冲区中每个页框中的状态，关键属性除了hashtable中的三元组，还有pinCnt（引用计数）、dirty（脏位）、valid（有效位）和refbit（引用位，用于时钟算法）以及Clear（清除页框）、Set（设置上述变量）和Print（打印信息，未在实现的函数中使用）函数。

我们绝大多数的操作基于这两张表和异常抛出进行。

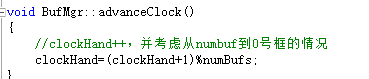
下面来分析各个函数的构建方法：

~BufMgr()：该函数要求我们“flush”所有脏页并取消分配缓冲区池和BufDesc表。

我们遍历缓冲区，找到所有脏位并调用flushFile函数，然后delete在构造函数中构建的bufPool（缓冲区）、BufDesctable和hashTable

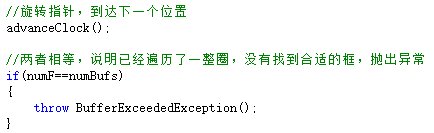


BufMgr::advanceClock()：该函数要求我们移动时钟臂，故而简单的将clockhand++并注意从最后一个到0的极限情况即可。

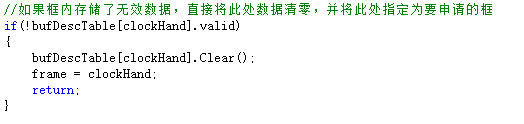


BufMgr::allocBuf：该函数要求我们使用时钟算法分配一个空闲帧，我们一步步的进行构造：

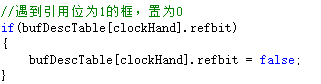
首先指定一个计数器，当其到达框的最大数时，代表我们没找到合适的空闲框，抛出异常：



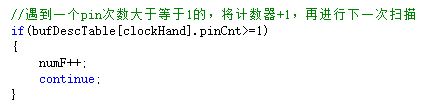
下一步，我们要确定时钟臂指向的框内的数据是有效的，如果无效，直接将其替换即可。



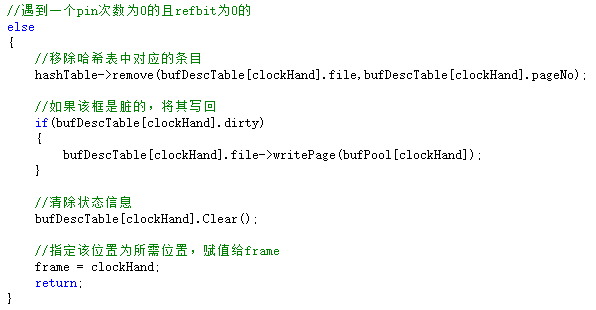
然后我们进行一般情况的处理，根据时钟算法，如果refbit为1，我们将其置零：



遇到被固定的框，我们将其跳过：



没有固定且refbit为0的框，则是我们要找的框，在选定这个框的同时要注意脏位写回的问题:

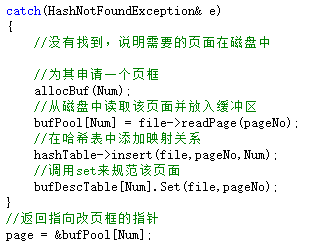


BufMgr::readPage：该函数模拟外界对于缓冲区中文件的读取，分为读到和没读到两种情况：

如果要读的文件在缓冲区中，则Num存储其页框标号，则将其refbit设置为true，并将pinCnt++：

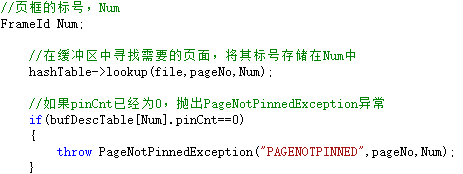


如果不在，则从磁盘中拉去所需的文件到缓冲区中，应使用allocbuf函数为该文件找到一个可用的空闲框，并调用两种哈希表的构造函数为页框中新的成员设置变量并建立哈希映射存储在表中，并在最后返回我们构建好的页面。



unPinPage：该函数是在模拟外部文件撤销对缓冲区中文件访问的情况，并利用dirty为模拟这次访问是否对缓冲区中的文件进行了更改。

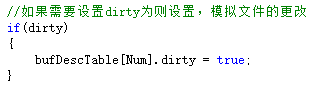
首先调用look函数寻找要撤销访问的页面，如果pinCnt已经为0，代表此时已经没有人访问这个页框，则抛出异常：



如果pinCnt正常，则将其递减：

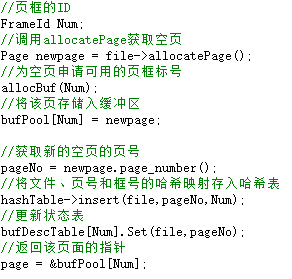


在最后对dirty进行处理，设置了则将dirty位置为true：

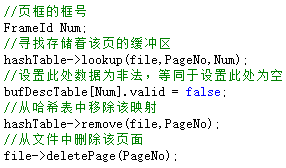


如果没找到则不仅任何处理。

allocPage：该函数是在磁盘中申请一个空页，并为空页配置其哈希映射，各项语句的作用已在注释中说明：

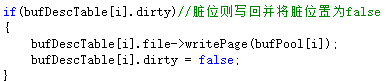


disposePage:该函数是在模拟从磁盘中删除一个已有页，并删除其相关的哈希映射关系，各项语句的作用已在注释中说明：



flushFile：该函数是在~BufMgr所调用的，所谓刷新函数，作用为在bufTable中扫描属于参数file的页框，细节为：

（a）如果页面是脏的，则调用file-> writePage（）将页面刷新到磁盘，然后将该页面的脏位设置为false：



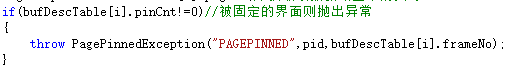
（b）从哈希表中删除该页面。 （无论页面是干净还是脏的）

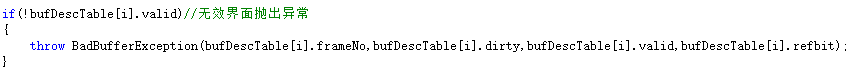


（c）为页框调用BufDesc的Clear（）方法



并且要抛出两个异常，分别是遇到了固定的界面和遇到了无效的页面时的情况：



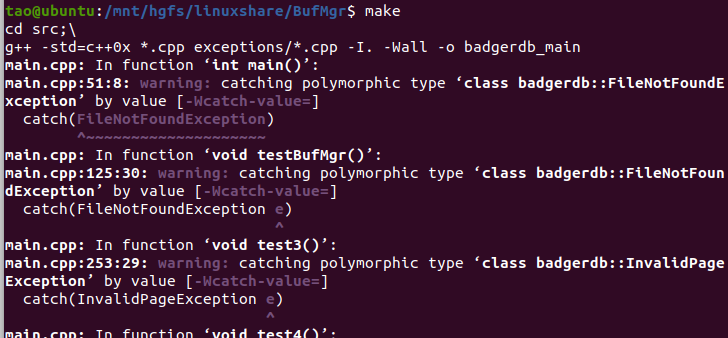


该部分的完整逻辑如下：

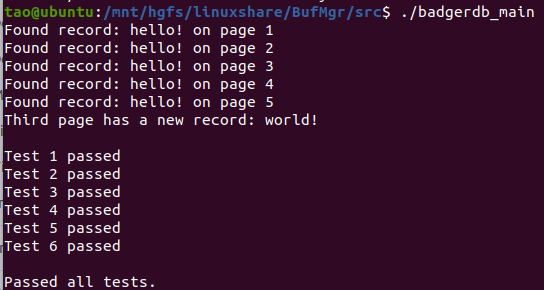


## 3.实验结果

在bufmgr使用make指令进行编译，虽然会抛出一些main函数中的异常，与同学讨论后发现是gcc版本的问题，不影响最终程序的运行：



在src目录下运行badgerdb\_main函数，成功通过了所有的测试：



## 4.心得体会

通过重复的阅读框架代码，完成了需要构建的八个函数，并从中了解了缓冲区管理器运行的机制，并复习了C++异常抛出的语法，收获很大。