2021 春《数据库系统》实验报告

实验 2：缓冲区管理器

姓名： 卢兑玑　　学号： L170300901　　班级：1803501

1. 实验目的

在这个项目中，需要在提供的存储管理器之上实现一个缓冲区管理器。需要实现的内容如下：

数据库缓冲池是一组固定大小的内存缓冲区，称为帧，用于保存已从磁盘读取到内存中的数据库页面（也称为磁盘块）。 页是磁盘和驻留在主内存中的缓冲池之间的传输单位。 大多数现代 DBMS 使用至少 8,192 字节的页面大小。 内存中的数据库页面是第一次读入时磁盘上相应页面的精确副本。一旦页面从磁盘读取到缓冲池，DBMS 软件就可以更新存储在磁盘上的信息。 页会导致缓冲池中的副本与磁盘上的副本不同。 这样的页面被称为“脏页”。

2. 实验准备

见实验指导书。根据实验指导书来完成。

3. 实验内容

在本次实验中，需要实现 buffer.cpp 文件中的若干函数来实现时钟置换算法，完成缓冲区的基本功能。具体需要实现如下：

```
BufMgr::BufMgr(std::uint32_t bufs)
    : numBufs(bufs) {
    bufDescTable = new BufDesc[bufs];
```

为具有 bufs 页框和相应 BufDesc 表的缓冲池分配一个数组。

在分配缓冲池时，所有帧的设置方式都将处于清除状态。

哈希表也将以空状态开始。 我们已经提供了构造函数。

```
BufMgr::~BufMgr() {
    //Clean dirty pages
    for(FrameId i = 0; i < numBufs; i++){
```

清除所有脏页并释放缓冲池和 BufDesc 表。

```
void BufMgr::advanceClock()
{
    clockHand += 1;
    if(clockHand > numBufs - 1){
```

将时钟提前到缓冲池中的下一帧。

```
void BufMgr::allocBuf(FrameId & frame)
{
    unsigned pinnedNum = 0;//number of pinned frame
    while(1){
        advanceClock();
```

使用时钟算法分配一个空闲帧；

如有必要，将脏页写回磁盘。

如果所有缓冲区帧都被固定，则抛出 BufferExceededException。

这个私有方法将由下面描述的 readPage() 和 allocPage() 方法调用。

确保如果分配的缓冲区帧中有一个有效的页面，

从哈希表中删除适当的条目。

```
void BufMgr::readPage(File* file, const PageId pageNo, Page*& page)
{
    FrameId FI;
    //Case 2
    try{
        hashTable->lookup(file, pageNo, FI);//Find the frame of page
        bufDescTable[FI].refbit = true;
        bufDescTable[FI].pinCnt += 1;
        page = bufPool + FI;//Return pointer
```

读取页的时候。首先通过调用 lookup()方法检查页面是否已经在缓冲池中，

当页面不在缓冲池中时，它可能会抛出 HashNotFoundException，

在哈希表上获取帧号。

根据 lookup() 调用的结果，有两种情况需要处理：

情况 1：页面不在缓冲池中。

情况 2：页面在缓冲池中。

```
void BufMgr::flushFile(const File* file)
{
    for (FrameId i = 0; i < numBufs; i++){
        if(bufDescTable[i].file == file){//Find the frame that belongs to the file
            if(!bufDescTable[i].valid){// Invalid page belonging to the file
```

扫描 bufTable 以查找属于该文件的页面。

对于遇到的每个页面，它应该：（a）如果页面脏了，

调用 file->writePage() 将页面刷新到磁盘，然后将页面的脏位设置为 false，

(b) 从哈希表中删除页面（页面是干净的还是脏的）并且 (c) 为页面框架调用 BufDesc 的 Clear() 方法。

如果文件的某些页面被固定，则抛出 PagePinnedException。

如果遇到属于文件的无效页面，则抛出 BadBufferException。

```
void BufMgr::allocPage(File* file, PageId &pageNo, Page*& page)
{
    Page p = file->allocatePage();//Assign new page
    FrameId FI;
    allocBuf(FI);//Assign frame
```

该方法的第一步是通过调用 file->allocatePage()方法在指定文件中分配一个空页。

此方法将返回新分配的页面。 然后调用 allocBuf() 获取缓冲池帧。

接下来，将一个条目插入到哈希表中，并在框架上调用 Set() 以正确设置它。

该方法通过 pageNo 参数将新分配的页的页码返回给调用者，并通过 page 参数返回指向为该页分配的缓冲区帧的指针。

4. 实验结果

Make 成功之后的结果如下：

```
[root@localhost L170300901 -卢兑玑-database_lab1]# make
cd src; ₩
g++ -std=c++0x *.cpp exceptions/*.cpp -I. -Wall -o badgerdb_main
[root@localhost L170300901 -卢兑玑-database_lab1]#
```

5. 总结与体会

通过这次实验，我对于时钟置换算法有了更加清晰的认识，有助于我更深入地了解具体步骤。同时我还对上课内容的理解有了更深的认识。这次实验设计的非常好。如果能再对所有需要用到的已实现方法进行介绍就更好了。