CSCI 3357: Database System Implementation

Homework Assignment 3

Due Thursday, September 21

The SimpleDB buffer manager is grossly inefficient in two ways:

- When looking for a buffer to replace, it uses the first unpinned buffer it finds, instead of doing something intelligent like LRU.
- When checking to see if a block is already in a buffer, it does a sequential scan of the buffers, instead of keeping a data structure (such as a map) to more quickly locate the buffer.

I would like you to fix these problems by modifying the class `BufferMgr`. Please use the following strategy:

a) Keep a list of the unpinned buffers, called `unpinned`. When a replacement buffer needs to be chosen, remove the buffer at the head of the list and use it. When a buffer's pin count becomes 0, add it to the end of the list. This implements LRU replacement.

b) Keep a map, called `inMemory`, whose entries describe the blocks that are currently stored in buffers. The key of each map entry is a `BlockId`, and its value is the `Buffer` object holding that block. The map is initially empty. When you need to know if a block is currently in a buffer, you simply look it up in the map. If you want to read a block into a chosen buffer, you add an entry to the map. If the chosen buffer already contains a block, then you must also remove the entry for that block. Recall that each `Buffer` object has a method `block()`, which tells you the block (if any) that is in that buffer.

c) Get rid of the `bufferpool` array. You no longer need it. You also won't need the `available` variable, because you can determined the number of available buffers by looking at the unpinned list.

I have written a test program called `HW3Test` that I will use to grade your code. You should download it to help with debugging. This program pins and unpins buffers, occasionally calling the buffer manager's `printStatus` method to display its status. The status consists of the id, block, and pinned status of each buffer in the `inMemory` map, plus the ids of each buffer in `unpinned` list. For example, here is what the output of the method should look like for a database having 4 buffers, in which blocks 0 to 3 of file "test" were pinned, and then blocks 2 and 0 were unpinned.

```
Buffers and their Contents:
   Buffer 1: [file test, block 1] pinned
   Buffer 0: [file test, block 0] unpinned
   Buffer 3: [file test, block 3] pinned
   Buffer 2: [file test, block 2] unpinned
Unpinned Buffers in LRU order: 2 0
```

Currently, the buffer manager doesn't have a `printStatus` method. You will need to write it. As shown above, each buffer has an id number to identify it. Your code for this method should ask each buffer for its id number. I modified the class `Buffer` to help you out. In particular, its constructor now has a third argument denoting the buffer's id, and there is a method `getId()` that returns its id. You should download and use this revised `Buffer` class. Of course, you will also have to modify the `BufferMgr` constructor so that its call to the `Buffer` constructor has three arguments: the file manager, the log manager, and the buffer's id number.

The allocated buffers are printed in seemingly random order because they were retrieved from a hash map; that's ok. The block information within brackets comes from calling the `toString` method of `BlockId`.

The make sure that your code works, you should compare its output on `HW3Test` with the output you expect to get. Make sure that you are clear about what the expected output should be! The sequence of operations in this test program contains a tricky special case that you may not have handled.

Submit to Canvas your revised code for *BufferMgr.java*. As before, do not change its package. Also, please do not change any other classes in SimpleDB. I need to be able to grade your file by simply adding it to my SimpleDB environment.

Although you won't need to write much code, I strongly suggest that you start early. This assignment requires a thorough understanding of LRU buffer management and the SimpleDB buffer management classes. You may find yourself getting very lost. If so, please see me early or ask questions in class.