哈尔滨工业大学

<<数据库系统>>

实验报告三

(2023 年度春季学期)

姓名:	樊启元
学号:	7203610712
学院:	未来技术学院
教师:	李东博

实验三

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

Windows XP 操作系统、MySQL 关系数据库管理系统、MinGW 编译器或 Microsoft Visual C++编译器。(可以用其他操作系统和编程环境，数据库尽量选择 MySQL)

三、实验过程及结果

在整个实验所给的源代码中，我们只需要完成其中的 buffer.cpp，实现缓冲区管理器类就可以了。buffer.cpp 的具体实现过程如下：

其中的 BufMgr(const int bufs)和~BufMgr()已经给出，他们的作用是：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。以及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。

The BadgerDB Buffer Manager

数据库缓冲池是一组固定大小的内存缓冲区，称为帧，用于保存从磁盘读入内存的数据库页（也称为磁盘块）。页是驻留在主存中的磁盘和缓冲池之间的传输单位。大多数现代 dbms 使用至少 8,192 字节的页大小。另一个需要注意的重要事项是，内存中的数据库页是第一次读入时磁盘上相应页的精确副本。一旦页面从磁盘读取到缓冲池，DBMS 软件就可以更新存储在页面上的信息，从而使缓冲池中的副本与磁盘上的副本不同。这样的页面被称为“dirty”。

由于磁盘上的数据库本身通常大于缓冲池可用的主内存，因此在任何给定时间，内存中只能容纳数据库页面的一个子集。缓冲区管理器用于控制哪些页面驻留在内存中。每当缓冲区管理器接收到对数据页的请求时，缓冲区管理器都会检查所请求的页是否已经在构成缓冲池的某个帧中。如果是，缓冲区管理器只返回一个指向该页的指针。如果没有，缓冲区管理器将释放一个帧(如果该页是脏的，则可能通过将该页包含的页写入磁盘)，然后将请求的页从磁盘读入已释放的帧。

简而言之，就是 I/O 层为 Unix 文件提供了一个面向对象的接口，其中包含打开/关闭文件和读取/写入文件页面的方法。现在，您需要知道的关键事情是打

开一个文件(通过传入一个字符串名称)返回一个 file 类型的对象。该类具有读取和写入文件页面的方法。

本次实验将使用这些方法在磁盘和缓冲池之间移动页面。

Buffer Replacement Policies and the Clock Algorithm（缓冲区替换策略和时钟算法）

当需要一个空闲框架时，有很多方法可以决定替换哪个页面。操作系统中常用的策略有 FIFO、MRU 和 LRU。尽管 LRU 是最常用的策略之一，但它有很高的开销，并且在数据库系统中出现的许多常见情况下并不是最好的策略。相反，许多系统使用近似 LRU 行为的时钟算法，并且速度快得多。

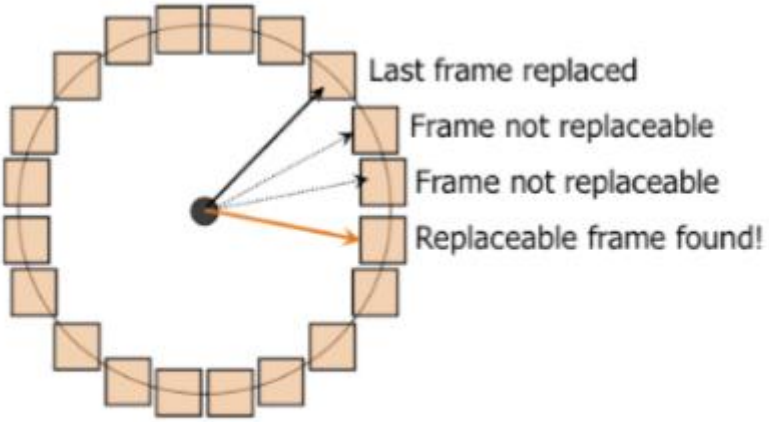

Figure 1: Structure of the Buffer Manager

图 1 显示了缓冲池的概念布局。在图 1 中，每个方框对应缓冲池中的一个帧。假设缓冲池中包含 numBufs 帧，编号为 0 ~ numBufs−1。

从概念上讲，缓冲池中的所有帧都排列在一个循环列表中。与每一帧相关联的位称为 refbit。每次访问缓冲池中的页面时（通过对缓冲管理器的 readPage() 调用），相应帧的 refbit 被设置为 true。在任何时间点，时钟指针（值在 0 和 numBufs−1 之间的整数）都以顺时针方式前进（使用模块化算法，因此它不会超过 numBufs−1）。对于时针经过的每一帧，refbit 都会被检查并清除。如果设置了该位，则对应的帧已被“最近”引用，而不会被替换。另一方面，如果 refbit 为假，则选择该页面进行替换(假设它没有固定-固定页面将在下面讨论）。如果选定的缓冲帧是脏的（即，它已被修改），则当前占用该帧的页面将被写回磁盘。否则，该帧将被清除，并从磁盘读入一个新的页面到该位置。图 2 演示了时钟算法的执行。时钟算法的细节如下所示。

Figure 2: The Clock Replacement Algorithm

The Structure of the Buffer Manager（缓冲区管理器的结构）

BadgerDB 缓冲区管理器使用了三个 c++类:BufMgr、BufDesc 和 BufHashTbl。BufMgr 类只有一个实例。该类的一个关键组件是实际的缓冲池，它由 numBufs 帧数组组成，每个帧的大小与数据库页面的大小相同。除了这个数组之外，BufMgr 实例还包含一个 BufDesc 类的 numBufs 实例数组，用于描述缓冲池中每个帧的状态。散列表用于跟踪当前驻留在缓冲池中的页面。这个散列表是由 BufHashTbl 类的一个实例实现的。这个实例是 BufMgr 类的私有数据成员。下面将详细描述这些类。

`BufMgr(const int bufs)`

这是类的构造函数。为缓冲池分配一个数组，其中包含 bufs 页帧和相应的 BufDesc 表。当分配缓冲池时，所有帧都将处于清除状态。哈希表也将以空状态开始。我们已经提供了构造函数。

`~BufMgr()`

清除所有脏页并重新分配缓冲池和 buffdesc 表。

`void advanceClock()`

将时钟提前到缓冲池中的下一帧。

void allocBuf(FrameId& frame)

使用时钟算法分配空闲帧;如有必要，将脏页写回磁盘。如果所有缓冲帧都被固定，抛出 BufferExceededException。这个私有方法将被下面描述的 readPage()和 allocPage()方法调用。确保如果分配的缓冲帧中有一个有效的页面，则从哈希表中删除相应的条目。

按照上面图二的流程图实现即可。

`void readPage(File* file, const PageId PageNo, Page*& page)`

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，该方法可能会在哈希表上抛出 HashNotFoundException，以获取帧号。根据 lookup()调用的结果，有两种情况需要处理:

情况 1：Page 不在缓冲池中。调用 allocBuf()来分配一个缓冲帧，然后调用 file->readPage()方法将页面从磁盘读入缓冲池帧。接下来，将页面插入哈希表。最后，在框架上调用 Set()以正确设置它。Set()将为设置为 1 的页面留下 pinCnt。通过 page 参数返回一个指向包含该页的帧的指针。

情况 2：页面在缓冲池中。在这种情况下，设置适当的 refbit，增加页面的 pinCnt，然后通过 page 参数返回一个指向包含页面的帧的指针。

`void unPinPage(File* file, const PageId PageNo, const bool dirty)`

减少包含(file, PageNo)的帧的 pinCnt 值，如果 dirty == true，则设置 dirty 位。如果引脚计数已经为 0，则抛出 pagenotpin。如果在哈希表查找中没有找到页，则不执行任何操作。

`void allocPage(File* file, PageId& PageNo, Page*& page)`

该方法的第一步是通过调用 file->allocatePage()方法在指定的文件中分配一个空页。这个方法将返回一个新分配的页面。然后调用 allocBuf()来获取缓冲池帧。接下来，将一个条目插入哈希表，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 参数将指向为该页分配的缓冲帧的指针返回给调用者。

`void disposePage(File* file, const PageId pageNo)`

此方法从文件中删除特定的页面。在从文件中删除该页之前，它确保如果要删除的页在缓冲池中被分配了一个帧，那么该帧将被释放，并相应地从哈希表中删除条目。

`void flushFile(File* file)`

应该扫描 bufTable 中属于该文件的页面。对于遇到的每个页面，它应该:(a)如果页面是脏的，调用 file->writePage()将该页刷新到磁盘，然后将该页的脏位设置为 false， (b)从哈希表中删除该页(无论该页是干净的还是脏的)，以及(c)调用 BufDesc 的 Clear()方法来处理该页帧。

如果文件的某些页面被固定，抛出 PagePinnedException。如果遇到属于文件的无效页面，则抛出 BadBufferException。

pinPage 是为了在读取等从 disk 到 buffer 的操作时，Page 被锁定，不能做其他操作，来保证数据操作的安全。

```
fanqiyuan@fanqiyuan:~/DatabaseLab/lab3/BufMgr$ make
cd src;\
g++ -std=c++0x *.cpp exceptions/*.cpp -I. -Wall -o badgerdb_main
fanqiyuan@fanqiyuan:~/DatabaseLab/lab3/BufMgr$ ./src/badgerdb_main
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.
```

四、实验心得

1. 锻炼了英文文献阅读能力。
2. 掌握数据库管理系统的存储管理器的工作原理、掌握数据库管理系统的缓冲区管理器的工作原理、使用 C++面向对象程序设计方法实现缓冲区管理器。
3. 熟悉如何调用接口，添加功能函数 API 的操作，提前熟悉了拧螺丝的程序员生活。