Lecture #05: Buffer Pools

15-445/645 Database Systems (Fall 2019)

https://15445.courses.cs.cmu.edu/fall2019/

Carnegie Mellon University
Prof. Andy Pavlo

1 Locks vs. Latches

We need to make a distinction between locks and latches when discussing how the DBMS protects its internal elements.

Locks

- Protect the database logical contents (e.g., tuples, tables, databases) from other transactions.
- Held for transaction duration.
- Need to be able to rollback changes.

Latches

- Protects the critical sections of the DBMS's internal data structures from other threads.
- Held for operation duration.
- Do not need to be able to rollback changes.

2 Buffer Pool

The *buffer pool* is an in-memory cache of pages read from disk. The DBMS always knows better so we want to manage memory and pages ourselves

It is a region of memory organized as an array of fixed size pages. Each array entry is called a **frame**. When the DBMS requests a page, an exact copy is placed into one of these frames

Meta-data maintained by the buffer pool:

- **Page Table:** In-memory hash table that keeps track of pages that are currently in memory. It maps page ids to frame locations in the buffer pool.
- **Dirty-flag:** Threads set this flag when it modifies a page. This indicates to storage manager that the page must be written back to disk.
- **Pin Counter:** This tracks the number of threads that are currently accessing that page (either reading or modifying it). A thread has to increment the counter before they access the page. If a page's count is greater than zero, then the storage manager is not allowed to evict that page from memory.

Optimizations:

- **Multiple Buffer Pools:** The DBMS can also have multiple buffer pools for different purposes. This helps reduce latch contention and improves locality
- **Pre-Fetching:** The DBMS can also optimize by pre fetching pages based on the query plan. Commonly done when accessing pages sequentially.
- **Scan Sharing:** Query cursors can attach to other cursors and scan pages together.

Allocation Policies:

- **Global Policies:** How a DBMS should make decisions for all active txns.
- **Local Policies:** Allocate frames to a specific txn without considering the behavior of concurrent txns.

3 Replacement Policies

A replacement policy is an algorithm that the DBMS implements that makes a decision on which pages to evict from buffer pool when it needs space.

Implementation goals:

- Correctness
- Accuracy
- Speed
- Meta-data overhead

Least Recently Used (LRU)

- Maintain a timestamp of when each page was last accessed.
- DBMS picks to evict the page with the oldest timestamp.

CLOCK

Approximation of LRU without needing a separate timestamp per page.

- Each page has a reference bit
- When a page is accessed, set to 1

Organize the pages in a circular buffer with a "clock hand"

- Upon sweeping check if a pages bit is set to 1
- If yes, set to zero, if no, then evict
- Clock hand remembers position between evictions

Alternatives

Problems with LRU and Clock replacement policies:

- LRU and Clock are susceptible to **sequential flooding** where the buffer pool's contents are trashed due to a sequential scan.
- It may be that the LRU page is actually important due to not tracking meta-data of how a page is used.

Better solutions:

- LRU-K: Take into account history of the last K references
- Priority hints: Allow txns to tell the buffer pool whether page is important or not
- Localization: Choose pages to evict on a per txn/query basis