Database systems: Lab Assignment

Khalid Belhajjame

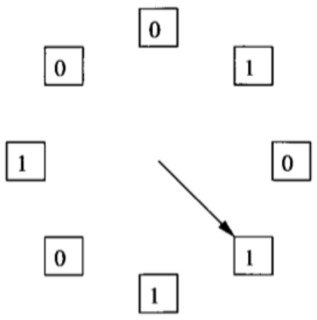
Buffer management strategies

The critical choice that the buffer manager must make is what block to though out of the buffer pool when a buffer is needed for a newly requested block. In what follows, we present three strategies:

Least Recently Used (LRU)
The LRU rule is to throughout the block that has not been used or written for the longest time. This method requires that the buffer manager maintain a table indicating the last time the block in each buffet was accessed. So there is a significant effort in maintaining this information. However, LRU is an effective strategy; intuitively, buffers that have not been used for long time are less likely to be accessed sooner than those that have been accessed recently.

First-In-First-Out (FIFO)
When a buffer is needed, under the FIFO policy the buffer that has been occupied the longest by the same block is emptied and used for the new block. In this approach, the buffer manager needs to know only the time at which the block currently occupying a buffer was loaded into that buffer. An entry into the table can thus be made when the block is read from disk, and there is no need to modify the table when the block is accessed. FIFO requires less maintenance then LRU, but it can make more mistakes. For example, a block that is used repeatedly, will eventually become the oldest bloke in a buffer. It will be written back to disk, only to be reread shortly thereafter into another buffer.

The "Clock" Algorithm
This algorithm is commonly implemented, is an efficient approximation to LRU. Think of the buffers as arranged in a circle, as suggested by the figure below. A hand point to one of the buffers, and will rotate clockwise if it needs to find a buffer in which to place a disk block. Each buffer has an associated "flag" which is either 0 or 1. Buffers with 0 flag are vulnerable to having their contents sent back to disk; buffers with a 1 are not. When a block is read into a buffer, its flag is set to 1. Likewise, when the contents of a buffer is access, its flag to 1.



When a buffer manager needs a buffer for new block, it looks for the first 0 it can find, rotating clockwise. If it passes 1's, it sets them to 0.

Requested work

Your job is to implement a buffer manager that supports the above described page replacement strategies. We will assume that there are four page slots (i.e., frames) that your buffer manager must manage: P1, P2, P3, and P4. All four slots are empty to start. When the buffer pool has unused slots (such as at the beginning, when all four slots are empty), it will put newly read data in the first unused slot. The pages to be read from disk are labelled A through G. To simplify the exercise, we will assume that for each access the page is pinned, and then immediately unpinned. (In general, a page may be pinned for any length of time.).

You task is:

1. Understand the three replacement strategies.
2. Implement them.
3. Test and compare the different replacement strategies.

For 3, we will use workloads that specify a sequence of page accesses.
For example, given the workload below, examine the output of your program for the three replacement strategies. To evaluate the three strategies, you will need to count the number of page misses, i.e., the number of times a page is not found in the buffer and needs to be fetched from the permanent memory.

Time	Page Read
T1	A
T2	B
T3	C
T4	D
T5	E
T6	A
T7	B
T8	C
T9	D
T10	E

Logistics

You will work in teams of two.
I will see what you have done the beginning of the next Lab Session.
Each team will show me their code, the example workloads they used, the results they obtained using their program, as well as the analysis or lessons they've learned about

the different replacement strategies. At end of your lab-assignement, put these element within your Team channel.