实验题目	缓冲区管理器实现			实验日期	2022/4/3
班级	1903102	学号	1190200122	姓名	袁野

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（介绍实验目的）

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（介绍实验使用的硬件设备、软件系统、开发工具等）

1、硬件设备：
i7-9750H CPU@2.60GHz 2.59GHz；1.8GHz；16G RAM;
2、软件系统
Ubuntu 20.04.4 LTS (GNU/Linux 5.10.102.1-microsoft-standard-WSL2 x86_64)
3、开发工具
MySQL

三、实验过程（介绍实验过程、设计方案、实现方法、实验结果等）

3.1 缓冲池替换策略

在缓冲池交换的部分，我们使用时钟算法作为缓冲池替换策略管理缓冲页。我们将所有的页看作一个环，然后维护一个变化的时钟指向当前的页，每次申请新页时，我们首先移动时钟指针，之后查看当前页是否空闲：如果为空，就直接返回当前空页；如果不为空，就查看是否空闲。如果空闲就写回缓冲（如果必要）再返回。循环一圈之后如果发现所有页都被占用着，就抛出一个异常。

3.2 析构函数

在析构函数中，我们需要写回脏页以及释放内存。程序会将所有合法的有效脏页写回到磁盘，之后将占用的所有堆内存释放（如 hashTable，bufPool 等）到堆上。

3.3 advanceClock

在 advanceClock 方法我们需要让时钟指针前进一步就可以了。由于缓冲页逻辑上是环形的，所以如果指针到达终点后面（clockHand == numBufs 之后），我们需要将指针移回至起点，相当于走了一个循环。

3.4 allocBuf

allocBuf 函数的作用是返回一个空闲的页来供给进程使用，这里我们用到了时钟算法来实现缓冲页的选取策略，同时对哈希表进行更新，如果找到了一个可用的页，那么我们将参数中的 frame 进行赋值，否则抛出 BufferExceededException 异常。

3.5 readPage

readPage 是用于读取特定页。其参数为 File* file, const PageId PageNo, Page*& page，其中 file，PageNo 用于定义查询对应的页。page 则用于返回对应页指针。程序会首先在哈希表中查询对应的页。如果找到，则表明该页已经在缓冲池中，那么只需要更新描述表中 refbit, pinCnt 两个变量即可，之后返回该页；如果没有找到，那么表明我们需要使用 allocBuf 方法来为该页申请一个新的缓冲池，之后更新描述表，在哈希表中插入对应项。

3.6 unPinPage

unPinPage 用于释放对于一个页的占用。首先在哈希表中查找对应的缓冲，将其对应描述表的 pinCnt 减一。如果在进行此操作之前 pinCnt 为零，则抛出 PageNotPinnedException 异常。如果该页不在哈希表中，则表明该页在缓冲池中没有对应的缓冲块，那么就不需要进行任何操作。

3.7 allocPage

allocPage 用于将给定文件的页读入到缓冲池中。我们可以使用 file 的 allocatePage 方法来在对应的文件中申请一个新的空页，之后利用 allocBuf 方法来申请新的缓冲块，将申请的新页与新的缓冲块建立其联系，并在哈希表中插入对应的项，同时更新描述表。在完成这一切之后，更新 PageNo, page 两个变量来传出新页和缓冲块的数据。

3.8 disposePage

disposePage 用于将指定页从文件中删除。如果该页在缓冲池中有对应的块，那么同时也需要同步清空对应的描述表，删除哈希表中对应项，同时释放该块。

实验题目	缓冲区管理器实现			实验日期	2022/4/3
班级	1903102	学号	1190200122	姓名	袁野

3.9 flushFile

flushFile 方法将给定文件的全部缓冲页全部写回，并在哈希表中删除对应的项，清空描述表中对应信息。若删除的块无效，则需要抛出 BadBufferException 异常；若该块正在被其他程序占用，则需要抛出 PagePinnedException 异常。

四、实验结论（总结实验发现及结论）

测试全部通过，如下图。

```
[Test 1]
Testing allocatePage() and testing readPage() when pages are in buffer.
Passed
[Test 2]
(1) Testing allocatePage() when the buffer pool is full.
(2) Testing readPage() when pages are not in buffer.
Passed
[Test 3]
Testing readPage() when reading a invalid page.
Passed
[Test 4]
Testing unPinPage() when unpining an unpinned page.
Passed
[Test 5]
Testing allocatePage() when the buffer pool is full.
Passed
[Test 6]
Testing flushFile() when pages are pinned.
Passed
[Test 7]
Testing disposePage().
Passed
[Test 8]
Testing if contents match after calling flushFile().
Passed
[Test 9]
Testing disposePage() and testing readPage() when reading a deleted page.
Passed
[Test 10]
Testing the clock algorithm and buffer space management
Passed
[Test 11]
Testing readPage() when pages in buffer pool has been updated.
Passed
[Test 12]
Testing the readingPage() when reading a page whose pid is the same with the pid of a deleted page.
Passed

Passed all tests.
```

本次实验之后我对于缓冲池管理机制有了更加深刻的理解，对数据库缓冲池的细节有了更新的认识，同时对自己的 C++代码能力有了一定的提升；数据库系统是一个层层相扣关系密切的系统，我们在构建时一定要做好清晰、简洁的程序设计；而对于数据库系统来说，一个好的缓冲池管理策略也十分重要，可以大大影响数据库系统的运行效率。