CSE5331 – DBMS Models and Implementation

REPORT – PROJECT 1 (BUFFER MANAGER)

Project Team:

Mahesh Hingane – 1001017122

Harsha Kosuru – 1000689168

- OVERALL STATUS OF THE PROJECT –

The project is ***completed*** and can be executed successfully. Here is a brief description of the implementation of some major components:

1. BufMgr constructor – Variable *bufpool* is initialized as an array of *Page* objects, variable *frametab* is initialized as an array of the *FrameDesc* objects. The variable *replacer* is an instance of the *Clock* class.
2. newPage – To allocate and deallocate pages, the respective methods in *DiskManager* class are invoked as *Minibase.DiskManager.allocate_page()* and *Minibase.DiskManager.deallocate_page()*.
3. freePage – This method removes the page from buffer if present, and then deallocates it from the disk.
4. pinPage – If the page being asked for pinning is already in the buffer pool, this method increments its *pincnt*. If not, a replacement candidate is found from the current buffer pool, with which the disk copy of the given page can be exchanged. The page to be replaced is copied to disk if it is dirty.
5. unpinPage – This method decrements the *pincnt* of the page. If it becomes 0 after decrement, its state is changed to *REFERENCED*.
6. flushPage – The page is written to the disk using *Minibase.DiskManager.write_page()* method.
7. flushAllPages – All the pages in the buffer pool are written to the disk.
8. getNumBuffers – Returns the size of the buffer pool.
9. getNumUnpinned – Returns the count of unpinned pages in the buffer pool.
10. pickVictim – This method runs a do-while loop that iterates through buffer pool twice looking for an *AVAILABLE* frame. Returns the frame index if found, or returns -1.

- DIVISON OF LABOR –

Mahesh Hingane:

Time spent - ~15 hours

Responsibilities – Algorithm, Coding, Design, Debugging

Harsha Kosuru:

Time spent - ~15 hours

Responsibilities – Report, Documentation, Debugging

- LOGICAL ERRORS –
  1. Using static methods of *DiskMgr* class – To allocate and deallocate pages, we need to invoke *allocate_page()* and *deallocate_page()* methods in *DiskMgr* class. As this class is instantiated as a static object *DiskManager* in *Minibase* package, these methods can be directly used as *Minibase.DiskManager.allocate_page()* and *Minibase.DiskManager.deallocate_page()*.

  2. Iterating the buffer pool twice in the *pickVictim* method – The *pickVictim* method returns an available candidate for replacement from the buffer pool. But, when a page has *pincnt* as 0, it is marked as *REFERENCED*, not *AVAILABLE*. So, the buffer pool needs to be iterated twice, first to mark all *REFERENCED* pages as *AVAILABLE* and then to look for the first *AVAILABLE* page.

  3. Adding code to *pinPage* and *unpinPage* methods in *Clock* class – We need to change the states of the pages once they are pinned and unpinned. As the states are defined in *Clock* class, it makes more sense to add this part of code to *pinPage* and *unpinPage* methods in *Clock* class and to invoke these methods from *BufMgr* class.