Lab 2: Buffer Pool 3

Instructor: Beom Heyn Kim

beomheynkim@hanyang.ac.kr

Department of Computer Science

Outline

- Buffer Pool Manager
- Assignment

Buffer Pool Manager

Next, you need to implement the buffer pool manager in your system (BufferPoolManager). The BufferPoolManager is responsible for fetching database pages from the DiskManager and storing them in memory. The BufferPoolManager can also write dirty pages out to disk when it is either explicitly instructed to do so or when it needs to evict a page to make space for a new page.

To make sure that your implementation works correctly with the rest of the system, we will provide you with some of the functions already filled in. You will also not need to implement the code that actually reads and writes data to disk (this is called the DiskManager in our implementation). We will provide that functionality for you.

All in-memory pages in the system are represented by Page objects. The BufferPoolManager does not need to understand the contents of these pages. But it is important for you as the system developer to understand that Page objects are just containers for memory in the buffer pool and thus are not specific to a unique page. That is, each Page object contains a block of memory that the DiskManager will use as a location to copy the contents of a **physical page** that it reads from disk. The BufferPoolManager will reuse the same Page object to store data as it moves back and forth to disk. This means that the same Page object may contain a different physical page throughout the life of the system. The Page object's identifer (page_id) keeps track of what physical page it contains; if a Page object does not contain a physical page, then its page_id must be set to INVALID_PAGE_ID.

Buffer Pool Manager (Cont.)

Each Page object also maintains a counter for the number of threads that have "pinned" that page. Your BufferPoolManager is not allowed to free a Page that is pinned. Each Page object also keeps track of whether it is dirty or not. It is your job to record whether a page was modified before it is unpinned. Your BufferPoolManager must write the contents of a dirty Page back to disk before that object can be reused.

Your BufferPoolManager implementation will use the LRUReplacer class that you created in the previous steps of this assignment. It will use the LRUReplacer to keep track of when Page objects are accessed so that it can decide which one to evict when it must free a frame to make room for copying a new physical page from disk.

You will need to implement the following functions defined in the header file (src/include/buffer/buffer_pool_manager.h) in the source file (src/buffer/buffer_pool_manager.cpp):

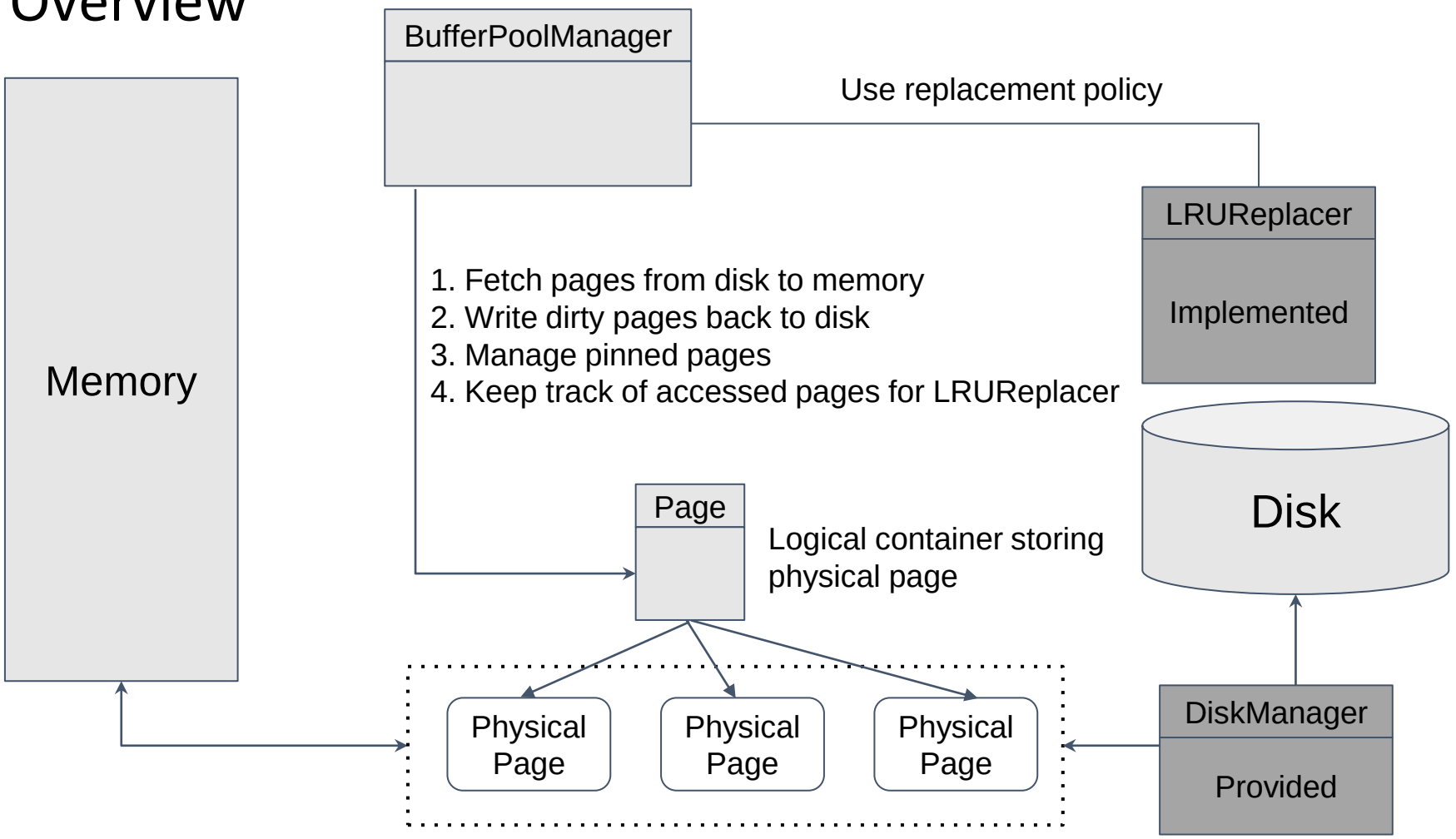
- FetchPageImpl(page_id)
- NewPageImpl(page_id)
- UnpinPageImpl(page_id, is_dirty)

For FetchPageImpl, you should return NULL if no page is available in the free list and all other pages are currently pinned. FlushPageImpl should flush a page regardless of its pin status.

Refer to the function documentation for details on how to implement these functions. Don't touch the non-impl versions, we need those to grade your code.

Buffer Pool Manager (Cont.)

Overview



Outline

- Buffer Pool Manager
- Assignment

Assignment: Buffer Pool Manager (Cont.)

- Go ahead and implement buffer pool manager following the specification provided in previous slides
- Also, make sure you **build&run the test** to check your code!

```
$ mkdir build
$ cd build
$ make buffer_pool_manager_test
$ ./test/buffer_pool_manager_test
```

The End