DS 660 Programming Assignment 0

Apoorva Gupta and Tanish Bhowmick

Submitted code link - https://github.com/tannyb28/cs660-Spring2025-pa/blob/main/docs/pa0.md

1. **Evicting a dirty page is more expensive than evicting a clean page. Propose a strategy to reduce the number of dirty pages that are evicted.**

   Evicting a dirty page is more expensive than evicting a clean page because the dirty page must be written back to disk before eviction, incurring additional I/O overhead due to writing and taking additional memory space. We can use following ways to reduce the number of evictions

   1. We can do batch flushing in which a vector of fixed length will store the information of dirty pages instead of writing a single page to disc everytime encountered, here the I/O operations for disk writing will be reduced and multiple dirty pages from a group will be flushed together.
   2. We can have a background flushing job that will scan the buffer pool periodically for dirty pages and flush them to disk, it prevents the writing of multiple dirty pages just before eviction reducing eviction time.
   3. To improve the background flushing process, we can use an adaptive approach based on dirty page ratio i.e we can set a threshold for the buffer pool and if dirty page percentage let's say increases by more than 70% we will start the flushing dirty page job before evicting them.

   With these strategies, we can significantly reduce the number of dirty pages that need to be evicted from the system, thereby improving buffer pool performance and reducing disk I/O overhead.

2. **The Database::remove flushes any dirty pages associated with the file before it is removed. What are the implications of leaving the pages in the buffer pool when a file is removed? Can you identify a possible bug?**

   If the pages associated with a file remain in the buffer pool after the file is removed, following issues might arise

   1. Since the buffer pool is still containing pages from the file that no longer exist we will get dangling references. If we try to access these pages, it will lead to erroneous behaviour or crashes due to accessing invalid memory
   2. It will lead to stale data usage in case of new file being added with same name, the buffer pool will still reference to the old pages leading to stale or corrupted data being returned
   3. Since buffer pool has the information from non-existent files, it will reduce the number of available slots for the other files this will lead to memory leak in bufferpool and affect the database performance

   Based on the points discussed above we can have a possibility of pages getting flushed from the database but does not remove them from the buffer pool. To fix this we need to explicitly discard all pages from the buffer pool that belong to the removed file after flushing the dirty pages,

3. **When would you need to discard a page from the buffer pool without writing it back to disk?**

   This can be done in following cases

   1. Transaction Rollback - If in a transaction a change is made on a page but later those changes are aborted, any of the made changes must be discarded. Since the changes were never committed, the page should be removed from the buffer pool without writing it to disk back again.
   2. Page Deletion - When a page or a set of pages is deleted, those pages should be directly removed from memory without writing back to disk. Writing them back would be unnecessary and wasteful since they are no longer available for querying.
   3. Corrupted or Invalid Pages - If we detect a page in memory to be corrupt or invalid, it should be discarded without writing it to disk to prevent overwriting. This scenario might occur due to an unexpected crash, or logical inconsistencies.

   Our code handled it as the existing discardPage method correctly removes a page from memory without flushing it to disk, ensuring that pages can be discarded safely whenever required.