Database System Implementation Assignment 1

Files Modified

1. BufferPool.java
2. HeapPage.java

Number of Hours Spent

6 Hours.

Part A

BufferPool.getPage() has been implemented. If the page exists in the pool then the given page is returned and _numhits is incremented. Else the page is accessed from the catalog and loaded into the buffer pool and _nummisses is incremented.

Part B

HeapPage.getSlot(): In the implementation we get the Header Byte Position(headerByte) by dividing by 32 and the bit position(headerBit) by taking the mod of it with 32. Then get the value of bit at position headerBit of integer at headerByte.

HeapPage.getNumEmptySlots(): Here a loop in run over the slots calling getSlot() function. If the function return false, increment the count and at the end of the loop return the count

HeapPage.setSlot(): We select he Header Byte Position and Bit position in same way as get slot. Then set the value of the bit at the position headerBit of the integer at headerByte position.

Part C

If the buffer pool is full we evict the page from the buffer. Their are two buffer policies we consider to implement i.e. LRU and MRU. We use Multiple Data Structures to implement these buffer eviction policies.

- Integer (maxPages) to store the maximum size of the pool
- ConcurrentHashMap<PageId, Page> to store the map of Pages and Page Ids
- Vector List construct of Page Ids to maintain access order of the Pages in the buffer. In case of LRU, the page with Page Id at position 0 in the list will be evicted. In case of MRU the page at the end of the list with be evicted. The evict page method returns the page id of the evicted page.

Observations & Issues

Initially I was storing using a map of <pid, timestamp> to keep track of how recent access to the page was. But instead of using this I implemented a much simpler data structure vector list of page ids. The number of comparisons and the space requirement are less in comparison to my earlier implementation. Instead of comparing the timestamps, in the vector list the pages that is accessed recently is put in at the end of vector list (If the page is already in the list it is removed from that position and add it to end). Similarly, the page that is least recently used will be available at the start of the list.