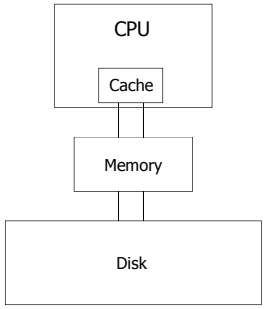
CS422 Principles of Database Systems
Buffer Management

Chengyu Sun
California State University, Los Angeles

Memory Hierarchy



Buffers in a Computer

- Disk cache
- Memory buffer
- L1, L2, and L3 caches

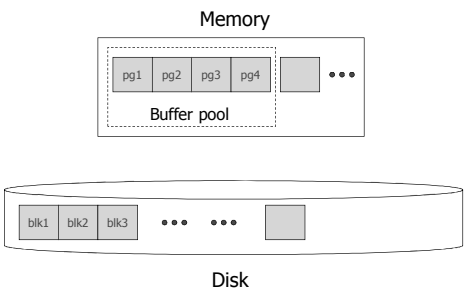
Why OS Memory Buffer Is Not Enough

- DBMS knows its data better
- Database buffer management must be coordinated with failure recovery mechanisms

Buffer Manager

- A buffer manager is a software component of a DBMS that manages a fixed set of pages, called a buffer pool
- Each page in the buffer pool is called a buffer page

Buffer Pool



Access Data on Disk

- Other DBMS components (i.e. client code) access data on disk through the buffer manager

```
// load block #1 into a buffer page
Page page = bufferManager.pin( 1 );
// read the int value at position 100
int i = page.getInt( 100 );
// set the int value at position 100
page.setInt( 100, i+10 );
// indicate this page is no longer used
bufferManager.unpin( page );
// save the changed data to disk
bufferManager.flush( page );
```

About Disk Access

- Disk access has to go through *buffer manager*
- Disk access is by block
  - Read
  - Write
- Buffer Manager API
  - `pin`, `unpin`, `flush`

Pin and Unpin

- `Pin`
  - Load a block into a buffer page
  - Indicate the buffer page is being used by some client code (i.e. pinned) – *how??*
- `Unpin`
  - Indicate the buffer page is no longer used by the client (i.e. not pinned, or unpinned)

Four Possible Cases for Pin

- The block to be pinned is already in the buffer pool
  - The buffer is not pinned
  - The buffer is pinned
- The block to be pinned is not in the buffer pool
  - There is at least one unpinned buffer
  - There is no unpinned buffer

Dirty and Flush

- If the data in a page is changed, the page is called a dirty page
- `Flush`
  - Write a dirty page to disk
- When to flush
  - Before the page is pinned to a different block
  - At the request of the failure recovery mechanism

Example: Buffer Replacement

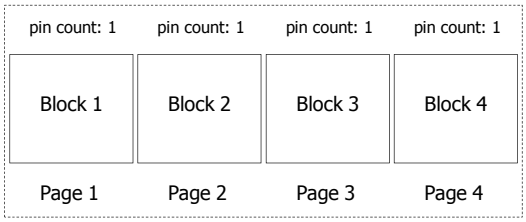
- Size of buffer pool: 4
- What does the buffer pool looks like after the following requests: `pin(1)`, `pin(2)`, `pin(3)`, `pin(4)`, `unpin(3)`, `unpin(1)`, `unpin(2)`, `pin(5)`, `pin(3)`

Buffer Replacement Policies

- Naïve
  - Sequentially scan the buffer pool and replace the first unpinned page
- Clock
- FIFO (First In First Out)
- LRU (Least Recently Used)

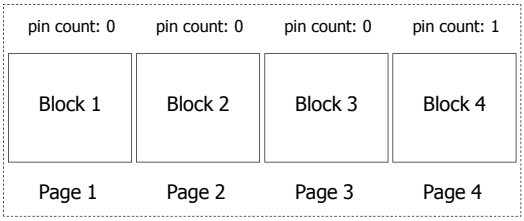
Naïve Policy Example …

After `pin(1), pin(2), pin(3), pin(4)`



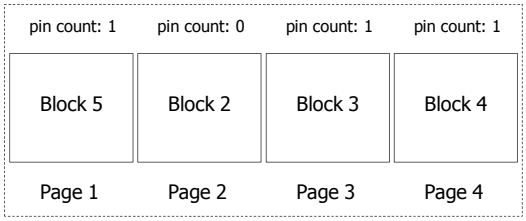
… Naïve Policy Example …

After `unpin(3), unpin(1), unpin(2)`



… Naïve Policy Example

After `pin(5), pin(3)`



Problem of the Naïve Policy

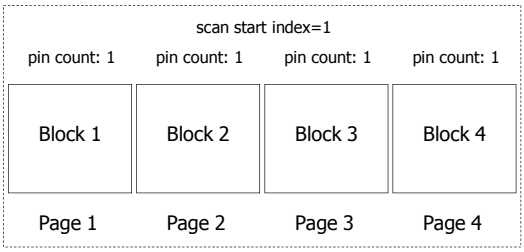
- `pin(1), unpin(1), pin(2), unpin(2), pin(1), unpin(1), pin(2), unpin(2)…`

Clock Policy

- Sequentially scan the buffer pool and choose the first unpinned page
- Start the next scan at the page after the previous replacement

Clock Policy Example

After `pin(1), pin(2), pin(3), pin(4)`

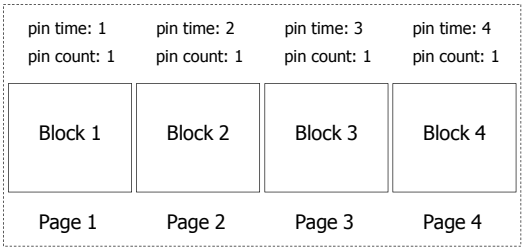


Implementing FIFO and LRU

- FIFO
  - For each buffer page, keeps the time when the block is pinned in
- LRU
  - For each buffer page, keeps the time when the page is unpinned

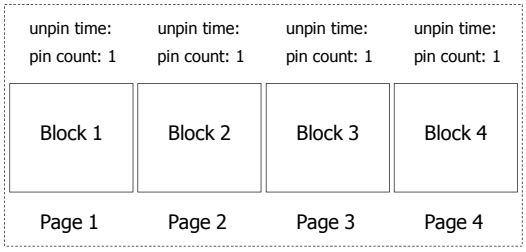
FIFO Policy Example

After `pin(1), pin(2), pin(3), pin(4)`



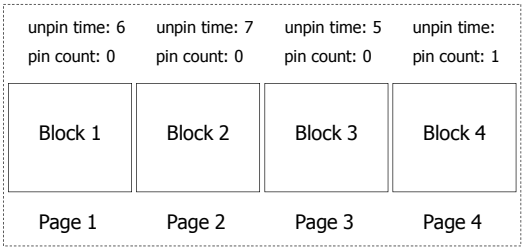
LRU Policy Example …

After `pin(1), pin(2), pin(3), pin(4)`



… LRU Policy Example

After `unpin(3), unpin(1), unpin(2)`



Readings

- Chapter 13.4 and 13.5 of the textbook