#### **CSED 2014**

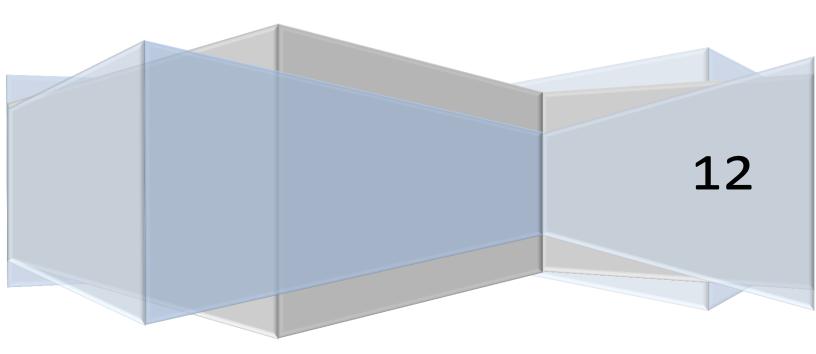
# MiniBase Buffer Manager

Ashraf Saleh Mohamed 20

Raed Ahmed Selim 35

Raymond Milad Faheem 39

Names are sorted by seat No



### **Problem Statement**

It is required to implement a simplified version of the buffer manager layer, without support for concurrency control or recovery given the code for the lower layer Disk Space Manger.

It allows a client(higher level program that calls the buffer manager) to allocate/deallocate pages on disk, bring a disk page into the buffer pool and pin it, and to unpin a page in the buffer pool.

The methods to be implemented

public BufMgr(int numbufs, String replacementArg)
public void unpinPage(PageId PageID\_in\_a\_DB, boolean dirty)throws ChainException
public void pinPage(PageId pin\_pgid, Page page, boolean empty)throws ChainException
public PageId newPage(Page firstpage, int howmany) throws IOException, ChainException
public void freePage(PageId globalPageId) throws ChainException
public void flushPage(PageId pageid) throws ChainException
public int getNumUnpinnedBuffers()
public void flushAllPages() throws ChainException

## Implementation Issues

The buffer pool is 2D of bytes rather than 1D of Pages due to limitation of Java Language.

The hash key of the hash table is Integer(id instance in the PageID object)rather than PageID because in the methods of DB it changes the PageID and we want to be affected by that.

#### Data Structure used;

**bufpool:** 2D Array of Bytes: The buffer Pool

bufDescr: 1D Aarray of BufDescr: contains the descreptor of each page in the pool.

hash: HashTable of Integer, Integer: key the integer PageID, the value the index of the

page in the pool

placementPolicy: an object that handle the replacement policy and how to insert unused frames

in it and get them back according to the policy used (MRU,LRU).

# Algorithms

```
UnpinPage(id,dirty)
       index<= hash.get(id)
       if index = null
              then HashEntyNotFoundException
       if bufDesc[index].pinCount = 0
              then PageUnpinnedException
       if dirty
              then bufDesc[index].dirtyBit=true
       pincount--
       if pinCount = 0
              add the page to the placementPolicy
pinPage(id,page,empty)
       if page in the pool
              then if pinCount = 0
                      then remove from placementPolicy
                 incrementPinCount
       else
              then
                      if usedPages<bufSize
                             then insertPos=usedPages++
                      else
                             insertPos=getFrame from placementPolicy
                              if the page was not freed
                                     then freePage
```

call DB to read the page

put page in hash

```
put pageDesc in bufDesc
```

end if

#### newPage(firstPage,howMany)

```
pid = call DB to allocate_page(fristPage,howMany)
pinPage(pid , firstPage , false)
return pid
```

#### freePage(pageId)

```
if page in the pool

if pinCount>1

then PagePinnedException

flushPage(pageID)

add page to placement policy

remove the descreptor

call DB to dellocate the page

remove from hash

end if
```

#### flushPage(pageId)

```
if page in the pool
then if dirtyBit is true
then call DB to writePage(pageID)
end if
else
then HashEntryNotFoundException
end if
```

### flushAllPages()

```
for each descreptor in bufDesc

if desc != null

then flushpage(desc.getId)

end if

end for
```

#### get Num Unpinned Buffers

return numOfCandidates in placementPolicy + bufsize – used

## Bonus

Support of Most Recently Used Replacement Policy