

实验题目	缓存区管理器实现			实验日期	2022.04.03
班级	1903102	学号	L190201901	姓名	王博

# CS33503 数据库系统实验

## 实验检查记录

实验结果的正确性(60%)		表达能力(10%)	
实验过程的规范性(10%)		实验报告(20%)	
加分(5%)		总成绩(100%)	

## 实验报告

### 一、实验目的

1. 掌握数据库管理系统的存储管理器的工作原理。
2. 掌握数据库管理系统的缓冲区管理器的工作原理。
3. 使用 C++ 面向对象程序设计方法实现缓冲区管理器。

### 二、实验环境

软件系统：Windows 11 22H2、Ubuntu 20.04.4 LTS

开发工具：Visual Studio Code 1.65.2

### 三、实验过程（介绍实验过程、设计方案、实现方法、实验结果等）

实验过程及实现方法：

buffer.cpp 文件中的方法介绍。

1. BufMgr(const int bufs)：BufMgr 类的构造函数。为缓冲池分配一个包含 bufs 个页面的数组，并为缓冲池的 BufDesc 表分配内存。当缓冲池的内存被分配后，缓冲池中所有页

实验题目	缓存区管理器实现			实验日期	2022.04.03
班级	1903102	学号	L190201901	姓名	王博

框的状态被置为初始状态。接下来，将记录缓冲池中当前存储的页面的哈希表被初始化为空。本实验已经实现了该构造函数。

2. `~BufMgr()`：BufMgr 类的析构函数。将缓冲池中所有脏页写回磁盘，然后释放缓冲池、BufDesc 表和哈希表占用的内存。
3. `void advanceClock()`：顺时针旋转时钟算法中的表针，将其指向缓冲池中下一个页框。
4. `void allocBuf(Frameld& frame)`：使用时钟算法分配一个空闲页框。如果页框中的页面是脏的，则需要将脏页先写回磁盘。如果缓冲池中所有页框都被固定了(pinned)，则抛出 BufferExceededException 异常。allocBuf() 是一个私有方法，它会被下面介绍的 readPage()和 allocPage()方法调用。请注意，如果被分配的页框中包含一个有效页面，则必须将该页面从哈希表中删除。最后，分配的页框的编号通过参数 frame 返回。
5. `void readPage(File* file, const PageId PageNo, Page*& page)`：首先调用哈希表的 lookup()方法检查待读取的页面(file, PageNo)是否已经在缓冲池中。如果该页面已经在缓冲池中，则通过参数 page 返回指向该页面所在的页框的指针；如果该页面不在缓冲池中，则哈希表的 lookup()方法会抛出 HashNotFoundException 异常。根据 lookup()的返回结果，我们处理以下两种情况。
  - 情况 1: 页面不在缓冲池中。在这种情况下，调用 allocBuf()方法分配一个空闲的页框。然后，调用 file->readPage()方法将页面从磁盘读入刚刚分配的空闲页框。接下来，将该页面插入到哈希表中，并调用 Set()方法正确设置页框的状态，Set()会将页面的 pinCnt 置为 1。最后，通过参数 page 返回指向该页框的指针。
  - 情况 2: 页面在缓冲池中。在这种情况下，将页框的 refbit 置为 true，并将 pinCnt 加 1。最后，通过参数 page 返回指向该页框的指针。

实验题目	缓存区管理器实现			实验日期	2022.04.03
班级	1903102	学号	L190201901	姓名	王博

6. `void unPinPage(File* file, const PageId pageNo, const bool dirty)` : 将缓冲区中包含(file, pageNo)表示的页面所在的页框的 pinCnt 值减 1。如果参数 dirty 等于 true , 则将页框的 dirty 位置为 true。如果 pinCnt 值已经是 0 , 则抛出 PAGENOTPINNED 异常。如果该页面不在哈希表中 , 则什么都不用做。
7. `void allocPage(File* file, PageId& pageNo, Page*& page)` : 首先调用 file->allocatePage() 方法在 file 文件中分配一个空闲页面 , file->allocatePage()返回这个新分配的页面。然后 , 调用 allocBuf()方法在缓冲区中分配一个空闲的页框。接下来 , 在哈希表中插入一条项目 , 并调用 Set()方法正确设置页框的状态。该方法既通过 pageNo 参数返回新分配的页面的页号 , 还通过 page 参数返回指向缓冲池中包含该页面的页框的指针。
8. `void disposePage(File* file, const PageId pageNo)` : 该方法从文件 file 中删除页号为 pageNo 的页面。在删除之前 , 如果该页面在缓冲池中 , 需要将该页面所在的页框清空并从哈希表中删除该页面。
9. `void flushFile(File* file)` : 扫描 bufTable , 检索缓冲区中所有属于文件 file 的页面。对每个检索到的页面 , 进行如下操作 : (a)如果页面是脏的 , 则调用 file->writePage()将页面写回磁盘 , 并将 dirty 位置为 false ; (b) 将页面从哈希表中删除 ; (c) 调用 BufDesc 类的 Clear()方法将页框的状态进行重置。如果文件 file 的某些页面被固定住(pinned) , 则抛出 PagePinnedException 异常。如果检索到文件 file 的某个无效页 , 则抛出 BadBufferException 异常。

下面我们将实现 buffer.cpp 文件中的方法。

1. `~BufMgr()` : 此函数为析构函数。扫描每个页框 , 若当前页框的 valid 值为 true , 则调用

实验题目	缓存区管理器实现			实验日期	2022.04.03
班级	1903102	学号	L190201901	姓名	王博

flushFile 函数将可能的脏页写回磁盘。再将 bufDescTable、bufPool 和 hashTable 删除。

2. void advanceClock() : clockHand 加 1 并将结果对 numBufs 取模即可。
3. void allocBuf(Frameld& frame) : 定义 bufPinned 表示固定的页框的数量，在每次遇到固定的页框时更新。首先调用 advanceClock 函数更新时钟指针，若当前指向的页框的 valid 为 false 则直接返回当前的 clockHand。否则，查看 refbit，若为 true，则将其置为 false，进入下一轮循环。否则，若当前页框 pinCnt 为 0，则先将脏页写回，再返回当前 clockHand。否则，更新 bufPinned，当 bufPinned 达到 numBufs 时，抛出 BufferExceededException 异常。返回前，若 clockHand 所指的页框的 valid 为 true，则调用 hashTable->remove 将该页面从哈希表中删除。
4. void readPage(File\* file, const Pageld PageNo, Page\*& page) : 首先调用 hashTable->lookup 查找缓存中是否存在该页面。若有，则将 refbit 置为 true，pinCnt 加 1。否则，调用 allocBuf 分配一个页框 frame，调用 file->readPage 获取该页面并存到 bufPool[frame] 中，调用 hashTable->insert 插入哈希表中，调用 bufDescTable[frame].Set 设置页框状态。最后，将 bufPool[frame]的地址给 page。
5. void unPinPage(File\* file, const Pageld PageNo, const bool dirty) : 调用 hashTable->lookup 尝试再哈希表中查找该页面，若没有该页面，则直接返回。否则，若当前页面的 pinCnt 为 0，则抛出 PageNotPinnedException。否则，将 pinCnt 减 1，更新 dirty 状态。
6. void allocPage(File\* file, Pageld& PageNo, Page\*& page) : 调用 file->allocatePage 分配一个空闲页面，并调用 page\_number 获取该页面的编号。调用 allocBuf 分配页框 frame，并调用 hashTable->insert 将新页面插入哈希表中。调用


实验题目	缓存区管理器实现			实验日期	2022.04.03
班级	1903102	学号	L190201901	姓名	王博

bufDescTable[frame].Set 设置该页框状态，并将页面存在 bufPool[frame]中。最后将 bufPool[frame]的地址给 page。

7. void disposePage(File\* file, const PageId pageNo) : 调用 hashTable->lookup 查找该页面是否在缓存区中，若在则调用 bufDescTable[frame].Clear 清除所在页框，并调用 hashTable->remove 从哈希表中删除该页面。最后调用 file->deletePage 将该页面删除。
8. void flushFile(File\* file) : 扫描每个页框，若当前页框的页面属于 file，则依次进行如下判断。若页框状态 valid 为 false，则抛出 BadBufferException。若 pinCnt 不为 0，则抛出 PagePinnedException。若 dirty 为 true，则调用 file->writePage 将 bufPool 中的数据写回磁盘，并将 dirty 置为 false。再调用 hashTable->remove 从哈希表中删除该页面，最后调用 bufDescTable[frame].Clear 将该页框状态删除。

实验结果：

实验题目	缓存区管理器实现			实验日期	2022.04.03
班级	1903102	学号	L190201901	姓名	王博

  
types.h

```

bendtion@ubuntu: ~/Desktop/BufMgr/src
bendtion@ubuntu:~/Desktop/BufMgr/src$ ./badgerdb_main
Found record: hello! on page 1
Found record: hello! on page 2
Found record: hello! on page 3
Found record: hello! on page 4
Found record: hello! on page 5
Third page has a new record: world!

Test 1 passed
Test 2 passed
Test 3 passed
Test 4 passed
Test 5 passed
Test 6 passed

Passed all tests.
bendtion@ubuntu:~/Desktop/BufMgr/src$ S

```

#### 四、实验结论 ( 总结实验发现及结论 )