CSCI 5559 - Database Systems
Spring 2015

Assignment 2
Due: 04/29/2015 8:59pm

Project Goals

Minibase is a single-user database management system (without support for concurrency control and recovery) intended for educational use. In this project you will implement the buffer manager layer for *Minibase* of a typical DBMS, without support for concurrency control and recovery.

You should follow the following procedure to perform this assignment:

I. Understand the Concepts:

1) Before starting with implementation, make sure you thoroughly understand the lecture notes as well as pages 318 to 330 of the textbook where "*buffer manager"* and the "*Clock buffer replacement policy"* are explained.
2) Answer the questions posed in Part I (see below).

II. Implement the code:

1) Download and install ***Java SE Development kit*** from here.
2) Download and install the ***Eclipse IDE for Java Developers*** from here. You can use this tutorial.
3) Open Eclipse and import the provided code into ***Eclipse***. There is a tutorial here.
4) Complete the requested parts of the provided code that includes a template for buffer management layer. The provided code also includes libraries for other components of *Minibase* that interacts with the buffer manager (mainly disk space manager). You can find more details here.

Project Description

Part I: Conceptual Questions (20 points)

After reviewing the relevant materials from lecture notes and textbook (see above for details), answer the following questions, making sure you understood the concepts you need to implement before starting with coding:

a) Briefly explain what buffer manager does to process a typical read request for a page. What happens if the requested page is in the pool but not pinned? (**4 points**).

b) When does the buffer manager write a page to disk? (**3 points**).

c) What does it mean to say that a page is pinned in the buffer pool? Who is responsible for pining and unpinning pages? (**3 points**).

d) When a page in the buffer pool is modified, how does the DBMS ensures that the change is made permanent in the disc? (**3 points**).

e) What happens if a page is requested when all pages in the buffer pool are dirty? (**3 points**)

f) Briefly explain how the Clock replacement policy works? (**4 points**)

Part II: Buffer Manager Implementation (80 points)

The buffer manager provides a uniform interface for allocating and de-allocating pages on disk, bringing disk pages into the buffer pool and pinning them, and unpinning pages in the buffer pool. Your buffer manager must implement the ***Clock replacement policy*** as described on page 321 of the textbook. The main methods of ***BufMgr.java*** that you will implement are the following:

```
public BufMgr(int numbufs)

public PageId newPage(Page firstpg, int run_size)
public void freePage(PageId firstid)
public void pinPage(PageId pageno, Page page, boolean skipRead)
public void unpinPage(PageId pageno, boolean dirty)

public void flushPage(PageId pageno)
public void flushAllPages()
public int getNumBuffers()
public int getNumUnpinned()
```

- **newPage(Page firstpg, int run_size)**
  Allocates a set of new pages, and pins the first one in an appropriate frame in the buffer pool.
- **freePage(PageId firstid)**
  Deallocates a single page from disk, freeing it from the pool if needed.
- **pinPage(PageId pageno, Page page, boolean skipRead)**
  Pins a disk page into the buffer pool.
- **unpinPage(PageId pageno, boolean dirty)**
  Unpins a disk page from the buffer pool, decreasing its pin count.
- **flushPage(PageId pageno)**
  Immediately writes a page in the buffer pool to disk, if dirty.
- **flushAllPages()**
  Immediately writes all dirty pages in the buffer pool to disk.
- **getNumBuffers()**
  Gets the total number of buffer frames.
- **getNumUnpinned()**
  Gets the total number of unpinned buffer frames.

Additional Information

How to import code into Eclipse?

1. Open Eclipse.
2. From the top menu click on *File → New → Java Project.*
3. Choose a name for your project and click on the *Finish* button.
4. Now you can see the project package under the *Package Explorer*.
5. Expand your project package. You will see an empty folder named *src*. Open the *Source* folder from the assignment folder and then proceed to the *Code* folder. Select all folders (*bufmgr, global and tests*), drag them and drop them into the *src* folder in Eclipse.
6. From the popped-up dialog, choose the first option (*Copy files and folders*) and click on the *OK* button.
7. Now you have to add JAR libraries to your application. These libraries, implement different components of the database that are needed for this assignment, such as disk manager ("*dm.jar*").
   Right click on ***root folder*** of your application and click on *properties*.
   From the left pane, click on *Java Build Path*.
   Click on the *Add External JARs* button.
   Browse to the Libraries folder in the Code folder from the assignment folder.
   Select all four *.jar* files and click on the *OK* button.
8. Click on the green (*run as*) button and choose *Java Application* from the popped-up dialog. You will receive some errors since the code is incomplete.

How to complete the code template provided?

You only need to modify the two following files to perform your assignment:

- *Source → Code → bufmgr → BufMgr.java*
- *Source → Code → bufmgr → Clock.java*

Below is a copy of the *pinPage()* method from Clock.java class:

```
/**
 * Notifies the replacer of a pined page.
 */
public void pinPage(FrameDesc fdesc) {
  /*
  ?
  ?
  ?
  */
}
```

You need to look for the lines marked with the **3 question marks** (highlighted in the above snapshot) and replace them by your code. You could use the search tool (press *CTRL+ F*) in the Eclipse and look for ***?*** to make sure no required part is missed.
Add **comments** to your code and explain what each line does. With this, you may still receive partial credit in case the final output is not completely correct.

What are the contents of the provided code?

The *Help* Folder

This folder contains documentations for the code. Although you just need to implement the buffer manager layer, reading this documentation helps you better understand how the whole system works.

The *Source* Folder

This folder contains the source code and JAR libraries:

- **Libraries:** Contains implemented components of the ***Minibase*** database which are needed for this assignment.
- **Code:** Contains the source code including the following components
  - **global** contains the kernel code for the ***Minibase*** database; you should NOT modify this!
  - **bufmgr** code for the buffer manager component of the ***Minibase*** database; you have to complete this.
  - **tests** contains a java application which tests your buffer manager component. BMTest.java acts as the main method of your application. You should NOT modify this. This module contains 3 tests as follows:

    Test 1: Performs a simple test of normal buffer manager operations as follows:
    - Allocates a number of new pages
    - Writes (something) on each page
    - Reads (that something) back from each page (because we're buffering, this is where most of the writes happen)
    - Releases the pages

    Test 2: Performs some illegal buffer manager operations, in particular:
    - Tries to pin more pages than there are frames

    Test 3: Evaluates some of the internals of the buffer manager as follows:
    - Allocates and makes dirty some new pages, one at a time, and leaves some pinned
    - Reads the pages

Submission Guideline

Please submit your assignment through **Canvas** before the deadline; late submissions are not accepted! You are allowed to submit your assignment multiple times, but only the last submission (**before the deadline**) will be recorded and graded.

Your submission should be a single file named <Your CU Denver Portal ID>_A2.zip. For example, if your CU Denver Portal ID is john, the file name would be
john_A2.zip. The submitted compressed file must contain the following files:

- ***Questions.pdf***
  This file must contain your answers to the conceptual questions presented in Part I.

- ***Java***
  This folder must contain the following:
  - Clock.java
  - BufMgr.java

- ***Output.pdf***
  This file must contain the output of your code.

- ***Readme.pdf***
  This file must contain your first and last name, CU Denver ID, 9 digit student id, and your UC Denver email address.

Please download your assignment after submission and check its files to make sure that they are not corrupted.

You are highly encouraged to ask your question on Piazza under the "Assignment 2" folder. Please DO NOT include your codes and solutions in the comments you share on Piazza. Feel free to help other students with general questions.