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）

|  |
| --- |
| 系统：Ubuntu 18.04  调试工具：gdb，valgrind等 |

三、实验过程（介绍实验过程、设计方案、实现方法、实验结果等）

|  |
| --- |
| 1. buffer.cpp中的主要变量   因为本次实验中主要的修改集中于buffer.cpp中，所以以下主要围绕着buffer.cpp中涉及的变量来讲解缓冲器的整体结构。  根据BufMgr类的构造函数，可以得到缓冲器的三个主要组成部分：   * bufDescTable：包含numBufs个页框（BufDesc）的数组，可以通过每一个BufDesc页框的属性来描述页框的实际状态。 * bufPool：包含numBufs个page的数组。代表实际的缓冲池。 * hashTable：将(File，Page)映射到页框中的哈希表。哈希函数如下：     缓冲区管理器的原理如下：  bufPool是由一组固定大小的内存缓冲区(buffer)构成的数组，用于存放从磁盘读入内存的page。缓冲池中每个固定大小的内存缓冲区称作页框。当磁盘上的页面被首次读入缓冲池时，缓冲池中的页面和磁盘上对应页面一样。一旦DBMS修改了缓冲池中该页面的内容，则缓冲池中的页面与它在磁盘上对应的页面就不再相同了。我们将缓冲池中被修改过的页面称为“脏”页面（对应dirty位为true）。缓冲区管理器(BufMgr)用于控制哪些页面驻留在缓冲池中。每当缓冲区管理器收到了一个页面访问请求，它会首先检查被请求的页面是否已经存在于缓冲池的某个页框中。如果存在，则返回指向该页框的指针；如果不存在，则缓冲区管理器会释放一个页框(如果页框中的页面是脏的，则需要将该页面先写回磁盘)，并将被请求的页面从磁盘读入刚刚释放的页框。  本实验中，缓冲区页面的替换策略选用时钟算法。   1. 对buffer.cpp的修改   下面介绍buffer.cpp中自行编写的具体函数。   1. 析构函数~BufMgr()：   实现先将脏页写回磁盘，然后清除占用内存的功能。注意在删除时为避免空指针出现，最好按指针的指向顺序删除。     1. void advanceClock()   通过参数clockHand在0 ~ numBufs-1之间循环自增，实现时钟算法中的表针转动的功能。每调用一次advanceClock()，clockHand在0 ~ numBufs-1之间自增1。   1. void allocBuf(FrameId& frame)   使用时钟算法分配一个空闲页框。函数逻辑如实验指导书所示。按照实验指导书设计函数即可。由于代码过长，此处不作展示，仅说明一些要点：   * 页框中包含有效页面的删除操作：      * 该函数不需要返回值，因为frame参数以指针形式传入，在函数内对其进行的修改可以作用于全局。  1. void readPage(File\* file, const PageId PageNo, Page\*& page)   分页面在缓冲池中和页面不在缓冲池中两种情况。用try catch语句实现其功能。    在try语句中，若lookup函数捕捉到HashNotFoundException，则转入catch语句块中的异常处理程序，实现了两种情况的readPage功能。   1. void unPinPage(File\* file, const PageId PageNo, const bool dirty)   实现将缓冲区中包含(file, PageNo)表示的页面所在的页框的pinCnt值减1的功能。仍然通过lookup函数实现查找特定页面的功能。该函数设计并无难点，按照实验指导书的程序逻辑设计即可。   1. void allocPage(File\* file, PageId& PageNo, Page\*& page)   按照实验指导书编写程序即可。如图所示，展示每一步实现的功能。     1. void disposePage(File\* file, const PageId pageNo)   通过deletePage从file中删除该页面。在此之前，需要根据页面是否在缓冲池中进行不同的操作。具体实现通过try catch语句完成。     1. void flushFile(File\* file)   通过for循环实现对缓冲区页框的遍历。通过bufDescTable[i].file == file条件判断当前页框的页是否属于某个文件。另外需要注意，对脏页面/无效页/页面被固定三种特殊情况的处理需要在调用remove和clear函数对哈希表和页框进行重置与删除操作前完成。  具体代码如下：     1. 新增测试用例   在此次实验给定的6个测试用例以外，我新增了1组测试用例test7，用于测试pageNo过大，造成InvalidPageException的情况。具体代码如下：     1. 编译与结果   若要通过编译，首先需要在main.cpp中将该部分代码注释掉，否则会报错（目前原因未知）：    然后，在BufMgr文件夹下打开终端，输入make执行Makefile，再执行src文件夹下的badgerdb\_main文件查看测试结果。对于总共7个测试用例，运行结果如下：    为方便起见，我在BufMgr文件夹下新建了run.sh脚本，执行该脚本即可完成make与执行badgerdb\_main的功能。 |

四、实验结论（总结实验发现及结论）

|  |
| --- |
| 通过此次实验，熟悉了数据库管理系统的存储管理器和缓冲区管理器的工作原理，了解了缓冲池的具体实现方法，同时，使用C++编程的能力得到了不小的提高。 |