CS539 Assignment 1

SimpleDB Heap Pages and Buffer Replacement Policies

Ashish Jindal (AJ523)

1 Design Decisions

1.1 Abstraction of page header implementation

Currently the code for page header's implementation (Store 32 bits in one integer) and related arithmetic for calculating storage is scattered all over the file. So I decided to move this into an inner class "Header". The class abstracts the implementation of header and users of its public api only need to deal with byte[ ] and slot data. This way all the bit arithmetic remains encapsulated and also leaves a scope for easy modification of header's implementation in future.

1.2 Encapsulating buffer replacement policy implementation

As there are multiple policies that can be applied to a buffer pool, each requiring different bookkeeping; so I decided to encapsulate this in class called "PolicyData", which maintains this bookkeeping information for various policies. This gives a benefit that instead of using if-else or switch-case constructs in all dependent functions of class BufferPool, it will now have to deal with only one class object which will handle the various cases inside itself. A better way of doing this can be to separate the Policy related implementation and bookkeeping in separate classes and then use decorator design pattern to use it over our buffer pool. This way would be more suitable when we have a lot more policies with complex and disjoint implementation schemes.

1.3 HashMap for BufferPool

HashMap provide O(1) access for all the elements in the data-structure so I decided to use it for storing our buffered pages. We can check for existence of a page in buffer pool in O(1), fetch the same in O(1) and also insert or remove an element from it in constant time.

1.4 Implementation of LRU and MRU using global counter scheme

Class BufferPool needs to work on both LRU and MRU buffer replacement schemes. So I decided to go for an implementation which can work in both cases. We keep track of usage of a page using a global counter. Every time a page is accessed, the global counter's value is stored in a hashmap corresponding to that page (Using pageId as key), after which counter's value is incremented for any further accesses. This way all the new buffered pages keep getting a value higher than any previous value. When a page needs to be evicted, we check if the policy for eviction is LRU, then find a page with the least count else if the policy for eviction is MRU then find a page with the highest count value and then return the page found for eviction.

2 Feedback on assignment

Assignment gave a great insight into the practical implementation of DB Catalogue, heap file, buffer pool and buffer replacement policies. Approximately it took me around 7hrs to finish the assignment; which includes time spent on code analysis, debugging and implementing the desired code features. One thing i struggled with was understanding the memory allocation for header in heap file. I think the implementation of header (Store bits in integers) and related arithmetic on the bits in the header, was scattered all over the file, which made it a bit confusing to analyse.