Assignment 3

Issue Date: November 5, 2018
Due Date: November 19, 2018, 10:00 A.M.
$\sum$ 40 Points

Database System Architecture and Implementation
INF-20210
WS 2018/19

Universität Konstanz

University of Konstanz
Database and Information Systems
Prof. Dr. Michael Grossniklaus
Leonard Wörteler

Buffer Management

i General Notes

Please observe the following points in order to ensure full participation in the lecture and to get full credit for the exercises. Also take note of the general policies that apply to this course as listed on the course website http://www.informatik.uni-konstanz.de/grossniklaus/education/inf-20210/.

- If you haven't done so yet, register yourself in the ZEuS, StudIS, and Ilias.
- Submit your solutions through Ilias before the deadline published on the assignment.
- Since this is a programming assignment, you will submit an archive (ZIP file or similar) that only contains the relevant source code files and a README.txt file (see Section Submission below).
- The use of external libraries is not permitted, except if they are explicitly provided by us.
- Submissions with compile-time errors due to syntax or Checkstyle problems will immediately be disqualified. Whereas solutions with run-time errors will be considered, they will not result in a good grade.

i Prerequisites

In order to successfully complete this project, you will need to install the following software on the computer that you use for software development.

- Java Development Kit, available at https://www.oracle.com/technetwork/java/javase/downloads/index.html (version 8 or greater).
- Apache Maven, available at http://maven.apache.org/ (version 3.5.2 or greater).
- Eclipse IDE for Java Developers, available at http://www.eclipse.org/downloads/.
- Eclipse Checkstyle plug-in, installation instructions at https://checkstyle.org/eclipse-cs/ (version 8.12 or greater).

i Minibase

Minibase is a miniature relational DBMS, originally developed at the University of Wisconsin by Raghu Ramakrishnan to support the practical exercises of the text book that we use in our course. In total, there will be four assignments that are based on Minibase. In all of these assignments, you will implement simplified versions of different layers of a typical DBMS, without support for concurrency control or recovery.

We use an updated version of the code written by Chris Mayfield and Professor Walif Aref of Purdue University as well as Professor Leonard Shapiro of Portland State University. More recently, members of the Database and Information Systems group at the University of Konstanz have refactored and extended the Minibase source code for use in this course. The assignments themselves are partly based on assignments provided by Professor Kristin Tufte of Portland State University, Christian Grün of the

University of Konstanz, Chris Mayfield of Purdue University and Raghu Ramakrishnan of the University of Wisconsin. We thank everybody for the development of these exercises and their work on the Minibase code!

After you have installed and configured the required software (see above), you can set up a local copy of the Minibase source code distribution by following the steps below.

- Download the source code distribution archive from Ilias and unzip it in a directory of your choice.
- Conveniently, the project has already been created as an Eclipse project. If you use any other IDE or editor, you may consult the respective documentation on how to import Eclipse projects.
- Now you are ready to launch Eclipse and import the Minibase project. In order to do so, follow the steps of the import project wizard (File → Import... → General → Existing Projects into Workspace). In the wizard select the directory where you have unzipped the source code distribution under Select root directory and make sure that the option Copy projects into workspace is not selected.
- Once Eclipse has finished importing and compiling the project, there should be no errors, but there might be a couple of warnings. Make sure that Checkstyle is activated for the project by right-clicking on the project and selecting Checkstyle → Activate Checkstyle.
- Congratulations! If you have made it to this point without errors, you are ready to begin your programming project.

Disclaimer: Please note that Minibase is neither open-source, freeware, nor shareware. Refer to the file COPYRIGHT.txt in the doc folder of the Minibase distribution for terms and conditions. Also do not push the code to publicly accessible repositories, e.g., GitHub.

i Programming Practices

Checkstyle will already check the formatting of the code and enforce a certain coding convention. The Checkstyle configuration that we use is a slightly more relaxed version of the default Java coding convention. Note that the Eclipse project is already configured to ensure that your code is formatted accordingly as you write it (Preferences... → Java → Code Style).

Apart from the rules enforced by Checkstyle your code also needs to adhere to the following (good) programming practices.

- Use meaningful names for classes, interfaces, methods, fields, and parameters. For example, use BufferManager instead of BMgr or page instead of p.
- Organize your code by grouping it into (conceptual) blocks that are separated by a new line.
- Write a (Javadoc) comment before each method and paragraph of code.
- Provide a comment before each non-obvious declaration.
- Your code must be understandable by reading the comments only.
- Comments should not simply paraphrase what the code does, but rather explain it. For example, your comment for the line of code clock = 0; should not read "Assign 0 to variable clock". A good comment would be "Initialize the clock to point to the first page in the buffer pool".
- Efficiency is considered, e.g., do not use an int if a boolean is appropriate, do not do a sequential search or a disk I/O unless necessary, etc.

i Assignment Overview

You will be given parts of the Minibase code, in Java, and asked to fill in other parts of it. You may spend 90% of your time understanding the given code and 10% of your time writing new code.

The best place to start understanding this assignment are the lecture slides and the text book (Sections 9.3 and 9.4) about disk space manager and the buffer manager. Following the reference architecture presented in the lecture, Minibase is structured in layers. Each layer corresponds to a Java package. For example, the disk manager is located in the minibase.storage.file package. All of these packages are managed as subdirectories of src/main/java, where directory src is directly located in the root directory of the Minibase distribution. For each package of the main Minibase system, there is a corresponding package of the same name that contains JUnit tests, which you will use to check the correctness of your

implementation. The JUnit tests are located in subdirectories of src/test/java.

In this assignment, you will be given the disk space manager layer (minibase.storage.file), which manages pages in a file on the disk. Before you start implementing you should study the existing code and make sure that you have an in-depth understanding of how it works. Your first task is to write the next higher layer, i.e., the buffer manager layer (minibase.storage.buffer), which builds on functionality from the disk space manager layer. Your second task is to implement alternative buffer replacement policies to be used in the buffer manager. The names of the files you are supposed to change and submit all have the suffix Group00.

Exercise 1: Buffer Manager (25 Points) 

As mentioned above, your first task is to implement the Minibase buffer manager. In the minibase.storage.buffer package, you will find a skeleton class called BufferManagerGroup00 that contains all methods you will need to implement. Rename the class according to your group number, e.g., change BufferManagerGroup00 to BufferManagerGroup42 if you are "Group42". Currently, all methods have been implemented to throw an UnsupportedOperationException, i.e., they will simply fail when invoked. Look through all the method signatures and read the comments provided for each method.

Once you understand the structure of the buffer manager and the functionality that it will provide, you will begin by implementing the constructor of the class BufferManager. The main task of this constructor is to initialize the buffer pool as an array of Page<?> objects, which will be referred to as buffer pages. The special interface PageType is a so-called phantom type which describes the current contents of the buffer page without the need for wrapping it with another object at run-time. Inside the buffer manager you can treat all pages as Page<?> objects without specifying concrete types. In BufferManager#pinPage(PageID) the returned page object must be cast to the type requested by the caller.

Each buffer page has certain information associated with it, which indicates whether a page is dirty, whether it includes valid data, i.e., data that reflects the data in a disk page, and, if it includes valid data, what the disk page number of the data is, and how many callers have pins on the data, i.e., the pin count. You have to make sure that all this is kept up-to-date in all situations. If you need to store additional information, try to include them in the classes you have to modify anyway (e.g. the ClockPolicy[...] instead of the Page) in order to keep your changes well encapsulated.

Your BufferManager class will need to determine, very efficiently, what buffer page a given disk page occupies. You may wish to use Java's HashMap class to map a disk page number to a frame. This mapping will also tell you if a disk page is not in the buffer pool. As you write methods such as pinPage() it is very important, and a prime source of bugs, to keep the buffer pool and your mapping current.

Your primary goal in this exercise is to fill in the stubs in BufferManagerGroup00.java so that the tests in BufferManagerTest.java run successfully, and so that the specifications above and in the documentation are met. If you absolutely need to create additional classes, put them in the minibase.storage.buffer package. Study the code for DiskManagerTest.java in the tests package. This may be useful to see how to call the methods in DiskManager, and to verify that DiskManager actually works. Look even more carefully at BufferManagerTest.java. Note that in BaseTest.java DB_SIZE is set to 20000 and BUF_SIZE is set to 100. You may wish to set these artificially lower for debugging purposes. You are encouraged to write additional unit tests, but you should not submit them.

Exercise 2: Buffer Replacement Policies (15 Points) 

You can find an existing buffer replacement policy in the minibase.storage.buffer.policy package that selects a random victim (RandomPolicy). The buffer replacement strategy can be configured in BaseTest.java. We recommend you use this replacement policy to test your implementation of the first exercise. In order for the buffer replacement policy to work correctly, you will need to call its method stateChanged() with the correct PageState at the correct positions in the code of BufferManager. This method is intended to inform the policy about the state of the buffer pool and to enable it to update its internal data structures.

Once you have made sure that the buffer manager and the policy are synchronized, you can use the following line of code in your implementation of the pinPage() method to figure out which page is best to replace.

```
final int frameNo = this.replacementPolicy.pickVictim();
```

As you can see from looking at the code of the ReplacementStrategy enumeration, Minibase has been designed to support a number of different buffer replacement strategies. As soon as you feel confident that your buffer manager works correctly with the random policy, your second exercise is to implement the three missing strategies, i.e., LRU, MRU, and Clock. The different algorithms for these policies have been discussed in the lecture, but are also repeated below. For each algorithm, you will need to implement the methods defined in the interface ReplacementPolicy according to these algorithms. The names of the classes are LRUPolicyGroup00, MRUPolicyGroup00, and ClockPolicyGroup00, where you should substitute Group00 with your group number, e.g., LRUPolicyGroup10 for "Group10".

LRU: Least Recently Used

Initialization: The LRU policy maintains a FIFO queue that is initialized with the frame descriptors $1 \ldots N$.

Update: If the pinCount of a page in a frame becomes 0, the frame descriptor is moved to the end of the queue.

Pick Victim: Return the index of the first frame in the queue containing a page with pinCount == 0, remove that frame from the queue, and append it to the end.

MRU: Most Recently Used

Initialization: The MRU policy maintains a LIFO stack that is initialized with the frame descriptors $N \ldots 1$.

Update: If the pinCount of a page in a frame becomes 0, the frame descriptor is moved to the top of the stack.

Pick Victim: Return the index of the first frame in the stack containing a page with pinCount == 0, remove that frame from the stack, and move it to the top of the stack.

Clock: Second Chance

Initialization: For every page, the Clock policy maintains a flag whether the page is referenced. Additionally, the clock hand current needs to be initialized to 0.

Update: If the pinCount of a page in a frame becomes 0, mark the page as referenced.

Pick Victim

```
begin
    round ← 0;
    while round < 2 · N do
        pos ← current position of the clock hand;
        advance the clock's hand one position;
        if the page at position pos is pinned then
            skip the page;
        else
            ref ← referenced bit of pos;
            if ref is false then
                return pos
            else
                set the referenced bit of pos to false;
        round ← round + 1;
    return −1
```

i Submission

Solutions are submitted electronically by Ilias. Your submission must consist of a single zipped archive that contains all the relevant Java files, i.e., all files that you have modified or created. You will also include a file called README.txt that should contain the following information.

Overall Status A one to two paragraph overview of how you implemented the major components. If you were unable to finish the project, please give details about what is and is not complete. Be short and to the point!

File Descriptions List all required files you submit. If you have created any new files, list their names and a short description.

Time Spent Please include how many hours you spent on this project. Note that the time spent has no impact on your grade. This information will only be used in planning future projects.

The name of the file containing your submission should be grp#-asg#.zip, where the first # is your group and the second # the assignment number.