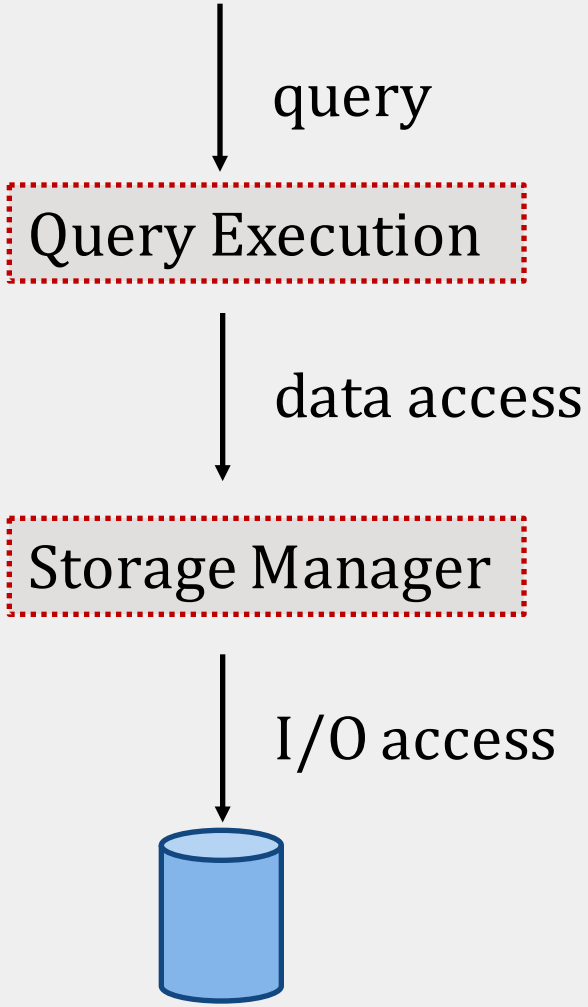
STORING DATA: DISK AND FILES

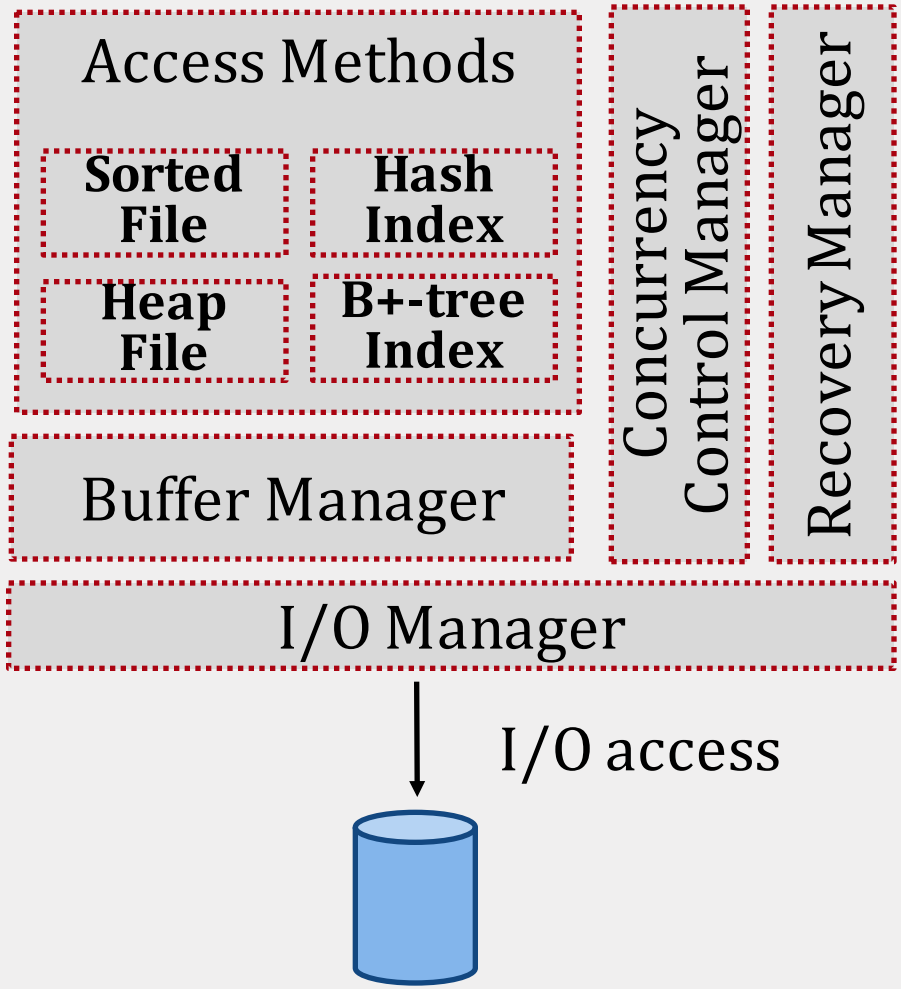
CS 564- Fall 2015

ACKs: *Dan Suciu, Jignesh Patel, AnHai Doan*

Architecture Of a DBMS



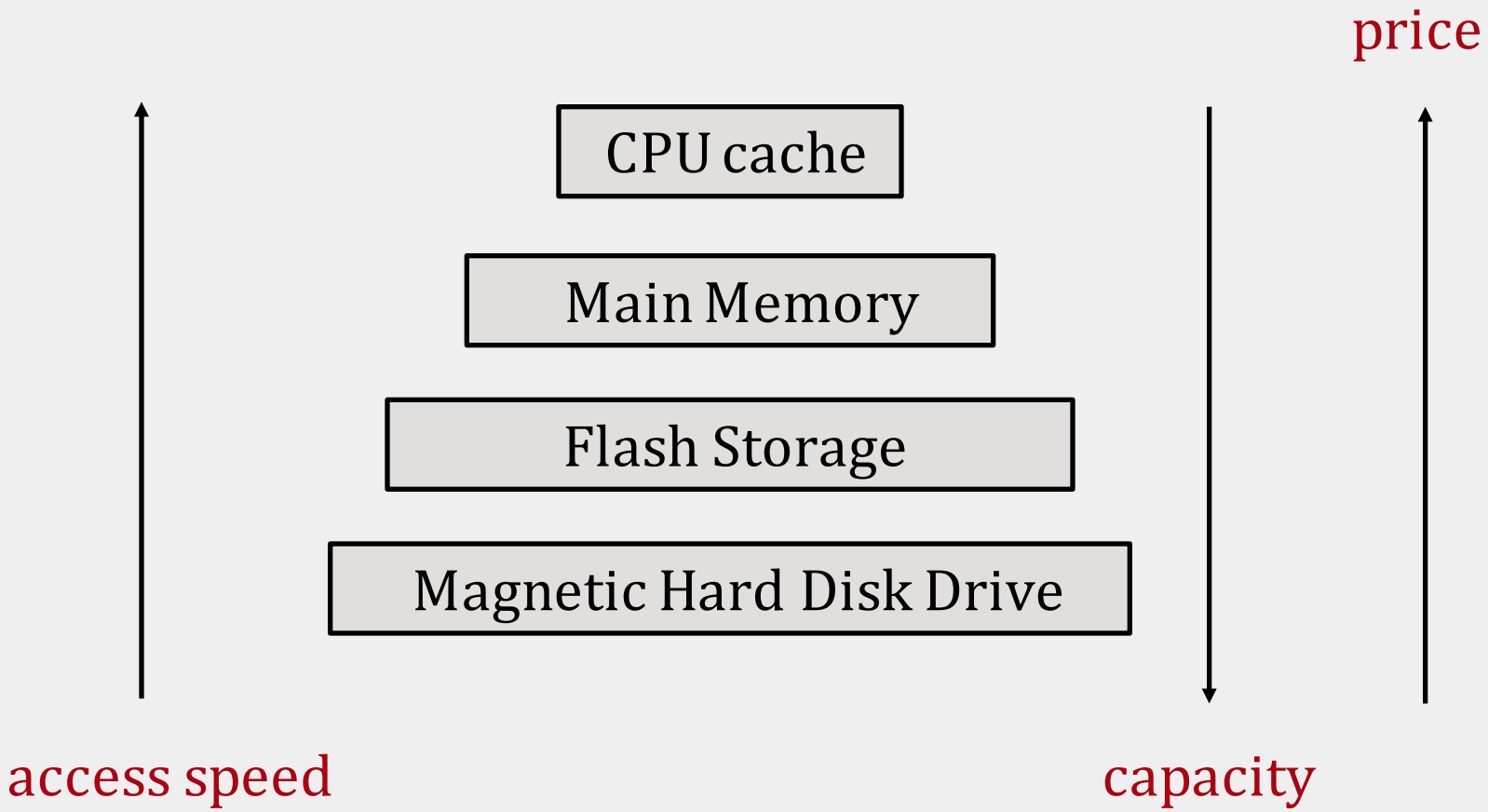
Architecture of Storage Manager



DATA STORAGE

- How does a DBMS store and access data?
  - disk
  - main memory
- How do we move data from disk to main memory?
  - pages
- How do we organize relational data into files?

Memory Hierarchy



Why Not Main Memory?

- Relatively high cost
- Main memory is not persistent!
- Typical storage hierarchy:
  - Main memory (RAM) for currently used data
  - Disk for the main database (secondary storage)
  - Tapes for archiving older versions of the data (tertiary storage)

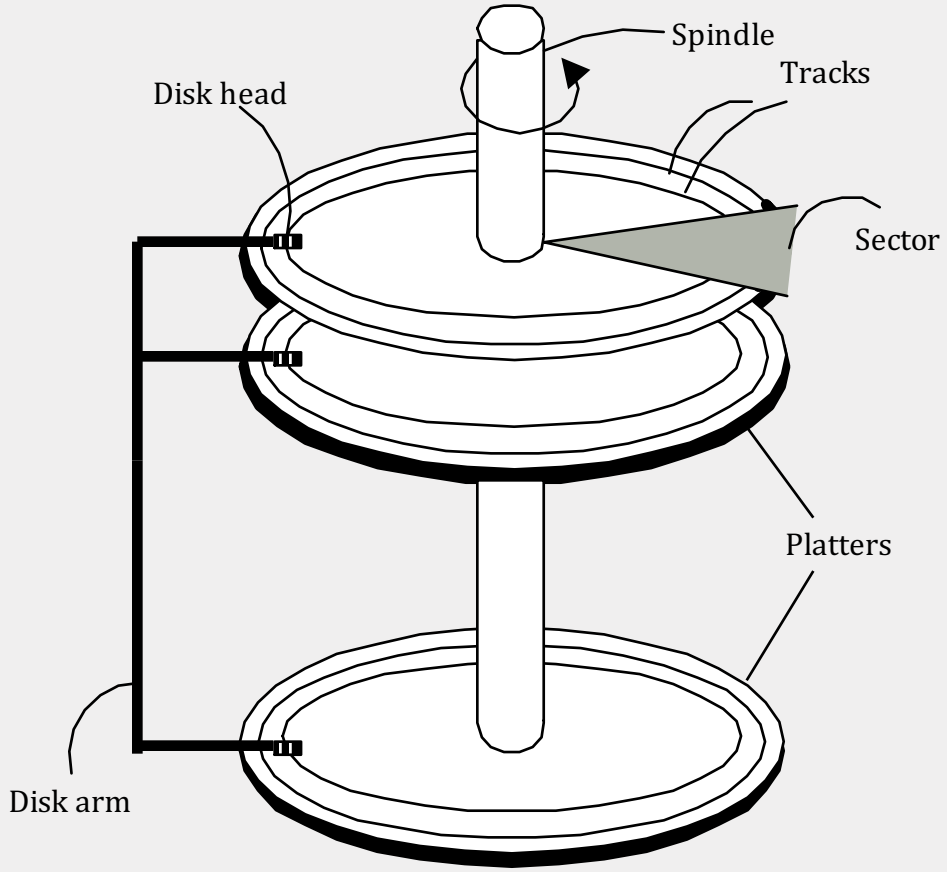
DISK

DISKS

- Secondary storage device of choice
- Data is stored and retrieved in units called disk blocks or pages
- The time to retrieve a disk page varies depending upon location on disk
  - The placement of pages on disk has major impact on DBMS performance!

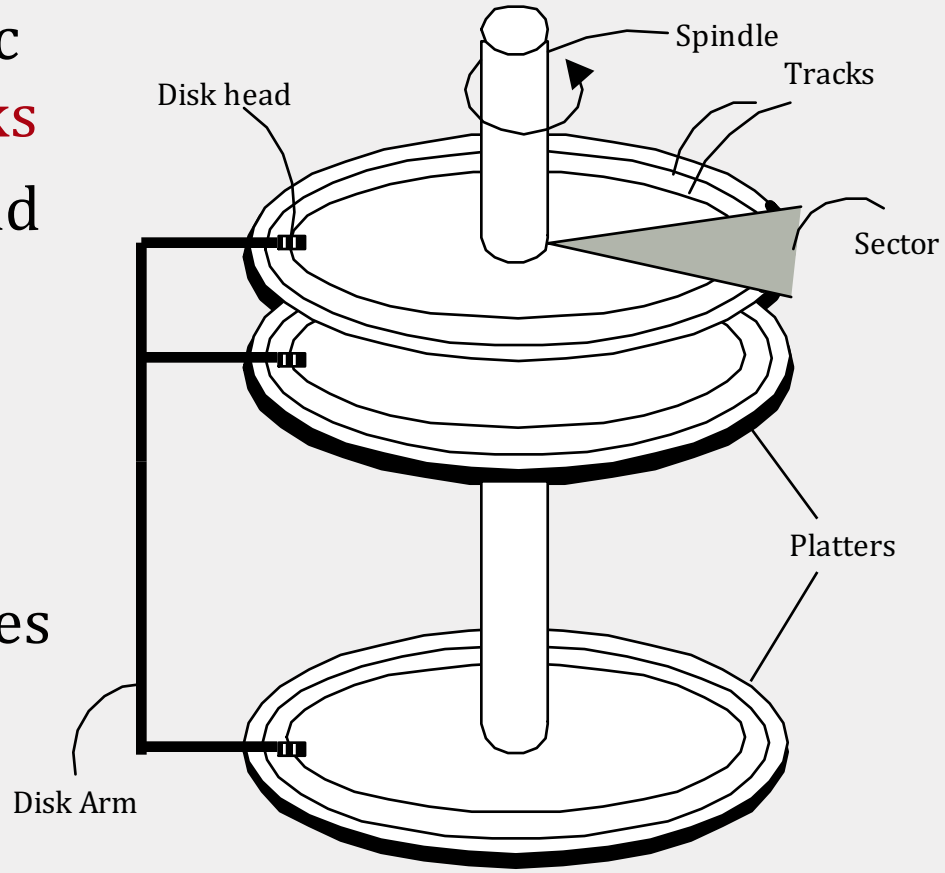
Components Of Disks

- platter: circular hard surface on which data is stored by inducing magnetic changes
- spindle: responsible for rotating the platters
- RPM (**R**otations **P**er **M**inute)
  - 7200 RPM – 15000 RPM



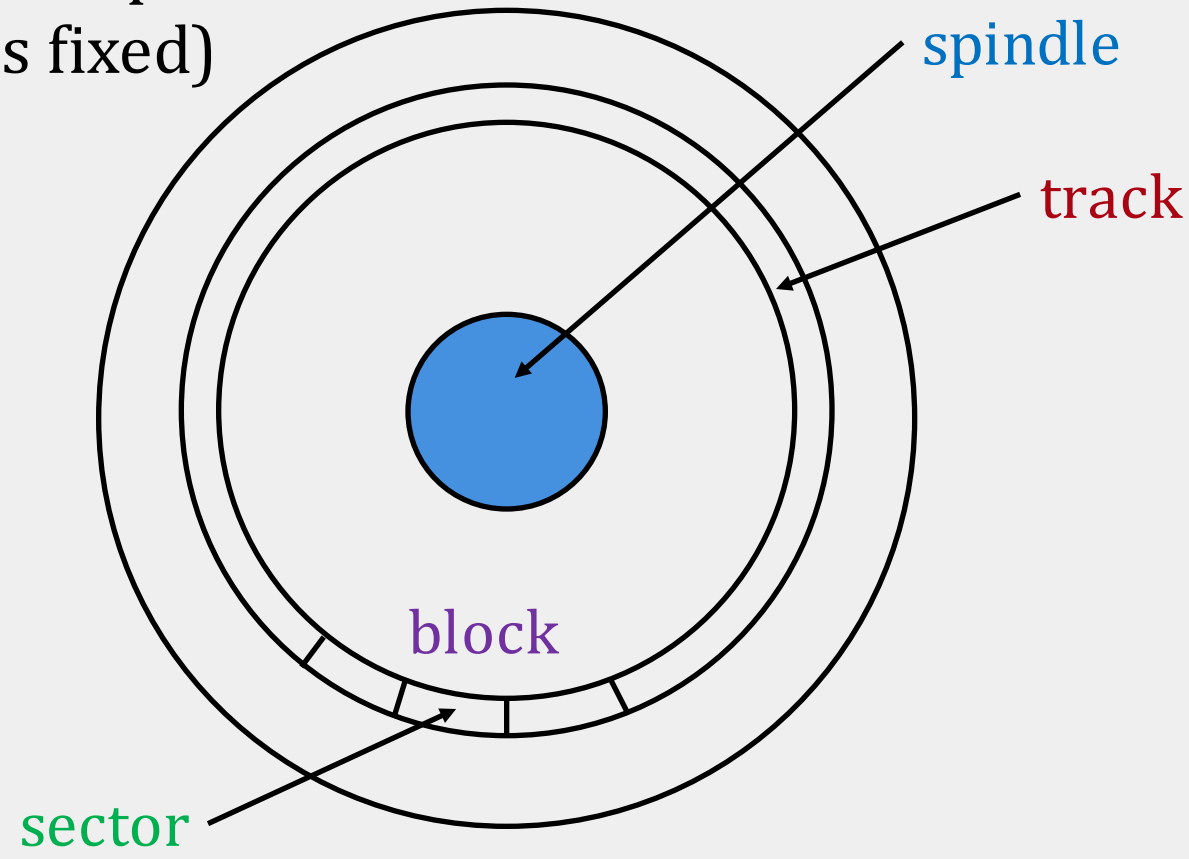
Components Of Disks

- data is encoded in concentric circles of sectors called tracks
- disk head: mechanism to read or write data
- The disk arm moves to position a head on a desired track
- exactly one head reads/writes at any time



Components Of Disks

block size : multiple of sector size (which is fixed)



ACCESSING THE DISK (1)

access time = seek time + rotational delay + transfer time

rotational delay: time to wait for sector to rotate under the disk head

- typical delay: 0 - 10ms
- average vs maximum delay

RPM	Average delay
5,400	5.56
7,200	4.17
10,000	3.00
15,000	2.00

ACCESSING THE DISK (2)

access time **=** seek time **+** rotational delay **+** transfer time

seek time: time to move the arm to position disk head on the right track

- typical seek time: ~9ms
- ~4ms for high-end disks

ACCESSING THE DISK (3)

access time **=** seek time **+** rotational delay **+** transfer time

data transfer time: time to move the data to/from the disk surface

- typical rates: ~100MB/s
- access time is dominated by seek time and delay!

EXAMPLE: SPECS

	Seagate HDD
Capacity	3 TB
RPM	7,200
Average Seek Time	9ms
Max Transfer Rate	210 MB/s
# Platters	3

What are the I/O rates for block size 4KB and:

- random workload (~ 0.3 MB/s)
- sequential workload (~ 210 MB/s)

ACCESSING THE DISK

- Blocks in a file should be arranged sequentially on disk to minimize seek and rotational delay!!

- `next' block concept:
  - blocks on same track, followed by
  - blocks on same cylinder, followed by
  - blocks on adjacent cylinder

MANAGING DISK SPACE

- The disk space is organized into files
- Files are made up of pages
- Pages contain records

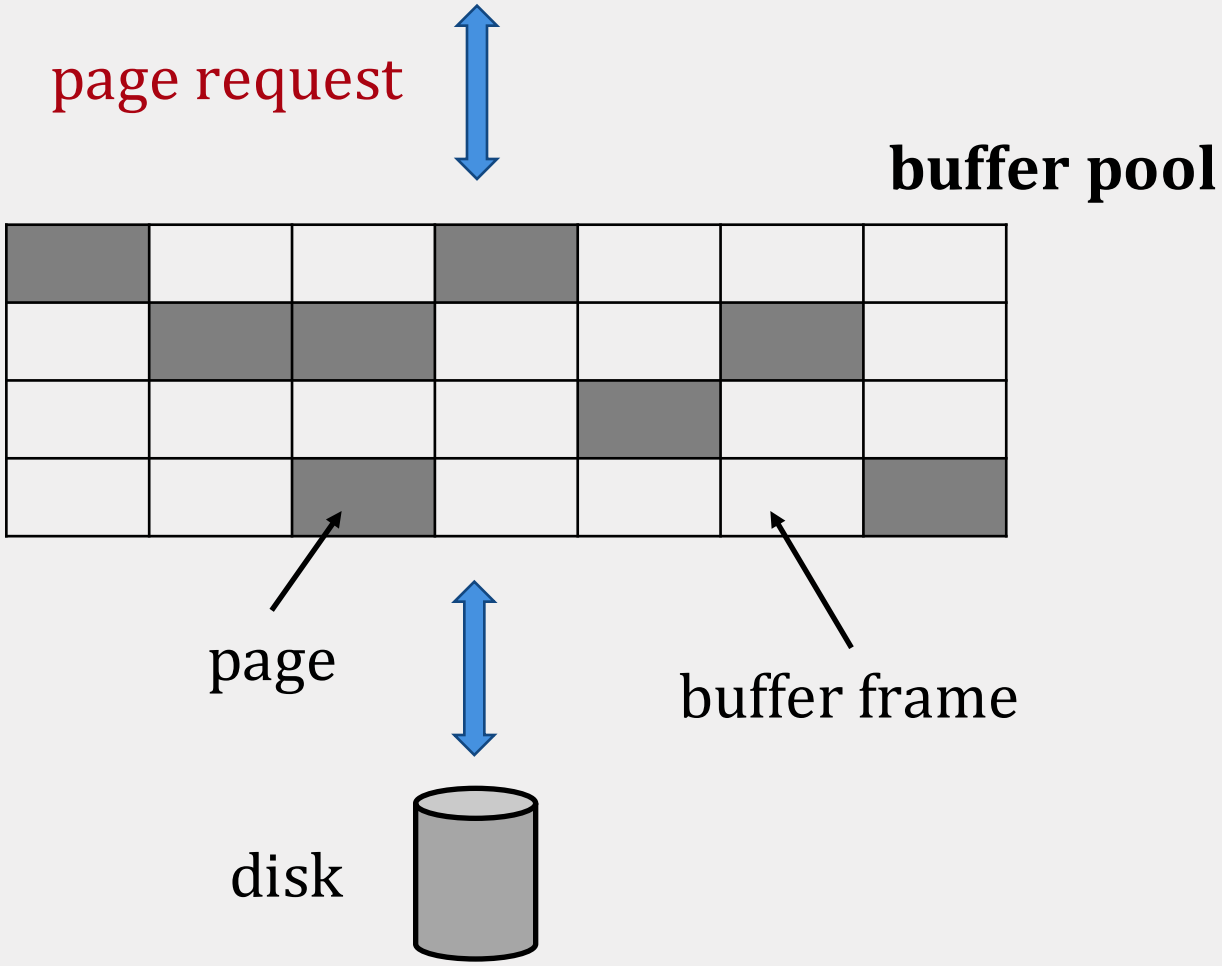
- Data is allocated/deallocated in increments of pages
- Logically close pages should be nearby in the disk

Buffer Management

Buffer Manager

- Data must be in RAM for DBMS to operate on it
- All the pages may not fit into main memory
- Buffer manager: responsible for bringing pages from disk to main memory as needed
  - pages brought into main memory are in the buffer pool
  - the buffer pool is partitioned into buffer frames: slots for holding disk pages

BUFFER MANAGER



Requests to Buffer Manager

The higher level of the DBMS can:

- request a page (w/o worrying if it is in memory)
- release a page when no longer needed
- notify when a page is modified

BOOKKEEPING

Bookkeeping per frame:

- pin count : # current users of the page
  - *pinning* : increment the pin count
  - *unpinning* : release the page (pin count is 0)
- dirty bit :indicates if the page has been modified (so changes must be propagated to disk)

PAGE REQUEST

- Page in buffer pool:
  - return the address to the frame
  - increment the pin count
- Page not in the buffer pool:
  - choose a frame for replacement
  - if frame is dirty, write it to disk
  - read requested page into chosen frame
  - pin the page and return the address

Buffer Replacement Policy

- How do we choose a frame for replacement?
  - LRU (**L**east **R**ecently **U**sed)
  - Clock
  - MRU (**M**ost **R**ecently **U**sed)
  - FIFO, random, …

- The replacement policy has big impact on # of I/O's (depends on the access pattern)

LRU

LRU (**L**east **R**ecently **U**sed)

- queue of pointers to frames with pin count 0
- add to end of queue, grab frames from front of queue

EXAMPLE

- Buffer pool with 3 frames
- 5 pages in disk: A, B, C, D, E
- Sequence of requests:
  - request A, modify A, request B, request B, release A, request C, release B, request D, modify D, release B, request A, request E

CLOCK

- Variant of LRU with lower overhead
- The N frames are organized into a cycle
- Each frame has a referenced bit that is set to 1 when pin count is 0
- A current variable points to a frame
- When a frame is considered:
  - If pin count > 0, increment current
  - If referenced = 1, set to 0 and increment
  - If referenced = 0 and pin count = 0, choose the page

SEQUENTIAL FLOODING

- Nasty situation caused by LRU + repeated sequential scans
  - # buffer frames < # pages in file
  - each page request causes an I/O !!
  - MRU much better in this situation

DBMS vs OS File System

Why not let the OS handle disk management?

- DBMS better at predicting the reference patterns
- Buffer management in DBMS requires ability to:
  - pin a page in buffer pool
  - force a page to disk (for recovery & concurrency)
  - adjust the replacement policy
  - pre-fetch pages based on predictable access patterns
- can better control the overlap of I/O with computation
- can leverage multiple disks more effective

RECAP

How a DBMS stores data:

- disk, main memory
- files, pages

Buffer manager:

- Controls how the data moves between main memory and disk
- Various replacement policies (LRU, Clock, etc)